Науковий пошук молодих дослідників

Збірник наукових праць студентів

№ 8

2014

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ЗАКЛАД
«ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА»

СТУДЕНТСЬКЕ НАУКОВЕ ТОВАРИСТВО

НАУКОВИЙ ПОШУК МОЛОДИХ ДОСЛІДНИКІВ

№ 8, 2014

Фізико-математичні та технічні науки

Збірник наукових праць студентів

Луганськ
ДЗ «ЛНУ імені Тараса Шевченка»
2014

У збірнику розкриваються напрямки наукових досліджень студентів з фізико-математичних та технічних наук.

Рекомендовано до друку Вченою радою
Луганського національного університету
імені Тараса Шевченка
(протокол № 9 від 25 квітня 2014 р.)

Видавництво Державного закладу
«Луганський національний університет імені Тараса Шевченка»
вул. Оборонна, 2, м. Луганськ, 91011. Тел./факс: (0642) 58-03-20.
e-mail: alma-mater@list.ru

ЗМІСТ

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ЕКОНОМІЦІ ТА ОСВІТІ

ТЕХНІКА

ФІЗИКА ТА МАТЕМАТИКА

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ЕКОНОМІЦІ ТА ОСВІТІ

УДК 378.016:004.4

В. Ю. Донченко, М. А. Габієв

ВІЗУАЛІЗАТОРИ АЛГОРИТМІВ СОРТУВАННЯ ЯК ЕЛЕМЕНТ ТЕХНОЛОГІЇ ВИКЛАДАННЯ ПРОГРАМУВАННЯ

Вирішення питань інформатизації навчання різних предметів, зокрема алгоритмізації та програмування, пов'язано з визначенням теоретичних засад організації процесу навчання із застосуванням ІКТ у психолого-педагогічному, дидактичному й методичному аспектах. Насамперед необхідними є наукове обґрунтування, розробка та апробація відповідних засобів навчання алгоритмізації та програмування і способів ефективного поєднання ІКТ із різними методами й формами організації навчання.

Потребують вивчення також методичні засади процесу організації навчальної діяльності студентів на заняттях з алгоритмізації та програмування із використанням ІКТ у контексті формування в них алгоритмічної компетентності.

Особливого значення при навчанні алгоритмізації та програмування набуває *принцип наочності.* За рахунок продуманого застосування засобів наочності можна посилити емоційний вплив на студентів, підвищити рівень доступності матеріалу, що вивчається, прискорити активізацію розумової діяльності студентів.

Якщо для удосконалення змісту освіти найважливішими задачами є її оптимізація, систематизація та проблематизація, для удосконалення методики навчання необхідна його індивідуалізація, диференциація та активізація, то найважливішим напрямом удосконалення дидактичних засобів визнається візуалізація. ЮНЕСКО в 2003 році об'явила про приорітет візуальної подачі матеріалу в освіті.

Наочність у навчанні – один з основних принципів дидактики, відповідно до якого навчання будується на конкретних образах, що безпосередньо сприймаються учнями.

Використання методу демонстраційних прикладів при навчанні програмуванню базується на концепції відомого методиста в навчанні програмуванню Н.Вірта «Програмування – це мистецтво конструювання. Як можна навчити конструкторській, винахідницькій діяльності. Існує такий метод виділити найпростіші будівельні блоки із багатьох вже існуючих програм і дати їх систематичний опис. Але програмування

являє собою велику і різнопланову діяльність, яка часто потребує складної розумової праці.

Помилково вважати, що її можна звести лише до використання готових рецептів. За метод навчання нам лишається обрати ретельний добір і розгляд характерних прикладів. Зрозуміло, не слід вважати, що вивчення прикладів усім однаково корисно. При цьому підході багато залежить від кмітливості й інтуїції учня».

Проблемам ролі візуалізації в підвищенні мотивації та активізації навчальної діяльності з алгоритмізації та програмування присвячені роботи М.С.Львова, Н.В.Морзе, О.В.Співаковського та ін.

Метою статті є аналіз вимог до візуалізаторів алгоритмів сортування для використання в навчальному процесі.

При вивченні алгоритмів обробки інформації, які представлені різними структурами даних [1,2], важливу роль відіграють візуалізатори алгоритмів, що дозволяють в наочній формі динамічно відображати деталі їхньої роботи. Це відкриває можливість використання новітніх технологій при при вивченні програмування [3].

Візуалізатор - це програма, в процесі роботи якої на екрані комп'ютера динамічно демонструється застосування алгоритму до вибраного набору даних. Візуалізатори дозволяють вивчати роботу алгоритмів в покроковому режимі, аналогічному режиму трасування програм.

Вони при необхідності допускають трасування укрупненими кроками, ігноруючи рутинну частину обчислювального процесу, що істотно, наприклад, для переборних алгоритмів.

Для деяких алгоритмів динамічний варіант демонстрації його роботи є більш природним, ніж набір статичних ілюстрацій. Для алгоритмів сортування візуалізація дозволяє наочно продемонструвати як загальний підхід, так і відмінність у механізмах їх дії.

Перерахуємо відмітні характеристики візуалізаторів.

1. Простота використання, обумовлена зрозумілістю інтерфейсу. Тому для роботи з візуалізатором зазвичай не потрібна спеціальна підготовка.
2. Чіткість і простота представлення візуалізуємого процесу.
3. Компактність візуалізаторів. Це при необхідності спрощує передачу візуалізаторів в мережі Інтернет, що особливо важливо при дистанційному навчанні.

Візуалізатори в розглянутій області вирішують наступні завдання, що виникають в процесі навчання:

1. Графічне та текстове роз'яснення дій алгоритму на конкретних наборах вхідних даних. При цьому розуміння алгоритму не потрібно, оскільки саме візуалізатор повинен пояснити дію алгоритму.
2. Надання користувачеві інструменту, що реалізує данний алгоритм. У результаті учень звільняється від необхідності виконувати кроки алгоритму, так як їх автоматично виконує візуалізатор.

Візуалізатор виконує зазвичай такі функції.

1. Покрокове виконання алгоритму.
2. Перегляд дії алгоритму при різних наборах вхідних даних, у тому числі і введених користувачем.
3. Перегляд дії алгоритму в динаміці.
4. Перезапуск алгоритму на поточному наборі вхідних даних.

Динамічна візуалізація наочно демонструє таку характеристику алгоритму, як трудомісткість (особливо при пошаговій демонстрації). Для деяких алгоритмів (наприклад, машини Тюринга) динамічний варіант демонстрації взагалі представляється більш натуральним, ніж будь-який набір статичних ілюстрацій.

У завдання візуалізатора алгоритму входить:

1. Відображення вхідних і вихідних даних у наочній формі-уявлення абстрактного поняття дано в відекартинках. Таке уявлення необхідно, як для учнів з початковим рівнем підготовки, так і для більш продвинутих учнів з метою спрощення сприйняття.
2. Відображення внутрішніх службових змінних.
3. Відображення процесу впливу алгоритму на вхідні дані та внутрішні змінні:
   a) Зміна даних.
   b) Копіювання і переміщення даних.
   c) Процес прийняття рішень.
4. Коментарі до кожної дії алгоритму.
5. Відображення роботи алгоритму по кроках.
6. Можливість введення даних з метою апробації алгоритму на різних наборах вхідних даних.

Для використання в навчальному процесі візуалізатори алгоритмів повинні задовольняти наступним вимогам:

1. Керованість - в учнів повинна бути можливість задавати власні набори вхідних даних і розглядати роботу алгоритму на них.
2. Інтерактивність - при роботі з візуалізатором повинна бути можливість здійснювати кроки як вперед, так і назад, також повинен бути передбачений режим автоматичної візуалізації.
3. Історія - візуалізатор повинен дозволяти зробити скільки завгодно багато кроків назад, відображаючи стан алгоритму на відповідний момент.
4. Можливість відображення ходу виконання алгоритму.
5. Можливість коментування виконання програми.
6. Простота використання - візуалізатор повинен бути зрозумілим для непідготовленого користувача.
7. Доступність – в учнів повинна бути можливість доступу до візуалізаторів не тільки на заняттях.
8. Платформонезалежність - візуалізатор повинен працювати незалежно як від платформи (IBM-PC сумісні комп`ютери, Apple Macintosh) так і операційної системи (Windows, Unix, Linux).

9. Незалежність від мережі - для роботи візуалізатора не повинен бути потрібен доступ до мережі.

10. Зручність створення візуалізаторів алгоритмів – простота використання системи візуалізаторів з точки зору написання нових візуалізаторів.

11. Керованість, інтерактивність і історія дозволяє більш глибоко вивчити роботу алгоритму, ніж візуалізація на заздалегідь обраних наборах данних.

Простота використання важлива при самостійному навчанні, так як в цьому випадку користувачі дуже рідко читають інструкції по застосуванню. Можливості відображення шагу алгоритму і коментування шагу программи є дуже важливими. Перша - для пояснення простих алгоритмів і введення в інформатику, а друга - для розуміння складних алгоритмів. Доступність, платформонезалежних і незалежність від мережі дозволяє використовувати візуалізатори як на лекціях і практичних заняттях, так і для самостійного навчання.

Зробимо огляд візуалізаторів алгоритмів на прикладі алгоритмів сортування, так як вони є однією з основоположних тем при навчанні алгоритмізації та програмування і входять як важлива складова частина до багатьох алгоритмів.

Розглянемо кілька підходів до візуалізації алгоритмів сортування:

1. «Числовий» - сортований масив відображається набором чисел, які містяться в ньому. При порівнянні і обміні елементів вони підсвічуються. У деяких випадках застосовується анімація обміну елементів [4];

2. «Гістограммний» - сортований масив зображується у вигляді гістограми. При цьому обмін елементів візуалізується як фізичне переміщення стовпців, відповідних чисел [5];

3. «Історичний» - весь процес сортування представляється у вигляді одного зображення. При цьому зазвичай значення елементів відображають кольором [4].

На рис. 1. наведені приклади зображень, які генерують візуалізатори, що використовують описані підходи.

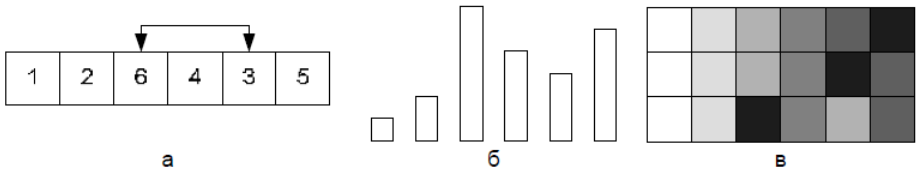


Рис. 1.1. Підходи до візуалізації алгоритмів сортування «числовий» (а), «гістограммний» (б) та «історичний» (в)

Зазвичай візуалізується наступний «стандартний» набір алгоритмів сортування: обмінна, бульбашкова, простого вибору, простої вставки, пірамідальна, злиттям і швидка [2].

Можливості відображення ходу алгоритму і коментування ходу програми є дуже важливими. Перша - для пояснення простих алгоритмів і введення в інформатику, а друга - для розуміння складних алгоритмів.

Доступність, платформонезалежних і автономність дозволяють використовувати візуалізатори як на лекціях і практичних заняттях, так для самостійного навчання.

В результаті візуальної підтримки алгоритмічної підготовки студентів краще засвоюється навчальний матеріал, його усвідомлення відбувається діяльнісним шляхом, і як наслідок, студенти знаходять можливості його широкого практичного застосування.

Використання візуалізаторів при формуванні алгоритмічних компетенцій дозволяє розвивати пізнавальні можливості студентів, вміння самостійно аналізувати та інтерпретувати результати, спонукає до дослідницької діяльності. Таким чином, у студентів формуються якісно нові професійно значимі вміння та навички, реалізується підготовка майбутнього спеціаліста для успішної професійної діяльності.

Література

1. Кнут Д. Искусство программирования. Том 1. Основные алгоритмы. / Д. Кнут - М.: Вильямс, 2000. **2. Кормен Т.** Алгоритмы. Построение и анализ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест - М.: МЦНМО, 1999. **3. Казаков М.А.** Визуализаторы алгоритмов как элемент технологии преподавания дискретной математики и программирования./ М.А. Казаков, С.Е. Столяр // Тезисы докладов международной научно-методической конференции "Телематика-2000". СПб.: СПбГИТМО (ТУ), 2000. **4. Brown M.**, Sedgewick R. A system for Algorithm Animation / Computer Graphics, Proceedings of the 11th annual conference on Computer graphics and interactive techniques, July 1984. **5. Demetrescu C.**, Finocchi I., Stasko J. Specifying Algorithm Visualizations: Interesting Events or State Mapping? // Proceedings of the International Dagstuhl Seminar on Software Visualization, Schloss Dagstuhl, May 2001, appears in Software Visualization State-of-the-Art Survey, LNCS 2269, Stephan Diehl (ed.), Springer Verlag, 2002.

Донченко В. Ю., Габієв М. А. Візуалізатори алгоритмів сортування як елемент технології викладання програмування

У статті розглянуто опис поточного стану в області визуализаторів і систем візуалізації. Зокрема, розглянуто застосування визуализаторів в навчальному процесі і виникаючі при цьому вимоги до візуалізаторів. Розглядається кілька підходів до візуалізації алгоритмів сортування.

Ключові слова: принцип наочності, візуальне середовище навчання, алгоритмічна підготовка, візуалізатор.

Донченко В. Ю., Габиев М. А. Визуализаторы алгоритмов сортировок как элемент технологии преподавания программирования

В статье рассмотрено описание текущего состояния в области визуализаторов и систем визуализации. В частности, рассмотрено применение визуализаторов в учебном процессе и возникающие при этом требования к визуализаторам. Рассматривается несколько подходов к визуализации алгоритмов сортировки.

Ключевые слова: принцип наглядности, визуальная среда обучения, алгоритмическая подготовка, визуализатор.

Donchenko V. Yu., Gabiev M. A. Visualization of sorting algorithms as part of teaching programming technology

In the article the description of the current state in the field of visualization and imaging systems. In particular, consider the use of visualization in the learning process and the concomitant requirements for visualizers. We consider several approaches to visualizing sorting algorithms.

Keywords: principle of clarity, the visual learning environment, algorithmic training visualizer.

УДК 00-4.942

В. Ю. Донченко Ю. В. Сыч

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ИНТЕРАКТИВНОЙ ТРИАНГУЛЯЦИИ

Большинство численных методов исследования напряженно-деформированного состояния деформируемого тела базируются на идее перехода от континуальной задачи к дискретной, когда исследуемая сплошная область заменяется некоторой конечной дискретной моделью. В методе конечных элементов (МКЭ) непрерывная область заменяется некоторой непересекающейся совокупностью конечных элементов, заполняющих весь объем тела.

Одна из главных проблем, возникающих при применении МКЭ – это построение дискретной модели исследуемой механической системы. Одной из главных частей любого программного комплекса численного анализа является программа, автоматизирующая построение геометрической модели исследуемого объекта с последующей ее дискретизацией на конечные элементы.

Проблема оптимальной дискретизации исследуемой области на конечные элементы в общем виде является весьма сложной (особенно для трехмерных областей). Это обусловлено тем, что на форму конечных элементов (КЭ) налагаются два основных ограничения: они не должны иметь слишком малых (или соответственно слишком больших) углов и объем КЭ не должен превышать некоторую наперед заданную величину. В первом случае при расчетах возникают значительные вычислительные погрешности. Во втором появляется риск потери точности вычислений при значительном изменении градиента исследуемой функции (например, в зоне предполагаемого концентратора напряжений).

Поэтому автоматическая генерация КЭ-сети представляет собой весьма сложную процедуру, являющуюся основой любого конечно-элементного пакета программ. На практике чаще используются КЭ в форме треугольника, прямоугольника, тетраэдра или параллелепипеда, т.к. они позволяют с высокой степенью точности аппроксимировать область произвольной формы.

Методы построения поверхностных сетей

Под сетью понимается топологическое множество точек, связанных между собой ребрами – отрезками прямых (кривых) линий так, что исходная область разбивается на элементы определенной формы. Методы построения треугольных сеток можно разбить на три большие группы [1]: ячеечный **(cell-based)**, предиктора-корректора **(predictor-corrector)** и мозаичные **(pre-tessellation methods & particle-based methods)**.

В методах ячеечного типа происходит разбиение области триангуляции на ячейки - параллелепипеды или треугольные пирамиды. Далее производится триангуляция поверхности в каждой ячейке отдельно. Причем каждая ячейка триангулируется одним из заданных ранее способов, т.е. значения координат для треугольников просто «подставляются» из заранее заданной таблицы.

Для применения методов этого типа необходимо задать допустимую ошибку аппроксимации, на основе которой выбрать размер ячейки - куба или тетраэдра (если быть точным - то треугольной пирамиды, т.к. тетраэдрами нельзя «замостить» пространство без пропусков и наложений). После этого с помощью уже известных таблиц триангуляции получить искомое множество треугольников. При этом процедура триангуляции каждой ячейки сводится к анализу значений функции в вершинах этой ячейки - другими словами, определяется, какие вершин лежат «внутри» поверхности, а какие - «снаружи». На основе этого можно сделать вывод о достаточности определения функции только в вершинах ячеек.

Наиболее известные ячеечные алгоритмы - Канейро, «MT6», Скалы, «Марширующие кубы». Алгоритм Скалы генерирует гораздо большее количество треугольников, чем алгоритм «Марширующие кубы». В алгоритме «MT6» выше количество генерируемых

треугольников, а средняя мера качества треугольников ниже, чем в алгоритме Канейро. Алгоритм «Марширующие кубы» имеет лучшие показатели по критериям количества сгенерированных треугольников и их качества.

Группа методов типа предиктор-корректор основана на добавлении к уже имеющемуся множеству точек триангуляции ещё одной, лежащей на касательной плоскости к заданной функции (это положение предиктора(predictor) - предсказанное) и затем передвижению её до визуализируемой поверхности (это положение корректора (corrector) - скорректированное).

При использовании методов из этого класса, необходимо знать значение функции во всех точках пространства и найти хотя бы одну точку, принадлежащую искомой поверхности. Метод заключается в «наращивании» треугольников - на каждой итерации метода к уже существующему множеству треугольников добавляется еще один, построенный на ребре крайнего треугольника и предсказанной (а затем скорректированной по кривизне поверхности) точки на поверхности.

Суть группы методов мозаичного типа заключается в разбиении искомой поверхности на части дальнейшей их триангуляции. Разбиение на части в pre- tessellation методах подразумевает разбиение поверхности на «примитивные » поверхности - фрагменты сфер и плоскостей. Разбиение на части в методах из плеяды particle-based - менее «интеллектуально» - ищутся только фрагменты плоскостей. При этом возникает проблема «соединения» уже «протриангулированных» частей. Чаще всего этот процесс сводится к подбору локальных по Делоне треугольников, соединяющих части искомой поверхности. Определение: треугольник локален по Делоне, если его самая маленькая сфера ограничения не содержит никакую другую точку триангуляции, которая имеет ту же самую поверхностную ориентацию.

Достоинства и недостатки методов триангуляции

Основа методов первого типа - независимая триангуляция каждой ячейки с помощью таблиц триангуляции - является одновременно их сильной и слабой стороной. Высокая скорость работы этих методов делает их наиболее привлекательными по отношению к другим методам и дает возможность использовать их в интерактивных приложениях, но большим минусом считается их относительная индифферентность к поведению функции вне выбранного множества точек. Другими словами это невозможность правильно визуализировать локальные искривления - масштаб треугольников всегда пропорционален размеру ячейки. Такие методы идеально подходят для визуализации трехмерных скалярных полей, заданных на регулярной сетке.

Методы второго и третьего типа применимы только при визуализации полей определенных в каждой точке той части пространства, которое нас интересует. Большим плюсом таких методов

можно считать их зависимость от локального искривления функции - в таких методах мелкие детали не «пропадут». Несмотря на сильную потерю в скорости по сравнению с методами первой группы и ограничениями на дифференцируемость функции и связность поверхности, привлекают высоким «качеством» получаемой поверхности.

Заметное различие между ячеечными методами и методами второго и третьего типа заключается еще и в том, что методы первого типа часто довольно просты в реализации, и предоставляют возможность визуализации «нетривиально» заданных скалярных полей. Так, к примеру, создать регулярную сетку на основе нерегулярной значительно проще, нежели восстановить функцию в каждой точке пространства. Это же относится и к проблеме восстановления поверхности по «срезам.

Методы слияния двух триангуляций

Слияние двух частичных триангуляций является весьма сложной задачей. Пусть необходимо объединить две выпуклые триангуляции. Тогда, учитывая выпуклость каждой, можно использовать следующие алгоритмы: «Удаляй и строй», «Строй и перестраивай», «Строй, перестраивая» [2].

В алгоритме «Удаляй и строй» вначале для двух триангуляций находятся две общие касательные, первая из которых становится текущей базовой линией. Затем от базовой линии начинается заполнение треугольниками промежутка между триангуляциями. К недостаткам данного алгоритма слияния относятся относительно большие затраты времени на поиск очередного узла-соседа. В целом трудоемкость алгоритма O(n) относительно общего количества точек в двух объединяемых триангулированных поверхностях.

В алгоритме «Строй и перестраивай» вначале строятся заполняющие зону слияния треугольники между касательными, а затем вновь построенные треугольники проверяются согласно выбранному условию построения с другими треугольниками. В целом трудоемкость в худшем случае O(n) относительно общего количества точек двух сливаемых сетей, а в среднем – O(n) или O(m), где m – общее количество точек вдоль границ сливаемых триангуляций.

Процедура слияния триангуляций «Строй, перестраивая» отличается от предыдущей тем, что перестроения выполняются не на втором этапе работы, а непосредственно после построения очередного треугольника. В целом этот алгоритм имеет такую же трудоемкость, что и предыдущий, но на практике работает немного медленнее.

Оценка качества и оптимизация сети

Для сравнения эффективности различных методов дискретизации необходим критерий качества построенной сетки. Критерий связывают с аппроксимационными свойствами сетки. Эти свойства в основном зависят от формы элементов. В частности, в оценки погрешности аппроксимации конечным элементом ω, как правило,

входит величина отношениярадиуса вписанного в ω шара к диаметру ω [3].

Прямое нахождение этой величины весьма трудоемко, и на практике используют различные альтернативные оценки. Оптимальной с точки зрения точности, полноты оценки качества сетки и удобства нахождения является следующая оценка [3]:

$$\mu = \frac{V}{abc}, \qquad (1)$$

где V – объем тетраэдра, а abc – наибольшее из произведений длин тройки ребер, выходящих из одной вершины.

Поскольку величина μ имеет порядок десятых и сотых, для наглядности ее удобно относить к значению идеального случая – правильного тетраэдра (тогда эта величина равна $\sqrt{2}/12 \approx 0{,}118$). Отношение μ из (1) к идеальному значению является "аппроксимационной характеристикой" (АХ) элемента [3]. Возможные значения АХ лежат в пределах от 0 до 1; чем ближе к 1, тем лучше.

Некоторые методы построения сеток не способны дать достаточно хорошую сетку для проведения анализа. Поэтому применяют трехэтапный метод улучшения качества сетки:

1. Если построенные элементы относятся не к тому типу, который требовался, то их разбивают на элементы нужного типа.
2. Если размер элементов не соответствует нужному распределению плотности ячеек, то они разбиваются на более мелкие.
3. Если форма элементов недостаточно хороша, применяют методы сглаживания сетки.

Результаты моделирования

В примере реализации алгоритма слияния "Строй и перестраивай" двух триангулированных поверхностей в качестве входных данных можно использовать множество узлов для разбиения с трехмерными координатами, либо задать набор точек на плоскости Oxy для их дальнейшего отображения в трехмерное пространство. Соответствующее поднимающее отображение $f = R^2 \to R^3$ имеет вид: $f(x)=(x_1,x_2,y(x_1,x_2))$.

Для создания проекта была выбрана структура данных "Узлы и треугольники" как наиболее удобная и достаточно простая в реализации.

Так как поверхность задается конечным набором точек, то удобно строить триангуляцию Делоне. Разбиение осуществляется итеративным алгоритмом "Удаляй и строй" [2] с использованием суперструктуры. Итеративный процесс предусматривает наращивание текущей триангуляции по одному треугольнику за один шаг, удовлетворяющему условию Делоне. Трудоемкость алгоритма равна $O(N^2)$.

Проводилась серия генераций сети на произвольно заданном наборе точек. Так, например, рассматривалась модель параболоида, чем большее количество точек задавалось, тем точнее и реалистичнее

получалось изображение. Каркасную модель можно использовать для проведения различных расчетов и вычислений, а сплошную – для представления полученной картины зрителю.

Далее проводилась генерация двух сетей с последующей операцией слияния. Результаты работы продемонстрированы на рисунке 1.

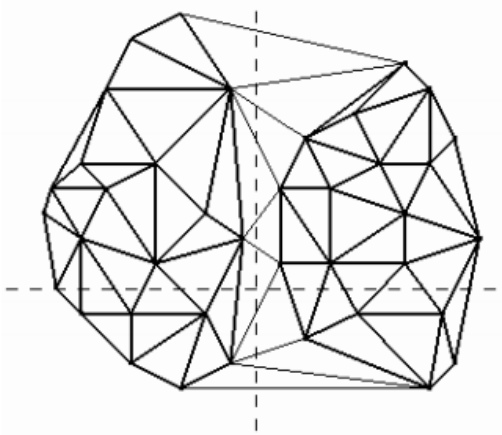


Рис. 1.Слияние двух триангулированных поверхностей, проекция на плоскость Oxy.

Литература

1. Семенихин А. Сравнительный анализ методов интерактивной триангуляции сеточных функций / А. Семенихин. Компьютерная графика и мультимедиа, 2004, №2(2). http://cgm.computergraphics.ru/content /view/63 **2. Скворцов А.В.** Триангуляция Делоне и ее применение / А.В. Скворцов Томск, Изд-во Том. ун-та, 2002. С. 128. **3. Frey P.J.** Mesh Generation Application to Finite Elements / P.J. Frey, P.L. George HERMES Science Europe Ltd, 2000.

Донченко В. Ю., Сыч Ю. В. Сравнительный анализ методов интерактивной триангуляции

В статье рассматриваются различные ячеечные алгоритмы визуализации трехмерных скалярных полей заданных на регулярной сетке. Проводится подробное исследование и описание методов визуализации, отмечаются преимущества и недостатки тех или иных подходов.

Ключевые слова: метод триангуляции, оптимизация сети, генерация сети.

Донченко В. Ю., Сич Ю. В. Порівняльний аналіз методів інтерактивної тріангуляції

У статті розглядаються різні коміркові алгоритми візуалізації тривимірних скалярних полів заданих на регулярній сітці. Проводиться

докладне дослідження і опис методів візуалізації, відзначаються переваги та недоліки тих чи інших підходів.

Ключові слова: метод тріангуляції, оптимізація мережі, генерація сіток.

Donchenko V. Yu., Sych Y. V. Comparative analysis of methods for interactive triangulation

In the article the different cell-type algorithms of visualization of the three-dimensional scalar fields are examined set on a regular net. The detailed research and description of methods of visualization is conducted, advantages and lacks of one or another approaches are marked.

Key words: method of triangulation, optimization of network, generations of network.

УДК 519.67:004.942

В. Ю. Донченко Е.Н. Черевань

ОБЗОР И АНАЛИЗ МЕТОДОВ ПОСТРОЕНИЯ ГЕОМЕТРИЧЕСКИХ МОДЕЛЕЙ СЛОЖНЫХ КОНСТРУКЦИЙ

Методы и программы для численного решения задач механики деформированного твёрдого тела развиваются на протяжении многих десятилетий. Развитие компьютерных технологий сделало вычислительный эксперимент мощным и эффективным способом решения проблем прикладной математики. Появление новых математических моделей и вычислительных методов, увеличение объема и сложности современных вычислительных задач, делают необходимым создание новых средств автоматизации построения компьютерных моделей и подготовки исходных расчётных данных.

Использование современной вычислительной техники для решения задач математического моделирования делает необходимым разработку и внедрение комплексного подхода к их решению. Программные средства, разрабатываемые для этой цели, должны обеспечивать поддержку различных видов вычислительной работы таких, как построение геометрической модели области решения задачи, генерацию расчетной сетки, составление системы сеточных уравнений для принятой математической модели, выполнение расчёта и обработку полученных данных. К программным средствам автоматизации научных исследований предъявляются требования робастности и устойчивости при изменении входных параметров математических моделей в широком диапазоне значений, а также при переходе от геометрически простых расчетных областей к сложным, многосвязным, разномасштабным областям.

Математические модели механических процессов обычно описываются дифференциальными, интегральными уравнениями или в виде вариационных принципов. Для решения прикладных задач в соответствующих постановках используются, в основном, численные методы, из которых наибольшее распространение получил метод конечных элементов (МКЭ).

В решении общей вычислительной задачи с использованием МКЭ можно выделить три этапа: предобработку, расчёт и постобработку. Первый этап состоит в задании геометрии расчётной области и определении начальных и граничных условий. Далее производится построение расчётной сетки для заданной геометрии.

Выбор типа сетки определяется характером решаемой задачи, геометрией области и требуемым объемом вычислений. Следующий этап – собственно расчёт, результатом которого являются некоторые распределения физических величин по элементам сетки. Заключительный этап предполагает постобработку результатов расчёта, нахождение физических и вспомогательных геометрических характеристик, а также их визуализацию.

Первый и третий этапы являются вспомогательными по отношению к основной задаче решения уравнений механики деформированного твёрдого тела. Однако, одним из важнейших элементов численного расчета напряженно-деформированного состояния (НДС) деформируемого тела является построение адекватной геометрической модели исследуемой области в конструкциях сложной формы. От точности построенной геометрической модели во многом будет зависеть качество полученного численного результата. Поэтому, вопросы подготовки и анализа геометрических данных настолько усложняются, что превращаются в самостоятельные проблемы, для эффективного решения которых требуются специальные средства автоматизации. Разработка таких средств является одной из проблем.

Вопросы геометрического моделирования и вычислительной топологии являются во многих случаях определяющими для систем автоматического проектирования (САПР), гео-информатики и всевозможных графических пользовательских интерфейсов (ГПИ), а их методология и аспекты программных реализаций стали предметами самостоятельного изучения [1,2].

Возможность задания реальной геометрии механических объектов – важнейшее требование к прикладному программному обеспечению.

Геометрическая информация формируется путем создания геометрических моделей, которые могут быть как двумерными, так и трехмерными.

Трехмерные модели подразделяются на:

1. каркасные или «проволочные» модели;
2. поверхностные модели;

3. объемные твердотельные модели или модели сплошных тел.

Каркасное моделирование

К первому и наиболее простому типу, объектно-ориентированного конструирования, относится каркасное моделирование низкого уровня. Каркасная модель полностью описывается в терминах точек и линий. Это моделирование самого низкого уровня и имеет ряд серьезных ограничений, большинство из которых возникает из-за недостатка информации о гранях, которые заключены между линиями, и невозможности выделить внутреннюю и внешнюю область изображения твердого объемного тела.

Однако каркасная модель требует меньше памяти и вполне пригодна для решения задач, относящихся к простым. Каркасное представление часто используется не при моделировании, а при отображении моделей как один из методов визуализации.

Наиболее широко каркасное моделирование используется для имитации траектории движения инструмента, выполняющего несложные операции по 2.5 или 3 осям. Понятие 2.5 оси связано с тем, что более простые системы могут обрабатывать информацию о формах только с постоянным поперечным сечением. Такую форму можно построить следующим образом - сначала создается вид , а затем каждой точке приписываются два значения координаты , характеризующие глубину изображения. Объекты, получаемые в результате данного типа визуального воспроизведения, называются каркасными или проволочными, которые в свою очередь состоят из связанных между собой наборов формообразующих линий, сегментов и дуг. Модели такого типа, не содержат информации о поверхности, объёме структурного предмета и используются в основной своей массе как один из методов визуализации.

Одним из преимуществ каркасных трёхмерных моделей, является минимальный объём занимаемой оперативной памяти компьютера. Каркасная визуализация часто используется для имитации траектории движения инструмента, в специальных CAM системах подготовки управляющих алгоритмов для машин с числовым программным управлением.

Недостатки каркасной модели:

1. Неоднозначность - для того, чтобы представить модель в каркасном виде, нужно представить все ребра (это эффект может привести к непредсказуемым результатам. Нельзя отличить видимые грани от невидимых. Операцию по удалению невидимых линий можно выполнить только в ручную с применением команд редактирования каждой отдельной линии, но результат этой работы равносилен разрушению всей созданной каркасной конструкции, т.к. линии невидимы в одном виде и видимы в другом);

2. Невозможность распознавания криволинейных граней - мнимые ребра (боковые поверхности цилиндрической формы реально не имеют ребер, хотя на изображении есть изображение некоторых мнимых ребер, которые ограничивают такие поверхности. Расположение этих мнимых ребер меняется в зависимости от направления вида, поэтому эти силуэты не распознаются как элементы каркасной модели и не отображаются на них);

3. Невозможность обнаружить взаимное влияние компонент (каркасная модель не несет информации о поверхностях, ограничивающих форму, что обуславливает невозможность обнаружения нежелательных взаимодействий между гранями объекта и существенно ограничивает использование каркасной модели в пакетах, имитирующих траекторию движения инструмента или имитацию функционирования робота, так как при таком моделировании не могут быть выявлены на стадии проектирования многие коллизии, появляющиеся при механической сборке);

4. Трудности, связанные с вычислением физических характеристик;

5. Отсутствие средств выполнения тоновых изображений (основным принципом техники выполнения тоновых изображений, т.е. обеспечение плавных переходов различных цветов и нанесение светотени, является то, что затенению подвергаются грани, а не ребра).

Поверхностное моделирование

Поверхностное моделирование в отличие от каркасного построения, помимо точек и линий входящих в состав основополагающих элементов объекта, в свой состав включают поверхности, которые образуют визуальный контур отображаемой фигуры. При разработке таких форм предполагается, что геометрические объекты ограничены наружными сторонами предмета, которые отделяют их от окружающего пространства.

Поверхностное моделирование имеет следующие преимущества по сравнению с каркасным:

1. способность распознавания и изображения сложных криволинейных граней;
2. изображение грани для получения тоновых изображений;
3. особые построения на поверхности (отверстия);
4. возможность получения качественного изображения;
5. обеспечение более эффективных средств для имитации функционирования роботов.

В основу поверхностной модели положены два основных математических положения:

1. Любую поверхность можно аппроксимировать многогранником, каждая грань которого является простейшим плоским многоугольником;

2. Наряду с плоскими многоугольниками в модели допускаются поверхности второго порядка и аналитически неописываемые поверхности, форму которых можно определить с помощью различных методов аппроксимации и интерполяции.

В отличие от каркасного моделирования каждый объект имеет внутреннюю и внешнюю часть.

Твердотельное моделирование

Твердотельное моделирование- это самое полное и самое достоверное построение реального объекта. Результатом построения геометрического тела таким методом является монолитный образец нового изделия, который включает в свой состав такие компоненты как линии, грани, а самое главное, создаётся участок поверхности в пределах геометрической формы объекта с такими важными параметрами как масса тела и объём.

Для работы с трёхмерными моделями используются специальные программы, обеспечивающие компьютерную поддержку проектирования.

Модели, получаемые вышеперечисленными программами по сути одинаковые. Твердотельная модель или сетчатая модель остаётся таковой не зависимо от программного продукта но, тем не менее, в виду отличая форматов файла несущего информацию об объекте, его не всегда можно открыть на сторонней программе.

Преимущества твердотельных моделей:

1. Полное определение объемной формы с возможностью разграничивать внутренний и внешние области объекта, что необходимо для взаимовлияний компонент.
2. Обеспечение автоматического удаления скрытых линий.
3. Автоматическое построение 3D разрезов компонентов, что особенно важно при анализе сложных сборочных изделий.
4. Применение методов анализа с автоматическим получением изображения точных весовых характеристик методом конечных элементов.
5. Получение тоновых эффектов, манипуляции с источниками света.

Анализ типов трехмерных геометрических моделей показывает, что модели поверхностей могут с успехом сочетаться с остальными типами геометрических моделей, как каркасными, так и объемными твердотельными, а иногда являются обязательными (авиация, судостроение, автомобилестроение). Поэтому способы генерации и модифицирования поверхностей чрезвычайно важны в процессе геометрического моделирования. При всем многообразии методов построения и модификации поверхностей, а именно: методы аналитических представлений, метод каркасов или плазовый метод (Осипов В.А.), параметрическое представление поверхностей (методы Кунса и Безье), методы сплайновой геометрии (Василенко В.А., Вермель

В.Д., Завьялов ЮС, Корнишин М.С., Стечкин СБ., Субботин Ю.Н., Якунин В.И.), B-сплайны и их дальнейшее развитие - NURBS (Non-Uniform Rational B-Splines) аппроксимация.

Применяют следующие подходы к построению геометрических моделей:

1. Задание граничных элементов – граней, ребер, вершин.
2. Кинематический метод, согласно которому задают двумерный контур и траекторию его перемещения; след от перемещения контура принимают в качестве поверхности детали.
3. Позиционный подход, в соответствии с которым рассматриваемое пространство разбивают на ячейки (позиции) и деталь задают указанием ячеек, принадлежащих детали; данный метод является очень громоздким и используется редко.
4. Метод конструктивной геометрии – представление сложной детали в виде совокупностей базовых элементов формы и выполняемых над ними теоретико-множественных операций.

В памяти ЭВМ рассмотренные модели обычно хранятся в векторной форме, т.е. в виде координат совокупности точек, задающих элементы модели. Операции конструирования также выполняются над моделями в векторной форме. Наиболее компактна модель в виде совокупности связанных базовых элементов формы, которая преимущественно и используется для хранения и обработки информации об изделиях в системах конструктивной геометрии.

Одним из распространенных методов описания пространственных тел, задаваемых с помощью параметрических кривых и поверхностей, является метод, основанный на граничном представлении. Граничное представление удобно применять для подготовки данных при решении задач механики деформированного твёрдого тела, т.к. сами границы в этом случае определяются функционально. Выбор вида параметрического представления производится из соображений универсальности. В качестве такого рода геометрических примитивов удобно использовать NURBS кривые и поверхности, позволяющие описывать объекты, как с криволинейными, так и с прямолинейными границами. NURBS широко применяются для задания кривых и поверхностей во многих современных CAD системах.

При задании геометрии также определяются граничные условия, каждое из которых сопоставляется определенной геометрической сущности. В частности, в двумерном случае условия ставятся в соответствие контурным кривым, а в трехмерном – граничным поверхностям. Сами граничные условия имеют произвольную структуру, что весьма удобно при решении различного рода задач.

Литература

1. Препарата Ф. Вычислительная геометрия. / Ф. Препарата, М. Шеймос. М.-Введение, Мир, 1989. **2. Фокс А.** Вычислительная

геометрия. / А. Фокс, М. Пратт -М., Мир, 1982. **3. Быков В.П.** Методическое обеспечение САПР в машиностроении. / В.П. Быков - Л.: Машиностроение, 1992. **4. Дементьев Ю.В.** САПР в автомобиле- и тракторостроении. / Ю.В. Дементьев - М.: Академия, 2004. **5. Норенков И. П.** Основы автоматизированного проектирования. / И.П. Норенков - М.: МГТУ им. Н. Э. Баумана, 2002. **5. Донченко В.Ю.** Моделирование препроцессора системы «МІРЕЛА+» / В.Ю. Донченко, С.И. Гоменюк, Р.В. Киричевский // Вісник східноукраїнського національного університету імені Володимира Даля. - Луганськ:2008. - №12[130] частина 2. – с. 11-18.

Донченко В. Ю., Черевань Е.Н. Обзор и анализ методов построения геометрических моделей сложных конструкций

В статье рассмотрен обзор и анализ методов построения геометрических моделей сложных конструкций.

Ключевые слова: каркасное моделирование, твердотельное моделирование, поверхностное моделирование.

Донченко В.Ю., Черевань О.М. Огляд і аналіз методів побудови геометричних моделей складних конструкцій

У статті розглянутий огляд і аналіз методів побудови геометричних моделей складних конструкцій.

Ключові слова: каркасне моделювання, твердотіле моделювання, поверхневе моделювання.

Donchenko V.Yu., Cherevan E.N. Review and analysis methods for constructing geometric models of complex structures

In article the review and the analysis of methods of creation of geometrical models of difficult designs is considered.

Keywords: frame modeling, solid-state modeling, superficial modeling.

УДК 004.4’2

Е. В. Зубков

УНИВЕРСАЛЬНЫЕ СРЕДСТВА РАЗРАБОТКИ ПО ДЛЯ МОБИЛЬНЫХ УСТРОЙСТВ

Разработка приложений и игр для мобильных устройств — наиболее активно развивающаяся сфера современных технологий, которая открывает совершенно новые возможности для рекламы и PR (англ. *Public Relation* — связи с общественностью). Рост числа людей, имеющих мобильные телефоны, смартфоны и планшеты продолжается. На данный момент, мобильными устройствами пользуется, большая

часть населения планеты. Людей, использующих Интернет через мобильные устройства, с каждым годом становится все больше.

Сейчас в мире насчитывается примерно:

- 5.3 миллиарда – Мобильных телефонов (конец 2012 года);
- Более 1-го миллиарда – Смартфонов (конец 2012 г.);
- 128 миллионов – Планшетов (конец 2012 г.) [1],[9].

Динамика роста мобильных устройств:

- С октября по декабрь 2012 г., по всему миру было отгружено 52,5 миллионов планшетов на различных платформах. Рост по сравнению с тем же периодом 2011-го, когда продажи равнялись 29,9 миллионов штук, достиг внушительных 75,3%;
- Прирост продаж смартфонов в 2012 г., составляет +46,1% по сравнению к прошлому году(2011);
- Количество продаж мобильных телефонов падает по сравнению с прошлым(2011г.) примерно на 4-5%.[3]

По прогнозам аналитиков падение продаж мобильных телефонов будет продолжаться и через год повыситься до 10%. А вот продажи смартфонов и планшетов будут расти. Ниже представлен график, предположения аналитиков по росту продаж смартфонов и планшетов, вплоть до 2017 г., (рис. 1). [8]

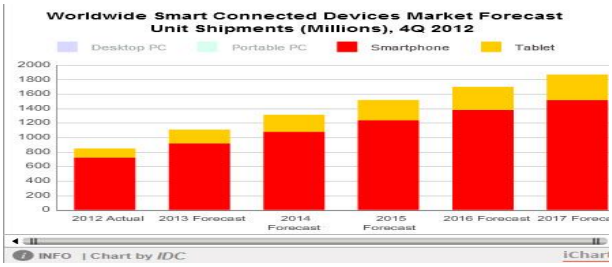


Рис. 1 смартфоны и планшеты

Такая статистика сама за себя говорит об актуальности создания мобильных приложений и игр, для большинства компаний на рынке с их последующим активным использовании как средства для продвижения своих товаров и услуг.

Преимущества создания приложений и игр для мобильных устройств:

- Возможность продвижений своих брендов среди потенциальных покупателей товаров или услуг, размещая баннеры, логотипы и любую другую рекламу в игры и распространяя их бесплатно.
- Реальное конкурентное преимущество деятельности компаний.
- Оптимизация бизнес-процессов компаний, например, за счет повышения скорости коммуникации между сотрудниками компании.
- И так далее ...

Статистика и динамика использования ОС мобильных устройств с марта 2012 г. по март 2013 г., (рис. 2, рис. 3).[6]

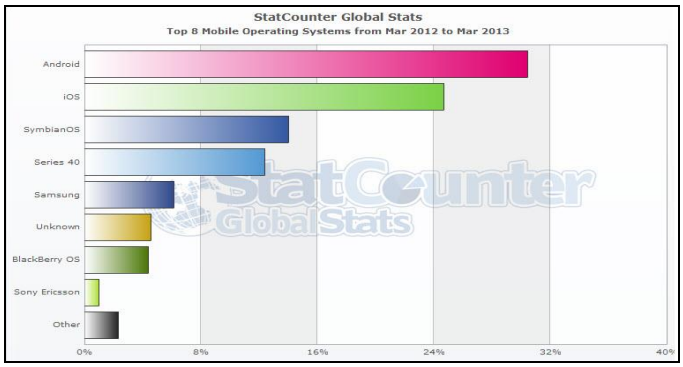


Рис. 2 статистика ОС моб. Устройств

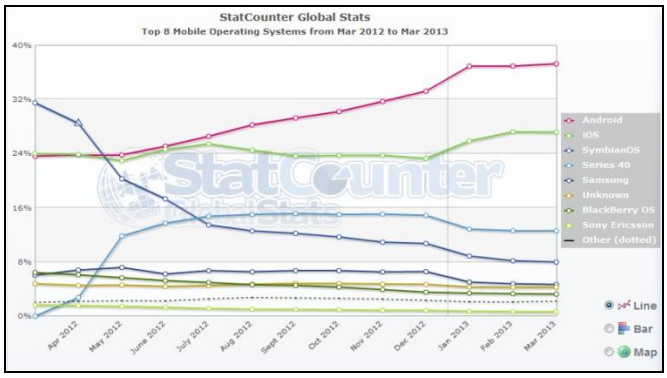


Рис. 3 динамика ОС моб. устройств

Как и другие традиционные инженерные дисциплины, разработка программного обеспечения имеет дело с проблемами качества, стоимости и надёжности. Некоторые программы содержат миллионы строк исходного кода, которые, как ожидается, должны правильно исполняться в изменяющихся условиях. [5] Ко всему этому, мобильные устройства имеют множество форм и стилей. Мобильные телефоны работают на разных операционных системах, таких как Apple iOS, Google Android, Windows Phone и др. Некоторые имеют большие экраны, физические клавиатуры и работают в сетях 3G, 4G или WiFi. Мобильные телефоны могут также иметь датчики ускорения, местоположения и даже платежей. Некоторые из этих устройств - даже не телефоны; это планшеты с более крупными экранами и сетевым подключением только для обмена данными. Поэтому разработка ПО и игр для мобильных устройств, ещё более усложняется. Так как нужно ориентировать на определенные платформы, ОС и конфигурации систем со своими наборами функциональности. И так получается, что нельзя написать программу которая будет работать на любом телефоне, для этого нужно переписывать код как минимум для каждой ОС.

Для упрощения разработки используются так называемые “Универсальные средства разработки” (УСР). Принцип их работы таков, мы пишем приложение на языке предусмотренном в УСР и затем переносим его для нужной нам платформы или ОС. Вот несколько УСР (RhoMobile Suite, Appcelerator Titanium, PhoneGap) которые будут рассмотрены далее.

RhoMobile Suite (разработчики: *Motorola,* тип: *open-source framework,* лицензия: *MIT,* Сайт: *rhomobile.com*) - Фреймворк для кросс-платформенной разработки. Главной его особенностью является использование *Ruby* и *MVC* архитектуры приложений. Созданный вами код на языке *Ruby* интерпретируется в нативный код устройства. Поддерживаются все основные мобильные платформы*: iPhone, Windows Mobile, RIM, Symbian и Android*. Подробнее о поддержке различных возможностей можно посмотреть в документации.

Разработчики фреймворка предлагают нам готовый платный *“backend”* для нашего приложения. Это не реляционная база данных типа ключ-значение, для которой существуют встроенные в фреймворк средства взаимодействия. Есть платный сервер предлагающий легкую интеграцию вашего приложения с уже существующими. Есть бесплатное для маленьких задач средство для создания ваших приложений в облаке, не нужно ничего устанавливать и настраивать на своем компьютере – все хранится и компилируется в облаке. Каждый проект в облаке представлен *git* репозиторием, поэтому писать код можно на своем компьютере в любимой среде разработки.

Из недостатков: сложно настроить фреймворк и релизить приложения, если не пользоваться облаком. Ваше приложение выполняется внутри *RubyVM*, что отрицательно сказывается на производительности. Нужно знать *Ruby*.

Appcelerator Titanium (разработчики: *Appcelerator, Inc.,* тип: Application f*ramework,* лицензия: *Apache Public License v2*,*Proprietary software,* Сайт: *www.appcelerator.com) -* Инструмент для разработки приложений для мобильных устройств с использованием в качестве основного языка *JavaScript/JSON*. Есть поддержка выполнения *Ruby, Python* и *PHP* скриптов. Большое количество различных библиотек, в том числе поддерживающих *Facebook, Yahoo API* и др. За небольшие деньги можно докупить дополнительных модулей, таких как: *Dropbox SDK* ($14), *Twitter Framework* ($12.99), *Google Analytics SDK* ($10) и др. Можно даже создать свой модуль и разметить его в магазине для продажи другим разработчикам.

Приложения созданные с помощью данного *SDK* будут действительно нативными. Контроллер навигации на *Android* будет выглядеть привычно и не так как на *iOs*. Причем не только вид, но и сам код приложения будет тоже нативный. Это кстати не мешает

вам создавать и классический *WebView* и наполнить его желаемым *web* контентом.

Разработчики на основе *Eclipse* создали специальную среду разработки для разработки с использованием данного *SDK*. На сайте можно найти исчерпывающую документацию и видео уроки хорошего качества. В общем, ребята постарались на славу.

Из недостатков: поддержка только *Android* и *iOs*. Нет простой возможности использовать стандартный стек *HTML/CSS/JavaScript* совместно с *Appcelerator API*. То есть, вы можете создать *WebView* и загрузить туда контент, но надо будет немного повозиться с доступом к *API SDK*.

PhoneGap (разработчики: *Adobe Systems,* тип: *mobile development framework,* лицензия: *Apache 2.0 License,* Сайт: *www.phonegap.com, cordova.apache.org) -* Это *HTML5* платформа для разработки мобильных приложений. Она позволяет разрабатывать веб-приложения и получать, посредством *JavaScript* библиотеки, доступ к нативным функциям устройства, таким как: файловая система, камера, уведомления и др. Используются стандартные веб-технологии, то есть ваше приложение работает внутри браузера как обычный сайт, плюс дается специальная *JavaScript* библиотека для доступа к нативным функциям устройства. Стоит отметить то, что вы можете спокойно использовать *JavaScript* библиотеки, которые вы использовали при разработки сайтов (*jQuery, Prototype, Sencha Touch* и др.).

Важной особенностью данного *SDK* является поддержка широкого списка мобильных платформ. Поддерживаются все современные платформы*: iPhone, Android, OS, WebOS, WP7, Symbian* и *Bada, W8, WP8, Tizen* . Такой длинный список не удивительным, ведь главным требованием *SDK* к платформе является наличие современного браузера. Поэтому если в будущем появится, какая-то новая *OS* – разработчики довольно оперативно адаптируют *PhoneGap* для работы на ней.

Есть простое средство релиза ваших приложений, с помощью которого можно легко получить бинарные файлы вашего приложения для всех требуемых платформ. Документация просто замечательная и очень дружелюбна к начинающим разработчикам. Можно расширять *SDK* и добавлять свои модули, предоставив их реализацию для каждой платформы на родном *SDK*.

Недостатки:

- Одинаковый интерфейс приложения на всех платформах, так как, вы это реализуете на ***HTML/CSS/JS***;
- Производительность, ограниченная браузером как следствие низкая;
- Плохая доступность нативных функций ОС (никаких родных диалогов, текстовых уведомлений и специфичных контролеров),

отсутствие специализированной среды разработки (подойдут ваши любимые инструменты для веб-программирования)

Преимущества:

- Позволяет создать приложения для мобильных устройств используя ***HTML/CSS/JS***, без необходимости знания «родных» языков программирования;
- Поддержка большого количества мобильных ОС.

Таблица поддерживаемых возможностей для различных ОС.[7]

Feature	iPhone /iPhone 3G	iPhone 3GS and	Android 1.0 — 4.2	WP	BlackBerry 5.x-6.0+	BlackBerry 4.6-4.7	Bada	Symbian	webOS	Tizen
Accelerometer	Да	Да	Да	Да	Да	Н/Д	Да	Да	Да	Да
Camera	Да	Да	Да	Да	Да	Н/Д	Да	Да	Да	Частично
Compass	Н/Д	Да	Да	Да	Н/Д	Н/Д	Да	Н/Д	Н/Д	Да
Contacts	Да	Да	Да	Да	Да	Н/Д	Да	Да	Н/Д	Да
File	Да	Да	Да	Да	Да	Н/Д	Н/Д	Н/Д	Н/Д	Да
Geolocation	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да
Media	Да	Да	Да	Да	Н/Д	Н/Д	Н/Д	Н/Д	Н/Д	Частично
Network	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да
Notification (Alert)	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да
Notification (Sound)	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да
Notification (Vibration)	Да	Да	Да	Да	Да	Да	Да	Да	Да	Да
Storage	Да	Да	Да	Да	Да	Н/Д	Н/Д	Да	Да	Да
Barcode Scanner	Да	Да	Да	Н/Д	Да	Н/Д	Н/Д	Н/Д	Н/Д	Н/Д

В работе представлен пример всплывающего (alert, dialog box) окна Пример из Apache Cordova API Documentation

```
<!DOCTYPE html>
<html>
 <head>
  <title>Notification Example</title>

  <script type="text/javascript" charset="utf-8" src="cordova-2.1.0.js"></script>
  <script type="text/javascript" charset="utf-8">
  // Wait for Cordova to load
    document.addEventListener("deviceready", onDeviceReady, false);
  // Cordova is ready
    function onDeviceReady() {
     // Empty
  }
  // alert dialog dismissed
  function alertDismissed() {
    // do something
  }
  // Show a custom alertDismissed
    function showAlert() {
     navigator.notification.alert(
        'You are the winner!',  // message
        alertDismissed,         // callback
        'Game Over',            // title
        'Done'                  // buttonName
     );
  }
  </script>
 </head>
 <body>
  <p><a href="#" onclick="showAlert(); return false;">Show Alert</a></p>
 </body>
</html>
```

Результат:

Рис. 4 запуск приложения

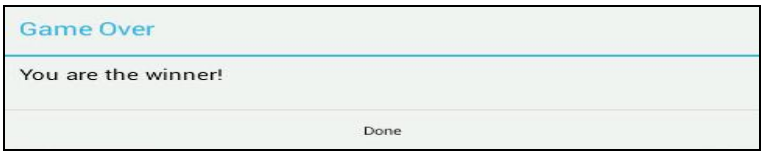


Рис. 5 всплывающее окно при нажатии на ссылку

Для отладки использовалась:
Eclipse IDE for Java and Report Developers
Version: Juno Release
Build id: 20120614-1722
Android SDK Manager
Revision 21.0.1
Эмулятор:
Oracle VM VirtualBox
Version 4.0.4. Edition
Платформа:
android-x86-4.2-20130228

Литература

1. Аналитические данные количества мобильных устройств. – [Электронный ресурс]. – Режим доступа к статье: http://slon.ru/fast/future/idc-v-2012-godu-prodan-milliard-devaysov-v-2013-m-planshety-obgonyat-pc-v-2014-m-noutbuki-924233.xhtml. **2.** Динамика продаж. – [Электронный ресурс]. – Режим доступа к статье: http://business.compulenta.ru/734316/. **3.** Смартфоны и планшеты динамика продаж. – [Электронный ресурс]. – Режим доступа к статье: http://itc.ua/news/idc-v-2012-godu-byilo-prodano-bolee-1-mlrd-umnyih-podklyuchennyih-ustroystv/. **5.** Понятие сложности разработки приложений. – [Электронный ресурс]. – Режим доступа к статье: http://ru.wikipedia.org/wiki/Разработка_программного_обеспечения. **6.** Статистика ОС мобильных устройств. – [Электронный ресурс]. – Режим доступа к статье: http://gs.statcounter.com/#browser-ww-monthly-201203-201303. **7.** Таблица поддерживаемых версий Phone Gap. – [Электронный ресурс]. – Режим доступа к статье: http://ru.wikipedia.org/wiki/PhoneGap.
8. Предположительный рост продаж планшетов и смартфонов в будущем. – [Электронный ресурс]. – Режим доступа к статье: http://www.icharts.net/chartchannel/worldwide-smart-connected-devices-market-forecast-unit-shipments-millions-4q-2012_m3aycxhc. **9.** Количество смартфонов в мире превысило 1 миллиард. – [Электронный ресурс]. – Режим доступа к статье: http://www.vestifinance.ru/articles/18538

Зубков Е. В. Универсальные средства разработки ПО для мобильных устройств

В статье представлена статистика популярности мобильных устройств таких как: мобильные телефоны, смартфоны и планшеты. Так

же преимущества создания и использования этого ПО. Представлены три разных УСР, рассмотрены их характеристики и выбрано одно средство разработки, а именно Phone Gap. Приведён пример из Apache Cordova API Documentation, и его реализация с отчётом работоспособности приложения.

Ключевые слова: мобильные устройства, ПО, УСР.

Зубков Є. В. Універсальні засоби розробки ПЗ для мобільних пристроїв

У статті представлена статистика популярності мобільних пристроїв таких як: мобільні телефони, смартфони і планшети. Так само переваги створення і використання цього ПЗ. Представлені три різних УСР, розглянуті їх характеристики і вибрано один засіб розробки, а саме Phone Gap. Наведено приклад з Apache Cordova API Documentation, та його реалізація із звітом працездатності додатка.

Ключові слова: мобільні пристрої, ПЗ, УЗР.

Zubkov E.V. Universal development tool software for mobile devices

The article presents statistics popularity of mobile devices such as mobile phones, smartphones and tablets. The same benefits of creating and using the software. Presents three different UDT (universal development tool), considered their characteristics and choose one development tool, namely Phone Gap. Is an example of Apache Cordova API Documentation, and its implementation with the application operation report.

Key words: mobile device, software, UDT.

УДК 004.434

Козуб В.Ю.

МЕТОДЫ ВИЗУАЛИЗАЦИИ ПРОСТРАНСТВЕННЫХ ОБЪЕКТОВ В ВЫЧИСЛИТЕЛЬНОМ КОМПЛЕКСЕ “МІРЕЛА+”

Вычислительный комплекс “МІРЕЛА+” [1] предназначен для исследования прочности, долговечности и разрушения конструкций из эластомерных и композитных материалов в условиях линейного и нелинейного вязкоупругого деформирования, исследования температурных полей диссипативного разогрева и параметров механики разрушения массивных эластомерных элементов конструкций и тонкослойных резинометаллических элементов с трещинами с изменяющимися физико-механическими параметрами в условиях циклического деформирования. Одним из наиболее эффективных методов решения задач механики и математической физики является метод конечных элементов (МКЭ).

Практическое применение МКЭ к расчету конструкций идет по пути усложнения, как в постановке задач, так и в их конкретной реализации с учетом особенности существующих ЭВМ. Необходимость анализа сложных систем на современном этапе развития вычислительной техники требует создания развитых универсальных вычислительных комплексов для ЭВМ, позволяющих подойти к решению задач с максимально возможной общностью, на основе единых позиций. Это позволяет сформулировать общие требования, предъявляемые к расчетным комплексам, реализующих метод конечных элементов и позволяющих в принципе решить любую задачу, независимо от ее сложности.

1. Требования на уровне проектирования: создание эффективных автоматизированных методов дискретизации исследуемого трехмерного объекта, задание физико-механических параметров материала конструкции.

2. Требования на уровне математического обеспечения: необходимость разработки алгоритмов, полностью автоматизирующих процесс расчета, начиная с выбора математической модели, построения и решения системы уравнений и заканчивая выводом и интерпретацией выходных данных.

3. Требования на уровне программирования: необходимость унификации всех вариантов проектирования в такой организации процедур обработки, которая позволила бы решать все задачи по единой достаточно удобной методике.

В рассматриваемом вычислительном комплексе “МІРЕЛА+” значительное внимание уделяется наряду с перечисленными выше возможностями вопросам графического представления исходной и конечной информации. Наиболее эффективным способом визуального представления по двух- или трехмерной расчетной области является способ изображения полутоновой или цветной картины, где каждому оттенку или цвету соответствует определенный диапазон числовых значений.

Существует достаточно большое количество различных способов визуализации результатов расчета. Среди наиболее распространенных – построение различных двумерных и трехмерных графиков, линий уровня. Однако наиболее эффективным методом визуального представления и восприятия распределения числовой величины по некоторой двух- или трехмерной расчетной области является изображение полутоновой или цветной картины, где каждому оттенку или цвету соответствует известный диапазон числовых значений.

В большинстве случаев видимая поверхность геометрической области, исследуемой методом конечных элементов, представляет собой конечную совокупность плоских непересекающихся геометрических фигур простой формы (чаще всего треугольников или

четырехугольников) (рис. 1). Поэтому, изображение конечно-элементного объекта обычно сводится к визуализации в пространстве некоторого количества плоских геометрических фигур – граней конечного элемента (КЭ).

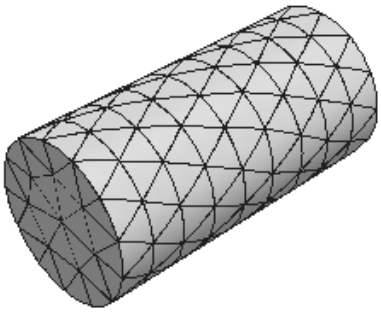


Рис. 1. Поверхность объекта как совокупность непересекающихся треугольников.

В общем случае проблема визуализации результатов численных расчетов задач механики методом конечных элементов сводится к решению двух следующих задач:

1) построение математического аппарата проецирования трехмерной геометрической области на экранную плоскость;

2) закрашивание получившейся проекции цветами или оттенками, соответствующими распределению по исходной геометрической области визуализируемой функции.

В компьютерной графике для задания места расположения 3D-объекта используется трехмерная ортогональная декартовая система координат.

Современные графические стандарты программирования трехмерной графики (OpenGL, DirectX и др.), оперируют тремя видами систем координат:

- мировой;
- видовой;
- экранной.

Мировая система координат – это правосторонняя декартовая система координат, центр которой W совпадает с положением наблюдателя сцены. Видовая система координат – это аналогичная левосторонняя система координат, центр которой E совпадает с геометрическим центром визуализируемого объекта. Экранная система координат – это плоская декартовая система координат, центр которой S совпадает с левым нижним углом экрана, а оси направлены естественным образом (рис. 2).

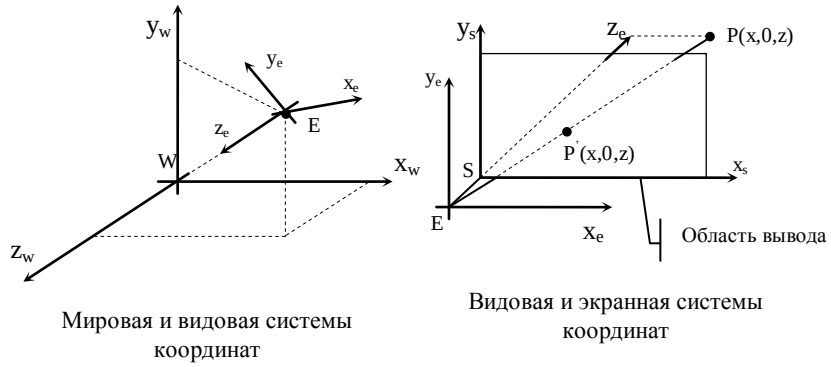


Рис. 2. Мировая, видовая и экранная системы координат.

Главная проблема заключается в построении математического аппарата преобразования мировых координат в экранные таким образом, чтобы добиться построения на экране монитора реалистичного образа. Исследованию данной проблемы посвящено большое количество работ, наиболее известными среди которых являются работы Л. Аммерала [2], Д. Роджерса [3] и др.

При программировании визуализации трехмерных компьютерных образов в общем виде необходимо задать объем видимости в трехмерном пространстве, вид проецирования на экранную плоскость и объем вывода на экран. При этом трехмерные объекты отсекаются по границе объема видимости и после этого проецируются на экран. Механизм преобразования мировых координат в экранные можно изобразить схемой, приведенной на рис. 3.

Рис. 3. Процесс преобразования мировых координат в экранные.

В процессе преобразования координат из мировых в экранные три измерения отображаются на два, что неизменно приводит к появлению искажений. От выбора типа проецирования зависит не только достоверность изображенного образа, но и адекватность его восприятия.

Проецирование выполняется с помощью проецирующих лучей, называемых проекторами, которые идут из центра проекции через все точки визуализируемого объекта до экранной плоскости. По

расположению центра проекции относительно экранной плоскости различают центральную и параллельную проекции.

В случае параллельной проекции ее центр располагается на бесконечном расстоянии от экранной плоскости. Таким образом, проекторы представляют собой параллельные пучки прямых. Вид данной проекции зависит от взаимного расположения осей координат, экранной плоскости и проекторов. В компьютерной графике чаще всего применяются ортогональная и аксонометрическая проекции.

При ортогональном проецировании проекторы параллельны главной оси наблюдения и ортогональны экранной плоскости.

При аксонометрическом проецировании возможны два варианта: проекторы перпендикулярны экранной плоскости, которая в свою очередь расположена под некоторым углом к главной оси. Такой тип проецирования называется прямоугольной аксонометрической проекцией. Если же, наоборот, проекторы не ортогональны экранной плоскости, а главная ось – ортогональна, то такой тип проецирования называется косоугольной аксонометрической проекцией.

Изображение, полученное при использовании таких видов проецирования, позволяет за счет отсутствия перспективы точно отобразить форму и размер трехмерного объекта, однако, оно не достаточно реалистично.

При центральном проецировании расстояние до центра проекции конечно, поэтому проекторы – это пучок прямых, исходящих из центра. Для задания такого вида проецирования необходимо определить местоположение центра проекции и картинной плоскости. Центральное проецирование за счет появления перспективы приводит к искажениям в форме и размерах объекта, однако, такой объект на экране выглядит более реалистично (рис. 4).

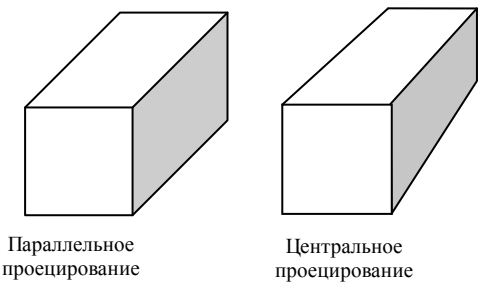


Рис. 4. Параллельное и центральное проецирование.

Процесс визуализации объектов разделяется на следующие этапы:

а) определение размеров области вывода;

б) определение первоначально объема видимости, включающего в себя все объекты;

в) определение оптимального положения точки наблюдения при выбранном пользователем в интерактивном режиме типе проецирования;

г) сдвиг, масштабирование и поворот сцены при необходимости;

д) последовательное занесение в Z-буфер всех объектов, которые необходимо визуализировать;

е) вывод содержимого Z-буфера в выбранный выходной поток (экран, файл, принтер).

Общая блок-схема процесса визуализации объектов приведена на рис. 5.

Процесс визуализации объекта зависит от типа используемых конечных элементов и принятого алгоритма визуализации данных. Изображение треугольника в пространстве также зависит от выбранного метода визуализации. Общая блок-схема визуализации треугольника имеет вид, приведенный на рис. 6. Визуализация других типов конечных элементов может быть сведена к изображению некоторого количества треугольников.

Визуализация объектов реализована с помощью графической библиотеки OpenGL.

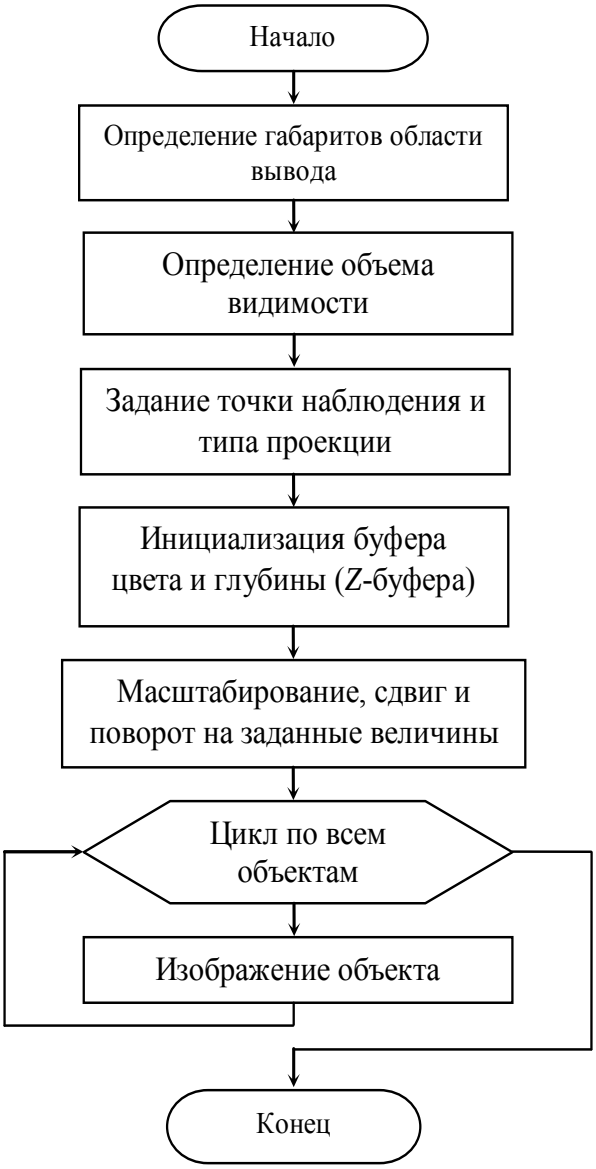


Рис. 5. Общая блок-схема изображения объектов.

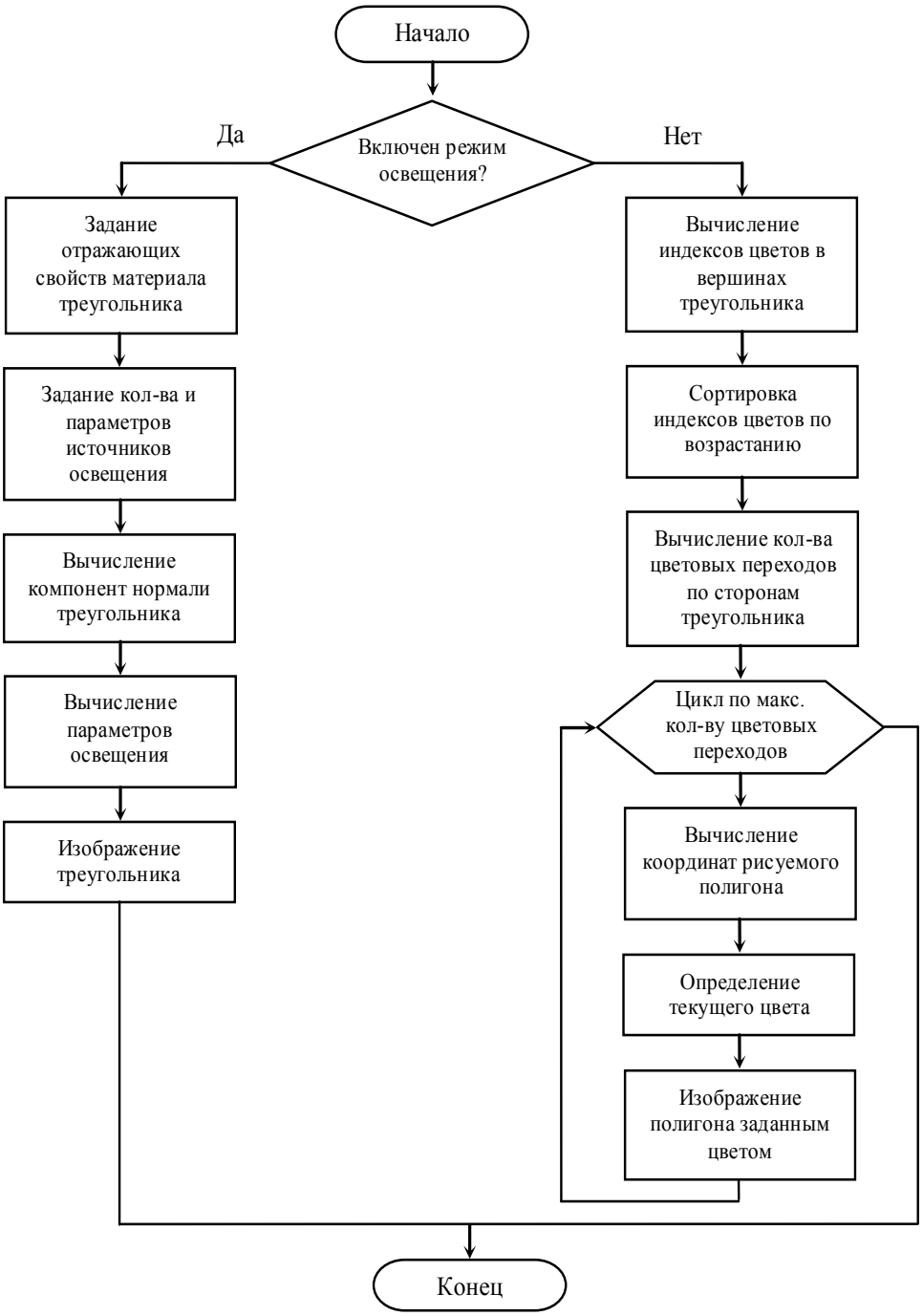


Рис. 6. Блок-схема изображения треугольника.

На рис. 7-10 приведены примеры визуализации различных плоских и объемных объектов и результатов расчета задачи о напряженно-деформированном состоянии куба со сквозным отверстием[4].

Рис. 7. Геометрическая модель объекта

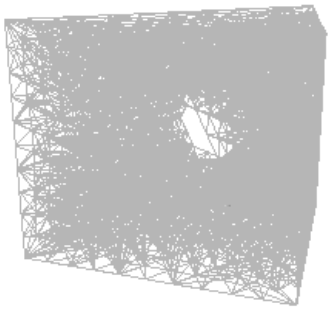


Рис. 8. Каркасная модель объекта

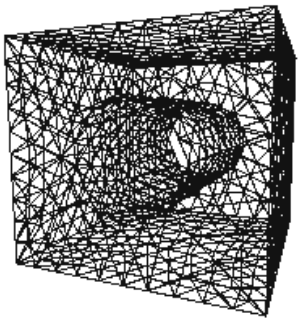


Рис. 9. Поверхностные ребра

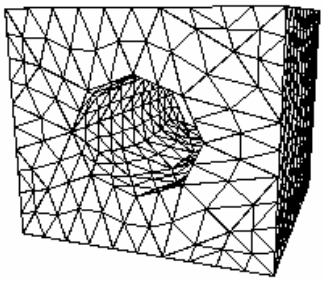


Рис. 10. Поверхностные грани.

Графическое представление входных данных, является более наглядным и эффективным, чем печать таблиц числовых данных. Кроме того, при представлении на экране компьютера исходных данных имеется возможность исправлять какие-то несоответствия в разбивке конструкции на элементы или вносить другие коррективы. Это позволяет, например, вносить изменения в жесткости элементов конструкции, учитывать наличие и распределение приложенных сил и т.д. При этом всегда можно оптимизировать количество изменений в зависимости от условий задачи и имеющего машинного времени.

Литература

1. Метод конечных элементов в вычислительном комплексе «МІРЕЛА+» / В. В. Киричевский, Б. М. Дохняк, Ю. Г. Козуб и др. ; под общ. ред. В. В. Киричевского. – К.: Наук. думка, 2005. – 403 с. **2. Аммерал Л.** Интерактивная трехмерная машинная графика / Л. Аммерал. – М.: Изд-во «Сол Систем». – 1992. – 313с. **3. Роджерс Д.** Математические основы машинной графики / Д. Роджерс, Дж. Адамс // Пер.со второго англ. изд. Монахова П.А., Олохтоновой Г.В. под ред. Баяковского Ю.М.и др. – М.: Мир. – 201. – 604 с. **4. Гоменюк С.И.** Объектно-ориентированные модели и методы анализа механических процессов / С.И. Гоменюк. – Никополь: Никопольская коммунальная типография, 2004. – 316 с.

Козуб В. Ю. Методы визуализации пространственных объектов в вычислительном комплексе “МІРЕЛА+”

В статье проведен анализ алгоритмов и поэтапного процесса визуализации объектов в вычислительном комплексе “МІРЕЛА+”. Приведены примеры визуализации различных плоских и объемных объектов и результатов расчета задачи о напряженно-деформированном состоянии куба со сквозным отверстием.

Ключевые слова: блок-схема, визуализация, 3D объект.

Козуб В. Ю. Методи візуалізації просторових об'єктів в обчислювальному комплексі “МІРЕЛА+”

У статті проведено аналіз алгоритмів і поетапного процесу візуалізації об'єктів в обчислювальному комплексі “МІРЕЛА+”. Наведені приклади візуалізації різних плоских і об'ємних об'єктів та результатів розрахунку задачі про напружено-деформований стан куба з крізним отвором.

Ключові слова: блок-схема, візуалізація, 3D об'єкт.

Kozub V. Yu. Metody visualizations of spatial objects in a calculable complex “МІРЕЛА+”

In the article the analysis of algorithms and stage-by-stage process of visualization of objects is conducted in a calculable complex “МІРЕЛА+”. The examples of visualization of different flat and by volume objects and results of calculation of task are resulted about the tense-deformed state of cube with the through opening.

Keywords: flow-chart, visualization, 3 D object.

УДК 004.421:004.651.5

А.С. Коломийцева, М.А. Юрченко

ЭЛЕКТРОННАЯ БИБЛИОТЕКА СПРАВОЧНОЙ ИНФОРМАЦИИ ДЛЯ УЧЕБНОГО ПРОЦЕССА

На сервере кафедры ИТС есть электронная библиотека справочной информации (БСИ). Так как электронная библиотека кафедры ИТС содержит большое количество книг и других учебных материалов (более тысячи), а сама библиотека не сортирована, то поиск необходимого материала становится затруднительным (тем более что часть книг на английском языке). В таком виде БСИ почти бесполезна. Необходима автоматизированная система просмотра БСИ. Необходимо каталогизировать электронную библиотеку для удобном поиска справочного материала. Можно написать программу сортировки книг по тематике для быстрого их поиска.

Т.к. на данном этапе имена книг в большинстве случайные и не предоставляют никакой информации об их авторах и тематике - необходимо привести их названия к информативному виду для удобного анализа. Первоначальной задачей стоит задать книгам информативные имена, в соответствие с тегами файлов. Для этих целей планируется написание программы LibInfo, генерирующей имена для файлов, содержащих теги (описания - автор, название, год издания и т.д.). Т.к. таким образом можно обработать не все книги (например pdf и djvu без тегов и текстового слоя не будут обработаны), то формируется отчет с перечнем необработанных книг.

После этого можно реализовать поиск по названиям книг и анализу описания к книгам. Сейчас описания делаются вручную:

1. В Total Commander выделяется файл и нажимается сочетание клавиш Ctrl+Z.
2. В появившемся окне записывается комментарий
3. Сохраняется клавишей F2.
4. Комментарий сохраняется в файле descript.ion (рис. 1, 2).

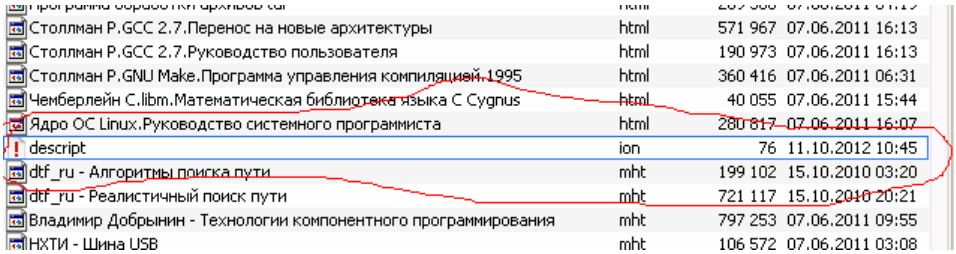


Рис. 1. Файл descript.ion

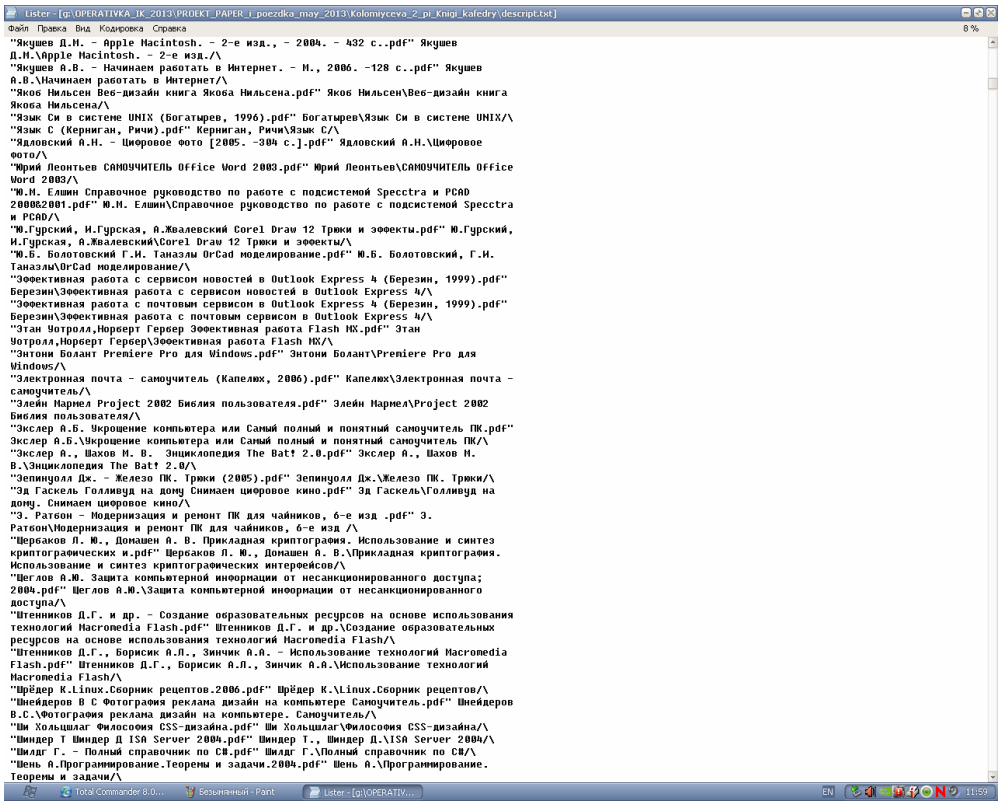


Рис. 2. Содержимое файла descript.ion

Библиотека расположена в H:\docs\new_doc (Рис. 3).

На рисунке видно, что БСИ содержит большой объем книг. Например, бегунок (scroll) на рисунке находится в самом верху и не все книги поместились в окно.

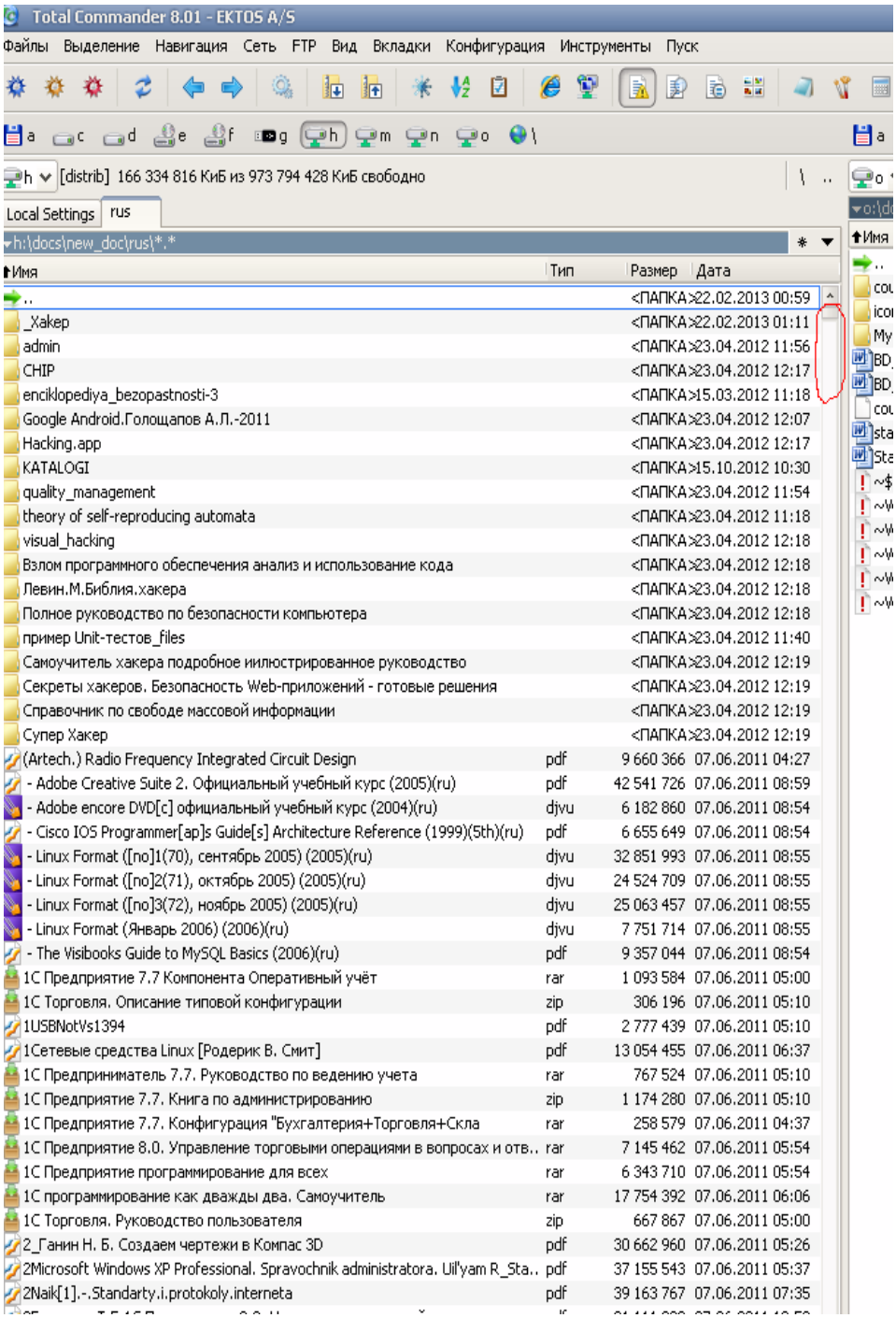


Рис. 3. Содержимое БСИ.

Для автоматической генерации комментариев предлагается программа LibAnalyzer для части книг. Генерация таких комментариев возможна для книг в текстовом виде (txt), гипертекстовых документов (html, chm), xml-книг (fb2), doc и docx форматов, а также файлов pdf и djvu при условии, что в них содержится текстовый слой или теги. Для

прочих форматов (не djvu и pdf) поиск также осуществляется по тегам файла либо анализом первых страниц книги.

В функции такой программы (LibAnalyzer) входит:

1. Сканирование книг в заданном пользователем каталоге.

2. Анализируем каждый файл. Если в нем содержатся теги, текстовый слой либо книга представлена в текстовом виде - возможна генерация комментария к книге. Если нет тегов, но доступен текстовый слой либо книга в текстовом виде - доступная информацию из первых страниц сокращается, оставляя краткий комментарий, который записываем в файл комментариев Windows.

3. Если никаких сведений к книге не содержится - выводится сообщение о том, что книга не может быть обработана. Файл в этом случае заносится в список необработанных книг, который впоследствии можно будет сохранить в виде отчета для дальнейшего анализа (возможно в программе будет добавлена функция поиска данных в интернете).

4. Программа в графическом виде представляет удобный просмотр количество обработанных и необработанных книг, с возможностью сформировать подробный отчет необработанных книг.

Последняя задача – каталогизировать книги для быстрого поиска материала и вывода похожей литературы. Каталогизация книг будет осуществляться по категориям: алгоритмизация, программирование, графика и т.д. Для этого можно создать программу LibSorter, которая помимо основной задачи также будет искать дублирующийся материал.

После работы вышеперечисленных программ библиотека становится упорядоченной и информативной. Теперь можно реализовать, собственно, поиск. Программа поиска по комментариям и именам файлов (LibSearcher) предоставляет пользователям возможность быстрого поиска необходимой литературы по запросу. Таким образом, программа LibSearcher включает следующие функции:

1. Обработка запроса пользователя, поиск ключевых слов запроса среди названий (имен) книг, а также в комментариях, сгенерированных автоматически на предыдущем этапе, при этом, если ни один файл не содержит метаданных - пользователь мог получить подробный отчет.

2. Т.к. на предыдущем этапе библиотека справочной информации отсортирована по тематике, осуществляется поиск категории материала, соответствующей запросу. Таким образом выводятся все книги, подходящие пользователю по тематике.

3. Если есть результаты (файлы были найдены) - выводим список найденных книг на экран, разделяя при этом графически максимально соответствующие запросу и похожие книги. Если поиск результатов не дал - сообщаем об отсутствии результатов.

Вывод:

Вышеописанные программы позволяют упорядочить библиотеку учебных материалов гораздо быстрее, чем делать это вручную, а также организует удобный поиск на основе запроса пользователя. В таком виде библиотека становится удобным инструментом для учебного процесса позволяющим без труда найти необходимый справочный материал для самостоятельных работ студентов.

Литература

1. Moore, P. (1988). Children's information seeking: Judging books by their covers. School library reviews, 8, 5. **2. Pejtersen, A.M.** (1986). Design and test of a database for fiction based on an analysis of children's search behavior.In Ingwersen, P. Kajberg, L., & Pejtersoen, M. ed., Information technology and information use: towards a unified view of information and information technology. London: Taylor Graham, pp. 125-46.

Коломійцева А.С., Юрченко М.А. Електронна бібліотека довідкової інформації для навчального процесу

Необхідно каталогізувати електронну бібліотеку для зручному пошуку довідкового матеріалу. Стаття присвячена вирішенню завдання написання програми сортування книг з тематики для швидкого їх пошуку

Ключові слова: електронна бібліотека, сортування даних, швидкий пошук

Коломийцева А.С., Юрченко М.А. Электронная библиотека справочной информации для учебного процесса

Необходимо каталогизировать электронную библиотеку для удобном поиска справочного материала. Статья посвящена решению задачи написания программы сортировки книг по тематике для быстрого их поиска

Ключевые слова: электронная библиотека, сортировка данных, быстрый поиск

Kolomiytseva A.S., Yurchenko M.A. E-library of reference information for the educational process

Electronic library must be cataloged for easy reference search. Article is devoted to the task of writing a program sort books by category to quickly find them

Keywords: e-library, sort data, a quick search

УДК 368.01

О. В. Корячкина, О. Н. Ие

СТРАХОВАНИЕ ПО НЕТРУДОСПОСОБНОСТИ

Каждый человек имеет право на уверенность в своём завтрашнем дне. Поэтому важной составляющей социальных гарантий в современном мире является страхование по частичной и полной потери трудоспособности.

Страхование по нетрудоспособности является одним из видов страхования жизни. Страховым случаем при данном виде страхования является временная или полная нетрудоспособность, наступившая вследствие заболевания или травмы. Это государственная система поддержки нетрудоспособных, престарелых и безработных за счет бюджетных средств, дотаций и специальных фондов. Социальное страхование является инструментом реализации государственной социальной политики. Впервые общеобязательное медицинское страхование появилось еще в XVII веке в Германии, в форме «страховых медицинских касс» для ремесленников. Такие кассы носили добровольный, общественный характер.

Актуальность исследования состоит в том, что практически все страны, экономическая система которых характеризуется как социально-ориентированная рыночная экономика, в той или иной мере используют обязательное медицинское страхование для решения проблем здравоохранения.

Цель статьи состоит в изучении методологических вопросов статистического исследования страхования по нетрудоспособности.

Страхование нетрудоспособности может быть как добровольным, так и обязательным.

При добровольном страховании страхователь заключает договор страхования по собственной инициативе. Для людей некоторых профессий, деятельность которых связана с угрозой здоровью и дальнейшей трудоспособности, такое страхование является обязательным. Страховщиком в подобных случаях будут выступать работодатели или государство. Это касается в первую очередь военнослужащих, сотрудников органов охраны правопорядка, пожарных. Существует так же обязательное социальное страхование нетрудоспособности, осуществляемое в государственных масштабах.

Договор добровольного страхования нетрудоспособности предполагает чёткий перечень страховых случаев. В договоре могут быть описаны условия наступления страхового случая.

Обязательное социальное страхование нетрудоспособности осуществляется Фондом социального страхования. Этот вид страхования является обязательным. Страховые взносы за рабочих и служащих вносятся работодателем ежемесячно в соответствии с утверждёнными

ставками: 32% от заработной платы для основной категории граждан и для 29% для льготных категорий населения. Начисления страховых взносов осуществляются из расчёта средней заработной платы застрахованного и выплачиваются Фондам социального страхования. Страховыми случаями являются: временная или полная нетрудоспособность, беременность и роды, протезирование в условиях протезно-ортопедического стационара. Работодатель не имеет к страховым выплатам никакого отношения, но страховые взносы, а, следовательно, и страховые выплаты начисляются только с официальной заработной платы. Размер пособия, которое будет получать безработный, зависит именно от размера его официальной зарплаты и трудового стажа. Объем помощи по безработице будет составлять 50% от средней зарплаты человека, если его рабочий стаж – менее 2 лет; 55% – для работающих от 2 до 6 лет; 60% от среднего дохода – для тех, кто отработал от 6 до 10 лет. На 70% от экзсзарплаты могут претендовать работающие более 10 лет. Помощь по безработице выплачивается на протяжении 1 года. Первые 3 месяца человек получает 100% суммы, рассчитанной по вышеприведенной схеме; следующие 3 месяца – 80%, и последние полгода – 70%. Например, начинающий специалист, не проработавший и двух лет, белая зарплата которого составляла 5000 грн., может получать от Центра занятости первые три месяца – по 2500 грн., вторые – по 2000 грн., и еще полгода – по 1750 грн. ежемесячно.

Фонд социального страхования по временной потере трудоспособности был создан в соответствии с Законом Украины «Про общеобязательное государственное социальное страхование в связи с временной утратой трудоспособности». Фонд является органом, который руководит и управляет общеобязательным государственным социальным страхованием в связи с временной утратой трудоспособности, проводит сбор и аккумуляцию страховых взносов и других средств, предназначенных для финансирования материального обеспечения и социальных услуг. Источниками формирования фондов социального страхования являются государственные дотации, а также страховые взносы, которые уплачивают трудящиеся граждане и работодатели. Средства государственного социального фонда расходуются исключительного на цели социальной защиты. Кроме оплаты пенсий и пособий, деньги могут направляться на оплату путевок в бальнеологические учреждения с целью санаторно-курортного лечения, диетическое (лечебное) питание, частичное содержание оздоровительных и санаторных детских лагерей, санаториев-профилакториев, находящихся на балансе страхователей.

Фонд социального страхования от несчастных случаев на производстве и профессиональных заболеваний Украины в 2012 году осуществил страховых выплат потерпевшим и членам их семей на сумму 4,749 млрд. грн., что на 28,8% больше чем в 2011году. Среднемесячный размер страховой выплаты в случае частичной или полной потери

трудоспособности, компенсирующей потерянную часть заработка, составил 1,057 тыс. грн, что на 39,4% больше, чем в 2011 году, одноразовой помощи – 60,8 тыс. грн, что на 25% больше.

За годы независимости наше государство так и не сформировало систему социального страхования европейского образца. Та, что имеется сейчас, показала себя как не очень эффективную систему защиты граждан. Проведя даже поверхностный анализ сравнения уровня здравоохранения в общем и медицинского страхования Украины с другими странами (США и страны ЕС), мы видим насколько Украина отстает по многим показателям. В данный момент, на наш взгляд, для нашей страны очень важно обратить внимание на многолетний опыт других государств в данной сфере и перенять для себя наиболее приемлемые черты. По уровню государственного финансирования здравоохранения наша страна находится на уровне стран с низким уровнем дохода. И это несмотря на то, что за последние 10 лет расходы на здравоохранение увеличились почти в 6 раз. Как следствие, налицо низкое качество медицинских услуг. Продолжительность жизни населения Украины составляет всего 68,2 года, что в среднем на 10 лет ниже, чем в странах ЕС. Уровень преждевременной смертности вследствие несчастных случаев на рабочих местах в 3 раза превышает аналогичный в странах ЕС (448 против 146 случаев на 100 000 чел.). В Украине в 2011 году расходы на здравоохранение одного жителя страны составили 920 грн., или чуть больше 115 долл., в то время как в США этот показатель был равен 1493 долл. Больше американцы тратят только на питание и жилье. При существующих конституционных гарантиях безвозмездности медицинской помощи значительная часть медицинских услуг является по сути платной. Пациенты оплачивают необходимые лекарства и расходные материалы, питание, постельное белье и даже услуги младшего медицинского персонала.

В данный момент необходимо провести реформирование страхования от нетрудоспособности. Задача, стоящая перед программами, предназначенными для помощи инвалидам, состоит в том, чтобы они были с одной стороны справедливыми и достаточными, а с другой стороны позволяли экономно расходовать средства. Финансирование систем страхования от нетрудоспособности должно быть таким, чтобы минимизировать неблагоприятные последствия, которые оно может создать для занятости и экономической деятельности. К сожалению, не существует ясных руководств на предмет того, как именно должна решать страна вопрос относительно введения программы социального страхования или же программы частного социального страхования от нетрудоспособности, дополненной по возможности системой государственных пособий. Выбор между социальным страхованием, обязательным частным страхованием или сочетанием этих двух систем должен быть тщательно продуман с учётом достоинств и недостатков каждого варианта. Так, например, существенным

недостатком исключительно бюджетного финансирования здравоохранения и медицинского страхования является склонность к монополизму, игнорирование прав пациентов, как правило, отсутствие возможности выбора врача, лечебного учреждения. А частное добровольное медицинское страхование является довольно убыточным продуктом, поскольку потребители услуг всегда стараются максимально использовать все медицинские услуги, которые предусмотрены страховкой. Это вынуждает страховые компании устанавливать высокий процент франшизы в случае, если клиент желает воспользоваться дорогими медицинскими услугами.

По нашему мнению, наиболее удачным сочетанием является второй вариант, где сочетается государственное и частное страхование. Так как существенным недостатком бюджетного финансирования здравоохранения и медицинского страхования является склонность к монополизму, игнорирование прав пациентов, как правило, отсутствие возможности выбора врача, лечебного учреждения.

Литература

1. Абрамов В. Ю. Страхование: теория и практика. / В. Ю. Абрамов. – М.: «Волтерс Клувер», 2007. – 221 с. **2. Гвозденко О. А.** Основы страхования. / О. А. Гвозденко. – Киев, 2009. – 234 с. **3. Дюжиков Е. Ф.** Страхование. / Е. Ф. Дюжиков, Е. А. Сплетухов. – ИНФРА-М, 2007. – 312 с. **4.** http://www.ukrstat.gov.ua.

Корячкіна О. В., Іє О. М. Страхування з непрацездатності

У статті розглянуто проблеми, пов'язані з недосконалістю сучасного обов'язкового соціально страхування в Україні.

Ключові слова: охорона здоров'я, соціальне страхування, система накопичувального страхування, Фонд соціального страхування.

Корячкина О. В. , Ие О. Н. Страхование по нетрудоспособности

В статье рассмотрены проблемы, связанные с несовершенством современного обязательного социально страхования в Украине

Ключевые слова: здравоохранение, социальное страхование, система накопительного страхования, Фонд социального страхования.

Koryachkina O. V., Ie O. N. Disability insurance

The problems associated with the imperfection of modern compulsory social insurance in Ukraine

Keywords: health care, social insurance, endowment insurance system, social insurance.

УДК 004.421:004.65

В.А. Кулешова, Н.В. Понамарев, А.С. Титаренко

РАБОТА С БАЗАМИ ДАННЫХ В С++

В современном мире одной из главных задач является создание систем для обработки информации, от которой зависит работа и деятельность различных предприятий, учреждений, научных заведений и др. Такие системы должны обладать рядом функций, в большей или меньшей степени удовлетворяющие запросам пользователя. Это и анализ информации, занесение новой информации, ее редактирование, получение отчетов, различных показателей и все это должно происходить без задержек и потери информации. Программирование базы данных является одним из наиболее приоритетных и востребованных направлений в сфере информационного обеспечения. В работе с базой данных известны некоторые недостатки:

- так как все поля одной таблицы должны содержать постоянное число полей заранее определенных типов, приходится создавать дополнительные таблицы, учитывающие индивидуальные особенности элементов, при помощи внешних ключей. Такой подход сильно усложняет создание сколько-нибудь сложных взаимосвязей в базе данных;
- высокая трудоемкость манипулирования информацией и изменения связей.

Цель этой статьи и заключается в том, чтобы разработать приложение гибкое и удобное для пользователя, работающее с БД. В данной статье мы используем язык программирования Visual C++ с большими возможностями для программирования баз данных, который позволит быстро создавать необходимые компоненты приложений, более быструю по скорости работу, которую трудно, а иногда невозможно разработать средствами «классических» СУБД, то есть технология «клиент-сервер». Мы стремились упорядочить информацию по различным признакам и быстро извлекать выборку с произвольным сочетанием признаков. Сделать это возможно, только если данные структурированы.

Структурирование — это введение соглашений о способах представления данных.

База данных (БД) — это поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Система управления базами данных (СУБД) — это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации.проектирование структуры базы данных

В программных комплексах различного назначения основную роль играют интегрированные базы данных.

Использование баз данных обеспечивает:

- представление в памяти персонального компьютера сложных структур информации, когда объектом хранения являются не только данные, но и структуры, в которых они организованы;
- сокращение дублирования информации за счет структурирования данных, что приводит к экономии памяти на внешних носителях и повышает надежность информации;
- целостность и сохранность данных от неавторизованного доступа;
- независимость прикладных программ от изменений описаний данных и наоборот;
- повышение достоверности информации и сокращение затрат на обслуживание системы;
- интеграцию информации, в результате чего эта информация становится объектом обработки множества пользователей и программ.

Поскольку необходима сетевая БД, для реализации была выбрана база данных MySQL [1].

Основные объекты предметной области

Исходя из анализа предметной области ателье и документооборота, можно определить основные объекты предметной области: клиенты, заказы, модели, материал.

Анализ определенных выше объектов и атрибутов позволяет выделить сущности проектируемой базы данных и построить ее инфологическую модель на языке «сущность – связь». Инфологическая модель предметной области призвана формализовать представление о предметной области.

Инфологическая модель структуры данных

На основе приведенного выше анализа предметной области построена инфологическая модель, описанная выше и представленная в форме диаграммы «сущность – связь».

Сущность «Материалы» (код материала, название, ширина, цена);

Сущность «Модели» (код модели, рекомендуемая ткань, расход ткани, цена);

Сущность «Заказы» (ФИО покупателя, информация о модели, информация о ткани, ФИО закройщика, дата приема заказа, дата примерки, выполнение заказа, дата выполнения);Сущность «Склад» (код материала, метраж);

Сущность «Расход» (код материала, оставшийся материал).

Помимо самих сущностей в инфологической модели отображают связи между сущностями. Каждая связь между сущностями характеризуется названием и типом.

В инфологической модели предметной области представлены следующие связи между сущностями:

- «Выбор модели» – «Клиенты» и «Модели» (1:1);
- «Выбор материала» – «Клиенты» и «Материалы» (1:1);
- «Расходуется» – «Склад» и «Модели» (1:1);

Инфологическая модель предметной области представляет собой схему из 5 сущностей и 3 связей между ними.

Даталогическая модель базы данных

Даталогическая модель предметной области представлена 5 взаимосвязанными таблицами.

Таблица «material_catalog» содержит данные о материалах

Наименование	Тип	Размер	Ограничения
1	2	3	4
Id	int	16	Уникальное
Title	char	128	
Width	float		
Price	float		

Уникальным ключом в таблице material_catalog является поле Id – уникальный код каждого материала. Таблица является подчиненной и использует данные из таблицы «material_warehouse».

Таблица «model_catalog» содержит данные о моделях одежды.

Наименование	Тип	Размер	Ограничения
1	2	3	4
Id	int	16	Уникальное
Recommended_material	int	16	
Material_use	float		
Price	float		

Уникальным ключом в таблице model_catalog является поле Id – уникальный код каждой модели. Таблица является подчиненной и использует данные из таблицы «material_warehouse».

Таблица «order» содержит данные о заказах.

Наименование	Тип	Размер	Ограничения
1	2	3	4
Full_Name	char	64	Уникальное
Model_ID	int	16	
Material_ID	int	16	
Cutter's_Full_Name	char	64	
Date_of_admission	date		

Date_of_fitting	date		
Execution_mark	tinyint	1	
Execution_date	data		

Первичным ключом в таблице order является поле Full_Name – уникальное имя каждого клиента. Таблица является подчиненной и использует данные из таблиц «model_catalog» и «material_warehouse».

Таблица «material_warehouse» содержит данные о материалах на складе.

Наименование	Тип	Размер	Ограничения
1	2	3	4
Id	int	16	Уникальное
Metric_area	int	16	

Уникальным ключом в таблице material_warehouse является поле Id – уникальный код каждого материала.

Таблица «material_left» содержит сведения о расходе тканей.

Наименование	Тип	Размер	Ограничения
1	2	3	4
Id	int	16	Уникальное
Material_left	int	16	

Уникальным ключом в таблице material_left является поле Id – уникальный код каждого материала. Таблица является подчиненной и использует данные из таблицы «material_warehouse».

Архитектура БД клиент – серверная.

Разработка программы

Проектирование интерфейса

Для реализации программы использовалась среда для разработки приложений Microsoft Visual Studio 2010.

Интерфейс данной программы содержит достаточное количество элементов для того, чтобы полностью отобразить все данные необходимые для просмотра интересующей информации. Интерфейс программы не перегружен лишними компонентами.

Контейнером для размещения элементов интерфейса является форма, которая представляет собой окно приложения на этапе разработки и обеспечивает создание интерфейса пользователя.

Для создания формы используется пункт меню Project→Add element→Windows Forms главного меню VS 2010. После создания формы на ней можно размещать элементы управления и элементы модуля данных (рис. 1).

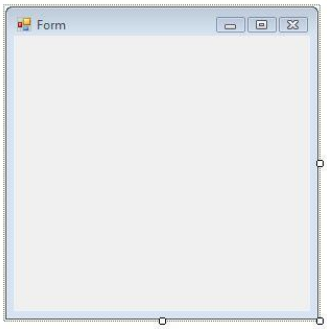


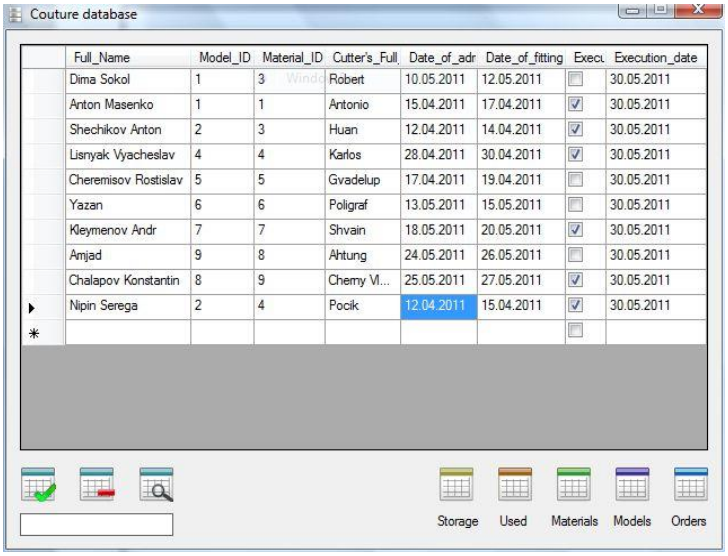
Рисунок 1 – Пустая форма

При запуске программы на экране монитора появляется главная форма, на которой расположены элементы управления программой. Элементы скомпонованы по смысловому признаку и дают описание всех предлагаемых пользователю возможностей. Для создания меню использовался элемент Button .

Элементы управления программы (см. рис. 2):

Рисунок 2– Элементы управления.

Для начала работы с приложением необходимо запустить .exe файл. После произойдет подключение к базе данных, и отобразиться форма приложения.



Full_Name	Model_ID	Material_ID	Cutter's_Full	Date_of_adr	Date_of_fitting	Execu	Execution_date
Dima Sokol	1	3	Robert	10.05.2011	12.05.2011	☐	30.05.2011
Anton Masenko	1	1	Antonio	15.04.2011	17.04.2011	☑	30.05.2011
Shechikov Anton	2	3	Huan	12.04.2011	14.04.2011	☑	30.05.2011
Lisnyak Vyacheslav	4	4	Karlos	28.04.2011	30.04.2011	☑	30.05.2011
Cheremisov Rostislav	5	5	Gvadelup	17.04.2011	19.04.2011	☐	30.05.2011
Yazan	6	6	Poligraf	13.05.2011	15.05.2011	☐	30.05.2011
Kleymenov Andr	7	7	Shvain	18.05.2011	20.05.2011	☑	30.05.2011
Amjad	9	8	Ahtung	24.05.2011	26.05.2011	☐	30.05.2011
Chalapov Konstantin	8	9	Cherny Vl...	25.05.2011	27.05.2011	☑	30.05.2011
Nipin Serega	2	4	Pocik	12.04.2011	15.04.2011	☑	30.05.2011
						☐	

Рисунок 3– Главная форма приложения.

По умолчанию .exe файл находится в папке Release.

Для перемещения по таблицам используется элемент Button под главным окном вывода таблиц.

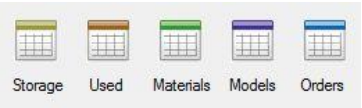


Рисунок 4 – «Кнопки» с названием таблиц БД.

Для работы с записями в приложении используется компонент dataGridView(см. рис. 5).

Id	Recommended_material	Material_use	Price
1	1	5	45
2	1	5	65
3	2	5	36
4	3	5	52
5	4	5	64
6	6	5	9
7	7	5	53
8	5	5	4
9	3	5	10
10	4	5	9

Рисунок 5 – Компонент dataGridView.

Он позволяет добавлять записи, выполнять перемещение по таблице, сортировать записи, перемащать колонки таблиц, изменять размр колонок, копировать данные из ячеек.

В приложении возможна сортировка в таблицах. Можно выбрать как поле сортировки, так и ее направление. Для сортировки используется элемент dataGridView.

Также возможно осуществление поиска. Необходимо ввести текст поиска и нажать кнопку поиска. Для организации поиска используются элементы textBox и Button.

После редактирования таблиц следует произвести их обновление кнопкой Update Для удаления строк существует кнопка Remove .

Описание кодов программы

При работе с базами данных удобно использовать хранилище данных dataSet. Модуль данных представляет собой контейнер для размещения таблиц из БД .

В программе используется 5 элементов mySqlDataAdapter для каждой из таблиц. С помощью этих элементов осуществляется выборка SELECT и заполнение dataSet данными из таблиц. Также создан элемент

mySqlConnection для соединения с БД. Элемент MySqlCommandBuilder осуществляет автоматическое создание команд DELETE и UPDATE.

Фрагмент кода на C ++ [2]для работы с БД

```
connection = gcnew MySqlConnection("server=localhost;User Id=root;database=couture");
adapter = gcnew MySqlDataAdapter();
adapter1 = gcnew MySqlDataAdapter();
adapter2 = gcnew MySqlDataAdapter();
adapter3 = gcnew MySqlDataAdapter();
adapter4 = gcnew MySqlDataAdapter();
ds = gcnew DataSet;
cb = gcnew MySqlCommandBuilder(adapter);
cb1 = gcnew MySqlCommandBuilder(adapter1);
cb2 = gcnew MySqlCommandBuilder(adapter2);
cb3 = gcnew MySqlCommandBuilder(adapter3);
cb4 = gcnew MySqlCommandBuilder(adapter4);
```

В данной работе, с помощью создания собственного примера, были рассмотрены основные возможности баз даних в Visual C++ . На собственном опыте убедились, что хоть и СУБД является универсальной, но использование дополнительного программного обеспечения делает работу с информацией намного удобнее и быстрее, экономя время пользователя, программиста или заурядного работника.

Литература

1. Р.Лафоре «Объектно-ориентированное программирование на с++». **2. Виктор** Гольцман «MySQL 5.0.Библиотека программиста».

Кулешова В.А., Понамарьов Н.В., Титаренко О.С. Робота з базами даних в С++

У даній статті ми використовуємо мову програмування Visual C++ з великими можливостями для програмування баз даних, який дозволить швидко створювати необхідні компоненти додатків, більш швидку по швидкості роботу, яку важко, а іноді неможливо розробити засобами «класичних» СУБД, тобто технологія «клієнт-сервер».

Ключові слова: СУБД, програмування баз даних, Visual C++

Кулешова В.А., Понамарев Н.В., Титаренко А.С. Работа с базами данных в С++

В данной статье мы используем язык программирования Visual C++ с большими возможностями для программирования баз данных, который позволит быстро создавать необходимые компоненты приложений, более быструю по скорости работу, которую трудно, а иногда невозможно разработать средствами «классических» СУБД, то есть технология «клиент-сервер».

Ключевые слова: СУБД, программирования баз данных, Visual C++

Kuleshov VA Ponomarev, NV, AS Titarenko Working with database in C++

In this article, we use the programming language Visual C++ with great opportunities for database programming, which enables you to quickly create the necessary components of applications, faster speed work, which is difficult and sometimes impossible to develop a means of "classical" database, ie technology "client-server".

Keywords: database, database programming, Visual C++

УДК 026:004.7

В.В. Рабой

РАЗРАБОТКА ЭЛЕКТРОННОЙ БИБЛИОТЕКИ НАУЧНОЙ И УЧЕБНОЙ ЛИТЕРАТУРЫ

Актуальность работы. В условиях современного динамического развития общества, информация становится таким же стратегическим ресурсом, как традиционные материальные и энергетические ресурсы. Наибольший экономический и социальный успех сегодня сопутствует тем странам, которые активно используют современные средства компьютерных коммуникаций и сетей, информационных технологий и систем управления информационными ресурсами. Становление современного информационного общества немыслимо без использования информационных ресурсов в электронном виде. Переведенные в электронную форму и собранные в общую систему информационные ресурсы приобретают новый статус, при котором реализуется качественно иной уровень производства, хранения, организации и распространения самой разнообразной информации (текст, графика, аудио, видео и др.), обеспечивая им более широкое распространение и эффективное использование.

Цель разработки: является совершенствование и повышение эффективности и оперативности обслуживания читателей, расширение способов сохранения культурного наследия, хранящегося в ее фондах, предотвращение износа физических экземпляров, восполнение лакун и приобретение дезидерат, расширение потенциально доступных источников комплектования библиотечных фондов, а также обеспечение дополнительных возможностей научно-исследовательской работы.

Для достижения этих целей в данном проекте выполняется разработка базы данных для информационной системы «Электронная библиотека».

Информационная система выполняет функцию автоматизации процесса сбора информации и создания необходимых документов. Интерфейс программы должен быть удобным и понятным для пользователя. Необходима удобная навигация по базе и наглядное отображение результатов работы. Система должна осуществлять хранение и обработку информации и формировать выходные документы.

Данная система должна функционировать под управлением операционной системы Windows 98, Windows 2000 и Windows XP и пакета программ Microsoft Office 2003 или Microsoft Office 2007.

Для начала работы с информационной системой необходимо заполнить базу Электронной библиотеки.

Выходные данные оформляются в процессе заполнения самой базы данных.

Для обеспечения целостности данных в информационной системе предусмотрена функция «пароль», которая блокирует несанкционированную модификацию каких-либо данных.

Для правильного функционирования информационная система «Электронная библиотека» была произведена нормализация отношений таблиц базы данных. Данные были разбиты на 3 таблицы. Что позволяет обеспечить правильность ввода и корректировки данных, выполняемых в процессе различных действий и операций по работе с данными, хранящимися в этих таблицах. Таблицы баз данных разрабатывались в среде разработки баз данных Access и имеют формат mdb.

Рисунок 6 - Схема взаимодействия таблиц базы данных

Контекстная диаграмма бизнес-процесса решаемой задачи «Электронная библиотека» выполненная в программе Computer Associates BPwin представлена ниже с использованием ER диаграмм.

Главная композиция контекстной диаграммы «Электронная библиотека» содержит следующие данные: вход – Информация о пользователе, Информационная система, Информация о книге;

управление – требуемые данные; механизм (ресурс) – пользователь и ПК, Оператор; выход – Архив, Книга. Диаграмма представлена на рисунке 2.

Декомпозиция первого уровня контекстной диаграммы «Электронная библиотека» представлена на рисунке 3.

Декомпозиция блока «пополнение базы данных библиотеки» представлена на рисунке 4.

Декомпозиция блока «Поиск книг» представлена на рисунке 5.

Декомпозиция блока «Формирование отчета» представлена на рисунке 6.

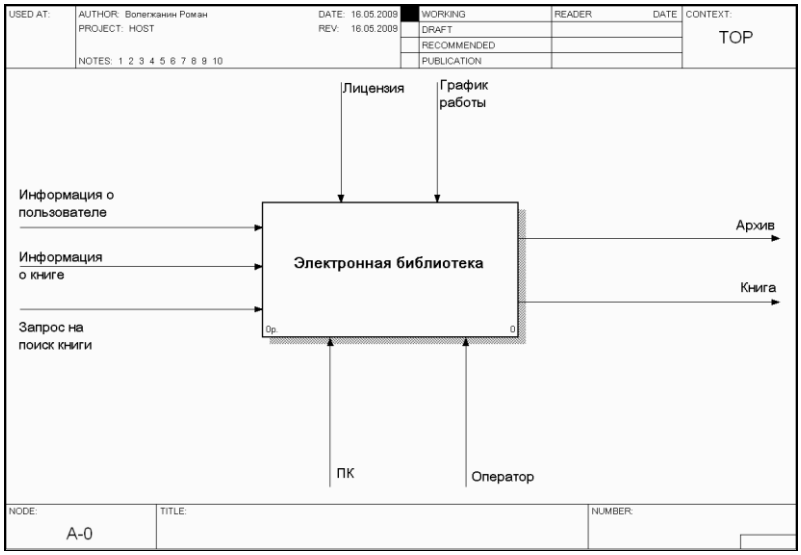


Рисунок 7 - Диаграмма "Электронная библиотека"

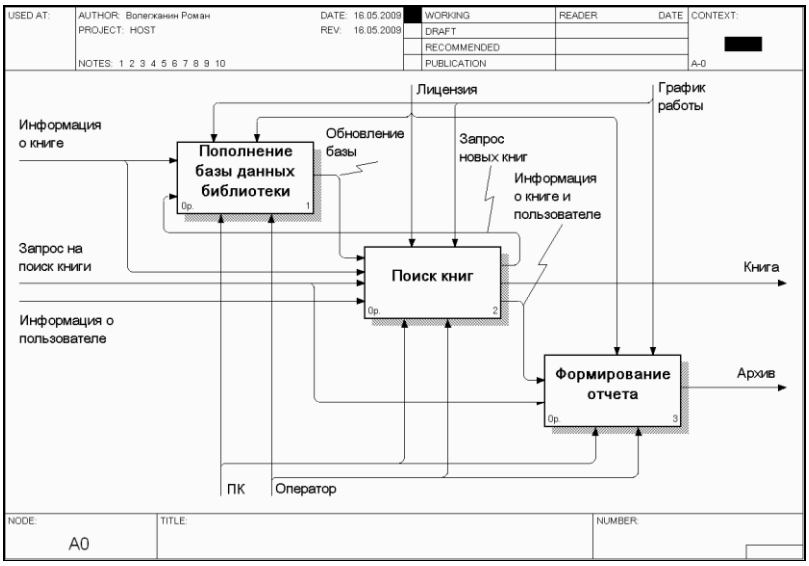


Рисунок 8 - Декомпозиция первого уровня контекстной диаграммы "Электронная библиотека"

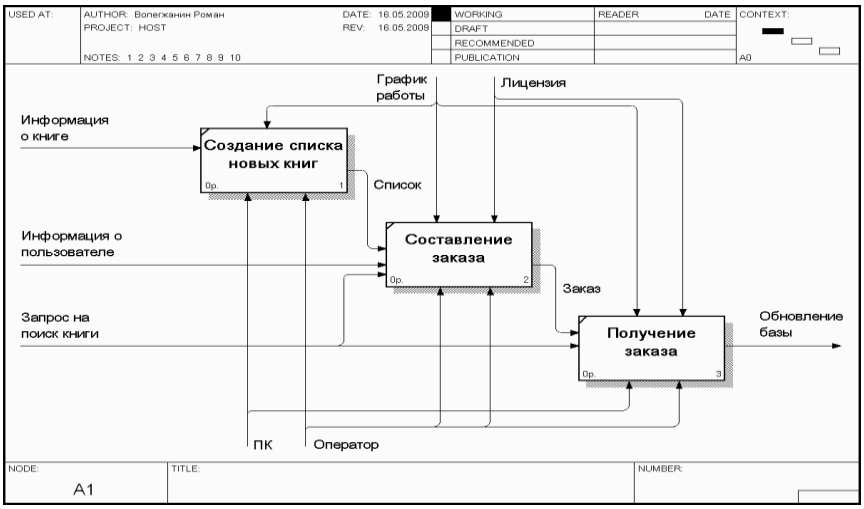


Рисунок 9 - Декомпозиция блока "пополнение базы данных библиотеки"

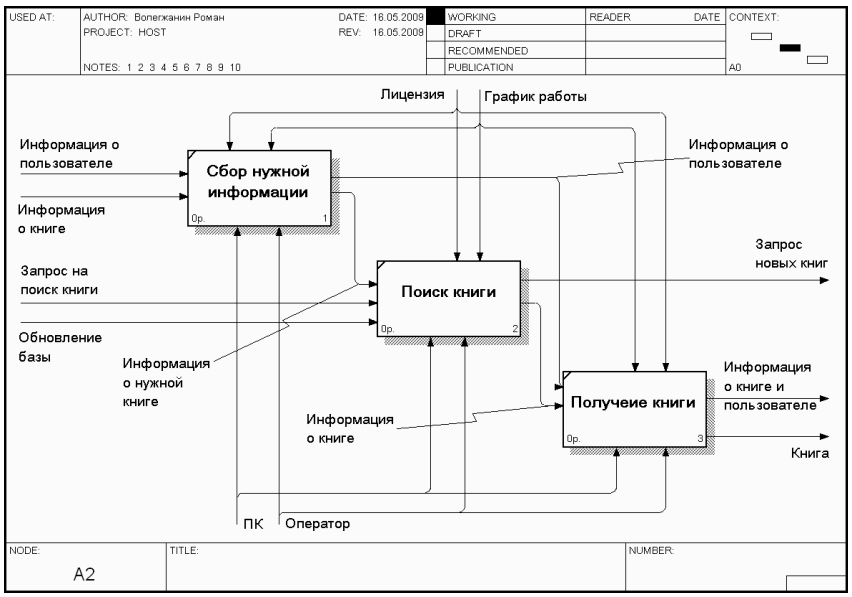


Рисунок 10 - Декомпозиция блока "Поиск книг"

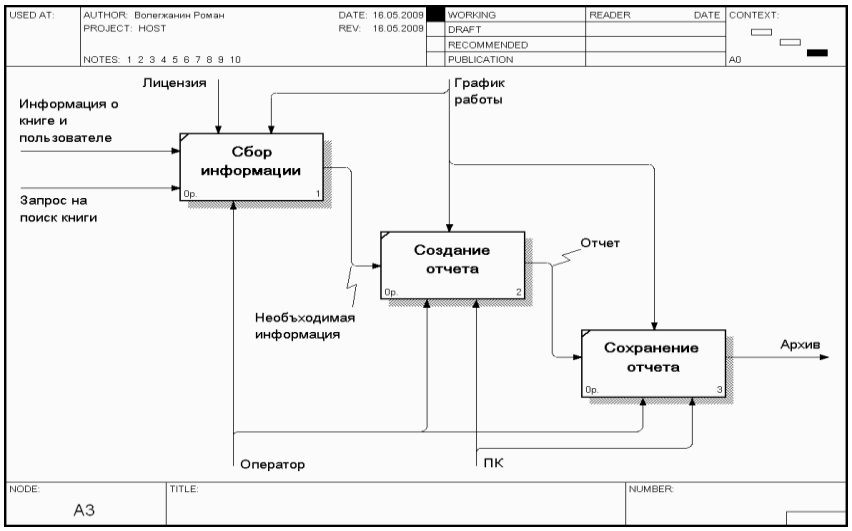


Рисунок 11 - Декомпозиция блока "Формирование отчета"

Программа «Электронная библиотека» предназначена для хранения и поиска информации о книгах. Система осуществляет хранение и обработку информации и формирует выходные документы. В дальнейшем предполагается, что программа будет использоваться постоянно и будут разрабатываться ее новые версии.

Программа разработана в системе Microsoft Office 2007. Занимает программа около 20 Мб памяти на жестком диске.

При входе в программе можно увидеть в главную форму:

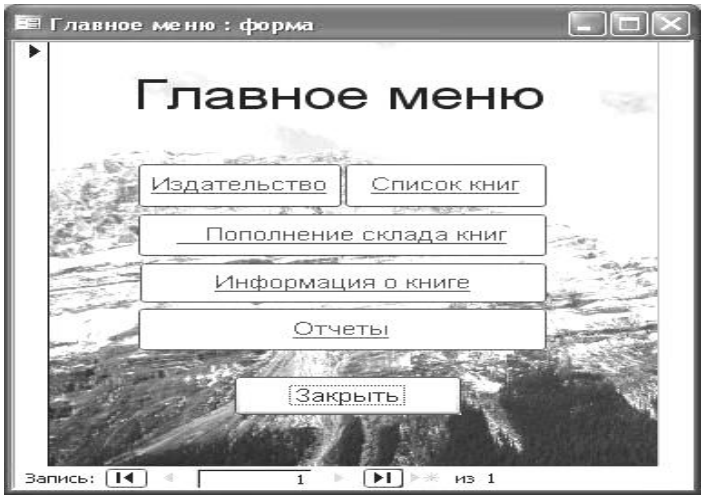


Рисунок 12 - Главная форма базы данных

Данная программа должна позволять пользователю информационная система «Электронная библиотека» облегчить ввод и редактирование, поиск и фильтрацию информации. Программа должна выполнять следующие функции: ввод и редактирование информации,

поиск информации по критериям; фильтрацию информации по показателям, формировать и распечатывать выходные документы.

Основные действия, которые пользователь может выполнить при работе с программой, приведены в таблице 1

Таблица 1 - Руководство пользователя

Операция	Действия
1	2
Запуск программы	Для запуска программы необходимо открыть базу данных «Электронная библиотека»
Выбор необходимого действия (открытие нужной формы)	Открытие необходимой формы производится путем нажатия соответствующей кнопки в главном окне программы
О программе	Для просмотра информации о программе нужно нажать на кнопку F1 программе
Печать отчетов	Для печати отчетов нужно нажать на кнопку печать отчетов и выбрать нужный отчет
Выход	Для выхода из программы нужно нажать кнопку Закрыть

Данная программа предназначена для удобного ввода и редактирования, поиска и хранения информации. Программа обеспечивает удобное отображение, добавление, редактирование, резервное копирование данных, а также формирование выходных документов.

Литература

1. Хабракен Дж. «Microsoft Access 2000», М.:ООО «Издательство АСТ», 2004, 349с. 2. Харитонова И., Вольман Н. «Программирование в Access 2002», Спб.: Питер, 2003, 480с. 3. Мейер М. Теория реляционных баз данных. – М.: Мир, 1987. – 608 с. 4. Хаббард Дж. Автоматизированное проектирование баз данных. – М.: Мир, 1984. – 294 с. 5. Тиори Т., Фрай Дж. Проектирование структур баз данных. В 2 кн., – М.: Мир, 1985. Кн. 1. – 287 с.: Кн. 2. – 320 с. 6. Кириллов В.В. Структуризованный язык запросов (SQL). – СПб.: ИТМО, 1994. – 80 с.

Рабой В.В. Разработка электронной библиотеки научной и учебной литературы

В статье представлена работа по разработке электронной библиотеки научной и учебной литературы. Определены основные требования к программе. Данная программа должна позволять пользователю облегчить ввод и редактирование, поиск и фильтрацию информации. Создан программный продукт на основе базы данных Access

Ключевые слова: электронная библиотека, информационная система, база данных.

Рабой В.В. Розробка електронної бібліотеки наукової та навчальної літератури

У статті представлена робота по розробці електронної бібліотеки наукової та навчальної літератури. Визначено основні вимоги до програми. Дана програма повинна дозволяти користувачеві полегшити введення і редагування, пошук і фільтрацію інформації. Створено програмний продукт на основі бази даних Access

Ключові слова: електронна бібліотека, інформаційна система, база даних.

Raboy V.V. Development of a slave digital library of scientific and educational literature

The paper presents work on developing an electronic library of scientific and educational literature. The basic requirements for the program. The program should allow the user to facilitate the entry and editing, searching, and filtering of information. Created a software product based on an Access database

Keywords: electronic library information system database.

УДК 004.421:004.651.5

А.А. Сычова

РЕАЛИЗАЦИЯ ФУНКЦИЙ РАБОТЫ ДИНАМИЧЕСКОГО ЦИКЛИЧЕСКОГО СПИСКА ДЛЯ БИБЛИОТЕКИ STL

Цель статьи проанализировать возможности решения задач посредством использования динамических структур, рассмотреть основные компоненты стандартной библиотеки STL и их возможности. Рассмотреть реализацию обработки данных в циклическом (кольцевом) списке и средствами контейнера list.

Актуальность. При разработке программного кода важно помнить о том, что существует ряд характеристик, определяющих его качество. Некоторые характеристики важны только для человека, как, например, читабельность и структурированность программного кода. К наиболее важным, относятся легкость поддержки, отладки, низкое использование ресурсов (памяти и процессорного времени), низкая сложность кода.

Среди наиболее значимых факторов качества программного обеспечения можно выделить эффективность - насколько рационально программа относится к ресурсам (память, процессор) при выполнении задач [1]. Повышают эффективность программного кода динамическое

распределение памяти, которое может быть реализовано через динамические структуры, частично реализованные в библиотеке STL.

STL (Standart Template Library) - стандартная библиотека шаблонов. Архитектура STL была разработана Александром Степановым и Менг Ли. STL содержит подборку классов коллекций для различных целей, а так же поддерживает ряд алгоритмов для работы с этими коллекциями. Все компоненты STL оформлены в виде шаблонов, поэтому могут использоваться с произвольными типами элементов. Так же библиотека формирует архитектуру для включения других классов коллекций и алгоритмов, работающих в сочетании с существующими коллекциями и алгоритмами. В библиотеке STL программно реализована работа со списками, стеками, очередями, итераторами и другими динамическими структурами, объектами и т.д.

К преимуществам библиотеки относятся[2]:

- Гибкость библиотеки.
- Код библиотеки разработан и отлажен профессиональными программистами, что исключает возможные ошибки.
- Код библиотеки написан эффективно с точки зрения использования оперативной памяти и быстродействия для типовых вариантов применения.
- Библиотека предлагает унифицированный интерфейс, однообразный для всех контейнеров и алгоритмов, что с наличием навыка использования библиотеки позволяет значительно повысить читаемость программы.
- Использование библиотеки позволяет приступить сразу к решению проектных задач.
- Библиотека хорошо документирована и описана.
- Код является встроенным в стандартную библиотеку C++.

Недостатки STL состоят в:

- Невозможности комбинировать операции для вручную созданного динамического списка и шаблонного. Так же библиотека не содержит такой динамической структуры данных, как циклический список.
- Сложности управления пулом памяти при работе с контейнерами STL.

Составляющие библиотеки

Работа с STL основана на взаимодействии различных структурных компонентов. Она содержит большое количество шаблонов классов и функций. Однако выделяют три основных компонента:

- Итераторы
- Контейнеры
- Алгоритмы

Рассмотрим подробнее некоторые элементы компонента контейнер.

Контейнеры предназначены для управления коллекциями объектов определенного типа. Контейнеры могут быть реализованы в виде массивов или связных списков, а каждый элемент может снабжаться специальным ключом. Доступ к данным находящимся внутри контейнера осуществляется с помощью итераторов.

Список (list) реализуется в виде двусвязного списка элементов. То есть, каждый элемент списка занимает отдельный блок памяти и содержит ссылки на предыдущий и последующий элементы. Списки не поддерживают произвольный доступ к элементам, однако в список очень удобно вставлять и удалять элементы. Для работы со списком необходимо подключить файл <list>.

Дек (deque) происходит от сокращения фразы “double-ended queue” (двусторонняя очередь). Дек представляет собой динамический массив, реализованный таким образом, что может расти в обоих направлениях. Очень быстро выполняются операции добавления и удаления элементов в конец и в начало дека, однако вставка и удаление из середины занимают больше времени, так как выполняется перемещение элементов. Для работы с деком необходимо подключить файл <deque>.

Циклический список не реализован в библиотеке STL.

Кольцевой (циклический) список— структура данных типа FIFO (First In, First Out — первым вошел, первым вышел); это структура данных, использующая единственный буфер фиксированного размера, в котором за последним элементом сразу следует первый.

То есть, в циклическом списке нет элементов, которые содержат пустые указатели, и крайние элементы выделить нельзя. Так же, как и линейные, циклические списки могут быть односвязными и двусвязными.

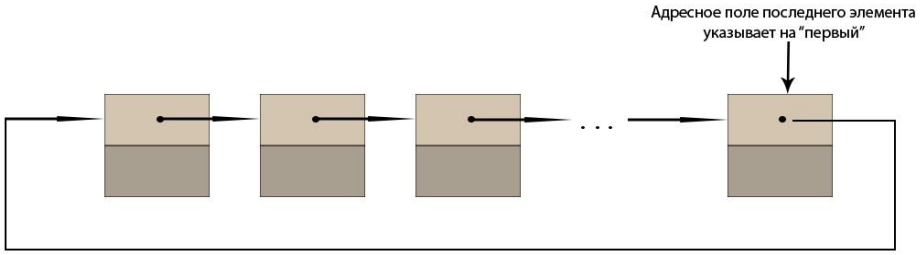


Рис. 1. Циклический однонаправленный список

Основные операции, выполняемые циклическим списком: создание списка; просмотр списка; удаление элемента; вставка элемента; поиск элемента; проверка на пустоту списка; удаление списка.

Подробнее по каждому из пунктов:

1) Создание списка:

Чтобы создать односвязный циклический список, нам потребуется функция, принимающая как аргумент количество элементов списка и два указателя:

void Make_List(int n, List** Head,List* Link)

До тех пор, пока переменная, равная количеству элементов в списке, будет больше нуля, создаем и инициализируем новый элемент. При создании каждого нового элемента, будем записывать его в следующее после. Функцию вызываем рекурсивно. С каждым проходом уменьшаем переменную, изначально равную количеству элементов, и переходим к созданию следующего элемента.

2) Просмотр списка

Создаем функцию, принимающую как аргументы указатель на «первый» элемент списка и количество элементов в нем:

void Print_List(List * Head, int temp)

Создаем вспомогательный указатель для перехода к следующему элементу. Пока переменная, изначально равная количеству элементов, больше нуля, выводим последовательно элементы на экран, переходя к следующему элементу с помощью цикла.

3) Удаление элемента:

Для удаления элемента из циклического списка требуется функция, возвращающая тип структуры. Как параметры функция принимает указатель на «голову» списка и номер (позиции) удаляемого элемента:

List* Delete_Item_List (List* Head, int Number)

По циклу проходим до нужного элемента, и перемещаем указатель на адресное после данного элемента на следующий. Для этого нам нужно создать два дополнительных указателя: первый – для хранения адресного поля данного элемента, другой – для предыдущего. Перемещаем указатель из адресного поля удаляемого элемента в предыдущий. Таким образом, элемент исключается из списка.

4) Вставка элемента:

Для вставки элемента требуется функция, возвращающая тип структуры. Она принимает как параметры указатель на «голову» списка, позицию вставляемого элемента и значение, которое будем записывать в поле данных:

List* Insert_Item_List(List* Head, int Number, int DataItem)

Чтобы вставить элемент в список, создаем два указателя типа динамической структуры. Полю данных первого присваивается значение вставляемого элемента, второй же нам требуется для того, чтобы по циклу дойти до предшествующей нужной позиции. Адресное поле предыдущего элемента указывает на новый элемент, а адресное поле нового – на следующий.

5) Поиск элемента

Для реализации данной операции нам нужна функция, возвращающая логический тип данных и принимающая как аргументы

указатель на первый элемент списка и значение элемента, который будем искать:

bool Find(List* Head, int DataItem)

Создаем вспомогательный указатель, благодаря которому будем передвигаться по циклу до тех пор, пока не дойдем снова до первого элемента. Если значение поля данных равно передаваемому значению, то возвращаем значение «true» (1). Если же элемент не найден, возвращаем значение «false» (0).

6) Проверка на пустоту списка

Проверка на пустоту списка осуществляется следующим образом:

Определяем функцию, возвращающую логический тип данных и принимающую как аргумент указатель на «первый» элемент списка:

bool Empty_Circle_Single_List(Circle_Single_List* Head)

Чтобы проверить, пуст ли список, нам достаточно определить пустоту первого элемента. Для этого рационально использовать тернарную операцию условного выбора, которая, если первый элемент равен нулю, возвращает значение «false» (0), а в случае его существования возвращает «true» (1).

7) Удаление списка

Для реализации данной операции определяем функцию, принимающую как аргументы указатель на «первый» элемент списка и количество элементов в нем:

void Delete_List(int n, List *Head)

До тех пор, пока переменная, равная количеству элементов, не станет равна нулю, последовательно удаляем элементы и уменьшаем значение переменной на единицу.

Главным достоинством связных списков (в частности циклического) является легкость операций вставки и удаления элементов. Главным недостатком – получение доступа к элементам. И чем больше в списке элементов, тем больше переходов придется совершить, в отличие от массива. Так же элемент связного списка занимает большее количество памяти, чем элемент массива.

Циклический список имеет большое преимущество перед линейным – он не имеет первого и последнего элемента, и чтобы добраться из «хвоста» списка в «голову» требуется совершить всего один шаг.

Разница между односвязным и двусвязным циклическими списками так же существенна. Используя двусвязный список, мы экономим время выполнения операции, но не экономим память, так как каждый элемент содержит по два адресных поля.

В качестве примера применения односвязного циклического списка ***решим следующую задачу***:

Создать динамический список, элементами которого являются символы. Создать функцию, которая возвращает единицу, если скобки

сбалансированы (каждой открывающей соответствует закрывающая скобка) и 0 в обратном случае.

Для реализации данной задачи используется циклический односвязный список в целях экономии памяти и так как каждая из созданных функций передвигается по элементам только в одном направлении.

Словесный алгоритм задачи:

Объявляем класс Circle. В private объявляем динамическую структуры типа List, содержащую поле данных типа char и адресное поле типа структуры. Так же объявляем указатели типа структуры на «первый» и «последний» элементы списка. В public объявляем переменную, которая хранит количество элементов списка и следующие функции: функцию заполнения односвязного циклического списка, вывода на экран, функцию проверки на сбалансированность скобок, а так же конструктор (для инициализации указателей в NULL) и деструктор (для освобождения динамической памяти после отработки программы).

В функции CheckBrackets проверяем, сбалансированы ли скобки (соответствует ли каждой открывающей скобке закрывающая). Функция принимает как параметр количество элементов списка и возвращает логический тип данных. Внутри создаем цикл, который проходит по всем элементам списка. Если поле данных содержит открывающую скобку, то увеличиваем счетчик, если закрывающую – уменьшаем. Цикл работает только в том случае, когда счетчик неотрицателен. Если после прохода по всем элементам счетчик равен нулю, то возвращаем 1 (скобки сбалансированы), если же содержит другое значение – возвращаем 0 (скобки не сбалансированы). В главной функции с помощью указателя последовательно вызываем функции создания, вывода списка на экран и проверки на сбалансированность скобок.

Листинг программы:

```
#include <iostream>
using namespace std;
 class Circle {
        struct List { char Data;
                List *Next; };
     List *Head,*Tail; //Указатели для 1-го и последнего элементов
  public:
         int n;  //к-во элементов
     Circle(); //Инициализация элементов в конструкторе
    ~Circle(); //Деструктор для освобождения памяти
     void Make_List(int Data); //Функция, заполняющая список
     void Show(int n); //Функция вывода списка на экран
         bool CheckBrackets(int n); //Функция, определяющая, сбалансированны ли скобки
};
 Circle::Circle(){ Head = NULL;
```

```
            Tail = NULL; }
        Circle::~Circle() {
          while (n!=0)    {    //Пока переменная, изначально равная к-ву элементов списка, не равна 0
            List *temp=Head->Next;
            delete Head; //Освобождаем память от активного элемента
            Head=temp; //Переходим к следующему
            n--; //и уменьшаем переменную
           } }
        void Circle::Make_List(int n){
         if(n>0){ List *temp=new List; //Выделение памяти для нового элемента списка
          temp->Next=Head;   //Замыкание контура. Последний элемент - это начало списка
          cout<<"-> > "; cin>>temp->Data; //Инициализация поля данных
        if (Head!=NULL) {              //Если список не пустой
             Tail->Next=temp;  //Запись данных в следующее за последним элементом поле
           Tail=temp; }            //Последний активный элемент - только что созданный.
        else Head=Tail=temp;    //Если список пуст то создается первый элемент.
        Make_List(n-1);         //функцию вызываем рекурсивно
            }}
        void Circle::Show(int n) {
            List *Curr=new List; //новый указатель для "первого" элемента списка
            Curr=Head;
            while (n!=0) {//пока переменная, равная количеству элементов списка, не равна 0
                cout<<Curr-> Data<<'\t'; //то выводим элемент на жкран
                Curr=Curr->Next;  //Переходим к следующему элементу списка
                n--;  }//Уменьшаем переменную n
            cout<<endl; }
        bool Circle::CheckBrackets(int n){ //функция определения соответствия открывающих скобок закрывающим
            int counter = 0;
            for(int i = 0; i < n; i++){
                if(counter>=0){       //если счетчик неотрицательный
                if(Head->Data == '(') {//если найдена открывающая скобка, счетчик увеличивается,
                        counter++;  }
```

```
                    if(Head->Data == ')') { //если закрывающая - уменьшается
                                        counter--; }
            Head = Head->Next; }
                    }
                    if (counter==0) return true; //если счетчик после прохода равен 0, то скобки сбалансированы
                    else return false; }          //если же нет, то возвращаем 0
            void main(){
                    setlocale(0, "");
              Circle *lst = new Circle;
              cout<<"Укажите количество пар элементов списка: "; cin>>lst->n;
              (lst->n)*=2;
              lst->Make_List(lst->n);
              lst->Show(lst->n);
              if(lst->CheckBrackets(lst->n) == 1) cout<<"Скобки сбалансированны";
              else cout<<"Скобки не сбалансированны";
              cin.get();
              cin.get(); }
```

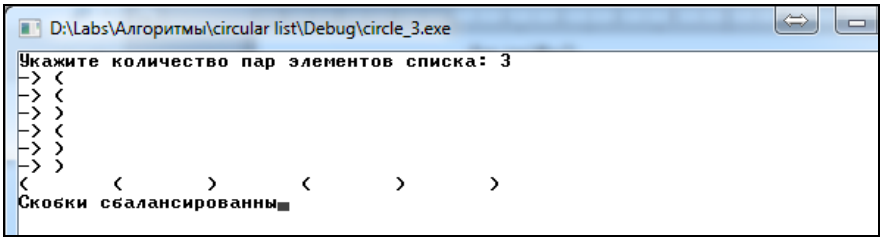


Рис 2. Результат выполнения программного кода.

Задача 2 представлена как пример работы с контейнером list библиотеки STL.

Словесный алгоритм:

Для создания динамического двусвязного списка средствами STL нам требуется использовать контейнер list, итераторы и алгоритмы.

Создаем две функции.

Функция printList выводит элементы списка на экран от первого до последнего через табуляцию.

Функция Menu предлагает выбрать операцию: в случае 1 выполняется удаление элемента из списка, в случае 2 – вставка элемента, 3 – сортировка элементов списка по возрастанию, 4 – перестановка элементов в обратном порядке, 5 - завершение работы. Каждая операция реализована средствами библиотеки STL. По завершении операции, кроме случая 5, вызываются функции вывода списка на экран и Menu (рекурсивно).

В главной функции определяем шаблон двусвязного динамического списка, количество элементов в нем. Заполняем список с помощью арифметического цикла for, вводя каждый новый элемент в «хвост» списка.

Последовательно вызываем функции printList и Menu, после завершения работы которых освобождаем память.

Листинг программы

```
#include <iostream>
#include <list>
#include <algorithm>
#include <iterator>
using namespace std;
void printList(list <int> list)
{
	cout<<"Список: ";
	copy(list.begin(), list.end(), ostream_iterator <int> (cout, "\t"));
	cout<<endl<<endl;}

void Menu(list <int> list){
	cout<<"Выберите операцию: "<<endl;
	cout<<"Удаление элемента - 1 "<<endl <<"Вставка элемента - 2 "<<endl
	  <<"Сортировка списка - 3 "<<endl<<"Переставить в обратном порядке - 4"<<endl
		<<"Завершить работу - 5"<<endl<<endl;
	int *counter = new int;  int *elem = new int;
	cin>>*counter;
	switch (*counter){
		case 1: {	cout<<"Введите элемент, который нужно удалить: "; cin>>*elem;
				list.remove(*elem);
				printList(list);
				cout<<endl<<endl;
				Menu(list); break; }
		case 2: {	cout<<"Введите элемент, который нужно вставить: "; cin>>*elem;
				list.insert(list.begin(),*elem);
				printList(list);
				cout<<endl<<endl;
				Menu(list); break; }
		case 3: {	list.sort();
				printList(list);
				cout<<endl<<endl;
				Menu(list); break; }
		case 4: {	list.reverse();
```

```
                    printList(list);
                    cout<<endl<<endl;
                    Menu(list); break; }
        default:  cout<<"Программа  завершила  работу"<<endl;
break;}
        delete [] counter, elem;
    }
    int main(){
        setlocale (0, "");
        list <int> list1;
        int *n = new int;
        cout<<"Введите кол-во элементов: ";      cin>>*n;
        int *x = new int;
        for(int i = 0; i < *n; ){                //организуем
арифметический цикл, в котором, пока i < *n ( n - количество элементов)
    cout<<"->> ";cin>>*x;  //вводим значение нового элемента,
    list1.push_back(*x);  //который добавляется конец списка;
    i++;                        //i  после  каждого  прохода  увеличивается
на единицу
        }
        printList(list1);
        Menu(list1);
        delete [] x, n;
        cin.get();      cin.get(); }
```

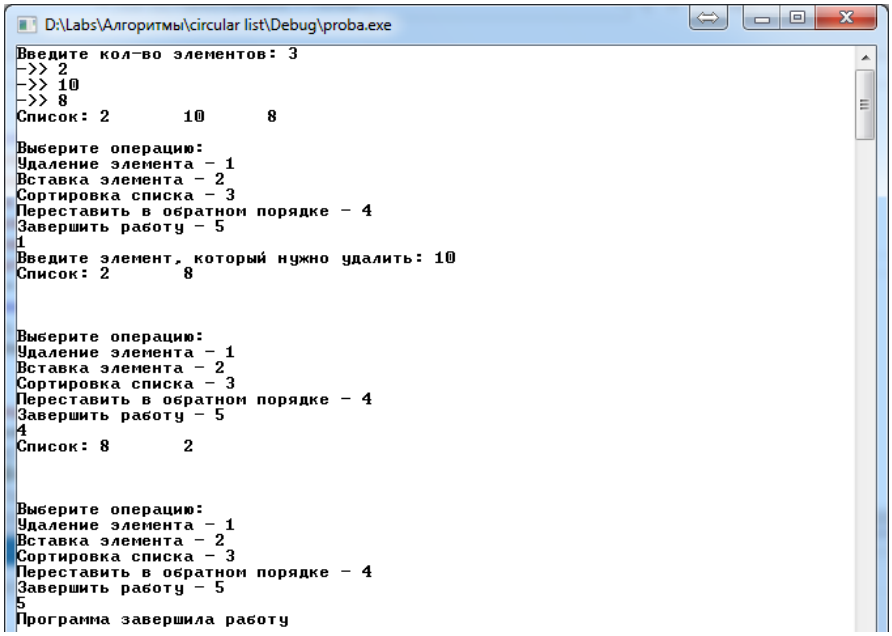


Рис. 3. Результат выполнения программы

Литература

1. Эффективность программного обеспечения – [Электронный ресурс]. – Режим доступа к статье: http://ru.wikipedia.org **2. Преимущества** и недостатки библиотеки STL – [Электронный ресурс]. – Режим доступа к статье: http://www.tenisheff.ru **3. Джосьютис Н.** C++. Стандартная библиотека. Для профессионалов / Н. Джосьютис. – СП Питер, 2004 -730 с.:ил. **4. Ключарев А.А.** Структуры и алгоритмы обработки данных: Учебное пособие/ А. А. Ключарев, В. А. Матьяш., С. В. Щекин. – СПБГУАП. СПб., 2003. 172 с.: ил.

Сычова А.А. Реализация функций работы динамического циклического списка для библиотеки STL

В статье рассматриваются вопросы организации работы циклического списка, дан краткий обзор библиотеки STL и ее функциональных возможностей, описаны ее слабые и сильные стороны. Проанализированы и обозначены достоинства и недостатки циклических списков. На практическом примере реализации функции определения сбалансированности скобок показаны преимущества циклического списка. На примере реализации операций удаления, вставки и сортировки элементов показана работа с контейнером list библиотеки STL.

Ключевые слова: Библиотека STL, динамические структуры, циклические списки, реализация.

Сичова Г.А. Реалізація функцій роботи динамічного циклічного списку для бібліотеки STL

У статті розглядаються питання організації роботи циклічного списку, дано короткий огляд бібліотеки STL та її функціональних можливостей, описано її слабкі та сильні сторони. Проаналізовано і позначені переваги і недоліки циклічних списків. На практичному прикладі реалізації функції визначення збалансованості дужок показані переваги циклічного списку. На прикладі реалізації операцій видалення, вставки та сортування елементів показана робота з контейнером list бібліотеки STL.

Ключові слова: Бібліотека STL, динамічні структури, циклічні списки, реалізація.

Sychova A.A. The implementation of the dynamic features of the circular list for the STL

The article describes the organization of the circular list, a brief overview of the STL and its strengths and weaknesses. Advantages and disadvantages of circular list are identified and analyzes. It is shown in the example of a program which does the checking of brackets in the sentence. It

shows the advantage of a circular list. On the example of the deletion, insertion, and sorting the work with the list – the STL container – is described.

Keywords: Library STL, dynamic structures, cyclic lists, implementation.

УДК 51-37

Я. В. Трофимов, А. В. Ореховский

СПОСОБЫ ОПТИМИЗАЦИИ ЭЛЕКТРОННОГО ТЕСТИРОВАНИЯ

В научной практике появилась новая форма проверки знаний – Интернет-тестирование. Тестирование внедряется не только в общеобразовательных школах, но стало обязательной формой контроля и в высших учебных заведениях. Несмотря на длительный период внедрения, оно вызывает противодействие не только со стороны учащихся и их родителей, но и со стороны преподавательского состава. В этой методике есть проблемы, затрудняющие образовательный процесс. Рассмотрим исторический аспект.

Первые тесты появились в начале XX века. Именно эти тесты вызывали настороженное к себе отношение у сторонников традиционной, "чистой" педагогической науки и практики.

Между тем тестовые методы не призваны заменять преподавателя и его личный опыт, а наоборот, призваны помогать ему, освободив его от рутинной работы и дав ему возможность сосредоточиться на повышении качества преподавания [4, с.51].

Тест обладает составом, целостностью и структурой. Он состоит из заданий, правил их применения, оценок за выполнение каждого задания и рекомендаций по интерпретации тестовых результатов.

Целостность теста означает взаимосвязь заданий, их принадлежность общему измеряемому фактору. Каждое задание теста выполняет отведенную ему роль и потому ни одно из них не может быть изъято из теста без потери качества измерения. Структуру теста образует способ связи заданий между собой. В основном, это так называемая факторная структура, в которой каждое задание связано с другими через общее содержание и общую вариацию тестовых результатов. Время нередко называется в качестве дополнительного системообразующего фактора. Каждый тест имеет оптимальное время тестирования, уменьшение или превышение которого снижает качественные показатели теста. Содержание теста можно определить как оптимальное отображение учебного материала в системе тестовых заданий. Уровень знаний выявляется при анализе ответов каждого ученика на все задания теста. Чем больше правильных ответов, тем выше

индивидуальный тестовый балл испытуемых. Один и тот же уровень знаний может быть получен за счет ответов на различные задания.

Существуют два основных класса тестов: традиционные и нетрадиционные.

Традиционный тест представляет собой единство, по меньшей мере, трех систем: содержательной системы знаний, описываемой языком проверяемой учебной дисциплины, формальной системы заданий возрастающей трудности и статистических характеристик заданий и результатов испытуемых.

К нетрадиционным можно отнести интегративные, адаптивные, многоступенчатые и так называемые тесты с критериально-ориентированной интерпретацией результатов [3, с. 20].

Ключевым моментом при этом является содержание теста. Он обязан включать такое количество контрольного материала, ответы на который с высокой степенью вероятности (больше 95 %) свидетельствовали бы об уровне подготовленности каждого учащегося.

На данный момент используется четыре формы заданий: задания с выбором одного или нескольких правильных ответов из числа предложенных, задания открытой формы что (предполагает ввод ответа в соответствующее поле), задания на установление соответствия, задания на установление правильной последовательности действий [1, с.1 , 2, с.23].

Но, как показывает практика, данный метод еще не настолько хорош, чтобы быть ярким примером и точным критерием для оценки знаний ученики, студента или даже преподавателя.

При решении подобного рода заданий при помощи Интернет-тестирования были замечены неточности. Так, например, в заданиях с указанием двух и более ответов из-за использования различных источников литературы один из вариантов ответов может быть пропущен. Тем не менее система позволяет переходить к следующему заданию. В результате верный, но неполный ответ будет признан ощибочным.

Приведем пример такого задания.

Укажите два или более вариантов ответа.

Сепаратный мирный договор, подписанный 3 марта 1918 года в Брест-Литовске представителями …

1) Советской России.
2) Германии, Австро-Венгрии.
3) Турции и Болгарии.
4) Китая и Англии.

В ответе были указаны первый и второй варианты ответов. Однако, как известно, туда же входили Турция и Болгария. При ограничении по времени тестируемый может упустить из виду, что дал неполный ответ. Заметим, что система позволяет перейти к следующему заданию. Подобная ситуация может неоднократно возникнуть при

прохождении теста. Наибольшая опасность возникает тогда, когда данная форма используется для отдельно взятой дидактической единицы. Это резко снижает шансы успешного прохождения теста.

В ряде случаев среди ответов нет правильных. Это касается в первую очередь задач по математике, где допускаются вычислительные ошибки. Иногда среди вариантов есть верный, но он указан два раза и неизвестно, который следует выбрать. Сложности возникают и в том случае, если разработчики ориентировались на малоиспользуемые источники литературы. Расхождения при этом бываю не только в трактовках, но даже в терминологии. И, наконец, уровень теста. Он может быть завышен настолько, что становится непроходимым.

Решение проблемы существует. Оно должно идти по трем направлениям: организационному, методическому и техническому.

Организационные методы должны включать проверку заданий независимыми экспертами, методические - создание единых норм и правил разработки тестов, технические – наличие кнопок обратной связи.

Рассмотрим последний метод более подробно. Наиболее рациональным представляется создание в тестовых системах дополнительной кнопки ***Пожаловаться***, которая поможет исправить ошибки и неточности, замеченные на этапе прохождения теста (см. рис. 1).

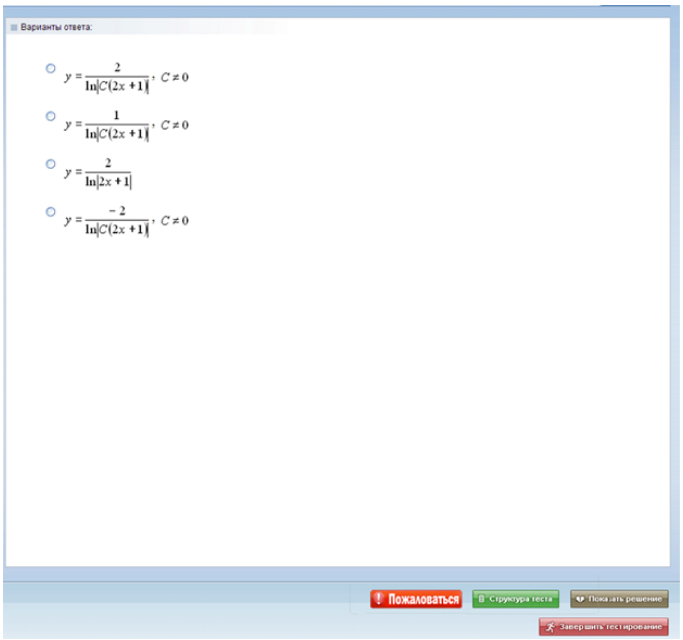


Рис. 1. Вид диалогового окна тестовой системы с кнопкой *Пожаловаться*

Щелчок по кнопке позволяет открыть диалоговое окно, в котором предлагается указать причину ошибки и дать подробное описание проблемы (см. рис. 2).

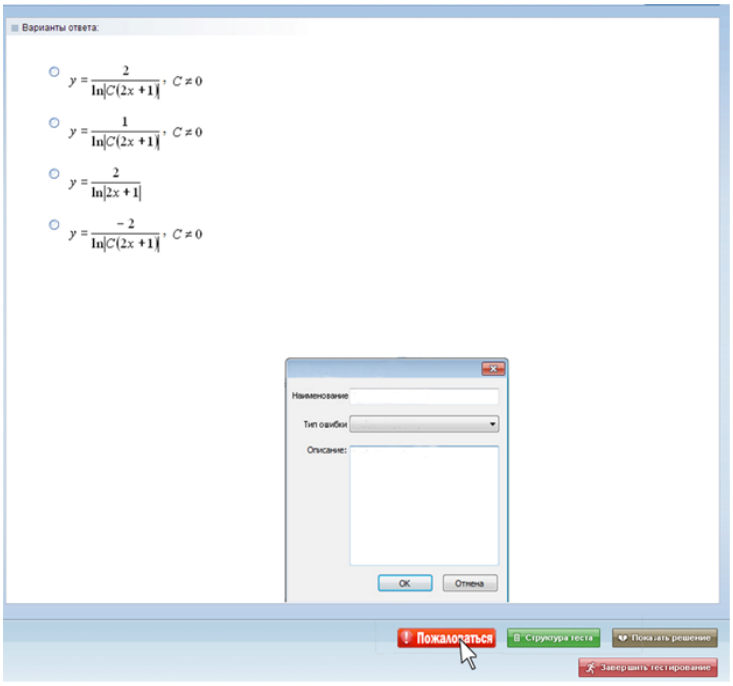


Рис. 2. Вид диалогового окна тестовой системы с окном обратной связи

Второй способ состоит в том, чтобы изменить тест таким образом, чтобы при выборе неполного набора ответов переход к следующему вопросу был бы невозможен. Для этого достаточно затемнить кнопку перехода к предыдущим и последующим вопросам теста. В данный момент тестирование продолжается независимо от количества выбранных ответов. Но имея указанную возможность, можно будет еще раз внимательно перечитать вопрос и найти ошибку, а значит и увеличить коэффициент точности ответов. Это программно достаточно легко реализуется. Необходимо учитывать и тот факт, что при повторной проверке ответов тестируемый может случайно изменить ответ. В этом случае также должна осуществляться блокировка кнопок.

Предложенный комплекс мер позволит повысить коэффициент точности проверки знаний и уменьшить психологическое давление, оказываемое подобными ситуациями. Личное участие преподавательского состава в разработке стандартов и проверке качества тестового материала уменьшит организационное противодействие системам Интернет-тестирования. И в заключении хочется добавить, что подобная проверка знаний еще находится на стадии внедрения, поэтому имеет довольно много ошибок и недочетов. Естественно, она будет развиваться, совершенствоваться, а значит и улучшать качественный показатель.

Литература.

1. В.С.Аванесов. История возникновения тестов. Научная статья 2000 года. Управление школой N12, март, 1999г. http://testolog.narod.ru

2. Акимова М.К. Психологическая диагностика**,** возникновение тестирования и тестовые методы. – СПб, Питер, 2005ю – 304 с. **3. Джон Макгрегор, Девид Сайкс.** Тестирование объектно-ориентированного программного обеспечения. Оригинальное название:«A Practical Guide to Testing Object-Oriented Software». - Киев: изд-во Юниор: ТИД "ДС", 2002. – 432 с. **4. Георгий Хубаев.** Информатика. Информационные системы. Информационные технологии. Тестирование. Подготовка к Интернет-экзамену. - СПб, издательский центр "МарТ" "Феникс", 2011. – 368 с.

Трофимов Я.В., Ореховский А.В. Способы оптимизации электронного тестирования

В статье рассмотрены проблемы внедрения системы Интернет-тестирования в высших учебных заведениях Российской Федерации. Предлагаемые пути решения затрагивают организационные, методические и технические стороны подготовки и проведения контроля знаний с использованием современных информационных технологий.

Ключевые слова: информационные технологии, Интернет-тестирование, оценка знаний.

Трофімов Я.В., Ореховський О.В. Способи оптимізації електронного тестування.

У статті розглянуті питання впровадження системи Інтернет-тестування у вищих навчальних закладах Російської Федерації. Шляхи розв’язання проблеми, що пропонуються авторами стосуються організаційних, методичних та технічних сторін підготовки та проведення контролю знань з використанням сучасних інформаційних технологій.

Ключові слова: інформаційні технології, Інтернет-тестування, оцінка знань.

Ja.V. Trofimov, A.V. Orekhovskji Optimization technique of e-assessment

The article is considered implementation of Internet – testing system in a higher educational establishment in Russian Federation. This way forward touches on organization, methodic and technological part of preparing and tests knowledge using modern IT tests.

Keywords: information technology, knowledge assessment.

УДК 004.421 : 004.65

Е.А. Фоменко

АНАЛИЗ ДИНАМИЧЕСКИХ СТРУКТУР ДАННЫХ НА ПРИМЕРЕ ДВУНАПРАВЛЕННЫХ СПИСКОВ ПРИ РЕШЕНИИ ЗАДАЧ

Цели статьи. Провести краткий аналитический обзор динамических структур, рассмотреть основные элементы библиотеки STL, их функциональные возможности и направления использования в решении задач, в частности кратко описать динамические структуры, такие как списки, двусвязные списки, деки и т.д. Рассмотреть реализацию возможностей двусвязных списков на примере решения задачи быстрой сортировки Хоара, оценить достоинства и недостатки двусвязных списков по сравнению с динамическими массивами.

Актуальность. Качество программы можно оценить несколькими критериями. Их достаточно много и они описывают разные подходы к оценке программного обеспечения. Среди распространенных характеристик: мощность алгоритма, временные характеристики, скорость выполнения и требования к памяти, ее объему и быстродействию. Причем последние две характеристики, которые взаимосвязаны, часто выходят на первый план при оценке качества программ и программных систем. В C++ существует ряд механизмов, которые позволяют улучшать и ускорять работу программы, за счет использования динамической памяти.

C++ обладает достаточно широкими возможностями для реализации работы с динамической памятью, в частности, он работает с динамическими переменными через указатели и операцию разыменования. В языке реализована возможность создавать как одномерные, так и многомерные (чаще всего двумерные) динамические массивы. Более того, механизм работы с массивами реализован в динамической памяти. Кроме этого, в C++ реализован набор динамических структур данных, таких как стеки, очереди, деки, однонаправленные и двунаправленные списки. В статье мы просто перечислим и очень кратко охарактеризуем каждую структуру.

Однонаправленный (односвязный) список – это структура данных, представляющая собой последовательность элементов, в каждом из которых хранится значение и указатель на следующий элемент списка. В последнем элементе указатель на следующий элемент равен NULL [2]. (рис.1).

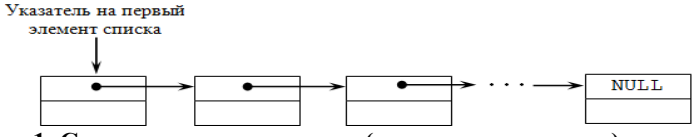


Рис.1. Структура односвязного (однонаправленного) списка

Двунаправленный (двусвязный) список – это структура данных, состоящая из последовательности элементов, каждый из которых содержит информационную часть и два указателя на соседние элементы. При этом два соседних элемента должны содержать взаимные ссылки друг на друга [2]. (рис.2).

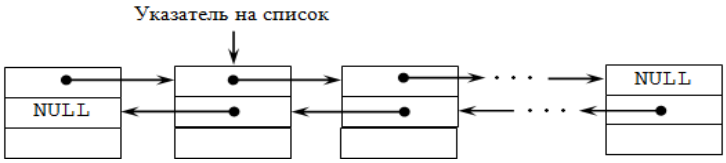


Рис.2. Структура двусвязного (двунаправленного) списка

Стек – это структура данных, в которой новый элемент всегда записывается в ее начало (вершину) и очередной читаемый элемент также всегда выбирается из ее начала. В стеках используется метод доступа к элементам LIFO (Last Input – First Output, "последним пришел – первым вышел") [2]. (рис.3).

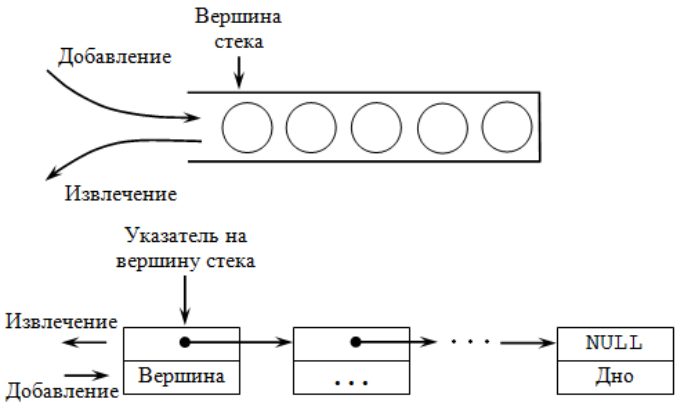


Рис.3. Структура стека

Очередь – это структура данных, представляющая собой последовательность элементов, образованная в порядке их поступления. Каждый новый элемент размещается в конце очереди; элемент, стоящий в начале очереди, выбирается из нее первым. В очереди используется принцип доступа к элементам FIFO (First Input – First Output, "первый пришёл – первый вышел") [2].(рис.4).

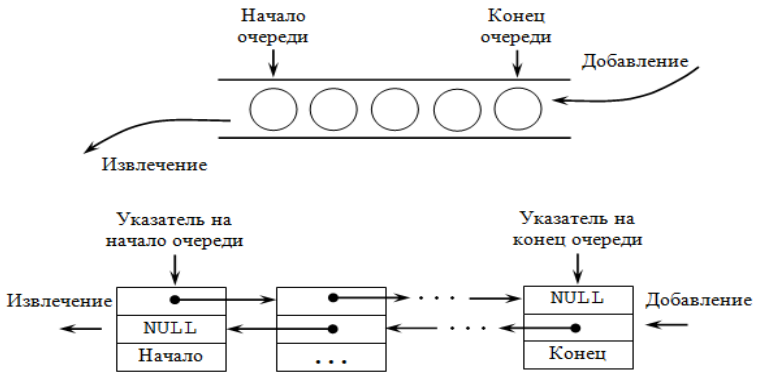


Рис.4. Структура очереди

Дек – это структура данных, представляющая собой последовательность элементов, в которой можно добавлять и удалять в произвольном порядке элементы с двух сторон. Первый и последний элементы дека соответствуют входу и выходу дека [2].(рис.5).

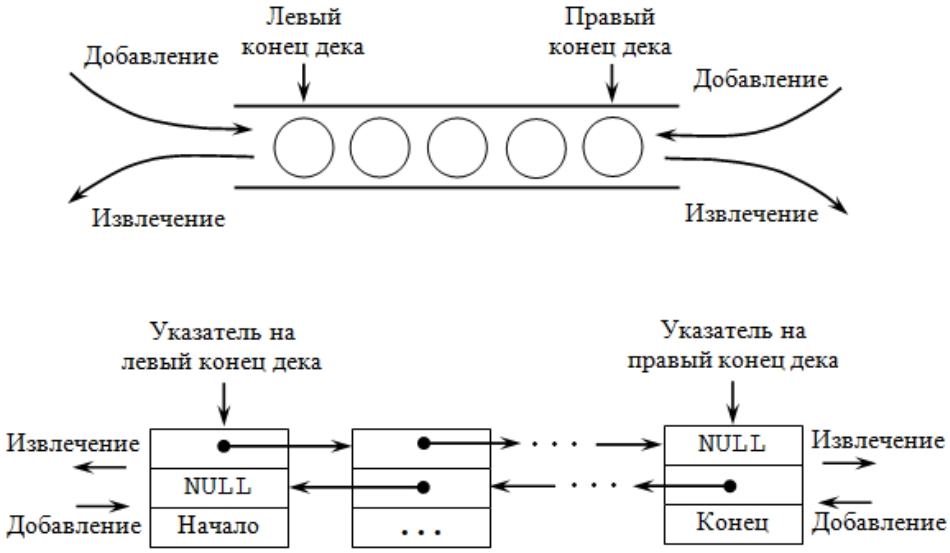


Рис.5. Структура дека

Многие из этих структур реализованы в библиотеке STL, которая была разработана Александром Степановым и Менг Ли. Стандартная библиотека шаблонов до включения в стандарт C++ была сторонней разработкой, вначале — фирмы HP, а затем SGI. Стандарт языка не называет её «STL», так как эта библиотека стала неотъемлемой частью языка. В библиотеке реализованы следующие структуры краткое описание их и некоторых их функций [1].

Цель включения библиотеки STL в стандарт языка – избавить пользователя от разработки рутинных общепринятых программ. Библиотека стандартных шаблонов содержит более сотни различных шаблонов и алгоритмов [3].

В библиотеке выделяют пять основных компонентов:

- Контейнер (англ. *container*) — хранение набора объектов в памяти.
- Итератор (англ. *iterator*) — обеспечение средств доступа к содержимому контейнера.
- Алгоритм (англ. *algorithm*) — определение вычислительной процедуры.
- Адаптер (англ. *adaptor*) — адаптация компонентов для обеспечения различного интерфейса.
- Функциональный объект (англ. *functor*) — сокрытие функции в объекте для использования другими компонентами.

Контейнеры библиотеки STL можно разделить на четыре категории: последовательные, ассоциативные, контейнеры-адаптеры и псевдо контейнеры [1].

Кратко рассмотрим некоторые контейнеры, которые наиболее часто используются при решении задач с динамической памятью.

vector	С-подобный динамический массив произвольного доступа с автоматическим изменением размера при добавлении/удалении элемента.
list	Двусвязный список, элементы которого хранятся в произвольных кусках памяти, в отличие от контейнера vector, где элементы хранятся в непрерывной области памяти.
deque	Дэк. Контейнер похож на vector, но с возможностью быстрой вставки и удаления элементов на обоих концах. Реализован в виде двусвязанного списка линейных массивов.
stack	Стек – контейнер, в котором добавление и удаление элементов осуществляется с одного конца.
queue	Очередь – контейнер, с одного конца которого можно добавлять элементы, а с другого – вынимать.
priority_queue	Очередь с приоритетом, организованная так, что самый большой элемент всегда стоит на первом месте.

А также set, multiset, map, multimap, bitset, basic string, valarray.

Данная библиотека настолько востребована и популярна среди разработчиков, что включена во все среды разработки и даже является частью стандартной библиотеки iostream.

Несмотря на удобство и полноту STL существует одна проблема при использовании стандартных структур, связанная с потерей гибкости программы. Разработчик не управляет распределением памяти и теряется

смысл организации динамической структуры, а с ним и эффективность программного кода.

Поэтому, в своей работе я хочу остановиться на возможности разработки динамических структур данных стандартными средствами C++ и их использования при решении типичных задач, а именно на создании и использовании двунаправленных списков.

В таком списке каждый элемент (кроме первого и последнего) связан с предыдущим и следующим за ним элементами. Каждый элемент двунаправленного списка имеет два поля с указателями: одно поле содержит ссылку на следующий элемент, другое поле – ссылку на предыдущий элемент и третье поле – информационное. Наличие ссылок на следующее звено и на предыдущее позволяет двигаться по списку от каждого звена в любом направлении: от звена к концу списка или от звена к началу списка, поэтому такой список называют двунаправленным.

При работе с двунаправленными списками обычно реализуют следующие стандартные операции:

- создание списка;
- печать (просмотр) списка;
- вставка элемента в список;
- удаление элемента из списка;
- поиск элемента в списке;
- проверка пустоты списка;
- удаление списка.

И некоторые другие, добавление группы элементов, удаление группы и т.д.

Для описания алгоритмов этих основных операций используется следующее объявление:

```
struct Double_List {//структура данных
                    int Data; //информационное поле
                    Double_List *Next, //адресное поле
                            *Prior; //адресное поле
                   };
. . . . . .
Double_List *Head; //указатель на первый элемент списка
. . . . . .
Double_List *Current;
//указатель на текущий элемент списка (при необходимости)
```

Для того, чтобы создать список, нужно создать сначала первый элемент списка, а затем при помощи функции добавить к нему остальные элементы. Добавление может выполняться как в начало, так и в конец списка. Реализуем рекурсивную функцию.

```
//создание двунаправленного списка (добавления в конец)
void Make_Double_List(int n,Double_List** Head,
      Double_List* Prior){
```

```
 if (n > 0) {
  (*Head) = new Double_List();
   //выделяем память под новый элемент
  cout << "Введите значение ";
  cin >> (*Head)->Data;
  //вводим значение информационного поля
  (*Head)->Prior = Prior;
  (*Head)->Next=NULL;//обнуление адресного поля
  Make_Double_List(n-1,&((*Head)->Next),(*Head));
 }
 else (*Head) = NULL;
}
```

По своим функциям и назначению списки похожи на динамические массивы, но при этом имеют существенные отличия по организации доступа к элементам и размещению в памяти. По сравнению с массивами списки имеют свои достоинства и недостатки.

Достоинства:

- лёгкость добавления и удаления элементов
- размер ограничен только объёмом памяти компьютера и разрядностью указателей
- динамическое добавление и удаление элементов

Недостатки:

- по его индексу (номеру) в списке
- на поля-указатели (указатели на следующий и предыдущий элемент) расходуется дополнительная память (в массивах, например, указатели не нужны)
- работа со списком медленнее, чем с массивами, так как к любому элементу списка можно обратиться, только пройдя все предшествующие ему элементы
- элементы списка могут быть расположены в памяти разреженно, что окажет негативный эффект на кэширование процессора
- над связными списками гораздо труднее (хотя и в принципе возможно) производить параллельные векторные операции, такие как вычисление суммы
- кэш-промахи при обходе списка.

Для предметного сравнения мы рассмотрели возможность реализации алгоритма быстрой сортировки Хоара, для этого мы разработали программу, который реализует эту сортировку через массив и двусвязный список.

Поставленная задача. Создать динамический список, элементами которого являются целые числа. Реализовать функцию QuickSort, которая упорядочивает список с помощью алгоритма быстрой сортировки.

Составим краткий словесный алгоритм.

1)Создаем динамическую структуру списка.

2)Объявляем необходимые указатели, которые понадобятся в функциях, такие как текущий, первый, последний и так далее.

3)Предопределяем функции.

4)В теле главной функции вызываем необходимые функции, при этом передаем туда значения нашего списка и другие необходимые значения.

5) Реализуем функции:

a) Функция, которая будет создавать двунаправленный список. Для того чтобы создать список, нужно создать сначала первый элемент списка, а затем при помощи функции добавить к нему остальные элементы. Добавление может выполняться как в начало, так и в конец списка. Реализуем рекурсивную функцию.

b) Функция, которая будет печатать созданный список, просматривать который можно в обоих направлениях.

c) Функция, которая будет отвечать за быструю сортировку списка. В этой функции мы вызываем ряд других функций, которые будут отвечать за поиск адреса медианы, поиск первого элемента и его адреса, последнего элемента и его адреса, а также функции движения влево, вправо и обмен элементов, которая будет происходить с помощью условий и циклов. Алгоритм быстрой сортировки будет описан ниже.

d) Функция, которая отвечает за движение вправо, идентична функции, которая отвечает влево, только за основу в первом случае берется адрес первого элемента и медианы, а во втором последнего элемента и медианы.

e) Функция, которая реализует поиск адреса медианы, работает почти аналогично функции поиска адреса последнего элемента. Происходит благодаря движению по списку до тех пор, пока в первом случае не станет равно медиане, во втором случае NULL.

f) Функция, которая отвечает за обмен элементов. Реализуется через третью переменную и переприсваивание.

Алгоритм быстрой сортировки(алгоритм Хоара):

Мы находим медиану списка, т.е. ее средний по счету элемент. Дальше мы двигаемся в двух направлениях, сравнивая значения элементов со значением медианы. Слева у нас остаются элементы меньше медианы, а справа больше ее. Если, элемент слева оказывается больше медианы, а элемент справа меньше, то мы меняем их местами. Далее, когда все меньшие элементы остались слева, а большие справа, мы еще раз находим медиану, но уже тех элементов слева и тех элементов справа и сортируем относительно нее. Для повышения скорости сортировки элементов списка медиана перемещается по списку случайным образом (через генератор случайных чисел), избегая краев списка.

Вот список необходимых функций для решения данной задачи, с кратким описанием основных задач, выполняемых данной функцией.

Также перечислим указатели типа структуры, которые были использованы для решения задачи.

```
//объявляем указатели типа струтуры
Double_List *Head;
Double_List *Mediana;
Double_List *Last;
Double_List *First;
Double_List *nFirst;
Double_List *nLast;
Double_List *Swapper;
//объявление функций
void Make_List(int n, Double_List** Head, Double_List *Prev);//функция, которая формирует сам список (выделяет память и записывает в адреса элементы)
void Print_List (Double_List* Head);//функция, которая выводит заполненный список
void Quick_Sort(Double_List* Head, int *n, int m);//функция, которая вызывает другие функции, отвечающие за сортировку списка
void Count_List(Double_List* Head, int *n);//функция, которая считает количество элементов списка
int Left_Go(Double_List* First, int median);//функция, которая отвечает за продвижение влево по списку
int Right_Go(Double_List* Last, int median);//функция, которая отвечает за движение вправо по списку
void Addres_Mediana(Double_List* Head, int number);//функция, которая находит медиану и ее адрес
void Addres_Last(Double_List* Head);//функция, которая находит последний элемент и его адрес
void Swap_Element(Double_List* Head, Double_List *nFirst, Double_List *nLast);//функция, которая отвечает за обмен элементов
```

Далее покажем фрагмент самой основной функции QuickSort, которая отвечает за вызов ряда других функций, реализующих отдельные части алгоритма быстрой сортировки. Эта функция является рекурсивной и также отвечает за количество итераций для более эффективной сортировки.

```
void Quick_Sort(Double_List* Head, int *n, int m){
 int *k=new int[10];//указатель целочисленного типа
 int iterac=0;//объявляем переменную, отвечающую за кол-во итераций
 int median=int((*n)/2);//получаем значение медианы
 do{
        Addres_Mediana(Head, median);//обращаемся к функции поиска адреса медианы и передаем туда саму медиану
        First=Head;
```

```
Left_Go(First, median);//вызываем функцию движения влево по списку и передаем туда значение медианы
Addres_Last(Head);//вызываем функцию поиска адреса последнего элемента
Right_Go(Last, median);//вызываем функцию движения вправо
//условия при которых мы будем перемешивать элементы
if(nFirst->Data>=Mediana->Data){//если значения слева больше или равны медиане
Swap_Element(Head, nFirst, Mediana);//передаем туда адрес элемента, который больше медианы и саму медиану
}
if(nLast->Data<=Mediana->Data){//если значения справа меньше или равны медиане
Swap_Element(Head, Mediana, nLast);//передаем туда медиану и адрес элемента, который меньше медианы
}
Print_List(Head);//выводим список
median=rand()%((*n)); if (median<=1) median+=2;
if (median==((*n)-1)) median--;
iterac++;//увеличиваем итерацию на 1
}
while((k>0)&&(iterac<(int)(*n)/2));//выполнили мы все это пока новая медиана больше нуля и итераци меньше 40
m--;//уменьшаем число повторений на 1
if(m!=0) Quick_Sort(Head, n, m--);//если число повторений не ноль, то вызываем функцию сортировки, куда передаем
//число повторений меньше на 1
}
```

Для определения преимущества динамических двусвязных списков по отношению к динамическим массивам был реализован алгоритм быстрой сортировки с использованием динамических массивов.

В результате проведенных исследований мы подтвердили предположение, что сортировка с использованием массивов эффективней с простыми элементами массива, такими как целые числа, символы, если же нам надо отсортировать более сложные структуры, для этого эффективней использовать двусвязный список. То есть практически всегда. Причина этого - мы не перемещаем элементы, а работаем с простыми числами в виде указателей.

Литература

1 Стандартная библиотека шаблонов**. – [Электронный ресурс]. – Режим доступа к статье:** *http://ru.wikipedia.org/wiki***.** 2. Динамические структуры данных **– [Электронный ресурс]. – Режим доступа к статье:** *http://www.intuit.ru/studies/higher_education/3407/courses/504/lecture/6555?page=4***.** 3. Стандартная библиотека шаблонов (stl) **– [Электронный ресурс]. – Режим доступа к статье:** *http://rpp.nashaucheba.ru/docs/index-48081.html***.** 4. Седжвик, Р. Фундаментальные алгоритмы на C++/ Р. Седжвик. - М.: DiaSoft, 2001. - 680 с. 5. Скиена, С. С. Алгоритмы. Руководство по разработке/С.С. Скиена. – С.-П.: БХВ-Петербург, 2011. – 713 с. 6. Гагарина, Л. Г. Алгоритмы и структуры данных / Л. Г. Гагарина, В. Д. Колдаев. - М.: Финансы и статистика, 2009. - 303 с. 7. Макконелл, Дж. Основы современных алгоритмов/Дж. Макконелл. – М.: Техносфера, 2004. – 368 с. 8. Структуры и алгоритмы обработки данных: учеб. пособие / В. А. Матьяш [и др.]; - Апатиты: КФ ПетрГУ, 2000. - 80 с.

Фоменко Е. А. Анализ динамических структур данных на примере двунаправленных списков при решении задач

В статье рассматриваются вопросы организации динамических структур данных, дан краткий обзор библиотеки STL, обозначены ее достоинства и недостатки. Проанализированы и обозначены достоинства и недостатки динамических структур данных в сравнении с динамическими массивами. На практическом примере реализации алгоритма Хоара показано преимущество динамических двусвязных списков по сравнению с массивами.

Ключевые слова: Библиотека STL, динамические структуры, двусвязные списки, реализация.

Фоменко Є.А. Аналіз динамічних структур даних на прикладі двоспрямованних списків при вирішенні завдань

У статті розглядаються питання організації динамічних структур даних, дано короткий огляд бібліотеки STL, позначені її переваги і недоліки. Проаналізовано і позначені переваги і недоліки динамічних структур даних у порівнянні з динамічними масивами. На практичному прикладі реалізації алгоритму Хоара показано перевагу динамічних двозв'язних списків порівняно з масивами.

Ключові слова: Бібліотека STL, динамічні структури, двозв'язні списки, реалізація.

Fomenko E.A. Analysis of dynamic data structures on the example of bidirectional lists in solving problems

The questions of organization of dynamic data structures, a brief overview of the library STL, marked its advantages and disadvantages. Analyzed and identified the advantages and disadvantages of dynamic data structures in comparison with dynamic arrays. On a practical example of the

implementation of the algorithm is shown Hoare advantage of dynamic linked list over arrays.

Keywords: Library STL, dynamic structures of doubly linked list implementation.

УДК 004.738.5.057.2(076)

О. В. Шевченко, Н. О. Цодікова

WEB 2.0 В ІНФОРМАЦІЙНОМУ ОБСЛУГОВУВАННІ

Ми живемо в інформаційній епосі. Ми маємо ідею про те, що сучасне життя буде характеризуватися широкими можливостями для людей вільно передавати, приймати інформацію і мати миттєвий доступ до знань, що було б важко або неможливо в попередні епохи розвитку людства.

Існує безліч визначень терміну інформація. Як інформаційних робітників, нас цікавить інформація як документовані або публічно оголошені відомості про події та явища, що відбуваються у суспільстві, державі та навколишньому природному середовищі.

В процесі спілкування – в повсякденному житті та в умовах виготовлення матеріальних та духовних благ – люди кожного дня займаються інформаційною діяльністю – сприймають, переробляють та видають інформацію, а кількість робітників розумової праці збільшується. Тенденція інформатизації людства привела до виникнення спеціальної сфери людської діяльності – сфери інформаційного обслуговування (інформаційного сервісу).

Тема статті є **актуальною**, адже інформація формує світогляд людини, допомагає їй опанувати різні науки, отримати важливі, інколи унікальні знання. В даний час проходить черговий етап розвитку цивілізації - перехід до „інформаційного суспільства", тож правильно організована інформаційна діяльність людей є важливим фактором сучасного життя.

Мета роботи полягає в аналізі поняття «інформаційне обслуговування», як професійної діяльності, функціонування якої є неможливим без активної в ній роль споживача та сучасних інформаційних технологій.

Інформаційне обслуговування (ІО) – діяльність, що забезпечує задоволення інформаційних потреб людей [2, с. 39].

Формування інформаційного середовища, яке є не чимось зовнішнім по відношенню до користувача, а сферою, що включає та інформує самого користувача – це основне завдання ІО.

ІО можна представити в вигляді задачі: умова – запит користувача, ресурси, рішення – виконання технологічного процесу,

відповідь – інформаційна послуга, перевірка – ступінь задоволення послугою.

Багаточисельні види інформаційних робіт, наприклад формування інформаційних масивів, створення інформаційно-пошукови систем, підготовка різноманітних бібліографічних покажчиків, фактографічних довідок, написання рефератів та оглядів, редагування, рецензування й переклад текстів та багато іншого, що традиційно розглядається як різновиди інформаційної діяльності, в дійсності являють собою різноманітні операції сфери інформаційного обслуговування [2, с. 40].

У процесі розвитку системи інформаційних комунікацій сформувалися три види інформаційного обслуговування – документальне, фактографічне і концептографічне, причому кожному з зазначених видів відповідає своя інформаційна система.

Центральне місце в документальній системі займають бібліотечно-бібліографічні та видавничі установи, які протягом вже багатьох століть забезпечують інформаційне обслуговування суспільства.

Сутність документального обслуговування полягає в тому, що інформаційні запити споживачів задовольняються шляхом надання їм первинних документів, необхідні відомості з яких вони вилучають самостійно. Зазвичай грамотне документальне обслуговування здійснюється в два етапи:

1) бібліографічне обслуговування – споживачеві надається деяка сукупність релевантних його запитом вторинних документів, що є результатом аналітико-синтетичної переробки одного або декількох первинних документів, наприклад, бібліографічний опис, анотація, реферат, рецензія, огляд, факторгафічна довідка (таблиця);

2) бібліотечне обслуговування - після відбору споживачем із сукупності певного числа вже пертінентних документів, йому надаються самі первинні документи, що відображають результати якої діяльності, наприклад, науковий звіт, стаття, опис до авторського свідоцтва, монографія, підручник, посібник, тощо. Таким чином, потреба в інформації при документальному обслуговуванні задовольняється через первинний документ [2, с. 41-42].

Фактографічне обслуговування передбачає задоволення інформаційних потреб безпосередньо, тобто шляхом подання споживачам самих відомостей (окремих даних, фактів, концепцій). Ці відомості, також релевантні запитам споживачів, попередньо видобуваються інформаційними працівниками з первинних документів і після певної їх обробки надаються замовникові.

Очевидно, однак, що повноцінне забезпечення фахівців фактографічною інформацією можливо сьогодні лише на основі використання спеціалізованих служб фактографічної інформації, які забезпечені обчислювальною технікою, тобто автоматизованими

фактографічними системами. У якості подібних систем в останні роки все частіше виступають так звані інформаційні банки фактичних матеріалів (часто звані просто ,,банки даних").

Як правило під поняттям ,,фактографічна інформація" розуміють інформацію, що характеризує конкретний факт, фактичну подію або їх сукупність [3]. Існує інша точка зору: ,,розуміти під фактографічною інформацією відомості не тільки фактичного характеру, але і теоретичного, передбаченого, оціночного тощо, тобто включати і факти, і концепції, і все те, що може бути об'єктом вилучення з тексту, описування на визначеній інформаційній мові, зберігання та пошуку в тій чи іншій інформаційній системі" [2, с. 43].

У наукових джерелах можна побачити, як правило, виділення двох видів фактографічних систем, пріоритетність яких визначається виходячи з функціональних особливостей і типу розв'язуваних завдань. Це або інформаційно-пошукові, або інформаційно-вирішальні фактографічні системи.

Слід зазначити що, якщо у випадку документального та фактографічного обслуговування споживачу інформації надаються документи або відомості, витягнуті з інформаційного потоку, так би мовити в ,,натуральному" вигляді (витягується те, що є в цьому потоці), то при концептографічному обслуговуванні все це (документи і відомості) піддається інтерпретації, оцінці, узагальненню з боку інформаційного працівника [2, с. 42].

Результатом такої інтерпретації є ,,так звана ситуативна інформація, що містить у собі оцінку розглянутих відомостей, тенденції та перспективи розвитку окремих наукових напрямів, рекомендації та інше. З цієї причини під концептографічним обслуговуванням можна також розуміти формулювання і доведення до споживачів ситуативної інформації, в явному вигляді не що міститься в аналізованих джерелах, а отриманої в результаті інформаційно-логічного та концептографічного аналізу деякої сукупності повідомлень" [2, с. 43].

Особливість концептографічного обслуговування полягає в тому, що воно, з одного боку, являє собою різновид інформаційного обслуговування поряд з документальним та фактографічних, а з іншого, – різновид наукової діяльності, що відповідає пояснювальному (концептуальному) етапу розвитку тієї чи іншої галузі знань.

Як правило, кожен з різновидів інформаційного обслуговування зводиться до створення вторинних документів і доведення їх до споживачів доступними засобами.

Таким чином проаналізовано видову сферу інформаційного обслуговування.

З розвитком новітніх інформаційних технологій, ІО вийшло на новий рівень.

Останнім часом спостерігається швидке зростання числа мереж, підключених до спільноти комп'ютерних мереж Internet. Популярність

Internet визначається наявністю простого у використанні програмного забезпечення, відпрацьованої технології міжмережевого обміну і великої кількості інформаційних матеріалів, розміщених у мережі за 25 років її існування.

Інтернет – глобальна обчислювальна мережа, яка об'єднує безліч регіональних, відомчих, приватних та інших інформаційних мереж каналами зв'язку і єдиними для всіх її учасників правилами організації [3].

Однією з найпоширеніших сучасних технологій, що допомагає організовувати ІО є Web 2.0.

Тім О'Рейллі, один із головних ідеологів Web 2.0, дає цьому терміну таке визначення: ,,Web 2.0 – це методика проектування систем, які шляхом мережевого впливу стає тим краще, чим більше людей ним користуються".

Для визначення суті підходить визначення Web 2.0 як комплексного підходу до організації, реалізації та підтримки Web-ресурсів. Пропонуємо розглянути найбільш відомі «прояви» Web 2.0, з якими, так чи інакше, иоже бути пов'язане ІО.

Вікіпедія. Сьогодні це ціла плеяда інтернет-сервісів, які позиціонують як довідники та енциклопедії. Вікіпедія являє собою базу довідкової інформації з наданням практично кожному користувачеві можливості редагувати дані.

Блоги (інтерактивні мережеві щоденники) представляють собою один з найяскравіших прикладів використання принципів Web 2.0. Значна частина Web-контенту створюється користувачами, а не власниками ресурсу. Для цього активно використовують технології RSS та FOAF, характерні для Web 2.0. Так само використовуються теги (мітки, tags) для тематичного структурування контенту.

Технологія FOAF (Friend Of A Friend) дає користувачеві можливість підписатися на новини і матеріали тих користувачів, які перебувають у так званому «списку друзів». Цим самим заохочується спілкування користувачів Мережі. Технологія FOAF є однією з найважливіших складових соціальних інтернет-мереж. Вона формує більш ,,тепле" інформаційне поле, що задовольняє потребу користувачів налагуджувати зв'язки між собою.

RSS (Really Simple Syndication або, дослівно, «дійсно просте об'єднання (інформації)») - це проста і ефективна технологія експорту гіпертексту, використовувана для створення новинних стрічок. RSS, як і інші технології Web 2.0, заснований на мові XML (eXtended Markup Language - англ., Розширена мова розмітки). Ця технологія задовольняє потреби користусвачів тим, що надає швидкий доступ до потрібної інформації.

AJAX (Asynchronous Javascript And XML - англ., Асинхронні Javascript і XML) - технологія комбінованого використання мов Javascript і XML для створення динамічного вмісту web-ресурсів. Однією з

важливих особливостей технології AJAX є те, що навантаження з обробки динамічних даних лягає на браузер користувача. При цьому обмін даними «сервер - користувач» зводиться до мінімуму, знижуючи витрати на трафік. Подібний підхід значно спрощує використання динамічних ресурсів, тому що користувачеві не потрібно перезавантажувати сторінки, щоб побачити зміни. AJAX допомагає швидше та зручніше сприймати та обробляти потрібну інформацію.

Web-додатки – частина концепції Web 2.0. Все більше додатків отримують свої Web-аналоги, позбавляючи користувачів від необхідності встановлювати ПЗ на локальному комп'ютері. Як правило, користування web-сервісом обходиться дешевше покупки аналогічних «не-web» програм. Такі сервіси допомагають задовольняти разові специфічні потреби користувачів.

Запити на природній мові – це технологія майбутнього. Вона полягає в тому, що комп'ютер приймає запит людською мовою, обробляє його, знаходить значення сказаних слів та видає результат. Це дає змогу користувачеві без всіляких зусиль отримувати потрібну інформацію.

Таким, чином, інформаційне обслуговування є невід'ємною частиною діяльності сучасної людини. І хоча багато з нас, користуючись послугами бібліотек, використовуючи Інтернет ресурси та інформаційні технології, не здогадуються про те, наскільки далеко просунувся інформаційний сервіс, ІО продовжує захоплювати все більшу частину нашого життя.

Література

1. **Закон „Про інформацію".** 2. **Блюменау Д.И.** Информация и информационный сервис. - Л: Наука, 1989.-192 с.- (Серия "Наука и технический прогресс"). 3. **Словарь терминов по информатике.** – М.: Наука, 1971.-359 с. 4. Что такое Web 2.0? **[Електронний ресурс]** // режим доступу: http://shkolazhizni.ru/archive/0/n-3081/

Шевченко А. В., Цодікова Н. О. Web 2.0 в інформаційному обслуговуванні

У статті висвітлюється поняття „інформаційне обслуговування", як професійної діяльності, яка формує інформаційне середовище. На сучасному етапі Web 2.0 є найпоширенішою технологією, що допомагає організовувати інформаційне облуговування.

Ключові слова: інформація, інформаційне обслуговування, інтернет, Web 2.0

Шевченко А. В., Цодикова Н. А. Web 2.0 в информационном обслуживании

В статье рассматривается понятие „информационное обслуживание", как профессиональной деятельности, которая формирует информационную среду. На современном этапе Web 2.0 является самой

распространенной технологией, которая помогает организовывать информационное облуживания.

Ключевые слова: информация, информационное обслуживание, интернет, Web 2.0.

Tcodicova N. A., Shevchenko A. V., Web 2.0 in information service

The article covers the concept ,,information service" as a professional practice, which forms the information area. Nowadays Web 2.0 is the most widespread technology that helps organize information service.

Keywords: information, information service, internet, Web 2.0.

ТЕХНІКА

УДК 004.49

А. Т. Агаджанов

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УЧЕТА РАБОТЫ ТАКСОПАРКА

Постепенно с развитием программного обеспечения ЭВМ появились идеи создания управляющих систем, которые позволяли бы накапливать, хранить и обновлять взаимосвязанные данные по целому комплексу решаемых задач. Эти идеи нашли свое воплощение в системах управления базами данных (СУБД). СУБД взаимодействуют не с локальными, а взаимосвязанными по информации массивами, называемыми базами данных. С появлением персональных компьютеров СУБД становятся наиболее популярным средством обработки табличной информации. Они являются инструментальным средством проектирования банков данных при обработке больших объемов информации.

База данных – совокупность специальным образом организованных данных, хранимых в памяти вычислительной системы и отображающих состояние объектов и их взаимосвязь. Информацию, хранящуюся в БД можно широко использовать в различных приложениях, причем способы использования данных можно легко и быстро изменять. Также обеспечивается возможность запрашивать, находить и изменять информацию в БД.

Грамотно составленная система очень сильно экономит время при обращении к необходимой информации. При правильном составлении и внесении информации в базу скорость поиска необходимой информации сводится до минимума. Создание такой базы данных поможет с легкостью работать с информацией, хранящейся в ней.

Разрабатываемую базу данных можно с легкостью использовать в работе таксопарка. Она является удобной и понятной для любого типа пользователей. База позволяет добавлять новых сотрудников, клиентов, вызовы, а также вести запросы и отчеты по ним.

Очень часто люди едут в места, отдаленные от дома. Но не всегда под рукой есть автомобиль. Поэтому они обращаются за услугами такси. Диспетчер таксопарка принимает вызов и вводит информацию в базу данных (телефон клиента, время приезда за ним, адрес и пункт конечного назначения). Далее диспетчер сообщает координаты клиента водителю, и он отправляется на вызов.

Те клиенты, которые часто пользуются такими услугами (например, если они совершили более десяти поездок в течении месяца),

становятся VIP клиентами. Они заключают договор, получают VIP-карточку и 10% скидку на проезд.

Информационная система «Таксопарк» предназначена для упрощения регулированием системы и для автоматизации её функций. База данных предназначена для диспетчеров и директора таксопарка.

Система содержит следующую информацию.

Учет сотрудников: код, ФИО, дата рождения, адрес проживания, номер телефона, должность, категория водительских прав, дата выдачи прав, семейное положение, сведения о паспорте (серия, номер, дата выдачи, кем выдан, код подразделения).

Учет принятых вызовов: код, код сотрудника (водителя), когда приехать за клиентом, телефон, адрес, пункт конечного назначения.

Учет VIP клиентов: код, ФИО, номер карточки, номер договора, скидка, телефон, адрес проживания, код сотрудника (водителя).

Учет транспортных средств сотрудников: номер, код сотрудника (водителя), марка, тип, категория, год выпуска, модель двигателя, номер двигателя, цвет, мощность двигателя, серия паспорта, разрешенная максимальная масса, масса без нагрузки.

Расценки: пункт назначения, километраж, цена.

Система предусматривает запросы:

1. Вывод информации, касающейся сотрудников.
2. Поиск информации об отдельном сотруднике по его фамилии.
3. Вывод сообщения о повторяющихся данных о VIP клиенте по его коду, фамилии, отчеству и номеру VIP карты.
4. Вывод информации о ценах проезда.

Описание выходных документов и сообщений. Выходными документами являются отчеты:

- список VIP Клиентов (Код клиента, Фамилия, Номер карты, Телефон, Адрес проживания);
- состав сотрудников (Должность, ФИО);
- цена проезда (Пункт назначения, Сколько до него км, Цена);
- визуальное представление транспортных средств (Номер ТС, Код сотрудника, Изображение ТС);
- пункты, расстояние между которыми менее 30 км (Пункт назначения, Сколько до него км, Цена);
- состав транспортных средств (Код сотрудника, Номер ТС, Марка ТС, Год выпуска, Цвет);
- сведения о принятых вызовах (Код сотрудника, Код вызова, Время приезда за клиентом, Телефон, Адрес).

Список ограничений. На данные, хранящиеся в базе данных, накладываются следующие ограничения:

- один водитель может выполнять только один вызов одновременно;
- номер ТС, номер двигателя, серия паспорта являются уникальными и соответствуют конкретному ТС;

- код VIP клиента, номер его карточки, номер договора являются уникальными и соответствуют конкретному VIP клиенту;

- код вызова является уникальным;

- код сотрудника, серия паспорта, номер паспорта являются уникальными и соответствуют конкретному сотруднику.

Что касается описания документооборота таксопарка, то он состоит из четырех основных технологических этапов: поступления информации; обработки информации; выдачи информации; хранения информации.

Каждый из названных этапов предусматривает выполнение определенных функций и операций, в совокупности составляющих процесс делопроизводства: прием входящей информации, первоначальную обработку, сортировку, регистрацию, доклад руководителям, передачу на исполнение, контроль над исполнением и т.д.; то же по исходящей или внутренней информации.

Основным фактором, определяющим единство форм и методов учета, контроля над исполнением, оперативного и архивного хранения документов, а также информационного обеспечения аппарата управления, является правильная систематизация документальных материалов, обращающихся в делопроизводстве предприятия.

Наиболее рациональной для таксопарка необходимо считать тематическую (предметную) систематизацию при карточной форме регистрации документов. Но при замене обычной карточки на перфокарту с краевой перфорацией поиск необходимого документа в картотеке не зависит от принятой формы систематизации и может осуществляться как по отдельным, так и по многоаспектным признакам. Незначительная стоимость технических средств, простота в работе с перфокартотекой, значительное снижение трудозатрат на поиск информации, облегчение контроля за исполнением и возможность анализа исполнительской дисциплины обеспечивают эффективность применения такой информационной поисковой системы (ИПС) .

Регистрационная перфокарта содержит всю необходимую информацию о документе: данные об авторе, когда создан документ, когда он поступил или отправлен, входящий или исходящий номер, краткое содержание документа, данные о его прохождении и исполнении и др. После оформления регистрационной карточки на ней производится кодирование (мелкие, глубокие вырезы или их комбинация) соответствующих признаков.

Кодирование, поиск и анализ информации осуществляются по разработанным для этой цели поисковым языкам, классификаторам и с помощью вспомогательного инструмента.

Правильная систематизация документальных материалов во многом зависит от квалификации секретаря, его умения быстро и четко ориентироваться во всем многообразии вопросов, возникающих в процессе управления предприятием. Большую помощь в решении этой

задачи может оказать специальный перечень вопросов и участков работы.

Таким образом, база данных – это совокупность структурированных и взаимосвязанных данных и методов, обеспечивающих добавление, выборку и отображение данных. Microsoft Access позволяет управлять всеми сведениями из одного файла базы данных. В рамках этого файла используются следующие объекты:

– таблицы для сохранения данных;

– запросы для поиска и извлечения только требуемых данных;

– формы для просмотра, добавления и изменения данных в таблицах;

– отчеты для анализа и печати данных в определенном формате.

Удачная разработка базы данных обеспечивает простоту ее поддержания.

Microsoft Access, обладая всеми чертами классической СУБД, предоставляет и дополнительные возможности. Access – это не только мощная, гибкая и простая в использовании СУБД, но и система для разработки работающих с базами данных приложений. С помощью Access можно создать приложение, работающее в среде Windows и полностью соответствующее потребностям по управлению данными. Используя запросы, есть возможность выбирать и обрабатывать хранящуюся в таблицах информацию. Можно создавать формы для ввода, просмотра и обновления данных, а также использовать Access для создания как простых, так и сложных отчетов. Формы и отчеты «наследуют» свойства базовой таблицы или запроса, так что в большинстве случаев необходимо указать форматы, условия на значения и некоторые другие характеристики данных только один раз. К числу наиболее мощных средств Access относятся средства разработки объектов – Мастера, которые можно использовать для создания таблиц, запросов различных типов форм и отчетов, просто выбрав с помощью мыши нужные опции.

С помощью Access также легко импортировать данные из текстовых файлов, документов текстовых процессоров и электронных таблиц (и экспортировать данные в них).

Таким образом, разработанная в данном проекте база данных для решения задач документооборота в таксопарке, позволяет автоматизировать весь документооборот, что позволяет разгрузить отчетный документооборот и сократить ошибки в документации. При появлении новых производственных задач разработчик может в кратчайшие сроки реализовать их в базе данных, путем добавления строк, столбцов и целых таблиц.

Литература

1. Михеева В. Д. Microsoft Access 2002 / В. Д. Михеева, И. А. Харитонова – СПб. : БХВ-Петербург, 2003. – 1040 с. **2. Диго С. М.**

Проектирование и использование баз данных; Учебник / С. М. Диго – М. : Финансы и статистика, 1995. – 208 с. **3. Рычка И. А.** Методические указания для курсового проектирования по базам данных для студентов специальности 080801 «Прикладная информатика (в экономике)» / И. А. Рычка. – Петропавловск-Камчасткий, КамчатГТУ, 2008. – 38 с. **4. Феддема Э.** Эффективная работа: Microsoft Access 2002. / Э. Феддема – СПб. : Питер, 2003. – 944 с.

Агаджанов А. Т. Автоматизированная система учета работы таксопарка

В статье рассматриваются вопросы организации автоматизированной системы учета работы таксопарка с помощью систем управления базами данных. Проанализирована предметная область таксопарка, выявлены группы параметров для каждого из объектов, информация о которых будет храниться в проектируемой базе данных. Проведен анализ требований к базе данных.

Ключевые слова: автоматизированная система, система управления базами данных, предметная область таксопарка.

Агаджанов А. Т. Автоматизована система обліку роботи таксопарку

У статті розглядаються питання організації автоматизованої системи обліку роботи таксопарку за допомогою систем управління базами даних. Проаналізовано предметна область таксопарку, виявлені групи параметрів для кожного з об'єктів, інформація про які буде зберігатися в проектованій базі даних. Проведено аналіз вимог до бази даних.

Ключові слова: автоматизована система, система управління базами даних, предметна область таксопарку.

Agadzhanov A. T. Automated accounting of taxis

The article deals with the organization of an automated accounting system of taxis using database management systems. Analyzed by subject area taxis, identified a group of parameters for each of the objects, details of which will be stored in the projected database. The analysis of the database requirements.

Keywords: automated system, database management system, subject area taxis.

УДК 004.49

Е. А. Дудник

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УЧЕТА РАБОТЫ ПОЧТОВЫХ ОТДЕЛЕНИЙ

Актуальность этой статьи заключается в том, что в деловой и личной сфере часто приходится работать с данными из разных источников, каждый из которых связан с определенным видом деятельности. Для координации всех этих данных необходимы определенные знания и организационные навыки.

В общем смысле термин база данных – это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области или разделе предметной области.

Увеличение объема и структурной сложности хранимых данных, расширение круга пользователей информационных систем выдвинуло требование создания удобных средств интеграции хранимых данных и управления ими.

Использование баз данных и информационных систем становится неотъемлемой составляющей деловой деятельности современного человека и функционирования преуспевающих организаций. В связи с этим большую актуальность приобретает освоение принципов построения и эффективного применения соответствующих технологий и программных продуктов: систем управления базами данных, CASE-систем автоматизации проектирования, средств администрирования и защиты баз данных и других.

От правильного набора инструментальных средств создания информационных систем, определения подходящей модели данных, обоснования рациональной схемы построения баз данных, организация запросов к хранимым данным и ряда других моментов во многом зависят эффективность функционирования разрабатываемых систем. Все это требует осознанного применения теоретических положений и инструментальных средств разработки баз данных и информационных систем.

Задание моей работы заключается в том, чтобы спроектировать и разработать базу данных «Почтовые отделения». Разрабатываемая база данных может быть использована для создания единой информационной системы почтовых отделений. В ней можно будет отслеживать пересылку писем, бандеролей, закупку печатных изданий у типографий. База данных будет ограничена закупкой печатных изданий у различных типографий. В базе данных будет отслеживаться информация об известных печатных изданиях, типографиях и почтовых отделениях. Создаваемая база данных облегчит поиск почтовым отделениям

необходимых изданий, а также можно будет выбрать типографию, закупка у которой данного печатного издания будет наиболее выгодной.

Почтовые отделения характеризуется большими объёмами информации. Основная цель почтовых отделений основывается на том, чтобы доставка газет, журналов, писем, телеграмм, бандеролей была во время. Для решения этой задачи необходима единая информационная система. В этой системе можно будет следить за получением и за доставкой всего товара (писем, телеграмм, и.т.д.), а также доставка газет и журналов по подписке на почтовых отделениях населению и организациям. Качество и своевременность доставки писем, телеграмм и бандеролей будет зависеть от того, насколько продуманы каналы обмена ими между почтовыми отделениями и непосредственная их доставка получателям. А доставка газет и журналов жителям и организациям, которые подписались на их получение, зависит от своевременного заказа необходимых газет и журналов на типографиях, оплаты и доставки их на почтовые отделения, а также от своевременной их доставки получателям.

Цель работы – разработать базу данных для почтовых служащих, работающих в единой системе почтовых отделений. Она должна отображать весь перечень газет и журналов, которые имеются в наличии, а также производство их на типографиях и заказы почтовых отделений газет и журналов у этих типографий с последующей оплатой и получением необходимых партий.

Анализ требований к БД. На информацию, хранящуюся в базе данных, накладываются следующие ограничения:

– по изданиям: один и тот же человек не может быть редактором нескольких изданий одновременно; у нескольких изданий не может быть одинакового кода издания;

– по типографиям: каждая типография должна иметь свой уникальный номер; несколько типографий не могут располагаться по одному почтовому адресу; один и тот же человек не может быть директором нескольких типографий одновременно;

– по почтовым отделениям: каждое почтовое отделение должно иметь свой уникальный номер; несколько почтовых отделений не могут располагать по одному почтовому адресу; один и тот же человек не может быть директором нескольких почтовых отделений одновременно.

С базой данных должны работать следующие группы пользователей: редакторы изданий; служащие и начальники типографий; служащие и начальники почтовых отделений.

Редакторам изданий может потребоваться следующая информация: об изданиях определенного типа с сортировкой их по популярности, которая основана на заказах почтовых отделений.

Служащим или начальнику типографии может потребоваться следующая информация: о тираже изданий, выпускаемых данной типографией, с указанием ФИО их редактора; об общем объеме заказа почтовыми отделениями каждого издания, выпускаемого на данной

типографии; о заказах почтовыми отделениями изданий у данной типографии.

Служащим или начальнику почтового отделения может потребоваться следующая информация: об изданиях, заказываемых у типографий, данным почтовым отделением; о заказах данным почтовым отделением изданий у типографий с указанием ФИО директора типографии; ФИО начальника почтового отделения, на которое поступает наибольшее общее число изданий.

Пользователям этой базы данных может потребоваться также следующая информация: об изданиях, учет которых ведется в базе данных; о типографии, в которых печатается данное издание и его тираж; о почтовых отделениях, в которые поступает данное издание и в каком количестве.

В базу данных могут вноситься следующие изменения: редакторы изданий могут добавлять информацию о своем издании, изменять или удалять ее из базы данных; служащие или начальник типографии могут добавлять информацию о своей типографии, изменять или удалять ее из базы данных; могут добавлять информацию о выпуске на своей типографии изданий, изменять или удалять ее из базы данных;

Служащие или начальник почтового отделения могут: добавлять информацию о своем почтовом отделении, изменять или удалять ее из базы данных; добавлять информацию о своем заказе на типографии изданий, изменять или удалять ее из базы данных.

Пользователям, работающим с данной базой данных, необходимы следующие отчеты:

– отчет по известным печатным изданиям, содержащий следующие данные: шифр, название, ФИО редактора этого издания; тираж и цена этого издания в различных типографиях;

– отчет о работе типографий, содержащий следующие данные: номер, адрес и ФИО директора типографии; шифры, названия, тиражи и цены выпускаемых изданий; адреса рассылки (номера и адреса почтовых отделений) для каждого печатного издания; объем заказа и цену доставки для каждого почтового отделения;

– отчет о работе почтовых отделений, содержащий следующие данные: номер, адрес и ФИО начальника почтового отделения; шифр, название, объем заказа и цена доставки покупаемых изданий; номера и адреса типографий, в которых заказаны издания.

– отчет о работе определенного почтового отделения, содержащий следующие данные: номер, адрес и ФИО начальника почтового отделения; шифр, название, объем заказа и цена доставки покупаемых изданий; номера и адреса типографий – поставщиков.

Что касается структуры почтовых отделений, то исследование направлено на выявление возможностей повышения эффективности управления почтовых отделений на базе использования средств вычислительной техники и современных методов, а так же на повышение

качества обслуживания населения почтовыми служащими при существующих технических средствах.

Любое предприятие решает поставленную задачу. Для того чтобы легче справляться с существующими задачами их разбивают между отделами предприятия или учреждения. В связи с этим современные предприятия состоят из нескольких отделов или даже множества отделов. Почтовое отделение, так же состоит из определенных отделов, которые выполняют разную функцию, а именно: приемное отделение, офис, склад.

Принцип работы почтовых отделений заключается в закупке печатных изданий, который составляет редактор типографии, их печатают и поставляют согласно заказам почтовых отделений.

Решение задач управления предусматривает интенсивный обмен информацией между всеми участниками системы, как между людьми, так и между людьми и персональными компьютерами (ПК). Возможны два вида обмена информацией – документированный и недокументированный.

Документированный – это обмен документами, главным образом на бумаге, подготовленными и заполненными людьми, либо ПК в виде распечаток.

Недокументированный обмен информацией может быть использован для:

– немедленного получения сведений от лица, являющегося их источником;

– устной передачи распоряжений;

– подтверждения выполнения полученных приказов или команд;

– сообщения о ситуациях, требующих безотлагательного вмешательства и т.п.

Его преимуществом является скорость получения информации, отсутствие ограничений на содержание вопросов и ответов, возможность введения любых уточнений и детализации. Основным недостатком недокументированных сообщений является отсутствие правовой и юридической ответственности источника информации за достоверность и точность выдаваемых сведений, невозможность последующей проверки содержания ранее переданного сообщения. Поэтому в случаях, связанных с материальной, финансовой или иной повышенной ответственностью, используют только документированные сообщения.

Основные типы документов с точки зрения отделов их ведения классифицируются на: входящие, исходящие и внутренние. К внутренним можно отнести организационно-распорядительную документацию. В системе организационно-распорядительной документации выделены три группы документов: организационная, в которую входят уставы, инструкции, правила; распорядительная – постановления, распоряжения, приказы, решения; справочно-информационная – письма, справки, протоколы, акты.

В процессе выполнения объединения потоков входящих и исходящих документов формируются типовые группы документов: договорные документы, бухгалтерские документы, документы поддержки бизнес и техпроцессов, отчетные документы.

В современных системах управления имеется программное обеспечение, позволяющее автоматизировать создание формы документа по минимальной информации, сообщаемой пользователем.

К примеру, можно рассмотреть какие должны присутствовать документы при приеме товара на склад. При получении используют самые разнообразные формы документов. В них фиксируются все действия по приёму товара, начиная с их разгрузки и кончая размещением на складе. Наличие таких документов обеспечивает детальное документирование всей деятельности по приёму товара. Главное назначение этих документов состоит в том, чтобы обеспечить тщательный учёт всего товара, поступающего на склад. Правильное безошибочное оформление документов имеет особо важное значение для приёмки товара. Ошибки в составлении и оформлении документов при приёмке товара отрицательно сказываются на всех дальнейших операциях технологического процесса складской обработки товара.

Таким образом, проектирование базы данных представляет собой длительный и трудоемкий процесс. Качество созданной БД напрямую зависит от анализа предметной области. При неполном анализе предметной области в процессе эксплуатации созданной базы данных может возникать избыточное дублирование данных, а так же различные аномалии, что, скорее всего, приведет к потере необходимых данных и повторному проектированию базы данных. Процесс последующего проектирования базы данных не менее ответственный, так как необходимо четко выявить необходимые сущности и согласно связям между ними сформировать отношения.

Задача почтовых отделений заключается в закупке печатных изданий. Эти печатные издания составляет редактор, типографии их печатают и поставляют согласно заказам почтовых отделений. В ходе анализа предметной области были выявлены группы параметров для каждого из объектов, информация о которых будет храниться в проектируемой базе данных. В дальнейшем программный комплекс может быть усовершенствован в процессе внедрения.

Литература

1. Информационные системы и базы данных: Учеб. пособие / Ю. С. Деордица. – Луганск : Изд-во ВУГУ, 1999. – 140 с. **2. Информационные** системы / В. Н. Перов. – СПб. : Питер, 2002. – 688 с. **3. Диго С. М.** Проектирование и использование баз данных; Учебник / С. М. Диго – М. : Финансы и статистика, 1995. – 208 с. **4. Четвериков В. Н.** Базы и банки данных: Учеб. для вузов по спец.

«АСУ» / В. Н. Четвериков, Г. И. Ревунков, Э. Н. Самохвалов; Под ред. В. Н. Четверикова. – М. : Высш. шк., 1987. – 248 с.

Дудник Е. А. Автоматизированная система учета работы почтовых отделений

В статье рассматриваются вопросы организации автоматизированной системы учета работы почтовых отделений с помощью систем управления базами данных. Проанализирована предметная область почтового отделения, выявлены группы параметров для каждого из объектов, информация о которых будет храниться в проектируемой базе данных. Проведен анализ требований к базе данных.

Ключевые слова: автоматизированная система, система управления базами данных, предметная область почтового отделения.

Дуднік Є. О. Автоматизована система обліку роботи поштових відділень

У статті розглядаються питання організації автоматизованої системи обліку роботи поштових відділень за допомогою систем управління базами даних. Проаналізована предметна область поштового відділення, виявлені групи параметрів для кожного з об'єктів, інформація про які буде зберігатися в проектованій базі даних. Проведено аналіз вимог до бази даних.

Ключові слова: автоматизована система, система управління базами даних, предметна область поштового відділення.

Dudnik E. A. Automated accounting of post offices

The article deals with the organization of an automated accounting system of the post office with the help of database management systems. Analyzed by subject area post office, identified a group of parameters for each of the objects, details of which will be stored in the projected database. The analysis of the database requirements.

Keywords: automated system, database management system, the subject area post office.

УДК 004.942

А. Душенко

МОДЕЛИРОВАНИЕ УСЛОВНОГО И БЕЗУСЛОВНОГО РЕФЛЕКСА В ПОВЕДЕНИИ ИСКУССТВЕННОГО АГЕНТА

Условные рефлексы – это реакции, приобретенные организмом в процессе индивидуального развития на основе «жизненного опыта». Они представляют собой выработанные в обучении и в практическом опыте привычные действия в знакомых или аналогичных условиях. Условные

рефлексы вырабатываются на базе безусловных рефлексов. Безусловные рефлексы относительно постоянны, условные же рефлексы непостоянны, и в зависимости от определенных условий они могут вырабатываться, закрепляться или исчезать, это их свойство отражено в самом их названии [1].

Безусловные рефлексы обеспечивают выживание организма и вида в постоянных условиях среды и на ранних этапах жизни. Для приспособления к новым условиям служат условные рефлексы. Они образуются при наличии определенных условий и обеспечивают наилучшую ответную реакцию [2]. Таким образом, можно сделать вывод, что адаптационные возможности организма строятся на базе рефлексов.

Простейшую схему безусловного рефлекса, построенную с использованием формальных нейронов представлена на рис 1.

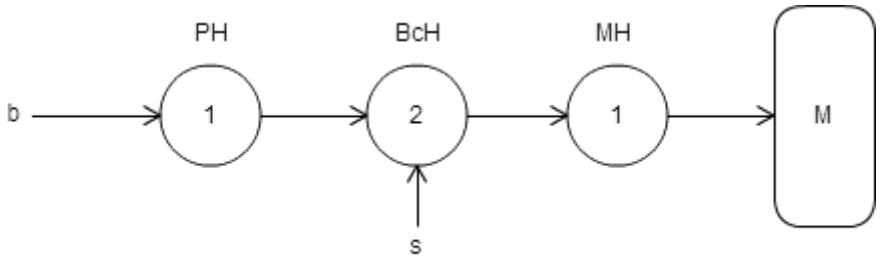


Рис. 1. Схема функционирования безусловного рефлекса на основе формальных нейронов

Рассмотрим принцип функционирования безусловного рефлекса. Воздействие среды *b* возбуждает рецепторный нейрон *PH*, передающий возбуждение на вставочный нейрон *BcH*. Сигнал *s* – характеризует состояние среды, и служит активатором вставочного нейрона *BcH*. При возбуждении вставочного нейрона передается сигнал мотонейрону *MH,* который вызывает сокращение мышц *M* [3].

Схема, приведенная на рис. 1. Полностью описывает сущность безусловного рефлекса, однако при ее построении приняты следующие упрощения:

- входное воздействие стимула *b* может выступать класс входных воздействий $b=\{b_i\}$, которые вызывают одинаковую реакцию;
- возбуждение группы мотонейронов может привести к различным сокращением мышц, различной амплитуде движений.

Рассмотрим назначение вставочного нейрона *BcH*. Исследования рефлексов показывают, что рефлекторная реакция зависит от состояния объекта. В зависимости от целей, либо мотивов живые существа имеют возможность подавлять рефлекторные реакции организма. В данных случаях объект находиться в состоянии готовности к стимулу, который вызывает рефлекс, но сигнал *s* оказывает тормозящее воздействие на вставочный нейрон.

Таким образом, при безусловном рефлексе, существуют иерархическая система различных, противоположно воздействующих стимулов, и наличие либо отсутствие сигнала *s* определяется тем, какой из одновременно действующих стимулов сильнее. Схема, реализующая механизм реакции на более сильный сигнал представлена на рис. 2.

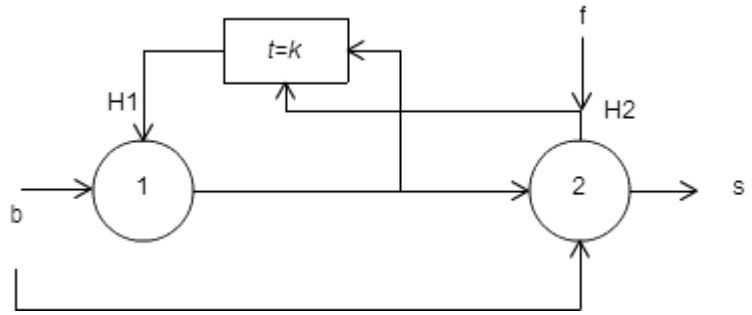


Рис. 2. Схема реализации торможения рефлекса

Пусть сигнал *b* поступает одновременно на два вставочных нейрона *H1* и *H2*. При наличии сигналов на двух входах нейрона 2 и отсутствии запрещающего сигнала *f* формируется сигнал *s* отпирающий схему представленную на рис. 1 и возбуждает мотонейрон *МН* вызывающий сокращение мышц. Итерации будут выполняться до тех пор, пока после *k* повторений на выход элемента памяти и на вход нейрона *H1* не поступит запрещающий сигнал. В результате на вход нейрона *H2* поступает только один возбуждающий сигнал, нейрон не возбуждается и сигнал *b*, разрешающий выполнение рефлекса не вырабатывается [4].

Рассмотрим ситуацию, когда возбуждающий стимул *b* имеет разрушающий характер или большую силу. В данном случае вырабатывается сигнал (признак сильного стимула) *f*, поступление которого, совместно со стимулом *b* достаточно для возбуждения нейрона *H1*. Таким образом, сильный стимул всегда вызывает рефлекс. Снятие привыкания осуществляется сбросом памяти слабых воздействий сигналом *f*.

На основе построенного алгоритма с помощью интегрированной среды разработки Embarcadero RAD Studio XE2 разработан программный продукт реализующий работу условных и безусловных рефлексов на основе двух нейронов.

Разработанное приложение позволяет:

1. моделировать условный и безусловный рефлекс;
2. реализовать механизм привыкания к периодически повторяющемуся сигналу, значение которого меньше критического, с прекращением рефлекторной реакции на стимул;
3. реализовать возобновление рефлекторного сигнала при поступлении на рецептор сигнала превышающего значение критического.

Литература

1. Самарский А.А. Математическое моделирование: Идеи. Методы. Примеры / А.А. Самарский, А.П. Михайлов – М. : Наука, 1997. – 320 с**. 2. Снитюк В. Є.** Еволюційні технології прийняття рішень в умовах невизначеності: дис. доктора техн. наук: 05.13.06. / Снитюк В. Є. – Київ , 2009. – 305 с. **3. Тарасов, В.Б .** От многоагентных систем к интеллектуальным организациям: философия, психология , информатика / В .Б . Тарасов // М.: Эдиториал УРСС, 2002 **4. Тимофеев А.В.** Мультиагентное и интеллектуальное управление сложными робототехническими системами // Теоретические основы и прикладные задачи интеллектуальных информационных.

Душенко А. Моделювання умовного і безумовного рефлексу в поведінці штучного агента

Стаття присвячена моделюванню умовного і безумовного рефлексу в поведінці штучного агента.

Ключові слова: умовний рефлекс, безумовний рефлекс, поведінка, штучний агент

Душенко А. Моделирование условного и безусловного рефлекса в поведении искусственного агента

Статья посвящена моделированию условного и безусловного рефлекса в поведении искусственного агента.

Ключевые слова: условный рефлекс, безусловный рефлекс, поведение, искусственный агент

Dushenko A. Simulation of conditioned and unconditioned reflex behavior in artificial agent

Article is devoted to modeling the conditional and unconditional reflex behavior in artificial agent.

Keywords: conditioned reflex, unconditioned reflex behavior, artificial agent

УДК 378.016:004.4

П.В. Гашенко

SOAP-СЕРВЕР НА JAVA ПРИ УЧАСТИИ APACHE CXF И SPRING

Web-сервисы применяются для обмена данными между сервером и клиентом; формат XML используется для "упаковывания" запросов и данных в целях взаимопонимания между обоими участниками общения. И сервер и клиент могут быть web-серверами или любыми

другими электронными устройствами, которые вам понадобятся. Смысл в том, что каждый запрос и ответ назапрос - XML-документ специального формата, у каждой службы он свой.

SOAP (Simple Object Access Protocol) — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Первоначально SOAP предназначался в основном для реализации удалённого вызова процедур (RPC). Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур.

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Так же существует форк для платформы .NET Framework, названный Spring.NET.

Первая версия была написана Родом Джонсоном, который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development» (Wrox Press, октябрь 2002 года).

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первый стабильный релиз 1.0 был выпущен в марте 2004. Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 3.2.2.

Несмотря на то, что Spring Framework не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring Framework предоставляет бóльшую свободу Java-разработчикам в проектировании, кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring Framework применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Apache CXF — открытый каркас для веб-сервисов Service Oriented Architectures (SOAs), работающий с несколькими стандартными протоколами, включая SOAP, WSDL 1.1 и 2.0, WS-адресацию, WS-Policy, WS-ReliableMessaging и WS-Security. Apache CXF — это комбинация из двух проектов с открытым исходным кодом Java Enterprise Service Bus Celtix и SOAP Framework XFire.

Apache CXF используется в качестве основного компонента платформы JBoss Enterprise Platform.

В данной статье будет рассмотрено, как настроить соединение с web-сервисом, передать запрос и получить ответ. Задача, которая перед

нами стоит: необходимо реализовать веб-сервис, соответствующий спецификации SOAP 1.1, основываясь на WSDL-схеме. В качестве протокола для передачи «конвертов» будем использовать HTTP. Таким образом нашу задачу можно разбить на две составляющие:

- создание java-интерфейса на основе WSDL-схемы;
- реализация веб-сервиса на основе полученного интерфейса.

По первому пункту стоит сделать следующую ремарку. В частном случае мы могли бы вручную описать необходимые нам операции, однако в общем случае удобнее воспользоваться инструментом для автоматической генерации интерфейса.

Создание SOAP-клиента обычно происходит по WSDL-документу, который представляет собой XML-документ в определенном формате, полностью описывающий тот или иной веб-сервис. Главное же, что необходимо знать для того, чтобы построить клиента к веб-сервису – это знать URL его WSDL-документа.

Вот так выглядит исходная WSDL-схема:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="CurrentTimeService" targetNamespace="http://artspb.me/cts"xmlns:tns="http://artspb.me/cts" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"> <wsdl:types> <xsd:schema targetNamespace="http://artspb.me/cts">
<xsd:element name="timeZoneId" nillable="true" type="xsd:string"/>
<xsd:element name="currentTime" type="xsd:dateTime"/>
</xsd:schema>
</wsdl:types>
<wsdl:message name="getCurrentTimeMsg">
<wsdl:part element="tns:timeZoneId" name="timeZoneId"/>
</wsdl:message>
<wsdl:message name="currentTimeMsg">
<wsdl:part element="tns:currentTime" name="currentTime"/>
</wsdl:message>
<wsdl:portType name="CurrentTimeService">
<wsdl:operation name="getCurrentTime">
<wsdl:input message="tns:getCurrentTimeMsg" name="getCurrentTime"/>
<wsdl:output message="tns:currentTimeMsg" name="currentTime"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CurrentTimeService_HttpBinding" type="tns:CurrentTimeService">
<soap:binding style="document"transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
            <wsdl:operation name="getCurrentTime">
              <soap:operation soapAction="http://artspb.me/cts"
style="document"/>
              <wsdl:input name="getCurrentTime">
                <soap:body use="literal"/>
              </wsdl:input>
              <wsdl:output name="currentTime">
                <soap:body use="literal"/>
              </wsdl:output>
            </wsdl:operation>
          </wsdl:binding>
          <wsdl:service name="CurrentTimeService_HttpService">
            <wsdl:port binding="tns:CurrentTimeService_HttpBinding"
name="CurrentTimeService_HttpPort">
              <soap:address
location="http://localhost:8080/YetAnotherService/service/currentTimeServic
e"/>
            </wsdl:port>
          </wsdl:service>
        </wsdl:definitions>
```

Сервис возвращает текущее время в указанном часовом поясе. В случае, когда пояс не указан, сервис возвращает текущее время сервера.

В процессе работы мы будем использовать следующие инструменты:

- Apache CXF;
- Apache Maven;
- Apache Ant;
- Apache Tomcat;
- soapUI.

Все их вместе очень удобно использовать с помощью IntelliJ IDEA, но этот инструмент не является обязательным.

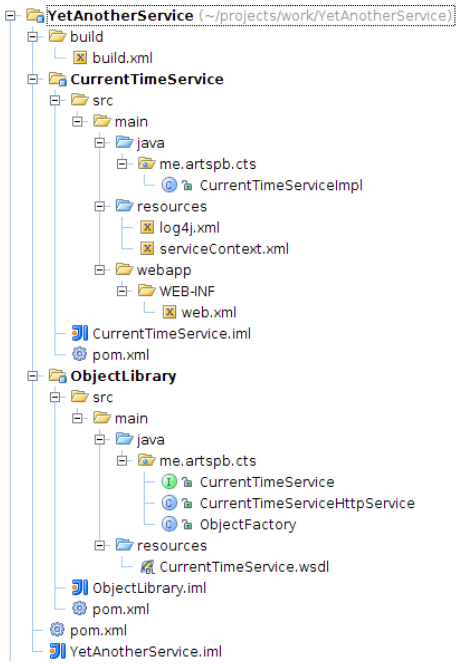


рис.1. Структура проекта

Определим структуру проекта. В папке build будем хранить скрипты для ant. В модуле ObjectLibrary будем держать исходную схему, а также сгенерированные по ней классы. Модуль CurrentTimeService будет основным, в нем будем держать реализацию сервиса, также по нему будем собирать war-файл.

В pom-файлах укажем необходимые модулям зависимости. Внешних зависимостей у нас будет не много: spring-web, log4j, cxf-rt-frontend-jaxws и cxf-rt-transports-http — все они доступны в http://repo1.maven.org/maven2. Для автоматической генерации java-интерфейса на основе WSDL-схемы воспользуемся утилитой wsdl2java из пакета Apache CXF. Немного модифицируем исходный ant-скрипт для того, чтобы он предварительно очищал директорию, в которую будут складываться java-классы. Это может быть полезно в том случае, если в будущем исходная схема будет меняться. По-умолчанию директория не очищается, что может приводить к появлению «мусора» из классов, которые больше не используются. Такая проблема характерна для схем, имеющих в своем составе сложные составные типы.

```
<?xml version="1.0"?>
<project name="BuildObject">
  <property name="cxf.home" value="/home/art/tools/apache-cxf-2.5.2"/>
  <path id="cxf.classpath">
    <fileset dir="${cxf.home}/lib">
      <include name="*.jar"/>
    </fileset>
```

```
        </path>
        <target name="cxfWSDLToJava">
          <java classname="org.apache.cxf.tools.wsdlto.WSDLToJava"
fork="true">
            <arg value="-verbose"/>
            <arg value="-d"/>
            <arg value="../ObjectLibrary/src/main/java"/>
            <arg
value="../ObjectLibrary/src/main/resources/CurrentTimeService.wsdl"/>
            <classpath>
              <path refid="cxf.classpath"/>
            </classpath>
          </java>
        </target>
        <target name="regenerate.object.library">
          <delete dir="../ObjectLibrary/src/main/java"/>
          <antcall target="cxfWSDLToJava"/>
        </target>
      </project>
```

Разместим этот скрипт в файле build/build.xml. Запустим таску regenerate.object.library, в результате работы которой в папке ObjectLibrary получим необходимый нам интерфейс.

```
package me.artspb.cts;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebResult;
import javax.jws.WebService;
import javax.jws.soap.SOAPBinding;
import javax.xml.bind.annotation.XmlSeeAlso;
@WebService(targetNamespace = "http://artspb.me/cts", name =
"CurrentTimeService")
@XmlSeeAlso({ObjectFactory.class})
@SOAPBinding(parameterStyle =
SOAPBinding.ParameterStyle.BARE)
public interface CurrentTimeService {
  @WebResult(name = "currentTime", targetNamespace =
"http://artspb.me/cts", partName = "currentTime")
  @WebMethod(action = "http://artspb.me/cts")
  public javax.xml.datatype.XMLGregorianCalendar
getCurrentTime(
    @WebParam(partName = "timeZoneId", name = "timeZoneId",
targetNamespace = "http://artspb.me/cts")
    java.lang.String timeZoneId);}
```

Стоит иметь ввиду, что с помощью утилиты wsdl2java мы сразу могли бы получить готовый сервер (ключ -server) и даже реализацию для

него (ключ -impl). Однако в рамках данной статьи мы будем использовать ее только для получения необходимого нам интерфейса, чтобы избежать жесткой привязки к сгенерированному коду.

Реализация веб-сервиса на основе полученного интерфейса

Реализацию интерфейса CurrentTimeService разместим в одноименном модуле. Код незамысловатый и в дополнительных комментариях не нуждается.

```
package me.artspb.cts;
import org.apache.log4j.Logger;
import javax.jws.WebParam;
import javax.xml.datatype.DatatypeConfigurationException;
import javax.xml.datatype.DatatypeFactory;
import javax.xml.datatype.XMLGregorianCalendar;
import java.util.GregorianCalendar;
import java.util.TimeZone;
public class CurrentTimeServiceImpl implements CurrentTimeService {
    private Logger logger = Logger.getLogger(CurrentTimeServiceImpl.class);
    public XMLGregorianCalendar getCurrentTime(
            @WebParam(partName = "timeZoneId", name = "timeZoneId", targetNamespace = "http://artspb.me/cts")
            String timeZoneId) {
        logger.debug("Operation getCurrentTime was requested.");
        XMLGregorianCalendar gregorianCalendar;
        try {
            gregorianCalendar = getXmlGregorianCalendar(timeZoneId);
        } catch (DatatypeConfigurationException e) {
            throw new RuntimeException(e);
        }
        logger.debug("Successful.");
        return gregorianCalendar;
    }
    private XMLGregorianCalendar getXmlGregorianCalendar(String id) throws DatatypeConfigurationException {
        TimeZone timeZone;
        if (!"".equals(id)) {
            logger.debug("TimeZoneId isn't null: " + id);
            timeZone = TimeZone.getTimeZone(id);
        } else {
            logger.debug("TimeZoneId is null. Will use default value.");
            timeZone = TimeZone.getDefault();
        }
        GregorianCalendar gregorianCalendar = new GregorianCalendar(timeZone);
```

return DatatypeFactory.newInstance().newXMLGregorianCalendar(gregorianCalendar); }}

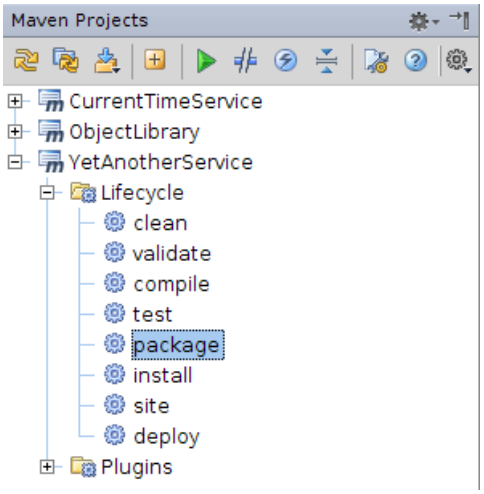


рис.2. Таски для выполнения

Для сборки проекта выполним задачу maven «package». Полученный war-файл развернем на сервере Tomcat. Будем считать, что он доступен на порту 8080.

Для проверки воспользуемся программой soapUI. WSDL-схема сервиса доступна по ссылке http://localhost:8080/CurrentTimeService/service/currentTimeService?wsdl, с ее помощью мы можем создать новый проект soapUI.

Таким образом поставленная задача достигнута: веб-сервис корректно обрабатывает запросы. Также нам удалось избежать «жесткой» привязки к генерируемой реализации, что делает код более гибким и позволяет прикладывать меньше усилий для внесения изменений в дальнейшем.

Литература

1. Герберт Шилдт. Java. Полное руководство. Java SE 7 = Java 7: The Complete Reference. - 8-е изд. - М:. Вильямс, 2012. - 1104 с. **2.** Кей С. Хорстманн, Гари Корнелл. Java 2. Библиотека профессионала = Core Java 2, том II - Расширенные функции. - 8-е изд. - М:. Вильямс, 2008. - Т. II: Тонкости программирования. - 992 с. **3.** Джошуа Блох. Java. Эффективное программирование = Эффективная Java. - М: Лори, 2002.. - 224 с. **4.** Роберт Седжвик, Кевин Уэйн. Алгоритмы на Java = Алгоритмы. - 4-е изд. - М:. Вильямс, 2012. - 848 с.

Гашенко П.В. SOAP-сервер на java за участю apache cxf і spring

У даній статті розглянуто, як налаштувати з'єднання з web-сервісом, передати запит і отримати відповідь. Завдання, яке перед нами

стоїть: необхідно реалізувати веб-сервіс, відповідний специфікації SOAP 1.1, грунтуючись на WSDL-схемі.

Ключові слова: SOAP-сервер, java, apache cxf, spring

Гашенко П.В. SOAP-сервер на java при участии apache cxf и spring

В данной статье рассмотрено, как настроить соединение с web-сервисом, передать запрос и получить ответ. Задача, которая перед нами стоит: необходимо реализовать веб-сервис, соответствующий спецификации SOAP 1.1, основываясь на WSDL-схеме.

Ключевые слова: SOAP-сервер, java, apache cxf, spring

Gashenko PV SOAP-server java involving apache cxf and spring

In this article you will learn how to configure a connection to the web-service, submit a query and get a response. The task that we face: it is necessary to implement a web service, a conforming SOAP 1.1, WSDL-based on the scheme.

Keywords: SOAP-server, java, apache cxf, spring

УДК 378.016:004.4

И.А. Гладкий

RESTFUL WEB SERVICE НА JAVA

Все актуальнее становится задача интеграции **программных модулей** с другими системами не только в локальных сетях, а и в сети интернет. О таком замечательном механизме, как **Restful веб – сервис,** я и постараюсь рассказать в этой статье.

Используемое ПО:

- JDK 1.7
- Apache_CXF
- Spring 3 Framework JDBC
- Apache Tomcat 7.0
- MySQL 5.1
- Eclipse 4.2 Juno
- Maven 3.0

ПО выбиралось по принципу — минимализма и практичности. Да, вместо MySQL для нагруженных сервисов без необходимости делать сложные запросы в базу, очень хорошо использовать MongoDB, ну, по крайней мере по этому поводу много написанно, да и опять же удобнее её использовать так как на входе тот же JSON.

Сервис будет работать с БД ,имея стандартные CRUD операции в своём составе, ну и, конечно же, получать записи списком или по Id.

Конечно же, хотелось бы иметь возможность параметризированного запроса на получение списка записей, не плохо было бы сделать «красивый» URL к сервису, прикрутить какой-нибудь интерсептор, чтобы, например, проверять права пользователя на доступ к сервису, или что-нибудь другое делать перед запуском сервиса, ну и как-то централизованно управлять кодами ошибок в ответах от сервера.

Таблица БД:

```
CREATE TABLE `customer` (
  `id` varchar(45) NOT NULL,
  `first_name` varchar(45) DEFAULT NULL,
  `last_name` varchar(45) DEFAULT NULL,
  `phone` varchar(45) DEFAULT NULL,
  `mail` varchar(45) DEFAULT NULL,
  `adress` varchar(45) DEFAULT NULL,
  `contract_id` varchar(45) DEFAULT NULL,
  `contract_expire_date` date DEFAULT NULL
)
```

WS Endpoints:

1. http://mysite.com/service/customer

2. http://mysite.com/service/customer/{id}

4 стандартных статуса, которые мы будем дополнительно обрабатывать (например, добавлять версию наших веб сервисов в ответ и при ошибке — наш код ошибки):

200 — Successful;
401 — Not Authorized;
404 — Not Found;
500 — Server error during operation.

Веб сервисы:

```
public class CustomersServiceJSON implements ICustomersService
{
   private ICustomersDAO customersDAO;
   public ICustomersDAO getCustomersDAO() {
      return customersDAO;
   }
   public void setCustomersDAO(ICustomersDAO customersDAO) {
      this.customersDAO = customersDAO;
   }
   @Context
   private HttpHeaders requestHeaders;
   private String getHeaderVersion() {
      return requestHeaders.getRequestHeader("version").get(0);
   }
   @GET
```

```
    @Path("/{id}")
    public Response getCustomer(@PathParam("id") String id) {
        Customer customer = customersDAO.getCustomer(id);
        if (customer != null) {
            return ResponseCreator.success(getHeaderVersion(),
customer);
        } else {
            return ResponseCreator.error(404,
Error.NOT_FOUND.getCode(),
                getHeaderVersion());
        }
    }
    @DELETE
    @Path("/{id}")
    public Response removeCustomer(@PathParam("id") String id) {
        if (customersDAO.removeCustomer(id)) {
            return ResponseCreator.success(getHeaderVersion(),
"removed");
        } else {
            return ResponseCreator.success(getHeaderVersion(), "no such
id");
        }
    }
    @POST
    @Consumes(MediaType.APPLICATION_JSON)
    public Response createCustomer(Customer customer) {
        System.out.println("POST");
        Customer creCustomer =
customersDAO.createCustomer(customer);
        if (creCustomer != null) {
            return ResponseCreator.success(getHeaderVersion(),
creCustomer);
        } else {
            return ResponseCreator.error(500,
Error.SERVER_ERROR.getCode(),
                getHeaderVersion());
        }
    }
    @PUT
    @Consumes(MediaType.APPLICATION_JSON)
    public Response updateCustomer(Customer customer) {
        Customer updCustomer =
customersDAO.updateCustomer(customer);
        if (updCustomer != null) {
```

```
                return ResponseCreator.success(getHeaderVersion(),
updCustomer);
            } else {
                return ResponseCreator.error(500,
Error.SERVER_ERROR.getCode(),
                    getHeaderVersion());
            }
        }
        @GET
        public Response getCustomers(@QueryParam("keyword") String
keyword,
            @QueryParam("orderby") String orderBy,
            @QueryParam("order") String order,
            @QueryParam("pagenum") Integer pageNum,
            @QueryParam("pagesize") Integer pageSize) {
          CustomerListParameters parameters = new
CustomerListParameters();
            parameters.setKeyword(keyword);
            parameters.setPageNum(pageNum);
            parameters.setPageSize(pageSize);
            parameters.setOrderBy(orderBy);
            parameters.setOrder(Order.fromString(order));
            List<Customer> listCust =
customersDAO.getCustomersList(parameters);
            if (listCust != null) {
              GenericEntity<List<Customer>> entity = new
GenericEntity<List<Customer>>(
                  listCust) {
              };
              return ResponseCreator.success(getHeaderVersion(), entity);
            } else {
              return ResponseCreator.error(404,
Error.NOT_FOUND.getCode(),
                  getHeaderVersion());
            }
          }
        }
```

Итак ,4 веб сервиса в зависимости от URI и метода которым этот URI передаётся, есть объект DAO, который подключается в beans.xml и доступ к заголовкам запроса, чтобы доставать для примера кастомный заголовок «version».

Код процедуры, которая отрабатывает перед тем как вызывается сервис:

```
public class PreInvokeHandler implements RequestHandler {
  int count = 0;
```

```
    private boolean validate(String ss_id) {
        count++;
        System.out.println("SessionID: " + ss_id);
        if (count == 1) {
            return false;
        } else {
            return true;
        }
    }
    public Response handleRequest(Message message,
ClassResourceInfo arg1) {
        Map<String, List<String>> headers = CastUtils.cast((Map<?,
?>) message
                .get(Message.PROTOCOL_HEADERS))
        if (headers.get("ss_id") != null &&
validate(headers.get("ss_id").get(0))) {
            return null;
        } else {
            return ResponseCreator.error(401,
Error.NOT_AUTHORIZED.getCode(), headers.get("version").get(0));
        }
    }
}
```

Здесь в методе validate() можно проверять какие-то пред условия, чисто для теста добавлена проверка кастомного заголовка в запросе идентификатор сессии «ss_id», ну, и с первого раза даже с этим заголовком будет падать 401.

Общий обработчик exceptions:

```
public class CustomExceptionMapper implements
ExceptionMapper<Exception> {
    @Context
    private HttpHeaders requestHeaders;
    private String getHeaderVersion() {
        return requestHeaders.getRequestHeader("version").get(0);
    }
    public Response toResponse(Exception ex) {
        System.out.println(ex.getMessage() + ex.getCause());
        return ResponseCreator.error(500,
Error.SERVER_ERROR.getCode(), getHeaderVersion());
    }
}
```

Также нужен вспомогательный класс для формирования ответа серверу и глобальный enum для хранения наших кодов ошибок. Да, дескриптор развёртывания и beans.xml :

web.xml:

```
<web-app>
  <display-name>service</display-name>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>WEB-INF/beans.xml</param-value>
  </context-param>
  <listener>
    <listener-class>
      org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
  <servlet>
    <servlet-name>CXFServlet</servlet-name>
    <display-name>CXF Servlet</display-name>
    <servlet-class>
      org.apache.cxf.transport.servlet.CXFServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>CXFServlet</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Тут основной интерес представляет подключение экшн сервлета от Apache — CXFServlet и стандартного спрингового ContextLoaderListener.

beans.xml:

```
<!-- Imported resources for cxf -->
  <import resource="classpath:META-INF/cxf/cxf.xml" />
  <import resource="classpath:META-INF/cxf/cxf-extension-jaxrs-binding.xml" />
  <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />

  <!-- Imported bean for dao -->
  <import resource="classpath:META-INF/spring/dao.xml"/>

  <bean id="customersService" class="com.test.services.customers.rest.CustomersServiceJSON">
    <property name="customersDAO" ref="customersDAO"/>
  </bean>
```

```
    <bean id="preInvokeHandler"
class="com.test.services.rest.PreInvokeHandler" />
    <bean id="customExceptionMapper"
class="com.test.services.rest.CustomExceptionMapper" />

    <jaxrs:server id="restContainer" address="/customer">
      <jaxrs:serviceBeans>
        <ref bean="customersService" />
      </jaxrs:serviceBeans>
      <jaxrs:providers>
        <ref bean="preInvokeHandler" />
        <ref bean="customExceptionMapper" />
      </jaxrs:providers>
    </jaxrs:server>
```

Здесь, собственно, задали нужные конфигурационные файлики для CXF, подключили DAO объект, наш предобработчик и обработчик исключительных ситуаций, конечно же, сам бин с сервисами и задали корень для сервисов.

Для того чтобы отработать сервисы я использовал REST Console 4.0.2 плагин для хрома, главное задать нужные ендпоинт, кастомные заголовки (как я уже говорил без «ss_id» всегда будет падать 401) и контент тип.

Литература

1. Любош Бруга. Java по-быстрому: Практический экспресс-курс = Luboš Brůha. Java Hotová řešení. - М:. Наука и техника, 2006. - 369 с. **2.** Кей С. Хорстманн, Гари Корнелл. Java 2. Библиотека профессионала = Core Java 2, том II - Расширенные функции. - 8-е изд. - М:. Вильямс, 2008. - Т. II: Тонкости программирования. - 992 с. **3.** Джошуа Блох. Java. Эффективное программирование = Эффективная Java. - М: Лори, 2002.. - 224 с. **4.** Герберт Шилдт. Java. Полное руководство. Java SE 7 = Java 7: The Complete Reference. - 8-е изд. - М:. Вильямс, 2012. - 1104 с.

Гладкий І.А. Restful web service на java

Все актуальнішою стає завдання інтеграції програмних модулів з іншими системами не тільки в локальних мережах, а і в мережі інтернет. Стаття містить опис сервісу Restful веб.

Ключові слова: Restful web service, java

Гладкий И.А. Restful web service на java

Все актуальнее становится задача интеграции программных модулей с другими системами не только в локальных сетях, а и в сети интернет. Статья содержит описание сервиса Restful веб.

Ключевые слова: Restful web service, java

Smooth IA Restful web service in java

All, the task of integrating software modules with other systems not only in local area networks, but also on the Internet. The article contains a description of the service Restful Web.

Keywords: Restful web service, java

УДК 378.016:004.4

Я.И. Громова

ТЕСТЫ В САПР ЭК

В системе САПР ЭК с использованием онтологии предметной области (ПО) готовится лекционный материал. Однако, для полноценного ЭК необходимы электронные работы и тесты. В данной работе рассматривается возможность подготовки тестов в САПР ЭК.

Система САПР ЭК предназначена для автоматизированного наполнения электронных курсов (ЭК) для системы управления обучением Moodle. В статье подробно рассмотрен процесс создания тестов в САПР ЭК Moodle [1].

Введение

Одним из компонентов e-learning являются ЭК. Проектирование ЭК предполагает формирование множеств концептов, их описаний, отношений, примеров, лабораторных работ, тестов с минимальным влиянием субъективного фактора. Построение указанных множеств вручную является трудоемким процессом, как по времени, так и по количеству вовлечённых в процесс проектирования высококвалифицированных специалистов. Понимание важности создания инструментальных средств поддержки процесса проектирования ЭК заданной ПдД пришло практически одновременно с принятием парадигмы e-learning [3].

Постановка задачи

Система автоматизации проектирования общезначимых электронных курсов (САПР ЭК), используя онтологию ПдО и БСИ должна построить ЭК соответствующей ПдД. При разработке программной модели САПР ЭК необходимыми этапами являются определение ее функций и ее функциональная модель. В работе предлагаются функции программной модели САПР ЭК и описание ее функциональной модели.

Решение задачи

Описание онтологии в БСИ дает основной учебный материал ЭК. Тесты вводятся при выборе соответствующих понятий ЭК вручную или из имеющихся электронных библиотек кафедры. На основе моделей

адаптивного управления качеством можно адаптировать ЭК к потребностям студента [3].

В конце работы САПР ЭК конвертирует справочники ЭК в соответствующие файлы Moodle [2].

Предлагаемая программная модель САПР ЭК выполняет следующие функции:

1. Чтение OWL файла описания онтологии и вывод онтографа предметной области на экран;
2. Обеспечение удобного интерфейса выбора ветви онтологии ниже выделенной вершины (при щелчке мышкой по вершине, ветвь ниже вершины выделяется красным цветом, остальные вершины становятся полупрозрачными);
3. Обеспечение автоматизированной добавки к понятию теста и запись его в файл «TEST.txt». Автоматизация предусматривает три варианта:
   - 3.1 Экспорт теста из электронного ресурса;
   - 3.2 Ввод текста теста вручную;
   - 3.3 Отсутствие тестов для данного понятия;
4. Конвертирование файлов тестов в файлы Moodle.

По выше описанной схеме удобно готовить лекционный материал. Однако, для полноценной ЭК необходимы электронные работы и тесты. В данной работе рассматривается возможность подготовки тестов в САПР ЭК.

Функциональная модель САПР ЭК.

Для проектирования функциональной модели был выбран язык функционального моделирования UML (Unified Modeling Language). ***UML*** – язык графического описания [4].

Функциональная модель САПР ЭК представляет собой набор диаграмм: диаграмма вариантов использования и диаграмма активности.

Исходными данными для системы САПР ЭК являются OWL-файлы содержащие описание онтологии предметной области (ПдО).

Описание Moodle: Система Moodle (Modular Object-Oriented Dynamic Learning Environment) отвечает специфике электронного обучения и требованиям обучаемых, стремящихся приблизится к западным моделям обучения. По своим возможностям Moodle выдерживает сравнение с известными коммерческими системами управления учебным процессом, в то же время выгодно отличается от них тем, что распространяется в открытых исходных кодах - это дает возможность использовать ее под особенности каждого образовательного проекта, дополнить новыми сервисами.

Moodle предоставляет разработчику тестов очень удобные средства для манипулирования тестовыми заданиями, а ее адаптивность обеспечивает индивидуальный подход к тестируемому.

При создании программы тестирования в системе Moodle учитывается возможность использования любой системы оценивания

(балльную и словесную); полная информация о работе тестирующихся (активность, время); возможность внесения изменений без тотального перепрограммирования; создания вопросов в закрытой форме (множественный выбор), с выбором верно/не верно, на соответствие, предполагать короткий текстовый ответ, а также числовой или вычисляемый.

Все вопросы хранятся в базе данных и могут быть впоследствии использованы снова.

Поскольку ЭК может использовать тесты разного типа и содержания, то программа должна иметь интерфейс, указывающий нахождение теста. Тесты берутся из библиотеки тестов.

Выводы

В работе предложена программная модель создания тестов в САПР ЭК, использующая результаты работы оригинального инструментального комплекса онтологического назначения (ИКОН). В автоматизированном режиме из заданной ПдО строится онтология предметной дисциплины (ПдД).

Для подготовки тестов в САПР ЭК были учтены следующие требования:

- Возможность выбора файла теста из библиотеки тестов или из других источников.
- Другие требования, описанные выше.

Литература

1. Eduard Zharikov, Topical questions of implementation of information services in a network of University ,TEKA Kom. Mot. I Energ. Roln. – OL PAN, 2010, 10B, 331-337. 2. Палагин А.В. Онтологические методы и средства обработки предметных знаний / А.В. Палагин, С.Л. Крывый, Н.Г. Петренко. – [монография] – Луганск: изд-во ВНУ им. В. Даля, 2012. – 323 с.. 3. Vladimir Tkach, The adaptive control of higher education process quality of a university on a “teacher-student” level ,TEKA Kom. Mot. I Energ. Roln. – OL PAN, 2010, 10B, 241-246. 4. C. Marshall, Enterprise Modelling with UML, ISBN 0-201- 43313-3, Addison-Wesley, Reading, MA, 2000. 5. T. Quatrani, Visual Modeling with Rational Rose and UML, Addison-Wesley, Reading, MA, 1998.

Громова Я.І. Тести в САПР ЕК

У роботі пропонуються функції програмної моделі САПР ЕК і опис її функціональної моделі.

Ключові слова: САПР ЕК, програмна модель, функціональна модель

Громова Я.И. Тесты в САПР ЭК

В работе предлагаются функции программной модели САПР ЭК и описание ее функциональной модели.

Ключевые слова: САПР ЭК, программная модель, функциональная модель

Gromov Y.I. Tests in CAD EC

This paper proposes a software function CAD model EC and description of its functional model.

Keywords: CAD EC programming model, functional model

УДК 004.434

Е.С. Ефремов, Р.В. Киричевский

КЕРАМИЧЕСКИЕ ДВИГАТЕЛИ. ИСПОЛЬЗОВАНИЕ КЕРАМИКИ В КАЧЕСТВЕ МАТЕРИАЛА ДЛЯ ДЕТАЛЕЙ ДВИГАТЕЛЯ. ПРЕИМУЩЕСТВА И НЕДОСТАТКИ

Среди больших и малых **сенсаций**, ежедневно облетающих планету, сообщения о создании «керамического» двигателя не затерялись: ведь речь шла о грядущей революции в мире моторов.

Местом ее рождения стала Япония. Сначала фирма НГК, известная производством свечей зажигания и изоляторов, объявила об успешных стендовых испытаниях 50-кубового двухтактного двигателя из керамических деталей. Почти в это же время на экранах телевизоров предстал японский автомобиль с трехцилиндровым «керамическим» двигателем... без системы охлаждения! Его «родители» — ведущая в своей области фирма «Киото керамике» и автомобильная компания «Исузу» явно торопились обнародовать сенсацию. Правда, это был только макетный образец автомобиля, но он двигался! Сообщалось, что двигатель машины — дизельный, рабочим объемом 2800 см3 и мощностью 55 л.с.

Это событие вызвало огромный интерес не только у специалистов, но и у широкого круга автолюбителей, что, в общем, неудивительно. Ведь речь шла о моторе, значительно более экономичном, чем двигатель внутреннего сгорания. А он, несмотря на далеко не идеальные экономические показатели, вот уже много лет удерживает в автомобилестроении позицию лидера, хотя настойчивые попытки изменить эту традицию не прекращаются.

При изготовлении транспортных двигателей, традиционно используются разные металлы и их сплавы, технология обработки которых доведена до высокой степени совершенства. В силу этого, а также массового характера производства только весьма серьезные причины могут заинтересовать моторостроительные фирмы в использовании нетрадиционных материалов.

С этих позиций посмотрим на достоинства и перспективы керамики.

Основные ее преимущества — более высокая, чем у металлов и их сплавов, термостойкость и прочность на сжатие, лучше теплоизоляционные свойства, меньшая объемная масса.

Керамика, выдерживает температуры порядка 1500°С (примерно на 600° выше, чем металл)

Высокая термостойкость и малая теплопроводность керамических деталей позволяет полнее использовать теплоту сгорания топлива на совершение полезной механической работы, и даже отказаться от системы охлаждения двигателя, а ее отсутствие не только существенно упрощает и облегчает двигатель, но, что важнее, позволяет намного сократить потери тепла, связанные с охлаждением деталей, и тем самым резко (на две трети) поднять термический КПД двигателя.

При работе без теплообмена с внешней средой термический КПД может возрасти на 60%.

Известно, что даже обыкновенное покрытие камеры сгорания ДВС тонким слоем теплоизолирующего, например, керамического покрытия, позволяет снизить температуру деталей в несколько раз, что обеспечивает уменьшение интенсивности изнашивания на 20…25 %, а расхода топлива на 3…5 %.

Повышенная температура поверхностей поршня и камеры сгорания и сама по себе создает лучшие условия для более полного и эффективного сгорания смеси. Отсюда — возможность применения обедненных смесей (а это дополнительный резерв экономии топлива), отсюда и более низкий уровень токсичности.

Такое достоинство керамики, как меньшая объемная масса, способствует облегчению силовой установки и уменьшению сил инерции ее движущихся частей.

Список достоинств выглядит весьма внушительно. И чтобы не создать впечатления о неограниченных возможностях керамики, пора вспомнить о ее недостатках.

Главный из них — малая прочность на растяжение и изгиб. Прекрасно работающие на сжатие, керамические материалы крайне плохо противостоят этим видам нагрузки (что, кстати, заставляет весьма осторожно отнестись к сообщениям о полностью керамическом двигателе).

Здесь уместно сказать, что сообщения об успехах НГК и «Киото керамике» являются сенсацией лишь отчасти. Их заслуга — в попытке использовать керамику для деталей поршневого ДВС. Но еще раньше такой материал начали осваивать создатели газотурбинных двигателей, где вопрос о повышении рабочей температуры для обеспечения экономичности стоял куда острее.

Тем не менее перспектива применения керамики в ДВС, как поршневых, так и газотурбинных, представляется заманчивой. Для ее достижения некоторые фирмы (среди них «Форд» и «Фольксваген») избрали более доступный путь: использование керамики не взамен

металла, а в сочетании с ним для изготовления именно тех деталей, где преимущества керамики наиболее ощутимы. Речь идет о теплоизолирующих вставках и пластинах для поршней, вставках для камер сгорания, направляющих втулок клапанов. Наряду с этим «Форд», исследуя возможность применения керамики в газовых турбинах, создал, например, керамический ротор для турбонагнетателя**.** Сообщается, что основным применяемым здесь материалом является нитрид кремния.

Были попытки применения керамики и на серийных двигателях, например никель-кремниевое (никосил) покрытие на блоках цилиндров, представляющее собой слой никеля толщиной 0,1…0,2 мм со сверхтвердыми частицами карбида кремния размером до 3 мкм.

Не стремясь, в отличие от японцев, привлечь к своей работе всеобщее внимание, довольно успешно экспериментирует с керамическими деталями (поршни, вставки в поршень и головку цилиндров) известная дизелестроительная фирма «Камминс» (США). «Скромность» ее легко объяснима: исследование свойств керамических деталей — составная часть долгосрочной программы по разработке адиабатического (неохлаждаемого) дизеля.

Несмотря на достижения в области создания более экономичного керамического двигателя, говорить о его серьезной конкуренции традиционным двигателям внутреннего сгорания еще рано.

Ресурс деталей двигателя из керамических материалов мал, и технология получения этих материалов и их обработки оставляет пока желать лучшего.

Это обстоятельство, несмотря на заложенные в керамическом двигателе значительные преимущества, и служит пока серьезным препятствием на пути его широкого распространения.

Литература

1. Систематологические основы инноватики, Акимов А.А., Гамидов Г.С., Колосов В.Г. – Политехника, 2002. – 596 с. **2. Н.Л. Маренков**, Инноватика. – М.: Либроком, 2009. – 304 с. **3. Рыбалкина**, М. Нанотехнологии для всех. /М.Рыбалкина. – М.: Nanotechnology News Network, 2006. – 444 с. **4. http://www.nanonewsnet.ru 5. Автомобили** будущего: ERL: http//avogop.com/site/publication-full/717 (2010, 9 декабря). **6. http://www.ntsr.info/**

Єфремов Є.С., Киричевський Р.В. Керамічні двигуни. Використання кераміки в якості матеріалу для деталей двигуна. Переваги і недоліки.

У статті описані і проаналізовані можливості кераміки в ДВС. Виділено основні моменти роботи з керамікою, її позитивні і негативні характеристики. Описано можливості для подальшого розвитку керамічних деталей.

Ключові слова: НГК, «Ісузу», кераміка в ДВС, «Кіото кераміці».

Ефремов Е.С., Киричевский Р.В. Керамические двигатели. Использование керамики в качестве материала для деталей двигателя. Преимущества и недостатки.

В статье описаны и проанализированы возможности керамики в ДВС. Выделены основные моменты работы с керамикой, ее позитивные и негативные характеристики. Описаны возможности для дальнейшего развития керамических деталей.

Ключевые слова: НГК, «Исузу», керамика в ДВС, «Киото керамике».

Efremov. E.S, Kirichevsky R.V. Ceramic engines. Use of ceramics as the material for engine parts. Advantages and disadvantages.

The article describes and analyzes the possibility of ceramics in engines. The basic aspects of working with ceramics, its positive and negative characteristics. Describes opportunities for further development of ceramic parts.

Keywords: NGK, "Isuzu", ceramics in the engine, the "Kyoto Ceramics."

УДК 004.434

Е.В. Зубков

ИСПОЛЬЗОВАНИЕ СКРИПТОВЫХ ЯЗЫКОВ ДЛЯ АВТОМАТИЗАЦИИ ПОСТРОЕНИЯ 3D МОДЕЛЕЙ

Необходимость перехода к новым технологиям на основе 3D проектирования очевидна для огромного большинства предприятий, проектных институтов в различных отраслях и сферах деятельности. Трёхмерная графика, активно применяется для создания изображений на плоскости, экрана или листа печатной продукции в науке и промышленности. Например, в системах автоматизации проектных работ(САПР; для создания твердотельных элементов: зданий, деталей машин, механизмов), архитектурной визуализации(сюда относится и так называемая «виртуальная археология»). Самое широкое применение 3D моделирование используется в современных системах медицинской визуализации, во многих современных компьютерных играх, а также как элемент кинематографа, телевидения и т. д. [1]

Следует отметить, что моделирование тех или иных объектов имеет свои особенности. Визуализация технического объекта имеет определённый алгоритм, который можно свести к следующим последовательным шагам: поиск чертежа, создание эскиза или общего вида будущей модели, мысленная разбивка или деление модели на

составные элементы, если выбранная модель является сложной, состоящей из различных частей и компонентов.[2]

Для упрощения моделирования в программных пакетах (ПП) существуют такие способы: использование языков сценариев (ЯС), поддержка других языков программирования (ЯП). Всё это используется для создания плагинов, дополнений, скриптов автоматизации и расширений программных модулей.

Те, кому нравится программировать и знакомы с такими ЯП как C или C++, вполне могут создавать дополнения для программ с помощью SDK(Software Development Kit). Некоторые программные продукты имеют свой собственный интерфейс прикладных программ (API — Application Programming Interface). ЯС прекрасно подходят для выполнения почти 95 процентов всех задач, этого вполне хватает большинству, а кроме того ЯС не слишком трудны в изучении. В отличии от ЯС для изучение API необходимо знать подробности применения SDK ПП для разработки дополнений. Существуют ЯС внутри другой программы. Их код интерпретируется построчно, а не компилируется. Поскольку сценарий интерпретируется “родительской” программой, выполняется он немного медленнее, чем откомпилированная программа, однако и дополнительных затрат они требуют намного меньше, чем откомпилированные программы.

ЯС позволяют автоматизировать выполнение сложных или повторяющихся задач. Скрипты, могут распространяться отдельно, и содержаться как в самой программе, так и отдельными файлами с расширениями (у каждого ПП своё, например в 3DS MAX язык MAXScript, расширение*.ms, *.vms, *.mcr).

С помощью ЯС можно выполнять действия над трехмерными объектами, создавать собственные интерфейсы, считывать и сохранять данные на диск, писать инструменты, дополнять уже имеющиеся и многое другое.

ПП позволяющие создавать трёхмерную графику, то есть моделировать объекты виртуальной реальности и создавать на основе этих моделей изображения, очень разнообразны. Вот несколько ПП, которые в последние годы являются устойчивыми лидерами в этой области : CINEMA 4D; Autodesk Maya; Blender; Autodesk 3ds Max .

CINEMA 4D или сокращённо C4D фирмы MAXON является универсальной комплексной программой для создания и редактирования трёхмерных эффектов и объектов. Позволяет рендерить объекты по “методу Гуро”. Поддержка анимации и высококачественного рендеринг, отличается более простым интерфейсом, чем у аналогов, и встроенной поддержкой русского языка, что делает её популярной среди русскоязычной аудитории. [3]

CINEMA 4D поддерживает следующие языки программирования

– Python, C++.

– C.O.F.F.E.E.— скриптовый язык программирования. Встречается только в программах CINEMA 4D и BodyPaint 3D.

– Xpresso — нодовая система программирования. Система основана на графическом интерфейсе. Представляет собой набор узлов (нодов), обладающие входными и выходными портами. Узлы являются отдельными функциями. Порты узлов можно соединять между собой, что приводит к последовательному просчёту функций. Новые узлы можно создавать наподобие плагинов при помощи C++. Также имеются два отдельных узла Python и C.O.F.F.E.E., в которых можно создавать код на соответствующих языках.

Autodesk Maya — редактор трёхмерной графики. В настоящее время стала стандартом 3D графики в кино и телевидении. Первоначально разработана для ОС Irix (платформа SGI), затем была портирована под ОС Linux, Microsoft Windows и Mac OS. В настоящее время существует как для 32, так и для 64-битных систем.[4]

Основные принципы языка MEL:

MEL — интерпретируемый процедурный язык, поэтому код на нем будет выполняться одинаково на всех платформах, на которых запускается Maya (за исключением команд работы с операционной системой).

Встроенный язык Maya (MEL— Maya Embedded Language) является основным уровнем взаимодействия пользователя с Maya. При запуске Maya, фактически выполняется несколько сценариев MEL, которые и формируют все окна. На самом деле Maya никакого интерфейса не имеет. Запустить Maya можно даже из командной строки операционной системы, достаточно ввести команду Maya -prompt! Практически все, что делается в Maya, выполняют сценарии MEL.

Но что же это означает для среднего пользователя Maya? Упрощенно, он имеет возможность сделать все, что уже сделали разработчики исходного интерфейса. Пользователь может создавать окна с ползунками, вкладки, текстовые поля и кнопки для них; может создавать атрибуты в панели каналов; может даже добавлять пункты в меню и новые панели. Тот факт, что Maya построена на языке MEL — уже является одним из наиболее существенных преимуществ этой программы.[5]

Blender — ПП для создания трёхмерной компьютерной графики, приложение для создания широкого спектра 2D- и 3D-контента. Нацелен на профессиональных 3D-аниматоров, графиков и художников. Предоставляет большие возможности для моделирования, текстурирования, освещения, анимации и пост-обработки видео. Blender может быть использован для создания 3D-визуализации, статических изображений, а также видео кинематографического качества, а интегрированный 3D-рендерер реального времени позволяет создавать интерактивный 3D контент для автономного воспроизведения.[6]

Blender обладает мощным, хотя и часто недооцениваемым достоинством: он располагает полноценным встроенным интерпретатором языка Python. Это позволяет любому пользователю пополнять функциональность пакета при помощи написания Python-скриптов.

Python — это не только огромная неядовитая змея, но и интерпретируемый, интерактивный, объектно-ориентированный язык программирования. Он включает поддержку модулей, исключений, динамического назначения типов, динамических типов данных очень высокого уровня и классов. Python соединяет изрядную мощь с ясным синтаксисом. Он был создан с очевидной целью, быть используемым в качестве языка расширения в приложениях, требующих программируемого интерфейса, вот почему Blender его и использует. Будучи, наряду с двоичными плагинами, одним из способов расширения функциональности Blender'а: сочинение скриптов на Python'е, является более мощным, гибким, и в то же время полнофункциональным и надежным средством. В общем случае написание Python-скрипта предпочтительней написания плагина.

Autodesk 3ds Max (ранее 3D Studio MAX) — полнофункциональная профессиональная программная система для создания и редактирования трёхмерной графики и анимации, разработанная компанией Autodesk. Содержит самые современные средства для художников и специалистов в области мультимедиа. Работает в операционных системах Microsoft Windows и Windows NT (как в 32 битных, так и в 64 битных). Весной 2012 года выпущена пятнадцатая версия этого продукта под названием «Autodesk 3ds Max 2013».[7]

MAXScript — это встроенный в 3ds Max язык макропрограммирования, обеспечивающий пользователям следующие возможности:[8]

- создание сценариев (скриптов) хранящихся в файлах типа *.ms, которые воспроизводят все функциональные возможности 3ds Max, такие как построение геометрических моделей, расстановка осветителей и камер, назначение материалов, визуализация и анимация объектов сцены;
- создание макросов, хранящихся в файлах типа *.mcr и описывающих свойства новых кнопок на панелях инструментов;
- создание ваших собственных свитков для командной панели Utilities (Утилиты) и окон диалога, имеющих стандартный для программы интерфейс;
- написание собственных модулей для работы с сетчатыми оболочками;
- организация обмена данными с другими приложениями Windows посредством механизма OLE;

– автоматическая запись всех действий, производимых пользователем, в виде набора макрокоманд и т.д.

3ds Max MAXScript – предназначенный для автоматизации рутинных задач, оптимизации использования существующего функционала, создания новых инструментов редактирования и пользовательского интерфейса. Через скриптовый API позволяет контролировать и модифицировать объекты трёхмерной сцены: геометрию, текстуры, анимацию и другое. Возможно создание различных плагинов и утилит для ускорения выполнения специфических задач. Инструменты Autodesk 3ds Max используют MAXScript во множестве функций, включая оригинальные утилиты и элементы интерфейса.

Преимущества использования MAXScript:

– можно настроить работу программы по своему усмотрению;

– использовать как видеомагнитофон, предназначенный для записи и дальнейшего воспроизведения определенной последовательности действий;

– позволяет пользователю «обсудить» с программой ту или иную сцену и в точности объяснить 3ds Max, что должно произойти. Для этого достаточно либо наглядно продемонстрировать операцию, либо ввести несколько инструкций, которые программа должна выполнить;

– простота использования. С помощью встроенного языка сценариев MAXScript можно управлять практически любой частью программы 3ds Max;

– некоторые из кнопок и разворачивающихся панелей являются частью встроенных сценариев;

– создавать с помощью MAXScript новую функцию и интегрировать в 3ds Max, а затем работать с ней, как с любым другим средством.

Недостатки:

– программа выполняется более медленно, нежели чем аналогичная программа на C++.

В работе представлен в виде панели пользователя “Свиток для быстрого доступа к выбору вывода и сохранения анимации” разработанный, с помощью макро-языка MAXScript. Созданный программный продукт является примером автоматизации доступа к операциям, которые часто используются и позволяют упростить последующую обработку данных.

Для реализации автоматизации 3d моделирования решены следующие задачи:

– создание простых примитивов;

– модификация моделей;

– использование модификаторов;

– использование материалов;
– реализация интерфейса.

В результате решения поставленных задач разработано программное приложение RENDER (рис. 1) для быстрого доступа к выбору вывода и сохранения анимации, с помощью которого можно автоматически осуществить следующие функции:

– выбор расширения (avi, jpg, bmp, png);
– выбор выбора размера видео или картинки(1920x1080, 1280x720, 480x270, 320x180);
– выбор стартового и конечного фрейма;
– выбор имени для конечного файла.

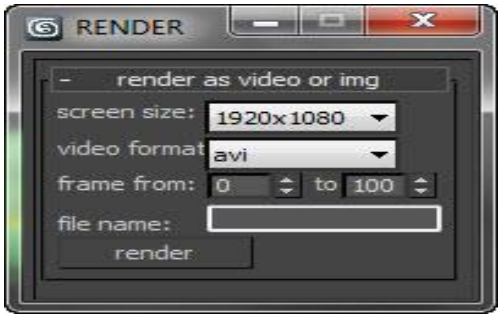


Рис. 13. Свиток для быстрого доступа к выбору вывода и сохранения анимации

В работе проведен анализ программных продуктов, позволяющих создавать трёхмерную графику. Рассмотрены доступные методы автоматизации моделирования, а именно, встроенные языки сценариев, на примере MAXScript, который интегрирован в программный пакет моделирования 3dsMax. Для достижения наиболее высокой эффективности по сравнению с обычным пошаговым 3d моделированием, без помощи скриптового языка, разработан “Свиток для быстрого доступа к выбору вывода и сохранения анимации”, который позволяет упростить последующую обработку данных.

Литература

1. Применение в различных сферах. – [Электронный ресурс]. – Режим доступа к статье: http://ru.wikipedia.org/wiki. **2. Сложность** создания 3-d объектов – [Электронный ресурс]. – Режим доступа к статье: http://www.news3d.biz/3d-modelirovanie-osobennosti-i-slozhnosti-postroeniya-texnicheskix-obektov/. **3. Описание** C4D, программирование в C4D – [Электронный ресурс]. – Режим доступа к статье: http://ru.wikipedia.org/wiki/Cinema_4D. **4. Описание** Maya – [Электронный ресурс]. – Режим доступа к статье: http://ru.wikipedia.org/wiki/Autodesk_Maya. **5. Основные** принципы языка MEL – [Электронный ресурс]. – Режим доступа к статье: http://www.render.ru/books/show_book.php?book_id=582. **6. Описание** ПП

Blender – [Электронный ресурс]. – Режим доступа к статье: http://hightech.in.ua/programs/3d-modeling. **7. Описание** 3ds Max – [Электронный ресурс]. – Режим доступа к статье: http://ru.wikipedia.org/wiki/3ds_max. **8. Язык** MAXScript Max – [Электронный ресурс]. – Режим доступа к статье: http://ru.wikipedia.org/wiki/MAXScript.

Зубков Е.В. Использование скриптовых языков для автоматизации построения 3D моделей

В статье проведен анализ программных продуктов позволяющих создавать трёхмерную графику. Рассмотрены доступные методы автоматизации моделирования, а именно встроенные языки сценариев, на примере MAXScript. Разработан свиток для быстрого доступа к выбору вывода и сохранения анимации при 3D - моделировании.

Ключевые слова: MAXScript, язык сценариев, 3D модель, автоматизация.

Зубков Є.В. Використання скриптовых мов для автоматизації побудови 3d моделей

У статті проведено аналіз програмних продуктів які дозволяють створювати тривимірну графіку. Розглянуто доступні методи автоматизації моделювання, а саме вбудовані мови сценаріїв, на прикладі MAXScript. Розроблено сувій для швидкого доступу до вибору виведення і збереження анімації при 3D - моделюванні.

Ключові слова: MAXScript, мова сценаріїв, 3D модель, автоматизація.

Zubkov E.V. Use of scripting languages to automate the construction of 3d models

The analysis of software products allow you to create three-dimensional graphics. Examined the available automated methods of modeling, and the built-in scripting languages for example MAXScript. Scroll designed for quick access to the selection of the output and save animations in 3D - modeling.

Keywords: MAXScript, the scripting language, 3D model, automation.

УДК 004.358

Я.С. Іщенко

ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОТОКОЛІВ ШИФРУВАННЯ ДАНИХ ДЛЯ ДОСТУПУ ДО СЕРВЕРА

Постанова проблеми: захист даних в мережах досить важке завдання. IP мережі сприйнятливі до великої кількості загроз, наприклад, отримання доступу облудним шляхом (т.зв. spoofing), втрата секретності, втрата цілісності даних, контроль з'єднання і відмова від обслуговування (denial-of-service). Тому потрібно відповідально підійти до проблеми обрання протоколу шифрування даних.

Формулювання мети дослідження: провести аналіз протоколів шифрування даних, виявити їх можливості та недоліки, порівняти їх функціональні властивості. На основі проведеного аналізу скласти таблицю з відображенням слабких та сильних сторін того чи іншого протоколу.

Мета засобів що забезпечують безпеку даних у мережі – це конфіденційність переданих даних, їх цілісність і аутентифікація.

Конфіденційність передбачає захист даних від несанкціонованого прослуховування зловмисником. Найбільш поширеним способом такого захисту є шифрування. Реалізація алгоритмів шифрування досить проста, незважаючи на всю складність математичних алгоритмів, які вони використовують. Труднощі виникають тільки в управлінні ключами захисту. Використання шифрування зменшує продуктивність системи. Рішенням проблеми може стати використання апаратно реалізованих засобів шифрування, тим самим ми звільняємо захисний пристрій від додаткового навантаження.

Цілісність передбачає ідентичність відправлених та отриманих даних. Для вирішення такого роду завдань використовується алгоритми хешування. На основі вихідного повідомлення обчислюється значення хеш-функції, приймальна сторона виробляє ті самі операції і порівнює отримане і розраховане значення. На підставі цього робиться висновок про цілісність отриманого повідомлення.

Механізм аутентифікації дозволяє визначити, достовірність того, що людина, що спілкується з вами дійсно той, за кого себе видає.

Всі ці властивості повинні бути притаманні будь-якого протоколу безпеки, але вони можуть бути реалізовані в кожному з протоколів в різній формі.

На даний момент найбільш надійними вважаються протоколи IPSec (IP Security) і SSL (Secure Socket Layer). Кожна з технологій має як недоліки, так і переваги.

IP Security

IPSec – це набір протоколів, які вирішують проблеми по шифруванню даних, їх цілісності та аутентифікації. IPSec працює на

мережевому рівні. Таким чином, захист даних буде прозорим для мережевих додатків.

Для протоколу IPSec було розроблено два види заголовків аутентифікаційний заголовок (Authentication Header) та заголовок безпечного приховування даних (Encapsulating Security Payload). Кожен із заголовків покликаний вирішувати завдання пов'язані з безпекою переданих даних.

Аутентифікаційний заголовок (AH) покликаний забезпечувати цілісність даних, аутентифікацію та захист від відтворення. Він не забезпечує таємності – передані дані не зашифровані. Він розташовується між основним заголовком IP-пакета і полем даних.

Для шифрування переданих даних використовується *протокол ESP*. Головним його завданням є забезпечення конфіденційності даних, з цією метою можуть застосовуватися різні алгоритми шифрування TCP-пакета. Після шифрування до пакету дописується заголовок ESP, який містить інформацію необхідну для розшифрування. Може також використовуватися з додатковим полем аутентифікації, що забезпечує встановлення автентичності.

Для роботи цих протоколів необхідна інфраструктура, яка б займалася узгодженням алгоритмів шифрування і характеристик ключів, розподілом ключів між спілкуючимися сторонами. Для цих цілей була розроблена група протоколів Internet Key Exchange (IKE). На неї так само покладені завдання щодо контролю виконання угод.

Встановлення з’єднання IPSec проходить в дві фази: Фаза 1 (ISAKMP SA) та Фаза 2 (IPSec SA).

Фаза 1

В процесі першої передачі повідомлень сторони домовляються про базові алгоритми і методи хешування. У другій здійснюється обмін відкритими ключами Діффі-Хеллмана і випадковими числами, які підписуються приймаючими сторонами і відправляються назад для ідентифікації. На третьому кроці за отриманими підписаних значеннях перевіряється дійсність сторін. Разом для встановлення потрібно 6 циклів обміну повідомленнями (Рис. 1.1).

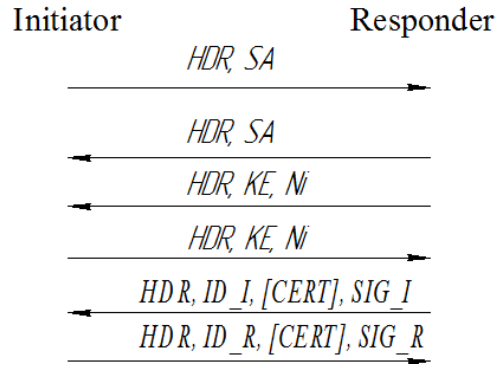


Рис. 1.1. Основний режим обміну повідомленнями

Фаза 2

В ході виконання другої фази відбувається узгодження параметрів SA (Security Association) і генерація нових ключів. У швидкому режимі всі передачі здійснюються по захищеному «тунелю». Його реалізація багато в чому схожа на реалізацію агресивного режиму: один цикл включає в себе передачу трьох пакетів (Рис.1.2).

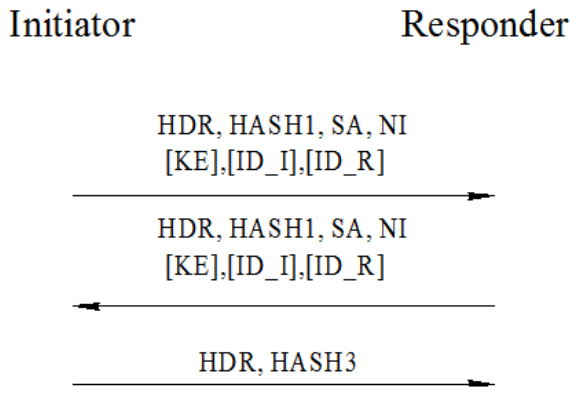


Рис. 1.2. Швидкий режим

Secure Socket Layer

SSL (Secure Socket Layer) протокол рівня додатків, в основному використовується для захищеного обміну інформацією між віддаленими програмами (здебільшого це звернення до Web-серверів).

Перевага протоколу SSL в тому, що він має два рівня: протокол записів (SSL Record Protocol) і протокол діалогу (SSL Handshake Protocol). В якості несучого середовища може вибиратися протокол TCP, який спільно з SSL Record Protocol, утворює так зване ядро SSL, поверх якого можуть накладатися інші протоколи.

SSL Handshake Protocol використовується для аутентифікації сторін, а так само для узгодження певного алгоритму шифрування і обміну ключами.

Однією з переваг є його незалежність від програм і платформ, на яких він використовується.

SSL набуває чинності, коли користувач намагається зайти на адресу яка починається з https. Типово SSL використовує 443 порт.

Схема аутентифікації:

На початку спілкування клієнт посилає повідомлення формату "Привіт", на яке сервер відповідає аналогічним вітальним повідомленням. Ці повідомлення містять дані для відкриття секретного каналу (версія протоколу, ідентифікатор сесії, спосіб шифрування, метод компресії і згенероване число). Далі сервер відсилає свій сертифікат. І після

остаточного уточнення всіх параметрів сторони можуть перейти до передачі зашифрованої інформації. Цей процес може зайняти досить тривалий час.

Технічне порівняння.

Архітектура

Основна відмінність полягає в тому, що IPSec – це протокол мережевого рівня, в той час як SSL – це протокол рівня додатків. IPSec однаково поводиться з пакетами протоколів більш високого рівня, тобто автентифіковані і шифруються не звертаючи уваги на їх вміст.

Для роботи SSL необхідний надійний транспортний протокол (наприклад, TCP).

Надійність IPSec ще гарантується тим, що інформація про порт, з яким встановлено з'єднання так само недоступна для зловмисника.

IPSec підтримує три види встановлення з'єднання

Шлюз – Шлюз;

Шлюз – Хост;

Хост – Хост.

SSL підтримує тільки з'єднання між двома host- ами або клієнтом і сервером.

Так як IPSec – це протокол мережевого рівня, то його реалізація легко може бути вбудована в ядро системи або як окремий пристрій.

Аутентифікації

У разі IPSec аутентифікація завжди двостороння, а для SSL вона може бути як взаємної, так і односторонньою (або відсутніми взагалі).

IPSec підтримує цифровий підпис та використання Secret Key Algorithm, в той час як SSL підтримує тільки цифровий підпис. Перевага IPSec в тому, що для малих систем, можна замість складного ключа використовувати preshared keys, що помітно спрощує завдання. Методи, які використовуються в SSL, ідеально підходять для встановлення захищеного з'єднання між сервером і клієнтом.

Основна відмінність між способами аутентифікації знову ж полягає в тому, що IPSec функціонує на мережевому рівні. Таким чином, є можливість простежувати адресу одержувача і джерела з тим же успіхом і аутентифікацію більш високих рівнів. SSL на відміну має доступ тільки до інформації транспортного рівня і вище.

У разі SSL ситуація зворотна. SSL підтримує єдине з'єднання за одну сесію. Кожна сесія незалежна, але продуктивність може падати з ростом числа сесій. Для кожного з'єднання шифрувальний ключ унікальний, що підвищує безпеку з'єднання.

Тунельний режим використовується в побудові VPN. Два кінцевих маршрутизатора шифрують дані таким чином, що від зловмисника ховаються не тільки передані дані, але так само джерело і пункт призначення. Декілька користувачів одночасно можуть використовувати один і той же тунель між двома кінцевими точками.

SSL ж не підтримує тунелювання, тобто якщо потрібно встановлювати VPN між двома підмережами, то потрібен деякий "зовнішній" спосіб / протокол тунелювання.

Шифрування.

IPSec та SSL можуть реалізовувати велику кількість криптографічних алгоритмів. Різниця полягає в послідовності виконуваних дій.

IPSec спочатку шифрує дані, а потім додає до них MAC. Якщо в середині транзакції скореговані дані були додані, IPSec перевірив би MAC перед виконання дій з розшифрування.

У разі SSL спочатку створюється MAC для вихідних даних, потім проходить шифрування. Таким чином, спочатку проходить розшифрування, а потім перевірка MAC. Це може призвести до додаткового навантаження процесора, у разі змінених пакетів.

Розмір службової інформації.

Один з недоліків IPSec великий обсяг додаткової інформації додається до вихідного пакету. У разі SSL цей розмір значно менше. У таблиці 1.1 представлені порівняльні характеристики.

Таблиця 1.1. – Розмір пакетів в різних режимах праці

Протокол	Режим	Розмір (в байтах)
IPSec Tunnel Mode	ESP	32
	ESP та AH	44
IPSec Transport Mode	ESP	36
	ESP та AH	48
SSL	HMAC-MD5	21
	HMAC-SHA-1	25

Використання алгоритмів стиснення.

У разі IPSec використання алгоритмів стиснення може призводити до різних результатів при використанні їх в різних умовах: продуктивність може як збільшуватися, так і зменшуватися. Результат залежить від співвідношення швидкостей шифрування, стиснення і зростання швидкості передачі даних. Більшість алгоритмів шифрування працюють швидше алгоритмів стиснення. Отже, це буде призводити до уповільнення роботи. Але у випадку низької швидкості передачі, використання стиснення помітно збільшить продуктивність.

Висновки. Ми проаналізували спільні та відмінні риси між протоколами IPSec та SSL. Кожен з них має унікальні властивості і вибір якогось із них має ґрунтуватися на вимогах, що ставлять до безпеки з'єднання. В результаті порівняльного аналізу склали таблицю характеристик яка покликана полегшити обрання протоколу відповідно до потреб користувача.

Таблиця 1.2. – Основні функціональні характеристики

Функція	**IPSec**	**SSL**
Конфігурація	складна	середня
Аутентифікація клієнта	обов'язково	вибірково
Попередній загальний ключ	так	ні
Проблема сумістності	так	ні
TCP підтримка застосунків	так	вибірково
UDP підтримка	так	ні
Пропускна властивість	висока	висока
Підтримка стискання	так	тільки в Open SSL
Час «Потискання рук»	довго	швидко

Основна перевага IPSec є те, що він працює на більш низькому рівні моделі OSI, що дозволяє йому вирішувати більш складні завдання. І поки SSL вдається на рівних конкурувати з IPSec.

Література

1. Шаньгин В.Ф. «Защита информации в компьютерных системах и сетях». **2. Alan O.** Freier, Philip Karlton, Paul C. Kocher « The SSL Protocol, Version 3.0. Internet Draft» **3. AbdelNasir** Alshamsi, Takamichi « A Technical Comparison of IPSec and SSL» **4. Павел** Иванов «Сети» Режим електронного доступу: http://www.osp.ru/text/302/140914.html **5. «Протоколы** защищенных каналов. часть 3. SSL» Режим електронного доступу: http://www.nestor.minsk.by/sr/2003/06/30613.html

Іщенко Я.С. Порівняльний аналіз протоколів шифрування даних для доступу до сервера

Стаття містить порівняльний аналіз протоколів шифрування даних для доступу до сервера

Ключові слова: протоколи шифрування даних, сервер

Ищенко Я.С. Сравнительный анализ протоколов шифрования данных для доступа к серверу

Статья содержит сравнительный анализ протоколов шифрования данных для доступа к серверу

Ключевые слова: протоколы шифрования данных, сервер

Ishchenko Y.S. Comparative analysis of data encryption for a service

This article contains a comparative analysis of data encryption for a service

Keywords: data encryption, server

УДК 681:004.4

В.А. Калиниченко

ПРОИЗВОДИТЕЛЬНОСТЬ HTML5 CANVAS

Многие разработчики используют HTML5 CANVAS в широком круге мультимедиа проектов, визуализаций и игр. С ростом сложности приложений, разработчики наталкиваются на проблему производительности.

Цель работы - изучить как фундаментальные оптимизации, которые относятся ко всем областям компьютерной графики, так и конкретные техники для CANVAS, которые меняются по мере развития реализаций CANVAS.

Основная часть. В современном мире HTML5 CANVAS JSPerf помогает проверить, работают ли сейчас все предложенные оптимизации. JSPerf – это веб-приложение, которое позволяет разработчикам писать тесты производительности JavaScript. Каждый тест сфокусирован на результате, который пользователь хочет получить (например, очистка холста) и включает различные подходы. JSPerf запускает каждый вариант как можно больше в течении короткого периода времени и отображает статистически осмысленное число итераций в секунду.

Отрисовывая похожие примитивы на экран на протяжении многих кадров (как это часто бывает при написании игры), можно получить значительные выигрыши в производительности, отрисовывая крупные части вне сцены. Предварительная отрисовка подразумевает использование виртуального (или виртуальных) холстов, на которых отрисовываются временные изображения, а потом копирование виртуальных холстов на видимый.

Эта техника особенно эффективна, когда отрисовка сложная. Хороший пример – отрисовка текста, которая является очень дорогой операцией. Вот как драматически увеличивается производительность при использовании pre-rendering текста.

Так как отрисовка – «дорогая» операция, гораздо эффективнее загружать drawing state machine длинными списками команд, а потом выгружать их в видео буффер.

Например, при отрисовке множества линий, гораздо лучше сделать один путь со всеми линиями и нарисовать его в один вызов. Другими словами, чем рисовать отдельные линии, лучше нарисовать одну ломаную. У CANVAS есть важное исключение из этого правила: если у примитивов отрисовываемого объекта небольшие окружающие прямоугольники – может оказаться, что эффективнее рисовать их отдельно [1].

CANVAS реализован на базе конечного автомата, который отслеживает такие вещи, как стили заполнения и штриха, как и

предыдущие точки, образовывающие текущий путь. При попытках оптимизации, возникает мысль сфокусироваться на рисовании. Но манипуляции с конечным автоматом также приводят к издержкам.

Используя несколько цветов заливки в сцене, выгоднее рисовать по цвету, чем по расположению на холсте. Для отрисовки текстуры в мелкую полоску вы можете нарисовать линию, сменить цвет, нарисовать следующую и так далее. Или же отрисовать все четные и нечетные полосы.

Получается, чем меньшую часть экрана отрисовывать, тем это выгоднее. Если у вас только незначительные различия между перерисовками, вы можете получить значительный рост производительности, отрисовывая только разницу.

Отрисовка больших изображений обходится не выгодно и ее стоит избегать. В дополнение к использованию внеэкранного буффера (секция предварительной отрисовки), можно использовать холсты, наложенные друг на друга. Используя прозрачность верхнего слоя, можно положиться на GPU для применения альфа-канала во время отрисовки.

Преимущество над единственным холстом в том, что, при отрисовке или очистке верхнего, мы не затрагиваем фон. Если игра или мультимедиа-приложение может быть разбито на 2 слоя, лучше отрисовывать их в разных холстах, чтобы получить значительный прирост производительности [2].

HTML5 CANVAS поддерживает суб-пиксельный рендеринг и нет никакой возможности его отключить. Если пользователь рисует с нецелыми координатами, он автоматически использует анти-алиасинг, чтобы сгладить линии. Вот визуальный эффект суб-пиксельной производительности из статьи Seb Lee-Delisle:

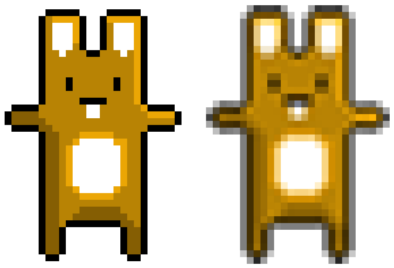


Рис. 1. Визуальный эффект суб-пиксельной производительности.

Если сглаженный спрайт – это не то, что вам нужно, намного быстрее будет переводить ваши координаты, используя Math.floor или Math.round.

Этот способ оптимизации больше не будет иметь смысла с того момента, как реализации CANVAS станут GPU-ускорены, что позволит быстро отрисовывать нецелые координаты [3].

Выводы. Таким образом, в статье рассмотрены основные приемы оптимизаций, которые могут помочь разрабатывать производительные HTML5 CANVAS-приложения.

Литература

1. Лоусон Б., Шарп Р. Изучаем HTML5. Библиотека специалиста. - СПб.: Питер, 2011. – с. 272. **2. Хольцнер С.** HTML5 за 10 минут, 5-е изд.: Пер. с англ. – М.: ООО «Вильямс», 2011. – с. 240. **3. Русаков М.** Учебник по HTML5. - СПб.: БХВ-Петербург, 2011. – 160с.

Калиниченко В.А. Производительность HTML5 CANVAS.

В статье представлена работа по изучению приемов оптимизации для повышения производительности приложений на HTML5 CANVAS.

Ключевые слова: HTML5, CANVAS, производительность.

Калініченко В.О. Продуктивність HTML5 CANVAS.

У статті представлена робота з вивчення прийомів оптимізації для підвищення продуктивності додатків на HTML5 Canvas.

Ключові слова: HTML5, CANVAS, продуктивність

Kalinichenko V.A. Productivity of HTML5 CANVAS

The article presented is work on the optimization of study for Improving productivity applications on HTML5 CANVAS.

Keywords: HTML5, CANVAS, productivity.

УДК 004.3'12

Е.А. Калиниченко

ДАТЧИК ОБРЫВА НИТИ НА ОСНОВЕ ЛИНЕЙНОЙ ПЗС МАТРИЦЫ

Одной из основных задач на ткацком производстве является слежение за обрывом нити, идущей на станок. Существует достаточно большое количество механических датчиков обрыва нити, однако, компьютеризация производства требует наличие электронных датчиков обрыва нити.

Приборы с зарядовой связью (ПЗС), как и транзисторы, обладают свойством универсальности, позволяющим использовать их в самых разнообразных устройствах. Они применяются в цифровых запоминающих устройствах большой информационной емкости. В оптоэлектронных приемниках изображений на основе ПЗС создают формирователи видеосигналов. В радиотехнических системах обработки информации ПЗС используют при разработке линий задержки, фильтров

различных типов, устройств спектрального анализа и обработки радиолокационных сигналов.

Исходя из всего вышесказанного, можно сделать вывод, что данная тема является **актуальной** и устройства на основе ПЗС матрицы могут служить для основы проектирования датчика обрыва нити.

Цель работы – изучить устройства сканирующих головок, принцип работы ПЗС – матрицы, спроектировать принципиальную схему датчика обрыва нити на основе ПЗС – матрицы.

Для достижения данной цели были выполнены следующие **задачи**:

- подбор, изучение и анализ литературы по данной проблеме;
- проанализированы существующие виды сканеров и сканирующих головок;
- изучен принцип работы ПЗС - матрицы;
- спроектировано принципиальную схему датчика обрыва нити на основе ПЗС – матрицы.

В качестве управляющей платы была выбрана отладочная плата STM32L-Discovery. Она содержит контроллер STM32L на ядре Cortex-M3. Эта плата – полноценный инструментарий, включающий в себя оценочную плату, программатор и отладчик с поддержкой самых популярных программных средств разработки от таких фирм как IAR, Keil и Atollic.

В основе STM32L-Discovery (рис. 1.) заложен 32МГц контроллер STM32F103C6T6 с 128 КБ Flash, 20 КБ RAM и 4 КБ EEPROM. Наличие встроенного программатора–отладчика ST-LINK, сигналы которого выведены на внешний разъем, позволяет начать разработку собственных приложений без дополнительных затрат.

Рис. 1. Платы STM32L-Discovery

STM32L-Discovery – плата, на которой установленный контроллер STM32F103C6T6, содержит для поставленной задачи

проектирования датчика обрыва нити все необходимое. Блок – схема контроллера представлена на рис. 2.

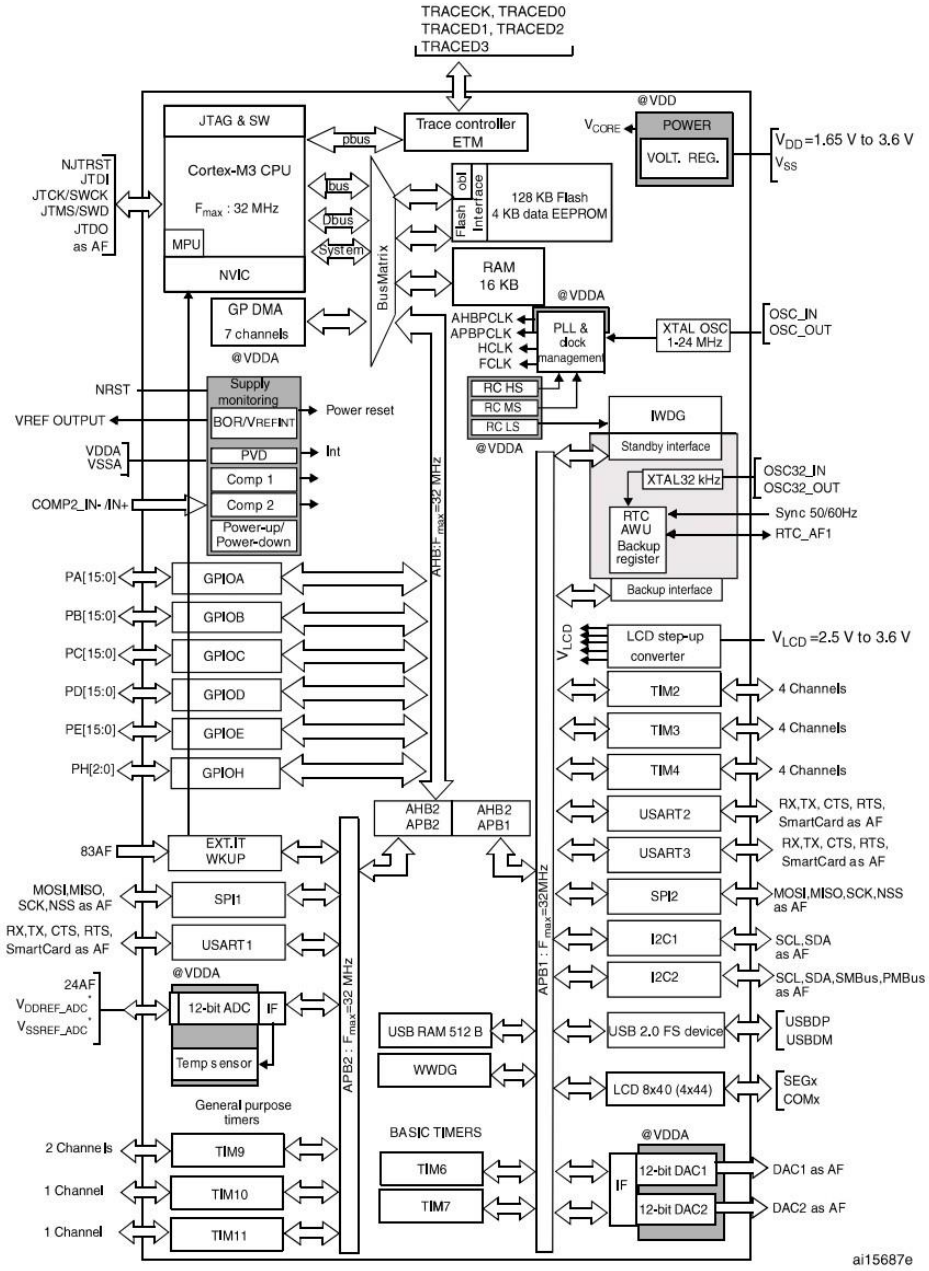


Рис.2. Блок-схема контроллера STM32L152RBT6

ПЗС – матрица является управляемым элементом, блок – схема которой представлена на рис. 3.

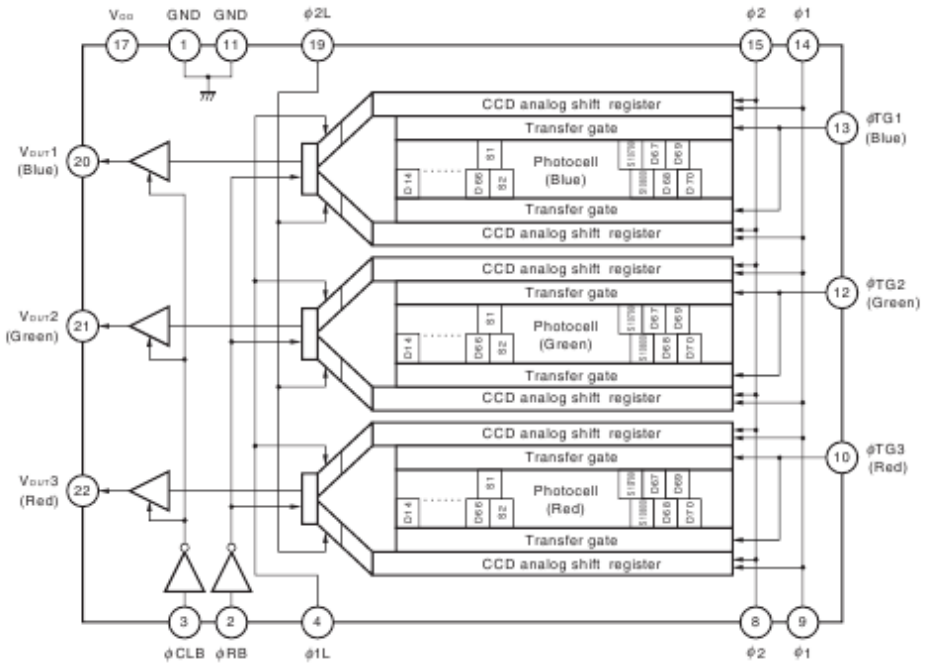


Рис. 3. Блок-схема ПЗС-матрицы

Изучив характеристики матрицы, необходимо «прозвонить» прибором контакты выходящие по шлейфу Составляется порядок входов-выходов шлейфа. Разъем соединения (1-22):

2 – выход синий;

4 – выход зеленый;

6 – выход красный;

8 – управляющий сигнал (ячейки);

14 – управляющий сигнал (строка);

10, 12 – сдвиговый регистр;

13, 18 – питание, +5В;

22 – питание, +12В;

1, 3, 5, …– «земля» (остальные).

Для включения матрицы необходимо подать +5В и +12В. +5В идут для логической части платы установленной на зеркальном блоке, а +12В идут для питания родного (правильного) освещения, лампа которого подбиралась на заводе изготовителе сканера.

Для запуска процесса чтения данных с ячеек необходимо подать:

1. Частоту 1MHz по шлейфу на RB.
2. «1» на CLB для запуска чтения строки.
3. «0» на TG1-3 для выключения обнуления.

При достижении конца строки по CLB передаем «0», на TG1-3 подать «1» для обнуления ячеек в строке, и повторяем цикл чтения.

Основной **проблемой** поставленной задачи является понимание (отличие) выходящих сигналов. То есть, так как информация о принципах работы матрицы является конфиденциальными данными,

необходимо подбирать многие параметры самостоятельно. В описании к матрице есть некоторые параметры, при помощи которых можно начать подсчитывать неизвестные, но необходимые для проектирования данные.

Ниже представлен пример входящих и выходящих сигналов с ПЗС-матрицы (рис. 4).

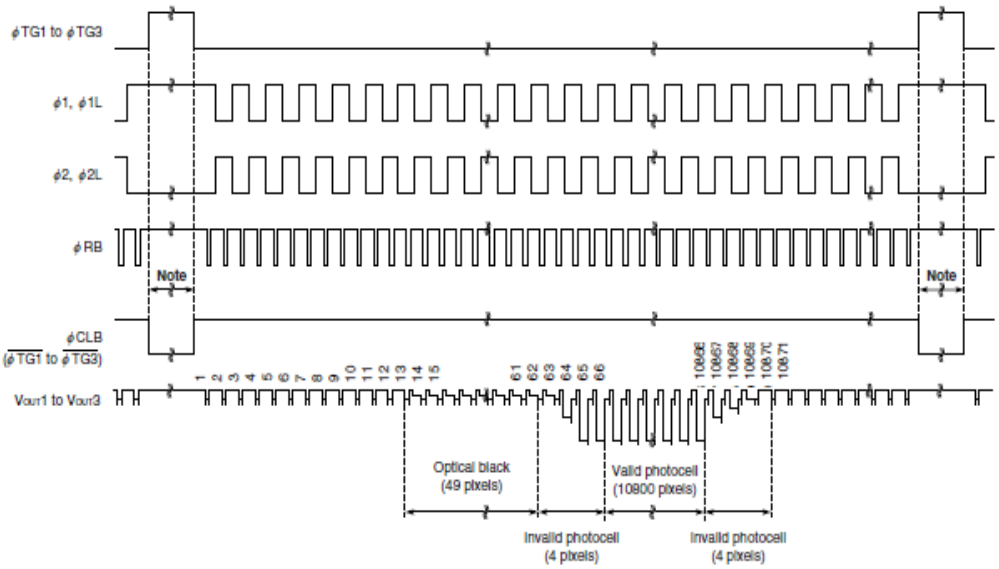


Рис.4. Входящие и выходящие сигналы с ПЗС-матрицы

Рассмотрим основные изи них:

1. RB – счетчик ячеек.
2. CLB – открывающий сигнал на чтение строки.
3. 1, 2 – сдвиговый регистр.
4. TG1 – TG3 – сигнал при помощи которого идет обнуление состояния всех ячеек в строке (сигнал инвертирован строчному).
5. $V_{out}1$ – $V_{out}3$ – выходной сигнал RGB (1,2,3).

Для преобразования выходящего аналогового сигнала в цифровой сигнал необходимо воспользоваться АЦП.

STM32F103C8T6 содержит два 12-битных АЦП, что позволяет с большой точностью преобразовать (оцифровывать) сигнал:

1. Сигнал $V_{out}1$ – синий.
2. Сигнал $V_{out}2$ – зеленый.
3. Сигнал $V_{out}3$ – красный.

Для различия двух цветов (фон и нить) необходимо и достаточно слежение за одним выходящим сигналом. После данного преобразования на выходе получается 2 типа (цвет фона и цвет нити) выходящих данных, которые записаны в исходный массив.

При старте первая строка запоминается как исходная, ее значение записывается в «стартовый» массив. При сканировании каждый считаный пиксель сравнивается с соответствующим элементом по

порядку в массиве, при несоответствии 50px за строку (одна строка 210мм либо 10800px) на выводящее устройство будет выведена информация о возможном сбое в качестве нитей. При несоответствии 100px и более подряд означает об изменении количества нитей (либо нить оборвалась, либо закончилась на катушке).

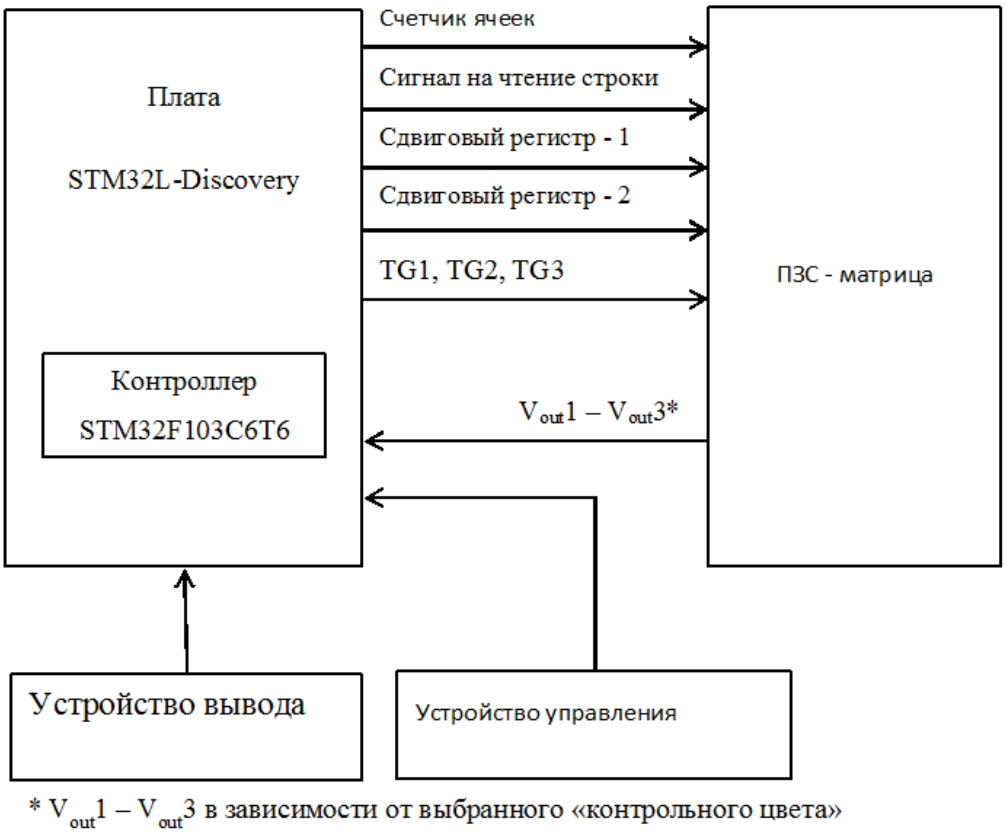


Рис. 5. Принципиальная схема устройства

Данное устройство может применяется для уменьшения брака в производстве ткацких полотен, его применение довольно широко - от плетения тросов и канатов до прядильного станка. Датчик обрыва нити может ставиться как в начале линии производства, так и перед ее концом. Для правильности работы необходимо в качестве фона устанавливать контрастный фон. Это обеспечит более четкое цветовое разграничение между фоном и нитями.

Выводы.На основе описания ПЗС – матрицы изучены ее выходы и определены их назначения. Представлен пример входящих и выходящих сигналов с ПЗС-матрицы. Предложен алгоритм проверки целостности нити. Разработана принципиальная схема датчика обрыва нити на основе ПЗС – матрицы.

Литература

1. Datasheet: STM32F103C6T6 [Электронный ресурс]: - Режим доступа к инструкции: http://www.datasheetdir.com/STM32F103C6T6+ARM. **2. Datasheet**: STM32L-DISCOVERY [Электронный ресурс]: - Режим доступа к инструкции: http://www.st.com/web/catalog/tools/FM116/ SC959/SS1532/PF250990 **3. Степаненко О.С.** Сканеры и сканирование. - СПб.: БХВ – Санкт-Петербург, 2000. – 528 с.

Калиниченко Е.А. Датчик обрыва нити на основе линейной ПЗС матрицы.

В статье представлена работа по проектированию датчика обрыва нити на основе линейной ПЗС матрицы.

Ключевые слова: ПЗС матрица, датчик обрыва нити, плата, контроллер.

Калініченко Є.О. Датчик обриву нитки на основі лінійної ПЗС матриці.

У статті представлена робота з проектування датчика обриву нитки на основі лінійної ПЗС матриці.

Ключові слова: ПЗС матриця, датчик обриву нитки, плата, контролер.

Kalinichenko E.A. The thread breakage sensor based on a linear CCD.

In the paper is describe of work of the design the thread breakage sensor which based on a linear CCD.

Keywords: CCD, sensor a thread breakage, board, controller.

УДК 005.82 : 004

В. С. Ковалева, Т.А. Крамаренко

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ СЕТЕВОГО ПЛАНИРОВАНИЯ И УПРАВЛЕНИЯ ПРОЕКТАМИ НА ПРИМЕРЕ БАЗЫ ДАННЫХ «ПРЕДПРИЯТИЕ ПО АВТОПЕРЕВОЗКИ ПАССАЖИРОВ»

Управление проектами за последнее время завоевало признание как наилучший метод планирования и управления реализацией инвестиционных проектов. По американским оценкам применение методологии управления проектами обеспечивает высокую надежность достижения целей проекта и на 10 – 15 % сокращает затраты на его реализацию.

Основной формой программного управления выступают целевые комплексные программы. Переход на программные методы связан с ликвидацией системы, основанной на планово-распределительных методах управления.

Цель исследования – проанализировать технологию сетевого планирования и управления проектами.

Рассмотрим вопрос сетевого планирования и управления проектами на примере базы данных «Предприятие по автоперевозки пассажиров».

Таблица 1

Таблица задач проекта

Номер работы	Название работы	Длительность
1	Начало реализации проекта	0
2	Постановка задачи	10
3	Разработка интерфейса	5
4	Разработка модулей обработки данных	7
5	Разработка структуры базы данных	6
6	Заполнение базы данных	12
7	Отладка программного комплекса	5
8	Тестирование и исправление ошибок	10
9	Составление программной документации	5
10	Завершение проекта	0

Для достижения поставленной цели необходимо решить следующие задачи:

1. Исследовать методологию управления проектами.
2. Проанализировать этапы сетевого планирования и управления проектами.

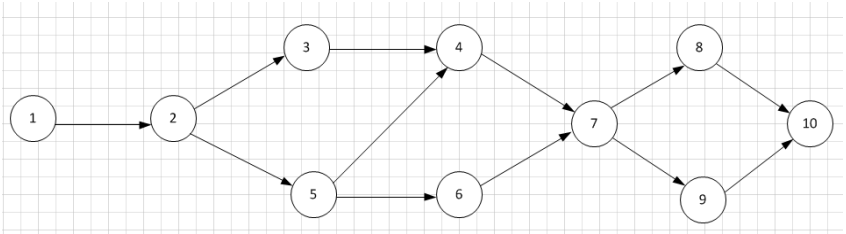


Рис. 1. Сетевой график проекта

Из-за большого объема планируемых действий, сложности методов распределения и оптимизации плана, осуществлять эффективный контроль, проводить качественное планирование без использования специальных программных средств невозможно. Системы управления проектами осуществляют технологию сетевого планирования и управления, составляющими которой являются метод диаграмм Гранта и сетевые методы планирования.

Сетевой график позволяет по заданным значениям длительностей работ найти критические работы проекта и его критический путь.

Нахождение критического пути сводится к нахождению критических работ и выполняется в два этапа:

1. вычисление раннего времени начала каждой работы проекта. Эта величина показывает время, раньше которого работа не может быть начата;

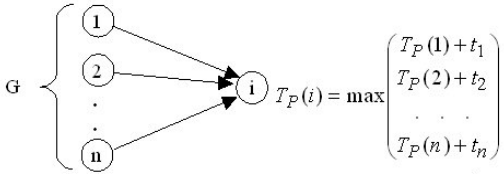


Таблица 2

Расчет раннего времени начала работы

№п/п	предшественник	Раннее время начала работы
1	-	-
2	1	-
3	2	10+0=10
4	3,5	Max(10+5;10+7)=17
5	2	10+0=10
6	5	10+6=16
7	4,6	Max(17+7;16+12)=28
8	7	28+5=33
9	7	28+5=33
10	8,9	Max(33+5;33+10)=43

2. вычисление позднего времени начала каждой работы проекта. Эта величина показывает время, позже которого работа не может быть начата без увеличения продолжительности всего проекта.

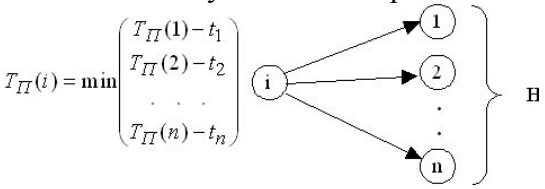


Таблица 3

Расчет позднего времени начала работы:

№п/п	предшественник	Позднее время начала работы
10	-	43
9	10	43-5=38
8	10	43-10=33
7	8,9	Min(33-5;38-5)=28
6	7	28-12=16
5	4,6	Min(21-6;16-6)=10
4	7	28-7=21
3	4	21-5=16
2	3,5	Min(16-10;10-10)=0
1	2	0

Таблица 4

Расчет резерва времени:

№п/п	1	2	3	4	5	6	7	8	9	10
Раннее время начала работы	0	0	10	17	10	16	28	33	33	43
Позднее время начала работы	0	0	16	21	10	16	28	33	38	43
Резерв времени	0	0	6	4	0	0	0	0	5	0

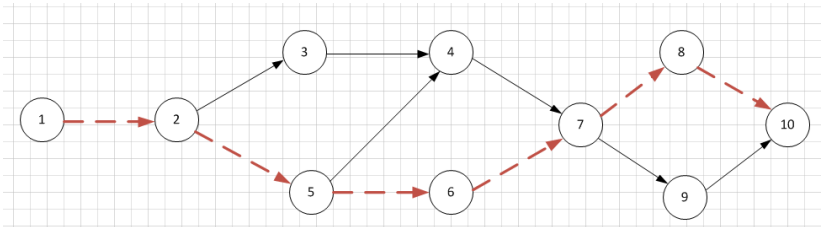


Рис. 2. Сетевой график проекта «Предприятие по автоперевозки пассажиров».

На основе сетевого графика строится календарный график (диаграмма Ганта). Диаграма Ганта представлена на рис. 1, где ромбиками обозначены вехи, сплошными линиями – продолжительность работ, сплошными линиями со стрелками – резерв времени работ, пунктирными линиями – связь между окончанием предшествующих и началом последующих работ.

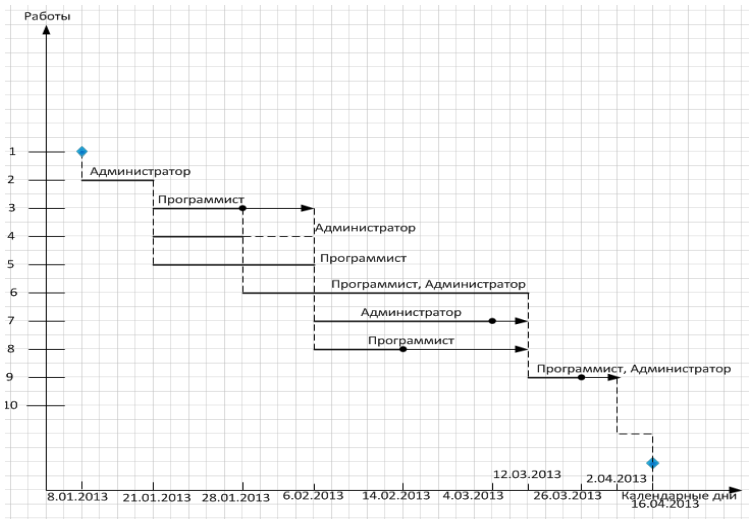


Рис. 3. Диаграмма Ганта

Реализация календарного графика представляет собой этап оперативного управления, который подразумевает контроль хода выполнения работ с внесением корректив в сетевую модель.

На основе полученных данных разрабатываем диаграмму Гранта средствами программы MsProject.

	Режим задачи	Название задачи	Длительно	Пред	Начало	Окончание
1		Начало реализации проекта	0 дней		Вт 08.01.13	Вт 08.01.13
2		Постановка задачи	10 дней	1	Вт 08.01.13	Пн 21.01.13
3		Разработка интерфейса	5 дней	2	Пн 21.01.13	Пт 25.01.13
4		Разработка модулей обработки данных	7 дней	3;5	Пн 28.01.13	Вт 05.02.13
5		Разработка структуры базы данных	6 дней	2	Ср 06.02.13	Ср 13.02.13
6		Заполнение базы данных	12 дней	5	Чт 14.02.13	Пт 01.03.13
7		Отладка программного комплекса	5 дней	4;6	Пн 04.03.13	Пт 08.03.13
8		Тестирование и исправление ошибок	10 дней	7	Вт 12.03.13	Пн 25.03.13
9		Составление программной документации	5 дней	7	Вт 26.03.13	Пн 01.04.13
10		Завершение проекта	0 дней	8;9	Вт 02.04.13	Вт 02.04.13

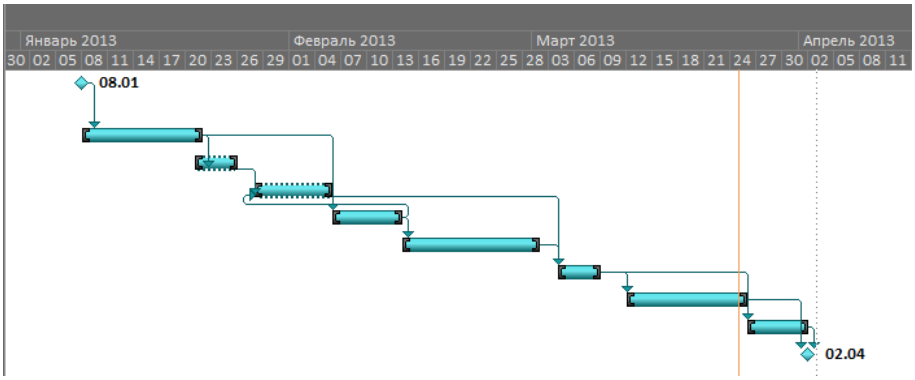


Рис. 4. Диаграмма Гранта в MsProject.

Таким образом, применение технологий управления проектами способствует своевременной реализации проектов в рамках выделенных

бюджетов и с требуемым качеством. В дальнейших исследований необходимо проанализировать программные средства, реализующие сетевые методы планирования.

Литература

1. Гультяев А. К. Microsoft Office Project Professional 2007. Управление проектами / А. К. Гультяев. – М. : Корона-Век, 2008. – 480 с.
2. Либерзон В. И. Основы управления проектами / И. В. Либерзон. – М. : Нефтяник, 1997. – 150 с.

Ковальова В.С., Крамаренко Т.А. Застосування технології мережевого планування та управління проектами на прикладі бази даних «Підприємство по автоперевезення пасажирів»

Стаття містить аналіз технології мережевого планування та управління проектами

Ключові слова: технології мережевого планування, управління проектами

Ковалева В.С., Крамаренко Т.А. Применение технологии сетевого планирования и управления проектами на примере базы данных «Предприятие по автоперевозки пассажиров»

Статья содержит анализ технологии сетевого планирования и управления проектами

Ключевые слова: технологии сетевого планирования, управления проектами

Kovalev V.S., Kramarenko T.A. Application of technology of network planning and project management for the sample database "Enterprise for trucking passengers"

The article contains the analysis of the technology network planning and project management

Keywords: technology network planning, project management

УДК 004.358

Ю.Ю. Кожемякіна

ДОСЛІДЖЕННЯ АЛГОРИТМІВ ОПТИМІЗАЦІЇ ОБ'ЄКТІВ З ЦІЛЬОВОЮ ФУНКЦІЄЮ ЯРНОГО ТИПУ

Проблема вибору оптимального варіанту рішення відноситься до числа найбільш актуальних проблем. У життєдіяльності та на виробництві поняття оптимізації виражається кількісними критеріями: мінімум витрат, мінімум часу, максимум прибутку тощо. Проведення аналізу методів оптимізації дозволяє виявити найбільш ефективні

способи вирішення окремої задачі, і тому обрана тема є актуальною. Класифікацію задач оптимізації наведено на рисунку 1 [1].

Метою роботи є дослідження оптимізації для об’єктів, які мають цільову функцію ярного типу. Функції ярного типу є функціями явно не лінійними, тому дослідження таких функцій проводиться засобами нелінійного програмування.

Поняття "рельєф функції" зручно розглядати на прикладі функції двох змінних *z = F(x,y)* [2]. Ця функція описує деяку поверхню в тривимірному просторі з координатами *x,y,z*. Задача *F(x,y)→min* означає пошук нижчої точки цієї поверхні. На рисунку 2 приведено геометричну ілюстрацію довільної нелінійної функції ярного типу, що має как максимум так і мінімум.

Рельєф функції $F(x,y) = 10\,(y - \sin x)^2 + 0{,}1x^2$ (Рис. 3) має яскраво виражений звивистий вирішуваний яр, "дно" якого має вигляд синусоїда, а точка мінімуму – початок координат [3].

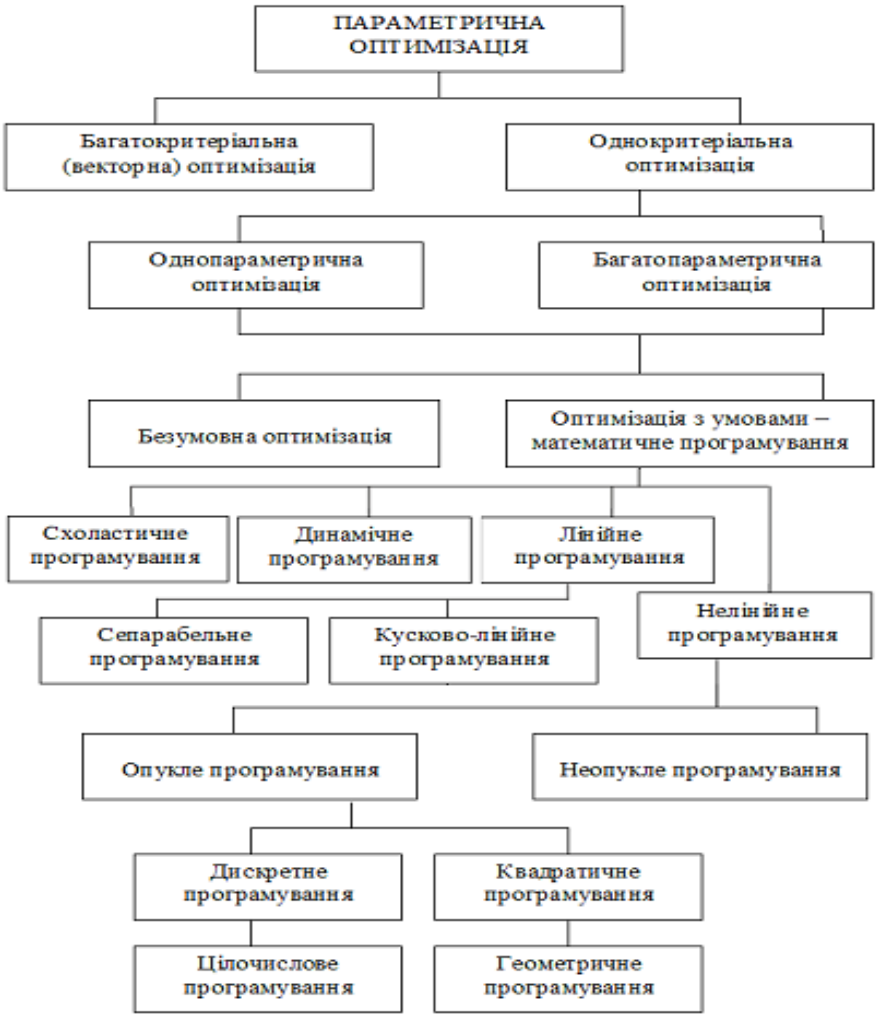


Рис. 1. Класифікація задач оптимізації

Для вирішення поставленої задачі обрано наступний ряд функцій, за допомогою яких визначено найбільш ефективні методи для вирішення об'єктів оптимізації з цільовою функцією ярного типу. Обрано функції ярного типу:

– $F(x,y) = 10\,(y - \sin x)^2 + 0{,}1x^2$,
– $F(x,y) = 4\,(x - 5)^2 + (y - 6)^2$,
– $F(x,y) = x^3 + 2\,y^2 - 3x - 4y$,
– $F(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$.

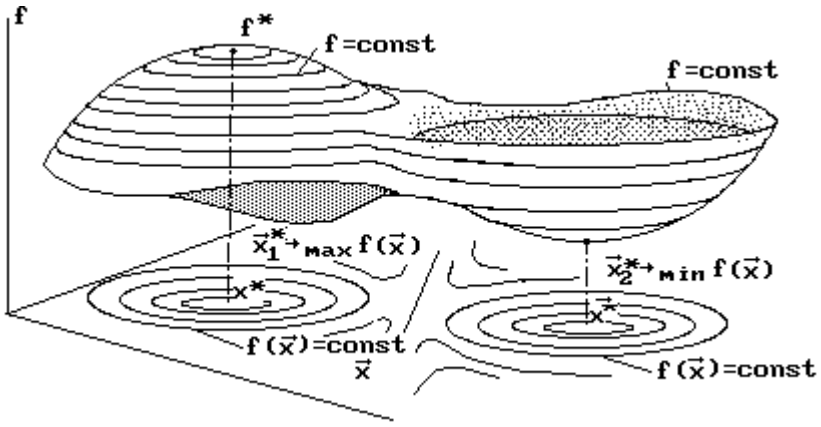


Рис. 2. Цільова функція, що має як мінімум, так і максимум

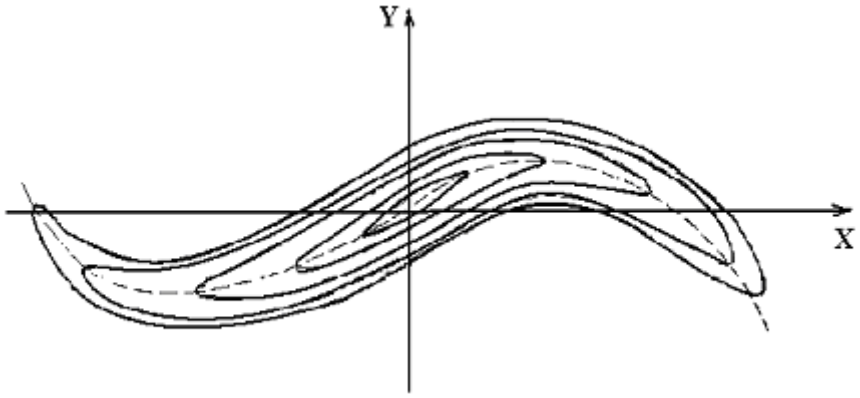


Рис. 3. Рельєф функції $F(x,y) = 10\,(y - \sin x)^2 + 0{,}1x^2$

Кожну із задач оптимізації вирішує окремий клас методів [4]. У роботі використані наступні методи:

1. Метод Хукка-Дживса,
2. Метод Розенброка,
3. Метод Пауела,
4. Метод покоординатного циклічного спуску,
5. Метод найскорішого спуску.

За відомими алгоритмами розроблено програмний продукт, що реалізує кожен з вказаних методів. Особливістю програми є можливість введення ярної функції з екрану в поле для розрахунку цієї функції. Загальний вид програми показано на рисунку 4.

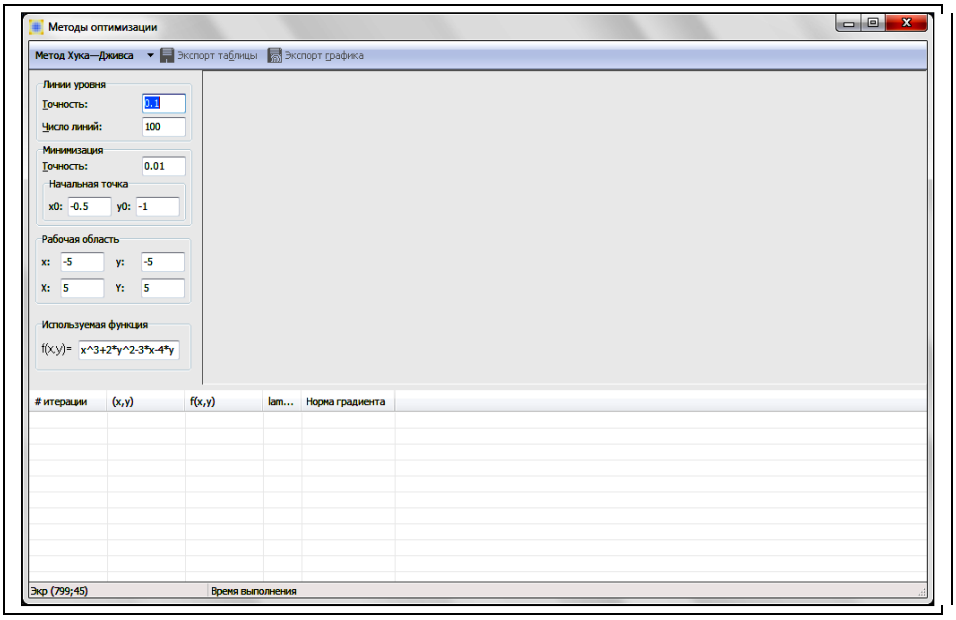


Рис. 4. Інтерфейс програми побудови графіку оптимізації ярної функції

Основне меню призначено для вибору метода оптимізації та збереження результатів роботи програми. В робочій області програмного продукту відображається результат методу в графічному вигляді: проходить розрахунок та прорисовка ліній рівнів ярної функції, за якими будується графік оптимізації введеної функції. Таблиця під графіком називається таблицею ітерацій і містить порядковий номер ітерації, за яку будується крок графіку, значення координат точок графіку ярної функції, значення функції при виконанні кожної ітерації, значення λ і норми градієнта, а також час, за який виконується процес оптимізації об’єкту з цільовою функцією ярного типу. Структуру виконання програми представлено на рисунку 5.

При реалізації кожного метода задаються початкові параметри:

– точність та кількість ліній рівнів,

– точність мінімізації та початкова точка,

– координати робочої області,

– введення ярної функції в поле введення використовуємої функції.

Таким чином программа розраховує кожну функцію ярного виду, що було введено в поле для введення функції, що використовується.

Прикладом функції, яку необхідно ввести в поле для введення фунції, може послужити будь-яка нелінійна функція, що має вид яру. Якщо обрано функцію $F(x,y) = 4(x - 5)^2 + (y - 6)^2$, то формат вводу цієї функції буде наступним: 4*(x-5)^2+(y-6)^2.

За замовченням встановлено початкові координати, які за небхідністю можна змінити (Рис. 6). Користувач має можливість занести дані з таблиці ітерацій в окремий файл для більш зручного їх використання. Файл зберігається у форматі .txt. Існує можливість збереження графіка-малюнку в окремий файл у форматі .bmp. Збереження таблиці ітерацій зображено на рис 7.

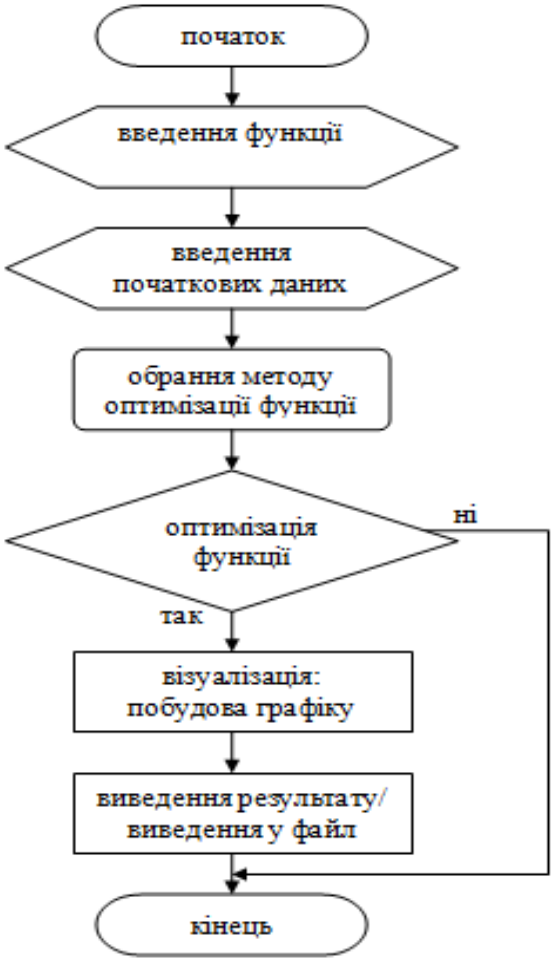


Рис. 5. Схема виконання програми пошуку оптимальних рішень для функцій ярного типу

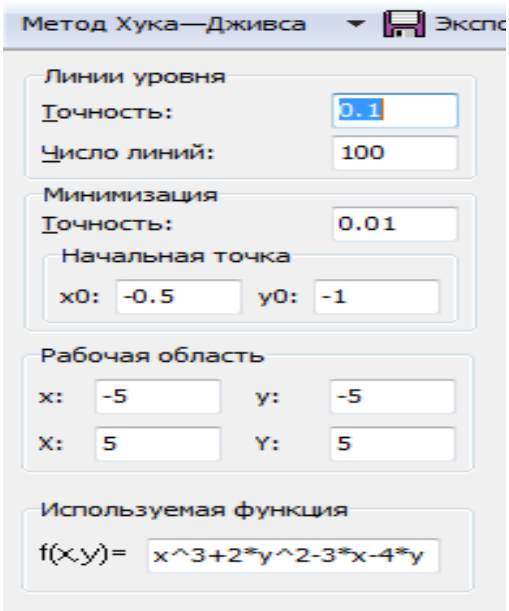


Рис. 6. Початкові координати для оптимізації функції

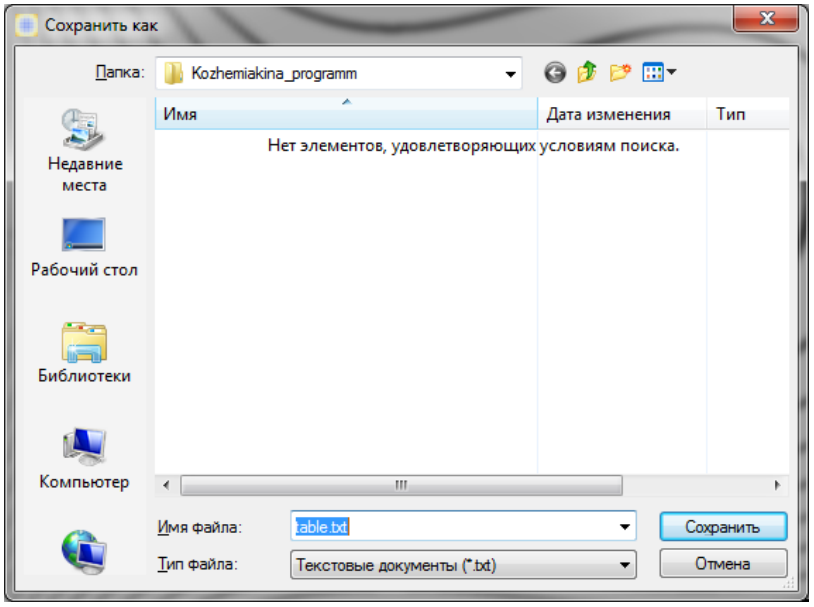


Рис. 7. Збереження результатів таблиці ітерацій

Для аналізу алгоритмів оптимізації вирішено провести статистичну залежність методів для обраних об’єктів з цільовою функцією ярного типу. Демонстрацію вирішення методів оптимізації у програмному вигляді показано на рисунку 8.

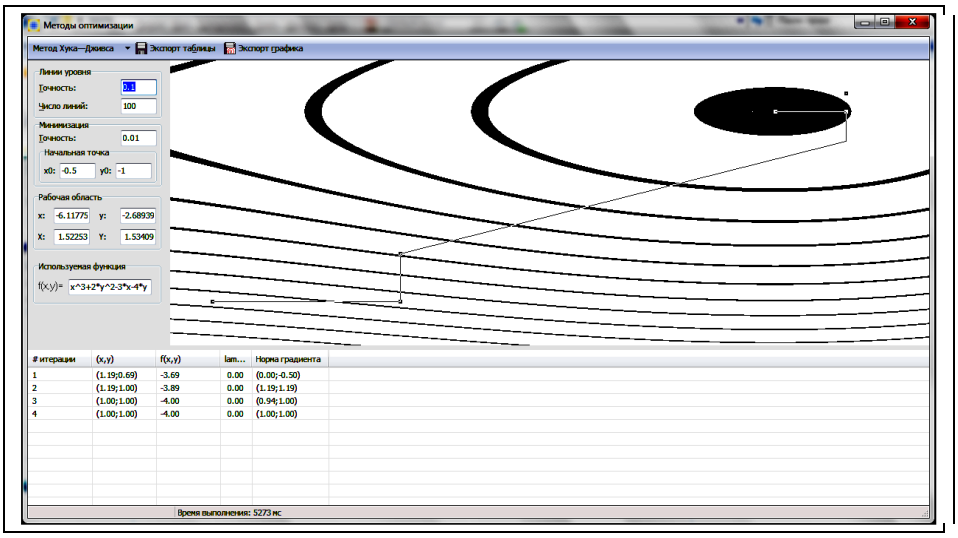


Рис. 8. Вирішення методів оптимізації для нелінійних фунцій

Результати проведених операцій занесено до таблиць. Таблиця 1 відображає результат за кількість ітерацій, таблиця 2 – за використаним часом, вираженому у мілісекундах.

Таблиця 1. Результати дослідження за кількість ітерацій

	Метод				
	Хука-Дживса	Розенб-рока	Покоорд. цикл. спуску	Пауела	Найшв. спуску
$F(x,y) = 10(y - \sin x)^2 + 0{,}1x^2$	21	47	-	6	-
$F(x,y) = 4(x - 5)^2 + (y - 6)^2$	6	1	2	2	10
$F(x,y) = x^3 + 2y^2 - 3x - 4y$	4	1	-	-	5
$F(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$	5	28	4	2	11

Таблиця 2. Результати дослідження за використаним часом, у мс.

	Метод				
	Хука-Дживса	Розенб-рока	Покоорд. цикл. спуску	Пауела	Найшв. спуску
$F(x,y) = 10(y - \sin x)^2 + 0{,}1x^2$	4902	4916	-	4879	-
$F(x,y) = 4(x-5)^2 + (y-6)^2$	4739	4712	4719	4730	4713
$F(x,y) = x^3 + 2y^2 - 3x - 4y$	5156	4819	-	-	4787
$F(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$	4859	4841	4863	4839	4815

Провівши дослідження зроблено висновки:

- Методи Розенброка та Пауела найкращі у вирішенні функцій ярного типу за критеріями часу та кількістю ітерацій.
- Метод Хука-Дживса рахує всі функції, але за великий проміжок часу.
- Методи покоординатного циклічного спуску та найскорішого спуску неточно рахують оптимальні точки функції ярного типу, тому їх краще застосовувати до функцій лінійного типу.

Література

1. Моисеев Н.Н. Методы оптимизации / Моисеев Н.Н., Иванилов Ю.П., Столярова Е.М. – М.: Наука, 1978. – 352 с. **2. Ларичев О.И.** Методы поиска локального экстремума овражных функций / Ларичев О.И., Горвиц Г.Г. – М.: Наука, 1990. – 95 с. **3. Д. Химмельблау.** Прикладное нелинейное программирование / Д. Химмельблау. – М.: Мир, 1975. – 536 с. **4. Милютин А.А.** Оптимальное управление в линейных системах / Милютин А.А., Илютович А.Е., Осмоловский Н.П., Чуканов СВ. — М.: Наука, 1993. – 268 с. **5. Канатиков А.Н.** Дифференциальное исчисление функций многих переменных / Канатиков А.Н., Крищенко А.П., Четвериков В.Н. – М.: Изд–во МГТУ им. Н.Э. Баумана, 2000. – 456 с. (Сер. Математика в техническом университете; Вып. V.) **6. Н. Елманова.** Delphi и технология COM / Елманова Н., Трепалин С., Тенцер А. – СПб., 2003, – 698 с. **7. Комп'ютерне моделювання** систем та процесів // http://posibnyky.vntu.edu.ua/k_m/t2/5..htm

Кожемякіна Ю.Ю. Дослідження алгоритмів оптимізації об'єктів з цільовою функцією ярного типу.

Стаття присвячена дослідженню алгоритмів оптимізації об’єктів з цільовою функцією ярного типу. Шляхом аналізу виявлено найбільш ефективні методи оптимізації, за якими проведено розрахунок нелінійних функцій.

Ключові слова: методи оптимізації, характеристика, метод хука-дживса, метод розенброка, метод покоординатного циклічного спуску, метод найскорішого спуску, метод пауела, програмний продукт.

Кожемякина Ю.Ю. Исследование объектов оптимизации с целевой функцией овражного типа.

Статья посвящена исследованию алгоритмов оптимизации объектов с целевой функцией овражного типа. Путем анализа обнаружено наиболее эффективные методы оптимизации, по которым проведен расчет нелинейных функций.

Ключевые слова: методы оптимизации, характеристика, метод хука-Дживса, метод Розенброка, метод покоординатного циклического спуска, метод скорейшего спуска, метод Пауэлла, программный продукт.

Kozhemiakina I. The research of facilities optimization with the objective function ravine-type.

The article is devoted to the development of algorithms for optimizing facilities with the objective function of the ravine type. By the analysis was found the most effective optimization techniques, along which the calculation of non-linear functions.

Keywords: optimization techniques, characteristic, Method hook-Jeeves, Rosenbrock method, technique-wise cyclic descent, the method of steepest descent, Powell's method, software product.

УДК 004.45

А.А. Лавренченко, В.В. Винник

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ МОНИТОРИНГА ПРОЦЕССОВ ОС

В наше время существует множество приложений для разных операционных систем, но неотъемлемой частью каждой операционной системы является программа контроля и мониторинга процессов этой операционной системы.

В данный момент существует множество операционных систем, которые используются как суперкомпьютерами так и обычными смартфонами. И в каждой из них встроен своеобразный монитор процессов и служб, который позволяет отслеживать все запущенные процессы (будь то процессы запущенные самой ОС или пользователем) и службы. Если раньше для этого нужно было очень хорошо знать

операционную систему и консольные команды этой ОС, то теперь есть возможность пользоваться приложениями для мониторинга без специфических знаний, так как большинство из них снабжены дружественным и интуитивно понятным графическим интерфейсом. Так же каждая программа сопровождается документацией и справкой, для того, чтобы пользователь, первый раз столкнувшийся с ней, смог быстро разобраться в интерфейсе и приступить к работе.

Спрос на эти приложения появился в связи с распространением компьютеров и развитием операционных систем. Сейчас почти в каждом доме есть компьютер и подключение к интернету. Но не каждый обладатель компьютера является опытным пользователем и не знает всех тонкостей работы ОС. Стандартные приложения для мониторинга процессов операционной системы могут дать доступ неопытному пользователю к системным процессам, это может привести к нестабильной работе ОС и компьютера в целом. Так же в связи с развитием интернета появились вредоносные программы – так называемые «вирусы». Эти приложения несут в себе опасность – они могут нанести вред как самому компьютеру и так и установленной на нём ОС. Стандартные приложения для не всегда дают завершить процесс вредоносной программы, здесь пригодятся программы сторонних разработчиков для опытных пользователей, которые имеют повышенный приоритет работы в ОС.

Цель данной статьи – проанализировать стандартное приложение для мониторинга процессов ОС Windows и программы сторонних разработчиков и, исходя из результатов анализа, разработать собственное приложение.

В данном конкретном случае стандартным приложением для мониторинга процессов ОС Windows является известный всем «Диспетчер задач». «Диспетчер задач» появился в семействе ОС Windows начиная с Windows 3.0 (Рис. 1).

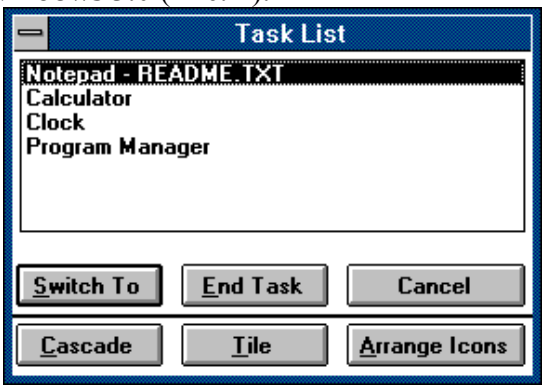


Рис.1 Диспетчер задач Windows 3.0

И начиная с Windows NT и до Windows 7 его интерфейс оставался фактически неизменным (Рис. 2 - 4).

Рис.2 Диспетчер задач Windows NT

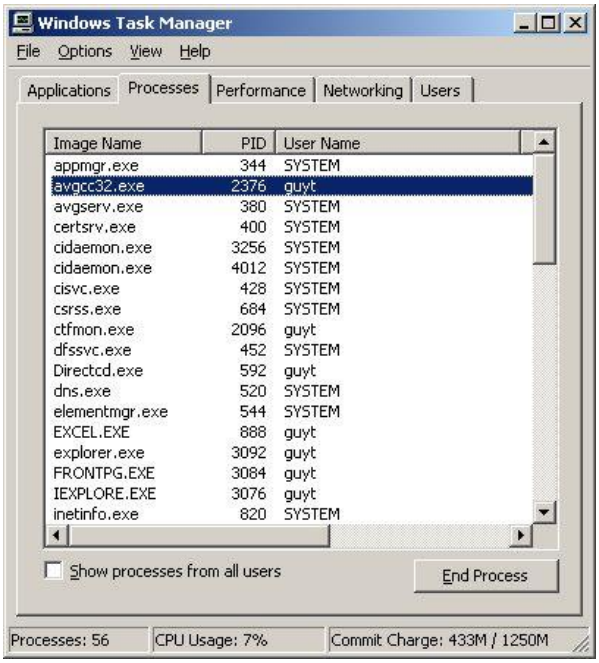


Рис.3 Диспетчер задач Windows XP

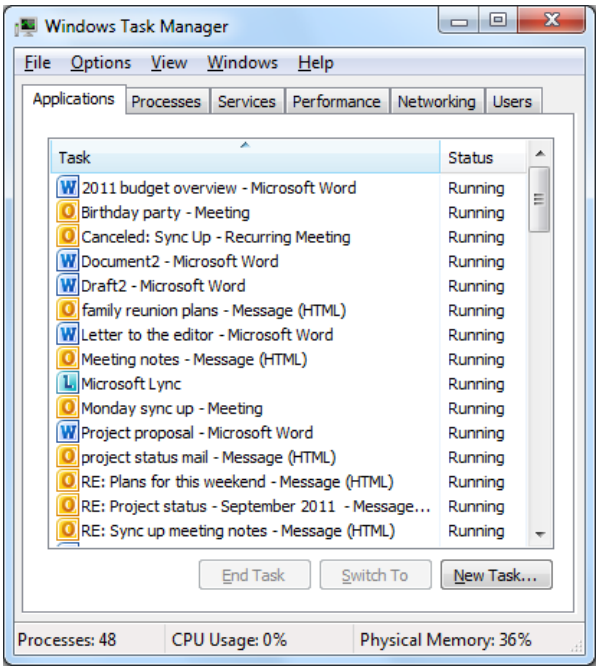


Рис.4 Диспетчер задач Windows 7

И только в недавно выпущенной корпорацией Microsoft ОС Windows 8 «Диспетчер задач» претерпел значительные изменения в интерфейсе (Рис. 5-6). Функциональность осталась практически неизменной.

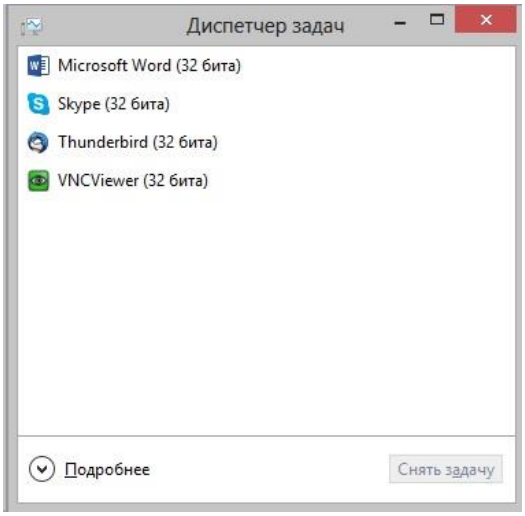


Рис.5 Диспетчер задач Windows 8 в свёрнутом виде

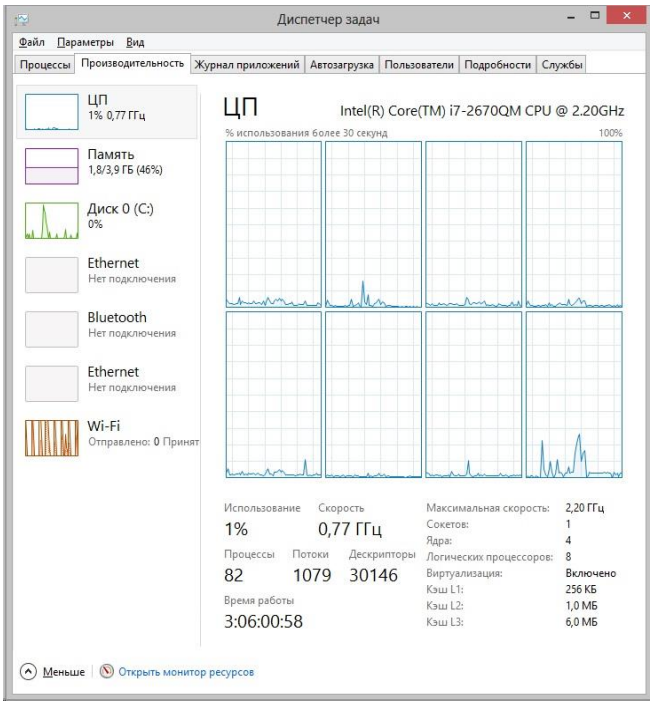


Рис.6 Диспетчер задач Windows 8 в развёрнутом виде

Теперь же рассмотрим приложения для мониторинга процессов ОС от сторонних разработчиков.

Process Explorer. Один из наиболее известных аналогов Диспечера задач. Process Explorer в режиме реального времени производит постоянный мониторинг всех запущенных в системе процессов. Для каждого процесса программа отображает подробную информацию (рис. 7):

- Объем используемой памяти;
- Процент загрузки процессора;
- Владелец процесса;
- Родственные отношения между процессами в виде дерева;
- Список библиотек, используемых выбранным процессом;
- Используемые протоколы TCP / IP;
- Расположение используемого файла.

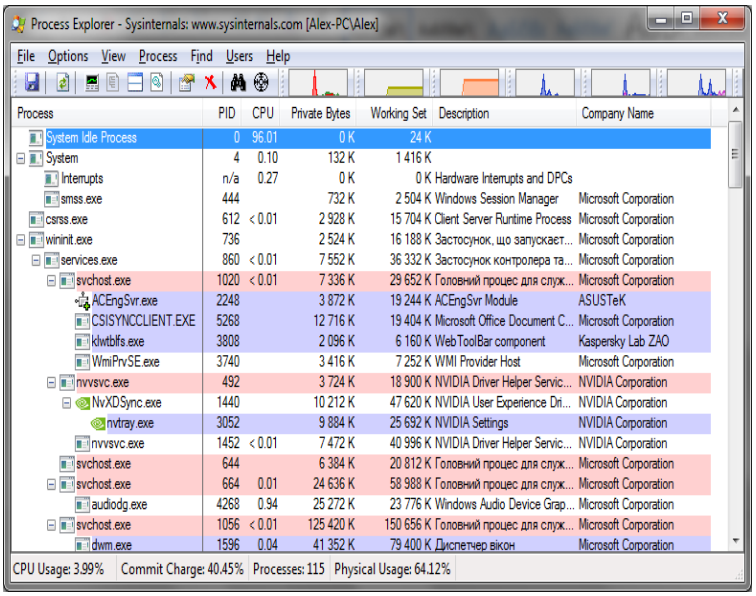


Рис. 7 Интерфейс программы Process Explorer

Кроме того, Process Explorer позволяет закрывать процессы, устанавливать их приоритет, а также искать DLL, используемых процессами. Интерфейс программы является простым в использовании и дружелюбным к пользователю.

AnVir Task Manager. Бесплатная системная утилита, разработанная для контроля над всем, что запущено на компьютере. AnVir Task Manager предоставляет полную информацию о процессах, отражает командную строку, загруженные DLL, входящий и исходящий трафик, нагрузку на диск, иконку в трее, подключения к Интернету, окна, потоки, открытые файлы и т.д.. Программа предоставляет все данные об автозагрузке программ и сервисов, в системном трее иконки мониторят использования процессора, температуру и загрузку жестких дисков, сетевой трафик, аккумулятор ноутбука и расход памяти. Применительно к процессам AnVir Task Manager позволяет выполнять весь набор основных операций - закрывать процесс, изменять приоритет и т.д. (рис. 8)

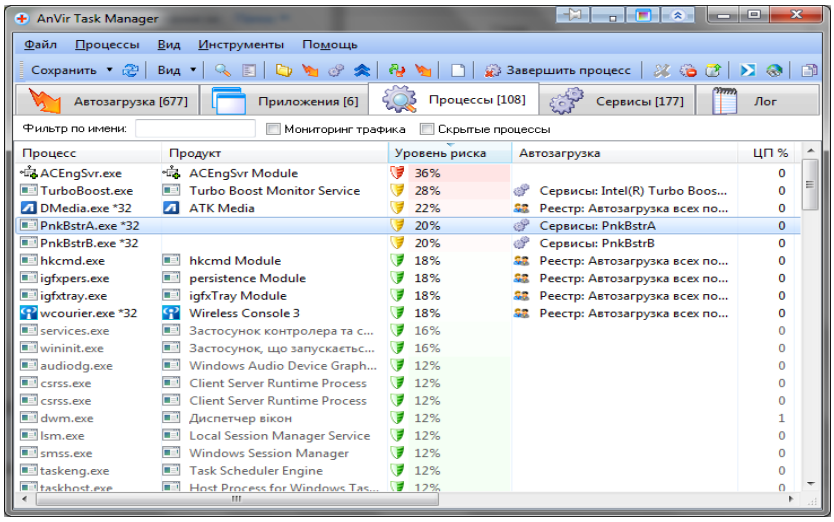


Рис. 8 Интерфейс программы AnvirTask Manager

Интерфейс программы имеет несколько вкладок [2]:

Автозарузка - мониторинг и управление программами, которые автоматически загружаются со стартом ОС.

Приложения - управление приложениями, запущенными на данный момент.

Процессы - управление и мониторинг процессов ОС.

Сервисы - управление запущенными сервисами.

Лог - лог системных событий.

Итак, данная программа имеет чрезвычайно информативный и разветвленный интерфейс, единственным недостатком которого является сложность для пользователя, который пользуется программой впервые.

Process Lasso. Основной задачей этой программы является автоматическое распределение циклов центрального процессора. Благодаря этому улучшается реактивность операционной системы при сильной загрузке CPU. Process Lasso может освобождать ресурсы для процессов за счет снижения приоритетов процессов, которым нужно много циклов.

Программа позволяет:

- Устанавливаться приоритеты;
- Отображать активные и вообще все процессы;
- Показывать загрузку процессора и количество потоков;
-Устанавливать максимальное количество копий процесса в системе;
- Завершать процессы;
- Создавать правила персонально для каждой программы;
- Вести блоги протоколирование всех выполняемых процессов (рис. 9).

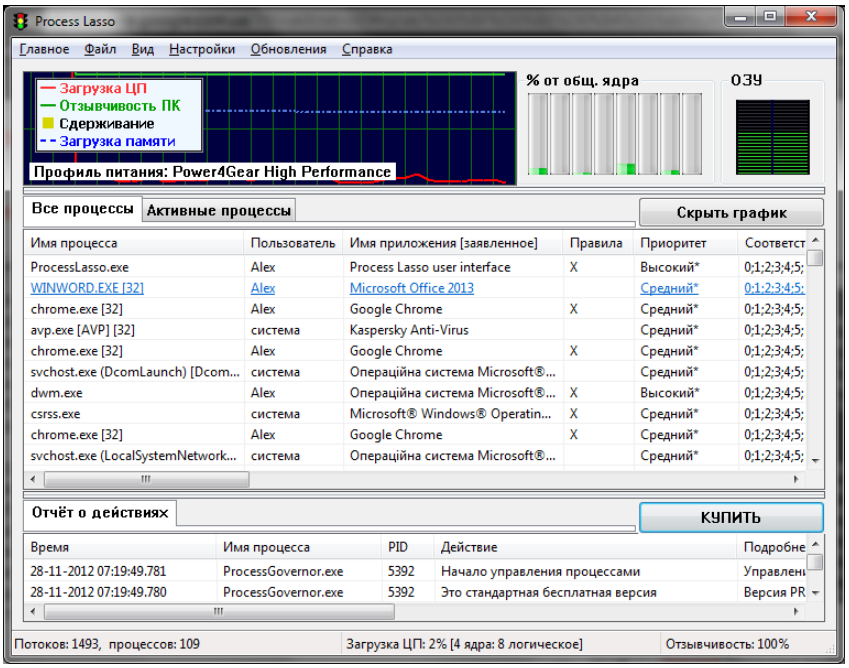


Рис. 9 Интерфейс программы Process Lasso

Starter. Starter разработан в первую очередь для управления запуском программ при загрузке Windows. Утилита позволяет редактировать список программ в автозагрузку, файл win.ini и ключи реестра. Кроме того, Starter показывает список запущенных процессов и служб, имеет удобную функцию просмотра библиотек, используемых процессом. Любой процесс можно остановить или установить для него приоритет. В списке виден путь к exe-файла процесса и объем используемой памяти. В программе предусмотрена возможность сохранения таблицы процессов в текстовый или HTML-файл (рис. 10).

Исходя из проведенного анализа, можно сделать следующий вывод. Проанализировано программное обеспечение является актуальным средством для мониторинга ОС. К недостаткам проанализированного программного обеспечения можно отнести нерешенность некоторых задач (см. табл. 1). При анализе не было выявлено ни одного программного средства, должен небольшой размер, портативность, не записывался в реестр, был бы построен с использованием WPF.

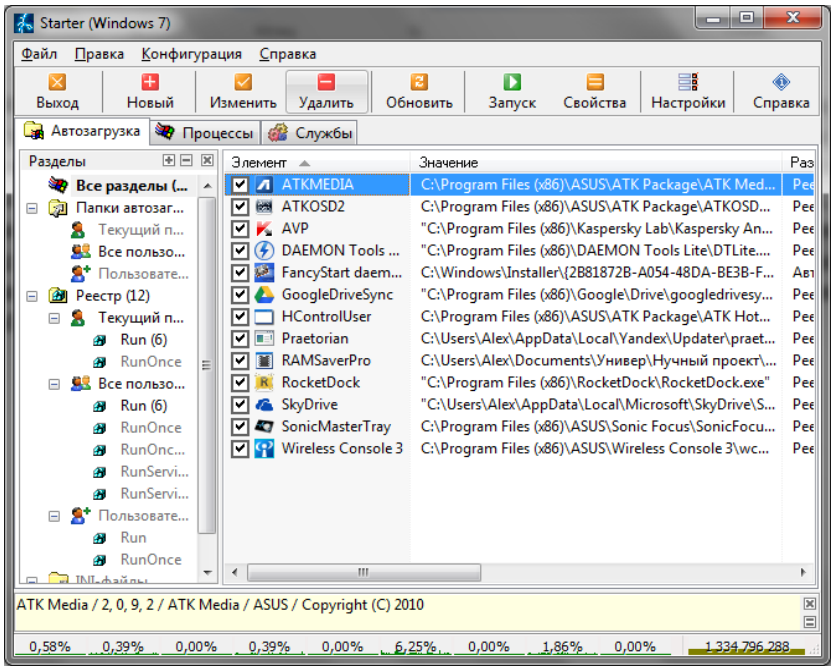


Рис. 10 Интерфейс программы Starter

Рассмотрим сравнительную таблицу преимуществ и недостатков проанализированных программ в представленной таблице. Как можно заметить, каждая из программ имеет свои плюсы и минусы.

Таблица 1 Сравнение рассмотренных программ

Задачи	AnVir Task Manager	Process Lasso	Starter
Отображение процессов	+	+	+
Представление дерева процессов	-	-	-
Процент загрузки процессора	+	+	-
Список используемых библиотек и служб	+	-	+
Установка приоритетов	+	+	+
Русскоязычный интерфейс	+	+	+
Бесплатная	+	-	+

Исходя из проведенного анализа, можно сделать следующий вывод. Проанализировано программное обеспечение является актуальным средством для мониторинга ОС. К недостаткам проанализированного программного обеспечения можно отнести нерешенность некоторых задач (см. табл. 2). При анализе не было выявлено ни одного программного средства, должен небольшой размер, портативность, не записывался в реестр, был бы построен с использованием WPF.

При анализе программ для мониторинга процессов операционной системы было обнаружено, что приложения в большинстве своем имеют много ненужных функций и большой размер, что не всегда является удобным. Также эти программы требуют базы знаний, которой не каждый пользователь.

Рассмотрим интерфейс разработанного нами приложения мониторинга системных процессов и процессов пользователя (рис. 11). Данная программа имеет возможность отображения загруженности процессора с помощью шкалы (ProgressBar), которая заполняется в соответствии с нагрузкой процессора. Также данные о нагрузке процессора выведены за экран с помощью объекта Processor Time.Информация о количестве свободной физической памяти отображается в блоке Memory free. Интерфейс программы позволяет просмотреть список запущенных процессов и служб.

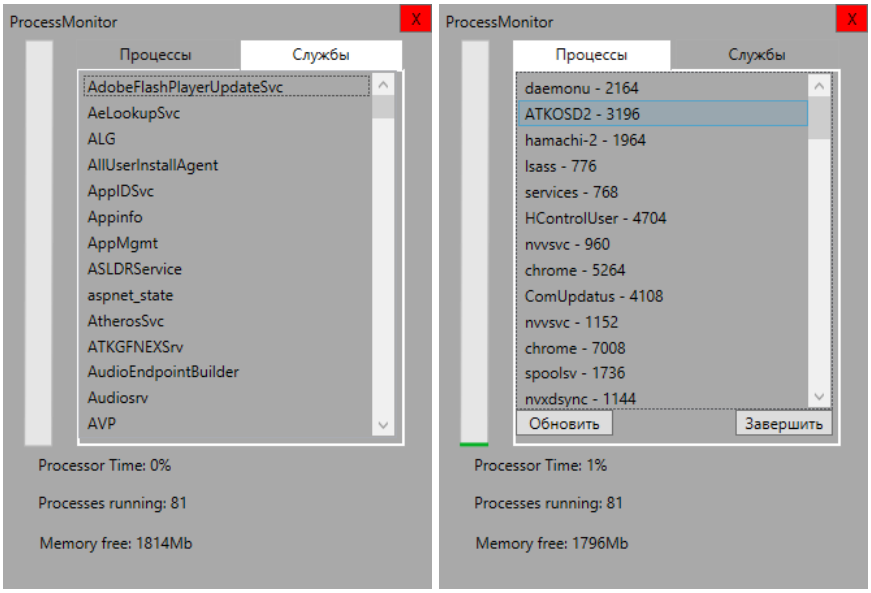


Рис. 11 Интерфейс программы ProcessMonitor

Программа написана на языке программирования C #. Интерфейс программы создан на базе Windows Presentation Foundation. Windows Presentation Foundation (WPF, кодовое название – Avalon) – графическая (презентационная) подсистема в составе. NET Framework 3.0, имеет прямое отношение к XAML. WPF вместе с. NET Framework 3.0 встроена в Windows Vista, а также доступна для установки в Windows XP Service Pack 2 и Windows Server 2003.

Это первое реальное обновление технологической среды пользовательского интерфейса со времени выпуска Windows 95. Оно включает новое ядро, которое должно заменить GDI и GDI +, используемые на сегодняшний Windows-платформе. WPF является фреймворком (высокоуровневым объектно-ориентированным

функциональным слоем), что позволяет создавать двумерные и трехмерные интерфейсы. Для запуска данного приложения необходимо Microsoft. NET Framework 3.0.

Таблица 2 Сравнение рассмотренных программ с Process Monitor

Задачи	Process Monitor	AnVir Task Manager	Process Lasso	Starter
Отображение процессов	+	+	+	+
Представление дерева процессов	-	-	-	-
Процент загрузки процессора	+	+	+	-
Список используемых библиотек и служб	+	+	-	+
Установка приоритетов	+	+	+	+
Русскоязычный интерфейс	+	+	+	+
Бесплатная	+	+	-	+

Таким образом, проанализированные программы для мониторинга процессов операционной системы объединяют в себе функции мониторинга процессов, на значение приоритета, определения PID процессов, отображение пути к файлу, выполняется, загрузка процессора, количество свободной оперативной памяти. В результате проведенного анализа было выявлено основные недостатки: отсутствует портативная версия, интерфейс, а также не имеют запрет на завершение системных процессов пользователем без прав администратора. Таким образом, на основании анализа существующего программного обеспечения были выдвинуты следующие требования к программам для мониторинга системных процессов и процессов пользователя ОС: дружественный интерфейс, портативность, запрет завершения системных процессов пользователем без прав администратора.

Разработана программа Process Monitor, которая удовлетворяет вышерассмотренных требованиям, предъявляемым к программам для мониторинга процессов ОС.А именно, мониторинг системных процессов и процессов пользователя ОС; PID процесса; загруженности процессора; отображение свободной оперативной памяти.

Программа ProcessMonitor может использоваться в высших и средних учебных заведениях, а также для частного пользования.

Согласно перспектив дальнейшего исследования проблемы мониторинга процессов ОС планируется реализация данного приложения как полноценного твикер ОС.

Лавренченко О. О., Віннік В.В. Розробка додатку для моніторигу процесів ОС

У статті описані і проаналізовані основні додатки для моніторингу процесів операційної системи. Виділені основні моменти роботі та ключові особливості додатків для моніторингу процесів ОС. Розроблена програма для моніторингу процесів та служб ОС на мові програмування C# з використанням графічної підсистеми Windows Presentation Foundation.

Ключові слова: операційна система, системні процеси, системні служби, Windows Presentation Foundation.

Лавренченко А. А., Винник В.В. Разработка приложения для мониторинга процессов ОС

В статье описаны и проанализированы основные приложения для мониторинга процессов операционной системы. Выделены основные моменты работы и ключевые особенности приложений для мониторинга процессов ОС. Разработана программа для мониторинга процессов и служб ОС на языке программирования C # с использованием графической подсистемы Windows Presentation Foundation.

Ключевые слова: операционная система, системные процессы, системные службы, Windows Presentation Foundation.

Lavrenchenko A. A., Vinnik V.V. Development of application for system monitoring

This article describes and analyzes the most popular applications for the monitoring of the system processes. Selected highlights the work of key features and applications for monitoring of the operating system Selected highlights the work of key features and applications for monitoring of the operating system. Was developed program to monitor the processes and services operating in the programming language C # using graphics Windows Presentation Foundation.

Keywords: operating system, system processes, system services, Windows Presentation Foundation.

УДК 519.853

С.В. Мисюра

РАЗРАБОТКА АЛГОРИТМОВ ДЛЯ УПРАВЛЕНИЯ ПОТОЧНО-ТРАНСПОРТНОЙ СИСТЕМОЙ КОЛЬЦЕВЫХ ШАХТНЫХ ПЕЧЕЙ

Актуальность работы. Обжиг извести в кольцевых шахтных печах представляет собой консервативный технологический процесс. Поэтому повышение эффективности производства извести

осуществляется за счет повышения эффективности работы инфраструктуры, обеспечивающей постоянную и бесперебойную работу печи. Одним из элементов инфраструктуры шахтных кольцевых печей является поточно-транспортная система (ПТС). Оптимальное функционирование ПТС обеспечивает система автоматического управления (САУ). Основополагающим фактором успешной реализации потенциала технологического оборудования и средств автоматизации САУ ПТС является разработка и реализация алгоритмов управления.

Цель работы – разработка алгоритмов для системы управления ПТС кольцевых шахтных печей. Для достижения поставленной цели решены следующие задачи:

1. Проведен анализ процессов современных кольцевых печей обжига известняка как объекта управления.
2. Разработаны алгоритмы управления поточно-транспортной си-стемой.

Постановка задачи упрваления. Управление ПТС должно отвечать требованиям действующих инструкций безопасности [1], обеспечивать технологическую потребность печи в сырье, отводить готовую продукцию в проектных объемах, обеспечивать оптимальные режимы эксплуатации оборудования, минимизировать время простоев и количество аварийных ситуаций [2].

Таким образом, поставлены следующие задачи управления:

– управление механизмами с взаимными блокировками, исключающими просыпь материала;
– задание оператором режимов работы в зависимости от технологических потребностей;
– автоматизированный пуск оператором всего комплекса в соответствии с заданием;
– автоматизированный останов оператором всего комплекса;
– организация обмена управляющими данными (пуск, останов трактов) между АСУ печи и ПТС;
– автоматический пуск/останов тракта по сигналу АСУ печи;
– автоматическое распределение произведенной извести по бункерам.

Для реализации поставленных задач определены уровни их решения:

– на уровне 1 проводится разбиение агрегатов по типам и разработка алгоритма проводится для каждого типа;
– на уровне 2 агрегаты объединяются в функционально и технически зависимые комплексы выполняющие совместно поставленную задачу и для них синтезируется комплексный алгоритм управления. На данном уровне реализуются поставленные задачи управления.

Уровень 1

Основными агрегатами ПТС являются:

- Вибропитатель;
- Конвейер;
- Грохот;
- Сбрасыватель.

Каждый тип технологических агрегатов имеет собственный набор входных параметров (табл. 1) и требует выдачи определенных управляющих воздействий.

Таблица 1– Перечень параметров агрегата Конвейер.

Вид параметра	**Тип параметра**	**Состояние наличия сигнала**
Входные параметры		
Автоматический выключатель включен	дискретный	1
Контроль напряжения	дискретный	1
Конвейер включен	дискретный	1
Отключение при перегрузке	дискретный	0
Скорость ленты низкая	дискретный	0
Кнопка "Пуск"	дискретный	1
Кнопка "Стоп"	дискретный	0
Кнопка "Аварийный стоп"	дискретный	0
Отключение канатным устройством1	дискретный	0
Отключение канатным устройством2	дискретный	0
Запрет включения	дискретный	1
Предпусковая сигнализация включена	дискретный	1
Предпусковая сигнализация. Включение	дискретный	1
Предпусковая сигнализация. Отключение	дискретный	0
Автоматический выключатель тормоза включен	дискретный	1
Показания тока двигателя	аналоговый	
Выходные параметры		
Включить предпусковую сигнализацию	дискретный	1
Пуск конвейера	дискретный	1

Алгоритмы управления агрегатами представляют собой линейные алгоритмы последовательного действия с циклическим выполнением (рис.1).

Рис 1 – Блок-схема упрощенного алгоритма управления механизмом.

Для агрегата типа Конвейер алгоритм представлен на рис. 2.

Для детализации операций алгоритма используются логические функции в базисе И ($\wedge$), ИЛИ ($\vee$), НЕ (!).Для проведения детализации операций введены условные обозначения, представленные в табл. 2.

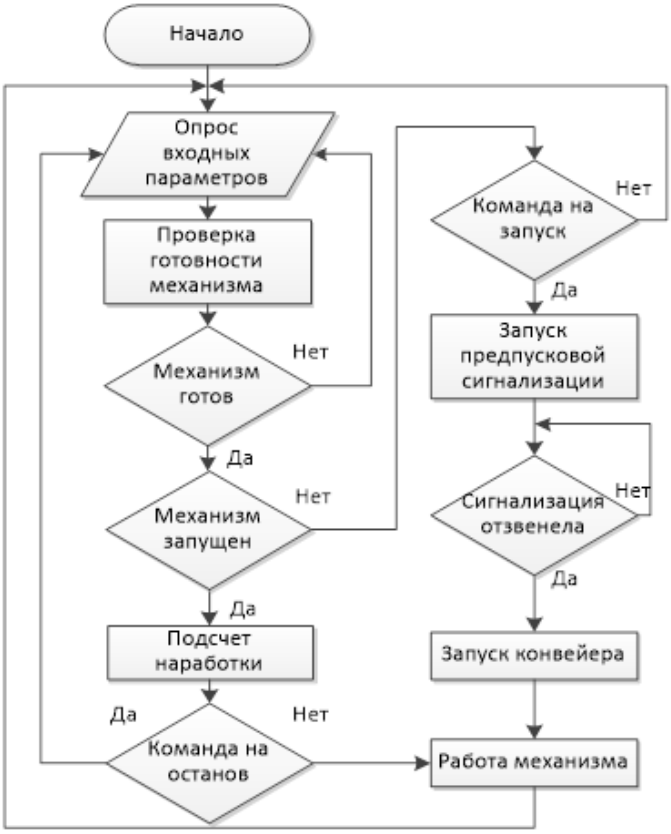


Рис 2 – Блок-схема алгоритма управления Конвейером.

Таблица 2 – **Условные обозначения параметров.**

Вид параметра	**Условное обозначение**
Входные параметры	
Автоматический выключатель включен	QF
Контроль напряжения	K1
Конвейер включен	KM1
Отключение при перегрузке	KK
Наличие скорости ленты	A1V
Кнопка "Стоп"	SB1_2
Кнопка "Аварийный стоп"	SBA
Отключение канатным устройством1	SQ1
Отключение канатным устройством2	SQ2
Запрет включения	S
Предпусковая сигнализация включена	KA
Автоматический выключатель тормоза включен	QF1
Показания тока двигателя	аналоговый
Сброс ошибок	SBROS_PC
Блокирвка включения	BLOCK
Выходные параметры	
Включить предпусковую сигнализацию	Q_KA
Пуск конвейера	Q_KM1
Сигнал готовности	G

Логическая функция операции проверки готовности механизма имеет вид:

$$G = QF \wedge K1 \wedge !KK \wedge SB1\_2 \wedge SBA \wedge !SQ1\_S \wedge !SQ2\_S \wedge !S \wedge QF1 \wedge !UA\_HI \wedge !KM\_NEVKL \wedge KA\_NEVKL \wedge SPL$$

где

$$SQ1\_S = (SQ1 \vee SQ1\_S) \wedge !SBROS\_PC$$

$$SQ2\_S = (SQ2 \vee SQ2\_S) \wedge !SBROS\_PC$$

$$KM\_NEVKL = Q\_KM1 \wedge !KM1 \vee KM\_NEVKL \wedge !SBROS\_PC$$

$$KA\_NEVKL = Q\_KA \wedge !KA \vee KA\_NEVKL \wedge !SBROS\_PC$$

Уровень2

Обобщенный алгоритм автоматизированного пуска тракта подачи известняка представлен на рис. 3. Данный алгоритм реализует задачу автоматизированного пуска оператором всего комплекса в соответствии с заданием. Для проведения детализации операций введены условные обозначения, представленные в табл. 3. Операция запуска тракта представляет собой систему логических уравнений:

$$KL6\_P1 = T\_P \wedge G \wedge B7 \wedge !B6$$

$$KL6\_P2 = T\_P \wedge G \wedge B6 \wedge !B7$$

$KL5\_P = T\_P \wedge G \wedge (KL6\_KM1 \vee KL6\_KM2)$
$GROH\_P = T\_P \wedge G \wedge KL5\_KM$
$KL4\_P = T\_P \wedge G \wedge GROH\_KM$
$KL3\_P = T\_P \wedge G \wedge KL4\_KM$
$KL2\_P = T\_P \wedge G \wedge KL3\_KM \wedge (B1 \vee B2)$
$KL1\_P = T\_P \wedge G \wedge KL2\_KM \wedge B1$
$V111\_P = T\_P \wedge G \wedge B3 \wedge KL3\_KM$
$V110\_P = T\_P \wedge G \wedge B2 \wedge KL2\_KM$
$V109\_P = T\_P \wedge G \wedge B1 \wedge KL1\_KM$

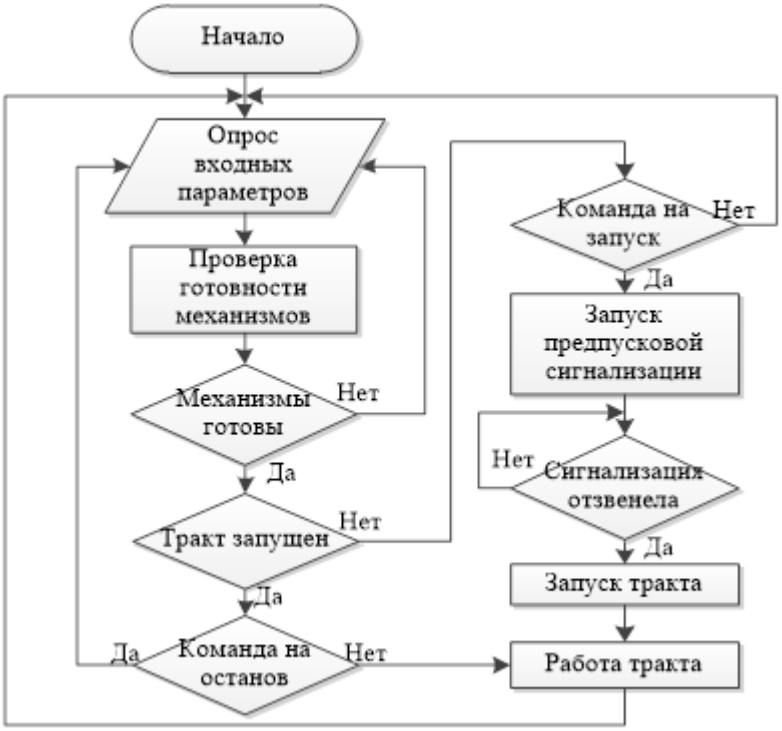


Рис 3 – Блок-схема упрощенного алгоритма управления трактом.

Таблица 3 – Перечень параметров.

Вид параметра	**Тип параметра**	**Состояние наличия сигнала**	**Условное обозначение**
Входные параметры			
Грузить в бункер Б-6	дискретный	1	B6
Грузить в бункер Б-7	дискретный	1	B7
Конвейер КЛ-1. Запущен	дискретный	1	KL1_KM
Конвейер КЛ-2. Запущен	дискретный	1	KL2_KM
Конвейер КЛ-3. Запущен	дискретный	1	KL3_KM
Конвейер КЛ-4. Запущен	дискретный	1	KL4_KM
Конвейер КЛ-5. Запущен	дискретный	1	KL5_KM

Вид параметра	**Тип параметра**	**Состояние наличия сигнала**	**Условное обозначение**
Конвейер КЛ-6. Запущен вперед	дискретный	1	KL6_KM1
Конвейер КЛ-6. Запущен назад	дискретный	1	KL6_KM2
Вибропитатель 109. Запущен	дискретный	1	V109_KM
Вибропитатель 110. Запущен	дискретный	1	V110_KM
Вибропитатель 111. Запущен	дискретный	1	V111_KM
Грохот. Запущен	дискретный	1	GROH_KM
Выходные параметры			
Пуск по тракту	дискретный	1	T_P
Включение предпусковой сигнализации	дискретный	1	KA_P
Конвейер КЛ-1.Пуск	дискретный	1	KL1_P
Конвейер КЛ-2.Пуск	дискретный	1	KL2_P
Конвейер КЛ-3.Пуск	дискретный	1	KL3_P
Конвейер КЛ-4.Пуск	дискретный	1	KL4_P
Конвейер КЛ-5.Пуск	дискретный	1	KL5_P
Конвейер КЛ-6.Пуск вперед	дискретный	1	KL6_P1
Конвейер КЛ-6.Пуск назад	дискретный	1	KL6_P2
Вибропитатель 109. Пуск	дискретный	1	V109_P
Вибропитатель 110. Пуск	дискретный	1	V110_P
Вибропитатель 111. Пуск	дискретный	1	V111_P
Тракт включен	дискретный	1	TRUCT_VKL

В результате получены алгоритмы в виде блок-схем для решения задачи управления отдельными агрегатами и логические функции, детализирующие операции, представленные на блок-схеме. Полученные алгоритмы реализуют полный функционал необходимый для работы всех типов агрегатов как самостоятельных единиц и позволяют синтезировать любой агрегат в общую систему управления всем трактом. Путем синтеза получены алгоритмы, управления САУ ПТС, реализующие комплексность и целостность системы, полностью отвечающие поставленным задачам. Детализация алгоритмов с помощью логических уравнений в базисе И, ИЛИ, НЕ позволила с минимальными изменениями реализовать алгоритмы в виде программы дискретного управления.

Литература

1. НПАОП 27.0-1.01-08 ПРАВИЛА охорони праці в металургійній промисловості. — 2008. **2. ГОСТ 24**.104-85 Автоматизированные системы управления. Общие требования. — 1985.

Мисюра С.В. Разработка алгоритмов управления поточно-транспортной системой кольцевых шахтных печей

В статье приведены результаты анализа процессов работы современных кольцевых шахтных печей для обжига известняка. Сформулированы основные задачи управления. Определены уровни решения задач с разбиением агрегатов по типам и объединением агрегатов в функциональные комплексы. Проведена детализация операций с помощью логических функций, которая максимально упрощает переход к реализации алгоритмов в программном коде. Предложены алгоритмы управления поточно-транспортной системой. Подтверждена работоспособность программы управления системой на базе алгоритмов.

Ключевые слова: система автоматического управления, поточно-транспортная система, алгоритм, синтез алгоритмов, siemens, step7, симуляция.

Місюра С.В. Розробка алгоритмів керування потоково-транспортною системою кільцевих шахтних печей

У статті приведені результати аналізу процесів роботи сучасних кільцевих шахтних печей для випалення вапняку. Сформульовані основні завдання управління. Визначені рівні рішення завдань з розбиттям агрегатів по типах і об'єднанням агрегатів у функціональні комплекси. Запропоновані алгоритми керування потоково-транспортною системою. Підтверджена працездат-ність програми управління системою на базі алгоритмів.

Ключові слова: система автоматичного керування, потоково-транспортна система, алгоритм, синтез алгоритмів, siemens, step7, симуляція.

Misyura S. The development of management algorithms of flow-transport system of annular shaft furnaces

Article includes the results of workflow analysis for the modern annular shaft furnaces for lime burning. Basic management goals were formulated. There was defined the levels of solving problems with aggregates division by type and combining the aggregates into functional complexes. Detailing of transactions was carried out with the help of logical functions, which greatly simplifies the transition to the implementation of algorithms in software code. The algorithms of controlling the system of traffic flows were proposed. Was confirmed the efficiency of the system control program based

on algorithms.

Keywords: automatic control system, thread-transport system, algorithm, algorithms synthesis, siemens, step7, simulation

УДК 004.421:004.65

Ю.Г. Могильная

АНАЛИЗ ВОЗМОЖНОСТИ ОРГАНИЗАЦИИ ИСПОЛЬЗОВАНИЯ ДИНАМИЧЕСКОЙ СТРУКТУРЫ ДЕК ПРИ РЕШЕНИИ ЗАДАЧ

Важнейшим этапом в процессе решения прикладных и системных задач является выбор эффективного алгоритма. Алгоритм должен быть простым для понимания и перевода программный код, эффективно использовать вычислительные ресурсы и выполняться по возможности быстро. Независимо от содержания и сложности алгоритма любые произвольные данные в памяти ЭВМ представляются в виде последовательности битов. Однако более крупные и содержательные "строительные блоки" для организации произвольных данных создаются на основе структур данных, под которыми понимается множество элементов данных и множество связей между ними. Поэтому целью данной работы является исследование возможности С++ для использования динамической структуры ДЕК.

Известно, что процесс стандартизации С++ начался в 1989 году и продолжался до конца 1997 года и закончился в 1998 году. Результатом этой работы стало справочное руководство, насчитывающее около 750 страниц и опубликованное международной организацией по стандартизации (ISO). Стандарт получил название «Information Technology — Programming Languages — C++».

Одной из составляющих стандарта является стандартная библиотека STL (STL - Standard Template Library) разработанная Алексом Степановым и Менг Ли[3]. Она содержит базовые компоненты для выполнения операций ввода-вывода, для работы со строками, контейнерами (структурами данных) и алгоритмами (сортировка, поиск, слияние и т. д.), для математических вычислений и, как и следовало ожидать от международного стандарта, для поддержки интернационализации.[1]

Стандартная библиотека шаблонов STL - это библиотека контейнерных классов, которая включает векторы, списки, очереди и стеки, а также ряд алгоритмов общего назначения и предоставляет набор хорошо сконструированных и согласованно работающих вместе обобщённых компонентов С++.

При ее разработке авторы предприняли попытку создания шаблонных алгоритмов и компонентов, основанных на прочной

теоретической базе, работающих не только со структурами данных в библиотеке, но также и с встроенными структурами данных C++ и имеющих обобщённую реализацию с целью обеспечения гибкости, эффективности и легкости для понимания.

Библиотека содержит пять основных видов компонентов:

- алгоритм (*algorithm*): определяет вычислительную процедуру.
- контейнер (*container*): управляет набором объектов в памяти.
- итератор (*iterator*): обеспечивает для алгоритма средство доступа к содержимому контейнера.
- функциональный объект (*function object*): инкапсулирует функцию в объекте для использования другими компонентами.
- адаптер (*adaptor*): адаптирует компонент для обеспечения различного интерфейса.

Такое разделение позволило уменьшить количество компонентов. Например, вместо написания функции поиска элемента для каждого вида контейнера обеспечивается единственная версия, которая работает с каждым из них, пока удовлетворяется основной набор требований.

STL обладает рядом преимуществ:[4]

- Код библиотеки написан профессиональными программистами, проверен и отлажен.
- Код библиотеки написан очень эффективно с точки зрения использования оперативной памяти и быстродействия для типовых вариантов применения.
- Библиотека предлагает унифицированный интерфейс, однообразный для всех контейнеров и алгоритмов
- Использование библиотеки позволяет приступить сразу к решению проектных задач, не задумываясь о реализации низкоуровневых контейнеров и алгоритмов.
- Библиотека хорошо документирована и описана в книгах.
- Код, написанный с использованием STL легко переносится на другие компиляторы, операционные системы и платформы.

К недостаткам STL можно отнести:

- Неприспособленность к работе со структурными типами данных.
- Низкая эффективность (быстродействие, память) при решении частных задач, где возможны целевые оптимизации кода.
- Неадекватный интерфейс шаблона для работы со строками.
- Сложность управления пулом памяти при работе с контейнерами STL.[7]

Библиотека STL реализует динамические структуры данных и функции и методы для работы с ними.

Несмотря на то, что библиотека STL реализует наиболее характерные динамические структуры, есть ряд структур (рис.1), которые она не описывает, и не могут быть реализованы через нее. Например, циклический список, двунаправленный список. Но есть ряд структур, которые могут быть реализованы через библиотеку STL, например, ДЕКи (от англ. Deq – double ended queue, т.е очередь с двумя концами) через стек и итераторы. Рассмотрим динамическую структуру данных ДЕК.

ДЕК - это такой последовательный список, в котором как включение, так и исключение элементов может осуществляться с любого из двух концов списка. Частный случай ДЕКа – ДЕК с ограниченным входом и ДЕК с ограниченным выходом. Логическая и физическая структуры ДЕКа аналогичны логической и физической структуре кольцевой FIFO-очереди. Применительно к ДЕКу целесообразно говорить не о начале и конце, а о левом и правом конце. ДЕК обладает большей общностью, чем стек или очередь; он имеет некоторые общие свойства с колодой карт (в английском языке эти слова созвучны). В ДЕКе все исключения и добавления происходят на обоих его концах. ДЕК по сути двунаправленный список. Первый и последний элементы ДЕКа соответствуют входу и выходу ДЕКа. Накладывая дополнительные ограничения на операции с началом и/или концом ДЕКа, можно осуществлять моделирование стека и очереди.[6]

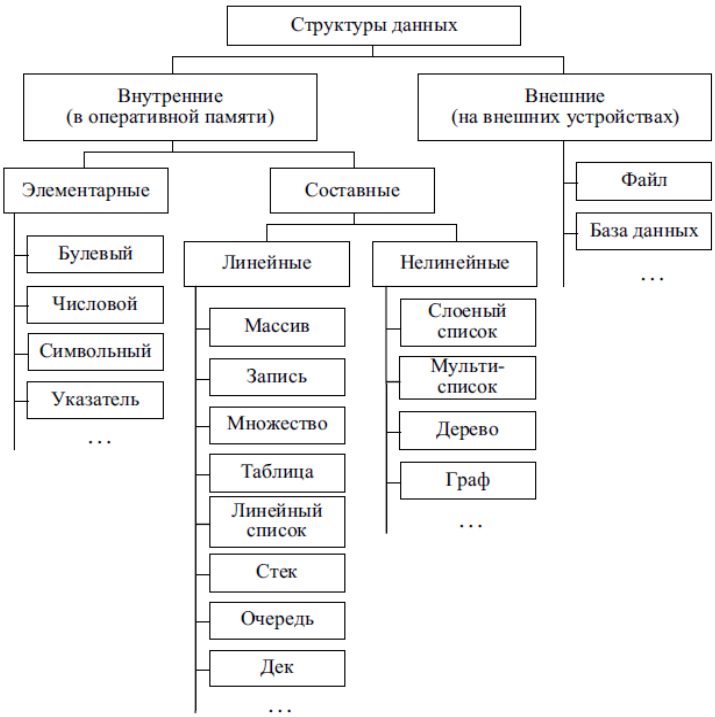


Рис. 1 Структуры данных

ДЕК также можно реализовывать как статическую структуру данных в виде одномерного массива, а можно как динамическую структуру в виде линейного списка (рис. 2).[2]

Операции над ДЕКом:

- включение элемента справа;
- включение элемента слева;
- исключение элемента справа;
- исключение элемента слева;
- определение размера;
- очистка.

Физическая структура ДЕКа в статической памяти идентична структуре кольцевой очереди. Динамическая реализация является очевидным объединением стека и очереди.

Задачи, требующие структуры ДЕКа, встречаются в вычислительной технике и программировании гораздо реже, чем задачи, реализуемые на структуре стека или очереди. Как правило, вся организация ДЕКа выполняется программистом без каких-либо специальных средств системной поддержки.

Примером ДЕКа может быть, например, некий терминал, в который вводятся команды, каждая из которых выполняется какое-то время. Если ввести следующую команду, не дождавшись, пока закончится выполнение предыдущей, то она встанет в очередь и начнет выполняться, как только освободится терминал. Это FIFO очередь. Если же дополнительно ввести операцию отмены последней введенной команды, то получается ДЕК.

В библиотеке STL ДЕК – это тип последовательного контейнера, который поддерживает итераторы произвольного доступа к элементам.

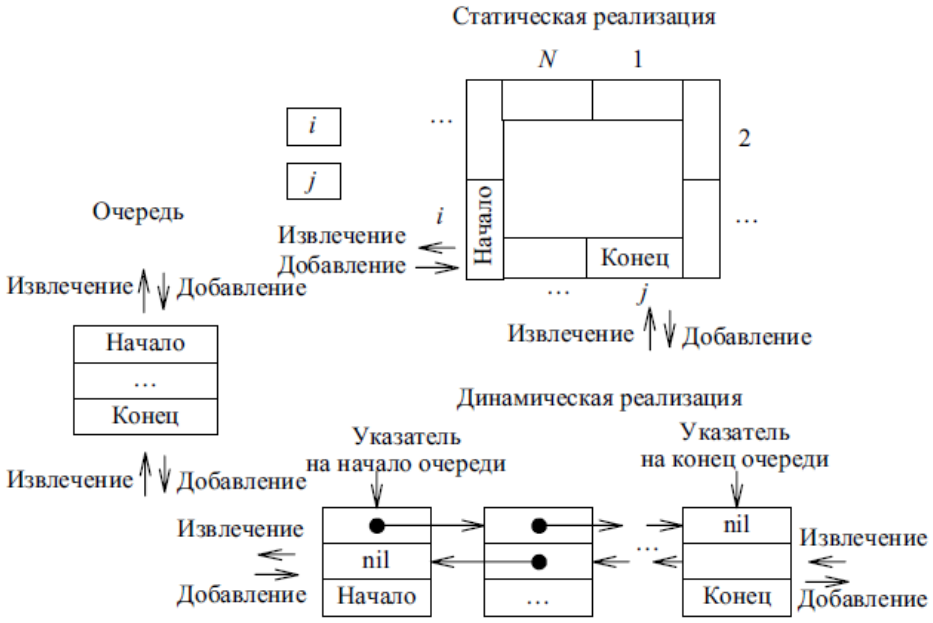


Рис 2 Способы реализации ДЕК

Для реализации ДЕК имеется класс **deque,** который определен в заголовочном файле **<deque>** в пространстве имен std. Контейнерный класс ДЕК позволяет выполнять и многое другое. На самом деле структуры данных ДЕК почти объединяют в себе возможности, предоставляемые векторами и списками. Как и вектор, ДЕК является индексированной совокупностью, то есть доступ к значениям возможен по индексу.

Однако, как и список, он позволяет легко добавлять элементы в начало или конец совокупности. Как и списки, и векторы, ДЕК поддерживает вставку элементов в середину хранимой совокупности. Такие операции в нем не столь эффективны, как в списке, однако все же более эффективны, чем в векторе.

Функции-члены этого класса <**deque**> – **begin** () и **end** () возвращают итераторы произвольного доступа, которые позволяют осуществлять доступ к *произвольным* элементам, а не двунаправленные итераторы, как в случае списка.

Анализ литературных источников показал, что ДЕК (класс **<deque>)** поддерживает операции удаления первого и последнего элементов **void pop_front() void pop_back()**. Вставка элементов выполняется функциями **insert (), push_front ()** или **push_back ().**

Из анализа установлено, что его использование имеет свои преимущества – быстрое написание кода и гибкость однако может потенциально приводить к неправильным значениям всех действующих итераторов и ссылок на элементы в ДЕКе. Поэтому для решения задачи

реализовали ДЕК через динамические структуры данных путем создание собственных функций.

Была написана программа, моделирующая использование ДЕК, который реализуется через динамические структуры. Программа моделирует примитивную игру – «Угадайка», в которой компьютер генерирует ряд случайных чисел заданного пользователем диапазона. Игрок должен угадать эти числа. В результате, в программе было реализовано целый ряд методов для работы с ДЕКом

Описание элементов дека аналогично описанию элементов линейного двунаправленного списка. Поэтому объявим дек через объявление линейного двунаправленного списка:

```
struct Deque{
        Double_List *Begin;//начало decka
    Double_List *End; //конец decka
        };
```

Реализацию этих операций приведем в виде соответствующих функций, которые, в свою очередь, используют функции операций с линейным двунаправленным списком. Кратко рассмотрим некоторые из них:[6]

```
//создание ДЕКа
void Make_Deque(int n, Deque* End_Deque){
  Make_Double_List(n,&(End_Deque->Begin),NULL);
  Double_List *ptr; //вспомогательный указатель
  ptr = End_Deque->Begin;
  while (ptr->Next != NULL){
    ptr = ptr->Next;
  }
  End_Deque->End = ptr;
}

//печать ДЕКа
void Print_Deque(Deque* Begin_Deque){
  Print_Double_List(Begin_Deque->Begin);
}

//добавление элемента в правый конец ДЕКа
void Add_Right_Item_Deque(int NewElem, Deque* End_Deque){
  End_Deque->End          =Insert_Item_Double_List(End_Deque-
>End,2,NewElem);
  End_Deque->End = End_Deque->End->Next;
}
```

В главном методе программа запрашивает у пользователя, на сколько чисел он хочет играть и попросит ввести самое максимальное число.. Далее, создается динамический список. В цикле for выполняется

генерация случайных чисел – числа, загаданные компьютером и добавляются в правый конец ДЕКа. В следующем цикле for вводятся числа пользователя, добавляя их в левый конец ДЕКа. В функции srav в цикле for я берется по одному элементу с каждого конца ДЕКа, (используя функции Extract_Right_Item_Deque, Extract_Left_Item_Deque, Delete_Item_Double_List) и если элементы были равны, то выводится на экран сообщение об этом, суммируется их количество и (в главной функции) выводится сообщение на экран.

В данной программе ДЕК применялся для хранения чисел сгенерированных компьютером и введенных пользователем с использованием операций добавлении элементов в правый и левый конец. Написанная программа, показывает работу динамической структуры ДЕК.

```
 Программа-угадайка
 Угадайте числа ,которые задумал компьютер
 Ведите на сколько чисел играем         15
 Введите максимальное число     25
 Ваши числа вводятся в левый конец дека, а числа компьютера в правый конец дека



 Введите  1 число
25
 Введите  2 число
14
 Введите  3 число
36
 Ошибка
 Введите  3 число
21
 Введите  4 число
15
 Введите  5 число
2
 Введите  6 число
1
 Введите  7 число
25
 Введите  8 число
14
 Введите  9 число
24
 Введите  10 число
13
 Введите  11 число
1
 Введите  12 число
0
 Введите  13 число
3
 Введите  14 число
19
 Введите  15 число
8
 Ваш дек

8       19      3       0       1       13      24      14      25      1
2       15      21      14      25      10000000        15      9       19
16      22      10      1       13      8       16      3       1       11
24      1


 Ха-ха-ха!!Вы лузер! Не угадали ни одного раза!
```

В результате проделанной работы были изучены свойства ДЕКа и разработана программа «Угадайка» моделирующая использование ДЕК.

Литература

1. Николай Джосьютис. C++ Стандартная библиотека Для профессионалов/ Н. Джосьютис. – СП Питер, 2004. – С. 23-24.: ил. **2. Ключарев А.А.** Структуры и алгоритмы обработки данных: Учебное пособие/ СПбГУАП. СПб., 2003. С 4-6, С 46-47. **3. Интервью** с создателем C++ STL. [Електронний ресурс] http://habrahabr.ru/post/166849/ **4. Руководство** по стандартной библиотеке шаблонов (STL) [Електронний ресурс] http://www.rsdn.ru/article/cpp/stl.xml **5. Линейные** списки. Стек. Дек. Очеред. [Електронний ресурс] http://www.zachetka.ru/referat/preview.aspx?docid=28914&page=5 **6. Фоменко А. В**. Курс лекцій з дисципліни Алгоритми та структури даних/ Луганськ., 2012. С 232-233. **7. Лингвистическое** и программное обеспечение автоматизированных систем. Библиотека STL. [Електронний ресурс]: конспект лекций http://tenisheff.ru/hgs/po/STL.htm

Могильна Ю.Г. Аналіз можливості організації використання динамічної структури ДЕК при вирішенні завдань

Стаття містить дослідження можливості C++ для використання динамічної структури ДЕК.

Ключові слова: динамічна структура, ДЕК

Могильная Ю.Г. Анализ возможности организации использования динамической структуры ДЕК при решении задач

Статья содержит исследование возможности C++ для использования динамической структуры ДЕК.

Ключевые слова: динамическая структура, ДЕК

Mogilnaya J.G. Analysis of the possibility of a dynamic organization structure for solving DEC

The article contains a study of the possibility of C+ + to use the dynamic structure of DEC.

Keywords: dynamic structure, DEC

УДК 004.514

Ю. Б. Мосейчук

ОСОБЕННОСТИ РАЗРАБОТКИ ИНТЕРФЕЙСОВ ПРИЛОЖЕНИЙ МОБИЛЬНЫХ УСТРОЙСТВ

Широкое распространение мобильных устройств (телефонов, планшетов, смартфонов), а также, стирание границ между этими понятиями, увеличение вычислительной мощности, постоянная доступность сети Интернет и высокое время автономной работы требует рассмотрения данных устройств как отдельного класса вычислительных машин, причем как в плане подхода к разработке приложения, так и в плане реализации интерфейсов.

При разработке мобильных платформ необходимо учитывать ключевые отличия мобильных приложений от стандартных компьютерных пользовательских GUI. Учитывая вышесказанное, и то, что рынок мобильных приложений является быстрорастущим и прибыльным, актуальным является выявление отличий в интерфейсах desktop-приложений и мобильных платформ, с разработкой требований к проектированию последних.

При переносе desktop-приложений на мобильную платформу необходимо, помимо сохранения функционала, учитывать особенности и возможности платформы. К основным особенностям мобильных платформ относиться:

- Ограниченная, по сравнению с обычными приложениями, функциональность. В массе случаев мобильная версия программы реализует базовые основные функции приложения.
- Размер экрана мобильных устройств значительно меньше, чем современных мониторов, а следовательно, необходима адаптация элементов управления под размер экрана.
- Пользователь взаимодействует с экраном с помощью стилуса (устаревшая форма) или пальца, следовательно, размер элементов приложения должен быть адаптирован под размер пальца.
- При взаимодействии с touch-интерфейсом возможны случайные нажатия. Критические действия для телефона или приложения должны быть защищены от случайного срабатывания.

Рис. 2. Прототип авторизации в Skype

Рис. 3. Основное окно и звонки

- Использование одного экрана для взаимодействия с программой. Это ограничивает логику взаимодействия и снижает наглядность переходов, что повышает требования к логике программы.
- На экране отображается в один момент времени только одна программа.
- Возможность смены ориентации экрана с горизонтальной на вертикальную. Интерфейс должен уметь адаптироваться под ориентацию.
- Возможность взаимодействия одним или несколькими прикосновениями одновременно, управление жестами.

Таким образом, сенсорный экран и принцип работы с ним вносит существенные коррективы в интерфейсы мобильных приложений. Принципы навигации на телефонах и планшетах существенно отличаются от аналогичных desktop версий и требуют переработки подходов к проектированию интерфейсов.

На основании ограничений описанных выше сформулируем набор рекомендаций к проектированию дизайна приложений:

1. Выделение основного функционала и его реализация и, как следствие, минимизация загруженности экрана.
2. Отказ от выделенного состояния. Навигация с помощью пальца или стилуса существенно отличается от навигации с помощью

мыши, необходимо использовать графические средства для того, чтобы продемонстрировать возможность управления элементом.

3. Использование хорошо читаемого крупного шрифта.

4. Отказ от перегруженности интерфейса приложения, вокруг активных элементов экрана необходимо оставлять пустое пространство, чтобы уменьшить количество ложных срабатываний и уменьшить ошибки взаимодействия.

5. Простая логика переходов между экранами, уменьшение областей с прокруткой.

На основании изложенных подходов был спроектирован прототип интерфейса для программы Skype. Логика размещения элементов и переходов показана на рисунке.

Вывод. При переносе desktop-версий программ на мобильные устройства необходимо учитывать особенности и ограничения подобных устройств, что требует пересмотра, как логики работы программы, так и разработки или адаптации интерфейса специально для мобильных версий. Разработка интерфейса приложения с учетом особенностей мобильных платформ позволяет существенно облегчить пользователю взаимодействие с программой.

Литература

1. Гращенко Л.А. Определение услуг защиты и защищенности графического интерфейса пользователя // Техника и технология связи. Сборник докладов – СПб: Санкт-Петербургский государственный университет телекоммуникаций им. М.А. Бонч-Бруевича, 2000. – С. 391 – 393. **2. Першиков В. И., Савинков В. М.** Толковый словарь по информатике / Рецензенты: канд. физ.-мат. наук А. С. Марков и д-р физ.-мат. наук И. В. Поттосин. — М.: Финансы и статистика, 1991. — 543 с. **3. Фисун А.П., Гращенко Л.А.** и др. Теоретические и практические основы человеко-компьютерного взаимодействия: базовые понятия человеко-компьютерных систем в информатике и информационной безопасности: Монография / Под ред. д.т.н. А.П.Фисуна; Орловский государственный университет. – Орел, 2004. – 169 с.: ил. **4. Гращенко Л.А.** Обобщенная модель угроз информационной безопасности визуальных интерфейсов пользователя // Информационные системы и технологии. - 2006. - №1. - С. 41-45.

Мосейчук Ю.Б. Особливості розробки інтерфейсів додатків мобільних пристроїв

У статті розглядаються особливості розробки інтерфейсів додатків мобільних пристроїв.

Ключові слова: розробки інтерфейсів, додатки мобільних пристроїв

Мосейчук Ю.Б. Особенности разработки интерфейсов приложений мобильных устройств

В статье рассматриваются особенности разработки интерфейсов приложений мобильных устройств.

Ключевые слова: разработки интерфейсов, приложения мобильных устройств

Moseichuk Y.B. Features interface design mobile applications

The peculiarities of the development of application interfaces of mobile devices.

Keywords: interface design, mobile application

УДК 007:004.439:004.5

А.С. Черніков

МОДЕЛЮВАННЯ САМОЛОКАЛІЗАЦІЇ МОБІЛЬНОГО АГЕНТА У ТОПОЛОГІЧНОМУ ОПЕРАЦІЙНОМУ СЕРЕДОВИЩІ

Сьогодні у світі відбувається інтенсивний розвиток робототехніки. Роботи класифікуються за багатьма ознаками, у тому числі, за умовами їх функціонування. Для функціонування у недетермінованих середовищах, якими є природні середовища або ті, що створюються аварійними ситуаціями, нині розвивається особливий клас робототехнічних систем, які в технічній літературі називаються мобільними роботами. Відмінною рисою мобільних роботів є наявність спроможності самостійно орієнтуватися і пересуватися у навколишньому середовищі.

У категоріях однієї із класичних проблем теоретичної кібернетики – проблеми взаємодії системи, що управляє, і системи, якою управляють, – мобільного робота називають мобільним агентом (МА). Середовище, в якому пересувається мобільний агент, називають операційним середовищем (ОС).

При плануванні поведінки мобільних агентів під час пересування виникають три взаємопов’язані фундаментальні задачі:

– задача побудови моделі невідомого середовища;

– задача визначення положення мобільного агента у відомому середовищі – задача самолокалізації;

– задача перевірки відповідності невідомого середовища і її моделі.

Центральною із цих задач є *задача самолокалізації*, оскільки від її розв’язання залежить розв’язання двох інших задач.

Для завдання операційного середовища в теорії геоінформаційних систем існує декілька підходів. Топологічні моделі, наприклад, дозволяють представити елементи карти середовища або усю

карту у вигляді графу. Прикладом такого графу є схема метро (рис. 1). Вершини графа, який описує операційне середовище мобільного агента, відповідають перетинам шляхів, ребра такого графа описують ділянки шляхів; довжина ребер при цьому може не нести жодного інформаційного навантаження. Операційне середовище мобільного агента, яке задано топологічною моделлю, будемо називати топологічним операційним середовищем (ТОС).

Сформулюємо задачу самолокалізації мобільного агента у топологічному операційному середовищі.

Нехай МА має карту ТОС. Карта середовища представлена скінченним неорієнтованим графом з поміченими вершинами. Вершини цього графу помічені заздалегідь і МА не змінює ці мітки. Спочатку МА знаходиться у довільній вершині графу. Він спостерігає мітки суміжних вершин і може пересуватись із поточної вершини у будь-яку суміжну вершину. Необхідно розробити алгоритм пересувань МА по графу так, щоб на основі аналізу міток вершин, по яких МА пересувається, можна було б визначити початкову вершину знаходження МА.

Рис. 4. Приклад топологічної моделі середовища

Основою комп'ютерної моделі самолокалізації мобільного агента є алгоритм, який базується на понятті графу пар, який відповідає заданому скінченному неорієнтованому графу, і понятті ідентифікатора вершини графу. Наприклад, графу G (рис. 2) відповідає граф пар $D(G)$ (рис. 3).

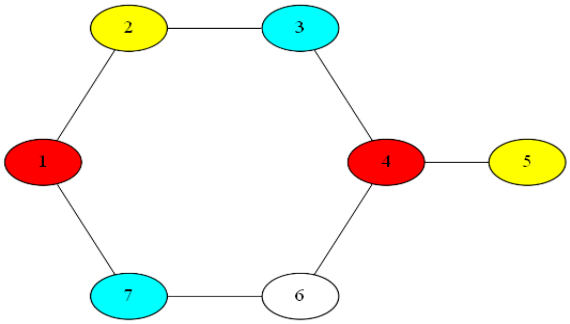


Рис. 5. Граф G

Для моделювання обрано інтегроване середовище розробки Microsoft «Visual Studio 2010» і мова програмування C++.

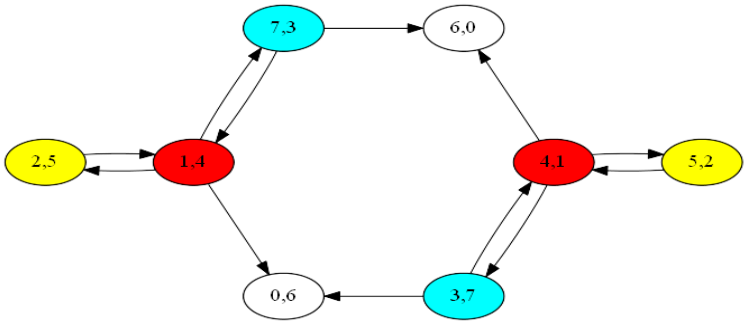


Рис. 6. Граф пар D(G)

В розробленому додатку у розпорядженні мобільного агента знаходиться карта топологічного простору, яка представлена у пам'яті комп'ютера матрицею суміжності вершин неорієнтованого скінченного поміченого графу.

Усі вершини графу помічаються за допомогою «жадібного» алгоритму. За вже поміченим графом будується матриця суміжності графу пар. І, нарешті, на основі графу пар модель будує ідентифікатори для кожної вершини графу.

Спочатку мобільний агент розміщується у випадковій вершині графу. Аналізуючи колір вершини, де він знаходиться, та перелік усіх ідентифікаторів вершин, агент обробляє лише ідентифікатори, які починаються з мітки вершини, де він знаходиться. Далі МА перевіряє шлях, описаний ідентифікаторами. У випадку неможливості пройти повністю шлях, описаний ідентифікатором, МА повертається у початкову вершину та починає перевіряти наступний ідентифікатор. Коли мобільному агенту вдається повністю пройти шлях, він, завдяки властивостям ідентифікаторів вершин, може однозначно відповісти на питання, куди його було розміщено спочатку.

Головне вікно розробленого додатку представлено на рис. 4.

Задати неорієнтований граф у додатку можна двома способами: виконати генерацію випадковим чином (кнопка «Генерація») або

завантажити заздалегідь підготовлений у текстовому форматі граф (кнопка «Завантажити граф»). Зберегти графи можна як у тестовому форматі (кнопка «Сохранить граф»), так і у форматі для генерації зображення у програмі GraphViz (кнопки «Граф» і «Граф пар» у компоненті «Сохранить для GrafhViz»).

У додатку передбачено також відображення кількості вершин, кількості ребер та списку ідентифікаторів усіх вершин графу.

Процес самолокалізації запускається кнопкою «Самолокалізація», результати розв'язання задачі відображаються у полях «Куда отправили» и «Где локализовался».

Додаток має невеликий розмір (1.02 Мб) та не потребує інсталяції; для його роботи на персональному комп'ютері необхідна наявність встановленого NET Framework 4 та набору бібліотек Microsoft Visual C++ 2010 Redistributable.

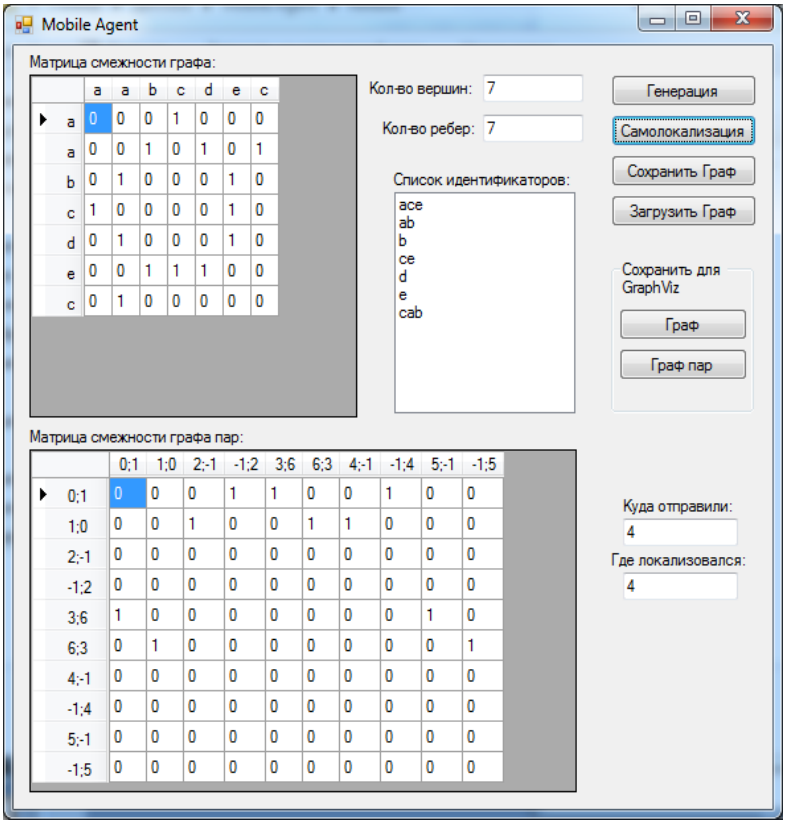


Рис. 7. Головне вікно додатку

За результатами дослідження можна зробити такі висновки:

- розроблено алгоритм побудови слів у алфавіті міток, які розрізняють пари однаково помічених вершин. Алгоритм є реалізацією методу графу пар;

- у процесі реалізації алгоритму виявлено необхідність попередньої обробки вихідного графу, а саме виділення вершин з унікальними мітками;

- метод графу пар було скориговано з урахуванням його програмної реалізації, а саме, додатково виконується перевірка ідентифікатора під час процесу самолокалізації;

- за допомогою графу пар розроблено алгоритм самолокалізації МА на графі його операційного середовища.

Література

1. Сапунов С.В. Анализ графов с помеченными вершинами: Дисс. канд. физ.-мат. наук: 27.09.07. – Донецк.: 2007. – 150 с. **2. С.В. Сапунов** О методе построения отношения неотличимости помеченных графов // Труды Института Прикладной Математики и механики НАН Украины: в 16 т. 2008. С 179-189. **3. И.С. Грунский, С.В. Сапунов** Идентификация вершин помеченных графов // Труды Института Прикладной Математики и механики НАН Украины: в 21 т. 2010. С 86-97. **4. Р. Хагарти** Дискретная математика для программистов. – М.: Техносфераб 2003. – 320с. **5. Dudek G., Jenkin M., Milios E. Wilkes D.** Map validation and Robot Self-Location in a Graph-Like World // Robotics and Autonomous Systems. – 1997. – Vol. 22, № 2. – P. 159–178.

Черніков А. С. Моделювання самолокалізації мобільного агента у топологічному операційному середовищі.

У цій статті описано постановку задачі самолокалізації мобільного агенту та процес її розв’язання, описані можливості розробленого автором засобами інтегрованого середовища Visual Studio 10 та мови програмування С++ додатку для моделювання самолокалізації мобільного агента у топологічному операційному середовищі.

Ключові слова: мобільний агент, операційне середовище, граф, граф пар, самолокалізація, моделювання, ідентифікатор.

Черников А. С. Моделирование самолокализации мобильного агента в топологической операционной среде.

В этой статье описаны постановка задачи самолокализации мобильного агента и процесс ее решения, описаны возможности разработанного автором средствами интегрированной среды Visual Studio 10 и языка программирования С++ приложения для моделирования самолокализации мобильного агента в топологической операционной среде.

Ключевые слова: мобильный агент, операционная среда, граф, граф пар, самолокализация, моделирование, идентификатор.

Chernikov A. Modeling of self-localization of a mobile agent in the topological operating environment.

This paper describes the statement of the problem of self-localization of a mobile agent and its solution, the capabilities of the softwear application for modeling the self-localization of a mobile agent in the topological operating environment developed by the author in the programming language C++ in the IDE Visual Studio 10.

Keywords: mobile agent, the operating environment, graph, pairs graph, self-localization, modeling, ID.

ФІЗИКА ТА МАТЕМАТИКА

УДК 004.421:004.65

Т. Ю. Романченко, Э. В. Черенкова

ИСПОЛЬЗОВАНИЕ МАТЕМАТИЧЕСКОГО ПАКЕТА MATHCAD ПРИ ИЗУЧЕНИИ МАТЕМАТИКИ В СТАРШЕЙ ШКОЛЕ

Постановка проблемы. Известно, что одной из задач изучения математики является приобретение математических знаний в их диалектическом единстве с другими научными дисциплинами, которые изучаются в школе, а также установление межпредметных связей. Из этого следует, что при изучении какой-либо темы учитель просто обязан не только изложить теорию, но и привести конкретные примеры применения изучаемого объекта. Очень важно, чтобы такая связь не была односторонней, а потому необходимо также привлекать технологии, более характерные для других предметов.

Достаточное количество отечественных вузов уже не первый год используют различные математические пакеты (Maple, Mathcad, Mathamatica, Mathlab и др.) при обучении студентов по современным математическим курсам. К сожалению, в школах ситуация является в точности противоположной. Учителя не заинтересованы в их применении на уроках информатики и математики. Следовательно, школьники не имеют никакого представления о современных математических пакетах.

На данный момент в школах на уроках математики не изучаются математические пакеты. Причиной этого есть то, что в программе по математике, утвержденной Министерством образования и науки Украины, не предусмотрены на это часы (уровень стандарта [1]) или же это предоставляется на выбор учителя (для классов с углублённым изучением [2]). Выходом из этой ситуации могло бы быть изучение математических пакетов на уроках информатики, но это так же не предусмотрено по программе [3].

В педагогических публикациях, повествующих о различных информационных технологиях на уроках математики особый интерес представляют исследования, направленные на изучение применения различных математических пакетов в учебном процессе (Скрыльникова Е. В, Королёва Е. С, Королёв С. О, Пипич П. В., Семёнова Г. А. и др.). Спектр различных тематик статей очень широк: от использования математических пакетов в личных, преподавательских целях [4] до постоянного применения в масштабах гимназии [5] и конкретных примеров проведения уроков, планов-конспектов[6; 7].

Таким образом, **цель статьи** – проанализировать использование математического пакета Mathcad на уроках математики в старших классах.

Из всего многообразия математических пакетов наиболее приемлемым для школьной программы является Mathcad. Mathcad – это система компьютерной алгебры из класса систем автоматизированного проектирования, ориентированная на подготовку интерактивных документов с вычислениями и визуальным сопровождением, отличается легкостью использования и применения для коллективной работы. Отличительной чертой Mathcad является интуитивный и простой для использования интерфейс пользователя. Для ввода формул и данных можно использовать как клавиатуру, так и специальные панели инструментов. Работа осуществляется в пределах рабочего листа, на котором уравнения и выражения отображаются графически, в противовес текстовой записи в языках программирования. При создании документов-приложений используется принцип WYSIWYG (What You See Is What You Get — «что видишь, то и получаешь»)[8]. Все вышеперчисленные харктеристики делают Mathcad достаточно удобным для обучения, вычислений и расчетов.

С помощью Mathcad можно повысить наглядность а, следовательно, и эффективность, изучения ряда тем курса алгебры и начал анализа и геометрии: построение графиков функций, решение систем уравнений и неравенств, нахождение площадей фигур, ограниченных графиками функций, построение сечений геометрических тел, вычисление объемов тел вращения и т.п.

Так как Mathcad не входит в обязательный курс математики, то работу с ним целесообразно поделить на две части:

1. Изучение функций программы;
2. Выполнение различных заданий из курса алгебры и начал анализа и геометрии.

Использование Mathcad даст возможность ученикам:

1) Сократить время на вычисления там, где они не являются основной задачей.
2) Самостоятельно проверять правильность выполнения своих действий, выполненных «вручную» по преобразованию выражений, решению уравнений и систем уравнений, неравенств и их систем, вычислению интегралов и производных функций.
3) Выполнять построения графиков функций в качестве, которое без помощи компьютера достичь нельзя.
4) Строить графики, а также изучать свойства функций, заданных в неявном виде.
5) Рассматривать примеры и задачи, выходящие за рамки школьного курса.
6) К моменту начала обучения в ВУЗе свободно владеть современным математическим программным продуктом.

Таким образом, применение Mathcad позволит ученикам всесторонне рассмотреть поставленную задачу, находя новизну в каждом решении, выявить все взаимосвязи в условии, произвести во время решения одной и той же задачи большее количество различных математических операций, приводящих к одному и тому же результату., а также развить у школьников интерес к дальнейшим исследованиям.

В свою очередь учителя получат следующие возможности:

1) Повысить интерес к математике среди учеников.

2) Перевести общение с учениками на более современный уровень [5].

3) Существенно облегчить ежедневно выполняемую работу: проверку домашних заданий, составление однотипных и одинаковых по степени трудности вариантов к контрольным работам и др. [4].

4) При всех положительных качествах и уникальных возможностях, которые предоставляются как ученикам, так и учителям при использовании Mathcad, необходимо найти оптимальный объем применения на уроках данного математического пакета. Учитель должен понимать, что Mathcad может быть только вспомогательным средством для наглядного представления и поиска информации или же для увеличения разнообразия и оптимизирования учебных задач. А потому не следует слишком увлекаться умением свободно оперировать указанным программно-техническим средством в ущерб основной задаче изучения математики – отработке у школьников соответствующих навыков мышления.

5) Учитывая это, при работе с Mathcad учитель должен подбирать задачи несколько исследовательского и творческого характера, либо трудно осуществимые на бумаге. Учитель должен научить детей не приводить задачи к «правильным» численным ответам, а мыслить правильными алгоритмами решения, складывая в голове цельную картину поставленной проблемы.

Таким образом, использование Mathcad на уроках математики позволит существенно повысить наглядность, а значит и доступность излагаемого материала. Постоянная работа с Mathcad даст возможность поднять интерес к математике среди учащихся. Перспективами дальнейшего исследования считаем изучение основ Mathcad через использование коротких учебных avi-фильмов.

Литература

1. Навчальна програма з математики для учнів 10-11 класів загальноосвітніх навчальних закладів (рівень стандарту) . – [Електронний ресурс]. — Режим доступа : http://old.mon.gov.ua/images/education/average/prog12/matem_st.pdf.

2. Навчальна програма з математики для учнів 10-11 класів загальноосвітніх навчальних закладів (для класів з поглибленим вивченням математики) . – [Електронний ресурс]. – Режим доступа :

http://old.mon.gov.ua/images/education/average/prog12/matem_pogl.pdf.
3. Навчальна програма з інформатики для учнів 10-11 класів загальноосвітніх навчальних закладів (рівень стандарту) . – [Електронний ресурс]. – Режим доступа : http://old.mon.gov.ua/images/education/average/prog12/inf_st.doc.
4. Скрыльникова Е. В. Использование системы Mathcad в работе школьного учителя/ Е.В.Скрыльникова //Педагогическая информатика. – 2001. – № 4 – С. 15 – 16. **5. Королёва Е. С**. Mathcad – рабочая среда современного школьника/Е.С. Королёва, С. О. Королёв// Компьютерные инструменты в образовании. – 2007. – № 5 – С. 55 – 61. **6. Mathcad** в преподавании физики : Статьи Фестиваля «Открытый урок»: – [Електронний ресурс]. – Режим доступа к статье : http://festival.1september.ru/articles/103956/. **7. План-конспект** урока „Некоторые графические возможности инструментального математического пакета MathCAD" : Статьи Фестиваля «Открытый урок»: – [Електронний ресурс]. – Режим доступа к статье : http://festival.1september.ru/articles/533465/. **8. Mathcad** – Википедия: — [Електронний ресурс]. – Режим доступа : http://ru.wikipedia.org/wiki/Mathcad. **9. Развитие** творческих способностей обучающихся посредством использования математических редакторов на уроках математики : Статьи Фестиваля «Открытый урок»: »: – [Електронний ресурс]. – Режим доступа к статье : http://festival.1september.ru/articles/596466.

Романченко Т. Ю., Черенкова Е. В. Використання математичного пакету Mathcad при вивченні математики у старшій школі

Стаття присвячена розробці концепції використання математичного пакету Mathcad з метою підвищення наочності та доступності занять, а також підвищення інтересу до предмету та розвитку творчого мислення учнів.

Ключові слова: Mathcad, наглядність, доступність, інноваційні технології.

Романченко Т. Ю., Черенкова Э. В. Использование математического пакета Mathcad при изучении математики в старшей школе

Статья посвящена разработке концепции использования математического пакета Mathcad с целью повышения наглядности и доступности занятий, а также повышения интереса к предмету и развития творческого мышления учащихся.

Ключевые слова: Mathcad, наглядность, доступность, инновационные технологии.

Romanchenko T., Cherenkova E. Using mathematical package Mathcad in the study of mathematics in high school

The article is devoted to the development of the concept of using mathematical package Mathcad to increase the visibility and accessibility of training, as well as increasing interest in the subject and the development of creative thinking of students.

Keywords: Mathcad, visibility, accessibility, innovative technology.

УДК 371.13

В.П. Сєдих

ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ НА УРОКАХ МАТЕМАТИКИ В ПОЧАТКОВІЙ ШКОЛІ

У сучасній школі дедалі важливішого значення набуває інформатизація освіти, підготовка підростаючого покоління до життя в суспільстві, де інформація швидко змінюється й оновлюється. Учителі повинні готувати учнів для роботи в умовах прискореного розвитку сучасного суспільства. Школярам потрібно не тільки мати необхідні вміння й навички, а й володіти новим стилем мислення, креативним підходом до справи, активністю й ініціативою. Вони повинні вміти орієнтуватися в потоці різноманітної інформації, обирати для себе головне і значиме.

Проблема широкого застосування сучасних інформаційних технологій у сфері освіти в останнє десятиліття викликає підвищений інтерес у вітчизняній педагогічній науці. Великий внесок у вирішення проблеми сучасної інформаційної технології навчання внесли російські та зарубіжні вчені: Г.Р. Громов, В.І. Гриценко, В.Ф. Шолоховіча, О.І. Агапова, О.А Кривошеєв, С. Пейперт, Г. Клейман, Б. Сенді, Б. Хантер та ін. [1, с. 10-12].

Учителі початкових класів використовують у методиці викладання математики різні методи навчання, що вважаються класичними усний виклад; демонстрування та ілюстрування; бесіди; вправляння; виробничо-практичні методи тощо. Однак ці методи всіх проблем та завдань не вирішують. Існує ще багато факторів, що суттєво впливають на якість навчання. Серед них не менш важливе значення має використання ІКТ.

Застосування комп'ютерних технологій навчання дозволяє видозмінювати весь процес викладання, реалізовувати модель особистісно-орієнтованого навчання, інтенсифікувати заняття, а головне - удосконалювати самопідготовку учнів. Позитивні сторони використання мультимедійних технологій на уроках математики полягають в тому, що якою б складною і не цікавою була тема уроку, вона стане цікавою школяреві, якщо навчальний матеріал на екрані

виражений в кольорах, зі звуком та іншими ефектами. Презентація з теми уроку в процесі пояснення нового матеріалу дозволяє вчителю не робити записів на дошці, а це означає, що у нього залишається більше часу на закріплення матеріалу, при виконанні практичних дій проявляється самоконтроль.

За цілями і завданнями навчальні комп'ютерні програми поділяються на ілюстраційні, консультуючі, програми-тренажери, програми навчального контролю, ігрові, операційні середовища тощо.

Одні з них призначені для закріплення знань і вмінь, інші орієнтовані на засвоєння нових понять. Є навчальні програми, які дозволяють учневі стати безпосередніми учасником відкриттів, композитором або художником.

Більшими можливостями володіють програми, які реалізують проблемне навчання. Особливо корисні програми, що моделюють і аналізують конкретні ситуації, тому що вони сприяють формуванню вміння приймати рішення в різних обставинах.

Нерідко в одній програмі з'єднуються кілька режимів (навчання, тренування, контролю). Таким чином, комп'ютер у навчальному процесі виконує *кілька функцій:* слугує засобом спілкування, сприяє створенню проблемних ситуацій, партнером, інструментом, джерелом інформації, контролює дії учня й надає йому нові пізнавальні можливості.

Ігрові програми сприяють формуванню мотивації навчання, стимулюють ініціативу й творче мислення, розвивають уміння спільно діяти, підкоряти своїм інтересам. Гра дозволяє вийти за рамки певного навчального предмета, спонукуючи учнів до здобутку знань у суміжних областях і практичній діяльності. В педагогічному процесі вибір способу використання комп'ютера прямо залежить від дидактичного завдання [3, с. 32-33]

Сучасний учитель має зважати на те, що інформаційно-комунікаційні технології навчання міцно ввійшли в життя. Використання нових інформаційних технологій розширює рамки освітнього процесу, підвищує його практичну спрямованість, сприяє підвищенню мотивації учнів в освітньому процесі, розвитку інтелектуальних, творчих здібностей учнів, їхніх умінь самостійно здобувати нові знання та створення умови для їх успішної самореалізації в майбутньому. Але при використанні комп’ютера на уроці педагогу не слід забувати про санітарно-гігієнічні вимоги до проведення занять з використанням ІКТ та тривалість застосування технічних засобів. Якщо протягом тижня кількість уроків із застосуванням ТЗН не повинна перевищувати 4 уроки, то безперервна робота з екраном ПК не повинна тривати більше 15 хв. для учнів 2 – 4 класів, не більше 10 хв. для першокласників [2, с. 65-69].

На уроках математики за допомогою комп'ютера можна вирішити проблему дефіциту рухливої наочності, коли діти під керівництвом вчителя на екрані монітора порівнюють способом накладання геометричні фігури, аналізують взаємовідносини множин.

Комп'ютер є найпотужнішим стимулом для творчості дітей. Екран притягує увагу, яку часом не можна добитися при фронтальній роботі з класом. На екрані можна швидко виконати перетворення в деформованому тексті, перетворивши розрізнені пропозиції в зв'язний текст. Але для того, щоб учні початкової школи могли відповідно до своїх бажань використовувати комп'ютер як помічник у навчанні, треба подбати про універсальність їхніх користувальницьких навичок. Діти мають право користуватися сучасними засобами праці вже сьогодні. За допомогою сучасних технічних і аудіовізуальних засобів та інтенсивних методів навчання можна зацікавити учнів, полегшити засвоєння матеріалу. На уроці використовуються два види інформаційних технологій: презентації та слайд-шоу. Вони дозволяють наочно і доступно пояснити дітям матеріал. Презентація є інформаційним забезпеченням фронтальної роботи вчителя з класом і складається зі слайдів. Основні форми цієї інформації - текст, малюнки, креслення.

Форми і місце використання презентації на уроці залежать від змісту цього уроку, від мети, яка ставиться на уроці. При вивченні нового матеріалу використання презентації дозволяє ілюструвати навчальний матеріал. При проведенні усних вправ презентація дає можливість оперативно пред'являти завдання. Навчальна презентація може представляти собою конспект уроку. У цьому випадку вона складається з основних складових традиційного уроку: зазначається тема, мета, план роботи на уроці, ключові поняття, домашнє завдання [4, с. 35].

На даний момент існує безліч комп’ютерних програм, які розвивають дитячий інтерес до навчання. Є підстави вважати, що комп'ютер дозволить більш глибоко розвити резерви дитини, дасть змогу вчителю працювати творчо, ініціативно, з більшою професійною майстерністю. В наш час в Україні склалася досить суперечлива ситуація: незважаючи на позитивні фактори, комп’ютер при вивченні навчальних предметів початкової школи, на жаль, використовується рідко, безсистемно і переважно з метою контролю знань, не торкаючи, як правило, інших етапів навчання. Основними причинами цього можна вважати недостатній рівень фінансування загальноосвітньої школи, недолік на ринку педагогічних програмних засобів, що відповідають дидактичним вимогам, відсутність методичних розробок щодо використання комп’ютера в цілому (і конкретних програм навчального призначення, зокрема), санітарно-гігієнічні умови використання персонального комп’ютера в початковій школі, а також невисокий рівень інформаційної грамотності вчителів початкової школи. У зв’язку з цим особливу значимість для інформатизації початкової освіти в нашій країні отримують дослідження, спрямовані на виявлення факторів ефективного використання комп’ютера в навчанні. Зростаючий інтерес до використання ІКТ не повинен зменшувати увагу педагога до особистості учня, навпаки, саме комп’ютеризація освіти звільняє вчителя від рутинної роботи з трансляції навчального матеріалу, дозволяючи йому

більше орієнтуватися на формуванні в дитини фундаментальних основ в області духовного життя особистості [5, с. 26].

Отже завдяки використанню комп'ютерних засобів навчання учні отримують змогу користуватись більшою кількістю інформації, що задовольняє їх потреби, збільшує мотивацію. За таких умов, вчитель може звільнити час та подати більше матеріалу, зробити належним чином диференційованість та індивідуалізацію в навчанні. Застосування мультимедійних засобів покращує роботу класу в цілому та кожного учня окремо за певних умов (правильна організація навчального процесу, використання сучасних засобів, методичної літератури тощо). Для того, щоб використання інформаційних технологій на уроці було значимим, необхідно дотримуватись ряду вимог: правильно визначати дидактичне значення і місце комп'ютера на уроці; застосовувати продумані форми уроку; раціонально поєднувати різні форми й методи використання ІКТ; ураховувати вікові особливості та дотримуватись санітарних норм під час роботи з комп'ютером.

Література

1. **Тимчина Н.** Використання інформаційно-комунікаційних технологій у початковій школі: переваги та недоліки / Н. Тимчина. – К. : Наук. світ, 2011. – 35 с. **2. Антипова О.** У пошуках нестандартного уроку / О. Антипова, В**.** Паламарчук. – К. : Рад. школа, 1991. - № 1. – 69 с. **3. Кривлюк О.П.** Використання комп'ютера на уроках математики в початковій школі: Комп'ютер в школі та сім'ї / О.П. Кривлюк. – Л. : – 2000. – № 4. – 33 с. **4. Молоков Ю.Г., Молокова А.В.** Актуальні питання інформатизації освіти / Ю.Г. Молоков, А.В. Молокова. – Львів : Освітні технології : наук. ст. Вип.1, 2011. – 35 с. **5. Белавина І.Г.** Сприйняття дитиною комп'ютера й комп'ютерних ігор. / І.Г. Белавина. – К. : - 1998. - №9 - 38 с.

Сєдих В.П. Використання інформаційних технологій на уроках математики в початковій школі

У цій статті визначається значення використання комп'ютера як засобу навчання, доцільність засобів ІКТ в процесі навчання математики. Наведені деякі види програм, які розвивають дитячий інтерес до навчання. Сформульовані основні санітарно-гігієнічні вимоги до проведення занять з використанням ІКТ та тривалість застосування технічних засобів.

Ключові слова: інформатизація освіти, операційні середовища, проблемне навчання.

Седых В.П. Использование информационных технологий на уроках математики в начальной школе

В этой статье определяется значение использования компьютера как средства обучения, целесообразность средств ИКТ в процессе

обучения математике. Приведены некоторые виды программ, которые развивают детский интерес к учебе. Сформулированы основные санитарно-гигиенические требования к проведению занятий с использованием ИКТ и длительность применения технических средств.

Ключевые слова: информатизация образования, операционные среды, проблемное обучение.

Sedykh V.P. Information technology for mathematics lessons in elementary school

In this article the value of using the computer as a medium of instruction, appropriateness of ICT in teaching mathematics. These are some types of programs that develop children's interest in learning. Formulated basic hygiene requirements for the classes using ICT and duration of use of technology.

Keywords: informatization of education, operating environments, problem teaching.

УДК 517.946

П.Ю. Шопін

ІНТЕГРАЛЬНІ НОРМИ ФУНКЦІЙ ЗІ СТУПЕНЕВОЮ ПОВЕДІНКОЮ НА НЕСКІНЧЕННОСТІ

Чимало питань різних розділів математики, зокрема, теорії нелінійних параболічних рівнянь, пов’язано з функціями, що характеризуються ступеневою поведінкою на нескінченності.

Для опису таких функцій використовують різноманітні функціональні простори. Так для функцій, що локально інтегруються за Лебегом, часто викистовують простір індукований нормою

$$f \;= \sup_{R\geq 1} R^{\theta}\left(|B_R(0)|^{-1} \int\limits_{B_R(0)} |f|^q \, dy \right)^{\frac{1}{q}}, \quad (1)$$

де $0 \leq \theta \leq \frac{N}{q}, q \geq 1, \quad f \in L_q^{loc}(\mathbf{R}^N), N \geq 1, \quad B_R(x_0) = \left\{ x \in \mathbb{R}^N : |x - x_0| < R \right\}$,

$|E|$- міра Лебега множини E на $\mathbf{R}^N$. Таку норму, зазвичай називають «нормою Морі». Ця норма досить добре підходить для характеристики початкових функцій задачі Коші для нелінійних вироджених параболічних рівнянь.

В роботі [1], для дослідження задачі Коші для нелінійних рівнянь з виродженням по просторовим змінним була введена норма

$$|||f||| = \sup_{x\in R^N}(1+|x|)^{\theta}\left(|B_{r(|x|)}(x)|^{-1}\int\limits_{B_{r(|x|)}(x)}|f|^q\,dy\right)^{\frac{1}{q}}, \quad (2)$$

де $0\le\theta\le\dfrac{N}{q}, q\ge 1,\ r(s)=(1+s)^{\alpha},\ s\ge 0,\ \alpha\in(-\infty,1)$.

Але для деяких рівнянь з виродженням по просторовим змінним більш сприятливою є норма

$$[f] = \sup_{x\in R^N}\sup_{R>r(|x|)}(R+|x|)^{\theta}\left(|B_R(x)|^{-1}\int\limits_{B_R(x)}|f|^q\,dy\right)^{\frac{1}{q}}, \quad (3)$$

де $0\le\theta\le\dfrac{N}{q}, q\ge 1,\ r(s)=(1+s)^{\alpha},\ s\ge 0,\ \alpha\in(-\infty,1)$. Норма (3) була запроваджена в роботі [2] також було показано, що при $\alpha\in(-\infty,1)$ норми (1) та (3) еквівалентні, але випадок $\alpha\ge 1$ вивчено не було. Отже, основна мета даної роботи дослідити властивості норми (3) для $\alpha>1$ (випадок $\alpha=1$ потребує дещо інших міркувань ніж ті, що використовуються нижче і тому залишається поза межами даного дослідження).

Як з'ясувалось, еквівалентність норм (2) та (3) порушується при $\alpha>1$, але у цьому випадку норма (3) еквівалентна нормі (1). Точні формулювання цих результатів містять у собі наступні теореми.

Теорема 1. *За умов* $0\le\theta\le\dfrac{N}{q},\ \alpha>1,\ q\ge 1$ *справедливі наступні твердження:*

1) $\left[(1+|x|)^{-s}\right]<\infty$, *лише при* $s\ge\theta$,

2) $|||(1+|x|)^{-s}|||<\infty$, *лише при* $s\ge\dfrac{\theta}{\alpha}$

і, як наслідок цього, норми (3) і (2) не еквівалентні.

Теорема 2. *Нехай* $0\le\theta\le\dfrac{N}{q},\ \alpha>1,\ q\ge 1,\ f\in L_q^{loc}(\mathbf{R}^N)$ *і* $|||f|||<\infty$ *або* $[f]<\infty$ *тоді виконані нерівності*

$$C_*\,|||f|||\le[f]\le C^*\,|||f|||, \quad (4)$$

де $C^*=\left(1+\dfrac{(\alpha-1)^{\alpha-1}}{\alpha^{\alpha}}\right)^{\frac{N}{q}}$ *та* $C_*=1$, *причому ці константи є точними.*

§1 **Доведення Теореми 1**

Для доведення нам знадобиться наступна лема.

Лема1. *Нехай* μ *- довільне дійсне число, тоді виконана нерівність*

$$C_1 R^N \left(R+1+|x_0|\right)^{\mu} \le \int\limits_{B_R(x_0)} \left(1+|x|\right)^{\mu} dx \le C_2 \left(R+1+|x_0|\right)^{\mu},$$

де сталі C_1 *і* C_2 *залежать тільки від* μ *і* N.

Доведення. Твердження леми доводиться так же як і лема 1 в [2]

Далі, використовуючи лему, отримаємо

$$C_1^{\frac{1}{q}} \sup_{x\in R^N} \sup_{R>r(|x|)} (R+|x|)^{\theta} \left(R+1+|x|\right)^{-s} \le \left[(1+|x|)^{-s}\right] \le C_2^{\frac{1}{q}} \sup_{x\in R^N} \sup_{R>r(|x|)} (R+|x|)^{\theta} \left(R+1+|x|\right)^{-s}.$$

Оскільки $\alpha > 1$, то $|x| < r\left(|x|\right)$, значить,

$$C_1^{\frac{1}{q}} 2^{-s} \sup_{R\ge 1} R^{\theta-s} \le \left[(1+|x|)^{-s}\right] \le C_2^{\frac{1}{q}} 2^{\theta} \sup_{x\in R^N} \sup_{R>r(|x|)} R^{\theta-s}.$$

З того, що $\sup\limits_{x\in R^N} \sup\limits_{R>r(|x|)} R^{\theta-s} \le \sup\limits_{R\ge 1} R^{\theta-s}$ остаточно маємо

$$C_1^{\frac{1}{q}} 2^{-s} \sup_{R\ge 1} R^{\theta-s} \le \left[(1+|x|)^{-s}\right] \le C_2^{\frac{1}{q}} 2^{\theta} \sup_{R\ge 1} R^{\theta-s}.$$

Таким чином $\left[(1+|x|)^{-s}\right] < \infty$ тільки при $s \ge \theta$.

Аналогічно отримаємо оцінку для норми (2)

$$C_1^{\frac{1}{q}} \sup_{x\in R^N} (1+|x|)^{\theta} (1+r\left(|x|\right)+|x|)^{-s} \le \left\|\left|(1+|x|)^{-s}\right|\right\| \le C_2^{\frac{1}{q}} \sup_{x\in R^N} (1+|x|)^{\theta} (1+r\left(|x|\right)+|x|)^{-s}$$

.

З того, що $\alpha > 1$ витікає $1+|x| < r\left(|x|\right)$, значить,

$$C_1^{\frac{1}{q}} 2^{-s} \sup_{x\in R^N} (1+|x|)^{\theta-s\alpha} \le \left\|\left|(1+|x|)^{-s}\right|\right\| \le C_2^{\frac{1}{q}} \sup_{x\in R^N} \left(1+|x|\right)^{\theta-s\alpha}.$$

Таким чином $\left\|\left|(1+|x|)^{-s}\right|\right\| < \infty$ тільки при $s \ge \dfrac{\theta}{\alpha}$.

□

§2 **Доведення Теореми 2**

Справедливість першої частини нерівності (4) випливає з тривіальної рівності

$$(R+|x_0|)^{\theta} \left(\left|B_R(x_0)\right|^{-1} \int\limits_{B_R(x_0)} |f|^q \, dy \right)^{\frac{1}{q}} \Bigg|_{x=0} = R^{\theta} \left(\left|B_R(0)\right|^{-1} \int\limits_{B_R(0)} |f|^q \, dy \right)^{\frac{1}{q}},$$

при цьому цілком зрозуміло, що $C_* = 1$ є точною сталою.

Доведемо другу частину нерівності (2). Нехай $R \ge r\left(|x|\right), x \in \mathbb{R}^N$, тоді з того, що $B_R(x) \subset B_{R+|x|}(0)$ випливає

$$(R+|x|)^{\theta}\left(|B_R(x)|^{-1}\int\limits_{B_R(x)}|f|^q\,dy\right)^{\frac{1}{q}}\le$$

$$\le\frac{\left|B_{R+|x|}(0)\right|^{\frac{1}{q}}}{|B_R(x)|^{\frac{1}{q}}}\cdot(R+|x|)^{\theta}\left(\left|B_{R+|x|}(0)\right|^{-1}\int\limits_{B_{R+|x|}(0)}|f|^q\,dy\right)^{\frac{1}{q}}\le$$

$$\le\frac{\left|B_{R+|x|}(0)\right|^{\frac{1}{q}}}{|B_R(x)|^{\frac{1}{q}}}\cdot\sup_{R\ge1}R^{\theta}\left(|B_R(0)|^{-1}\int\limits_{B_R(0)}|f|^q\,dy\right)^{\frac{1}{q}}=\frac{\left|B_{R+|x|}(0)\right|^{\frac{1}{q}}}{|B_R(x)|^{\frac{1}{q}}}\cdot\ \|f\|\ .$$

Таким чином, нам необхідно показати, що для довільного $x\in\mathbb{R}^N$ і $R\ge r\left(|x|\right)$ виконана нерівність

$$\frac{\left|B_{R+|x|}(0)\right|^{\frac{1}{q}}}{|B_R(x)|^{\frac{1}{q}}}\le C^*.$$

Насправді, враховуючи, що $R>r(|x|)$ отримаємо

$$\frac{\left|B_{R+|x|}(0)\right|^{\frac{1}{q}}}{|B_R(x)|^{\frac{1}{q}}}=\left(\frac{|B_1(0)|\cdot\left(R+|x|\right)^N}{|B_1(0)|\cdot R^N}\right)^{\frac{1}{q}}=\left(1+\frac{|x|}{R}\right)^{\frac{N}{q}}\le\left(1+\frac{|x|}{(1+|x|)^{\alpha}}\right)^{\frac{N}{q}}\le$$

$$\le\left(1+\frac{(\alpha-1)^{\alpha-1}}{\alpha^{\alpha}}\right)^{\frac{N}{q}}=C^*$$

Покажемо, що C^* є точною константою.

Нехай $a\in\mathbf{R}^N$ і $|a|=\frac{1}{\alpha-1}$, тоді розглянемо функцію $\varphi(x)$ яка задовольняє наступним вимогам:

1) $\operatorname{supp}\varphi(x)\subset B_{r(|a|)}(a)$,

2) для малого $\varepsilon>0$ існує стала C_3 залежна тільки від θ, q, N така, що для довільного додатного $R\le r(|a|)+|a|$ виконано

$$\int\limits_{B_R(0)}\left|\varphi(x)\right|^q\,dx=C_3R^{N-\theta q+\varepsilon}.$$

Існування зазначеної функції $\varphi(x)$ випливає, наприклад, з таких міркувань: для радіально симетричної функції $f(x)=f(|x|)$, носій якої зосереджено у шарі $B_{r(|a|)}(a)$, здійснюючі перехід до сферичних координат, отримаємо

$\int\limits_{B_R(0)} \left| f\left(x\right)\right|^q dx = \int\limits_0^R |\ f(\rho)\ |^q\ d\rho \int\limits_{\Omega(a,\rho)} \left|J(\rho,\omega)\right| d\omega$,де $\Omega(a,\rho)$ область зміни кутових змінних ω, $J(\rho,\omega)$- якобіан переходу до сферичних координат. Оскільки $\int\limits_{\Omega(a,\rho)} \left|J(\rho,\omega)\right| d\omega$ залежить тільки від ρ і a, то можна покласти

$$\varphi(x) = |\ x\ |^{\frac{N-\theta q+\varepsilon-1}{q}} \left(\int\limits_{\Omega(a,|x|)} \left|J(|\ x\ |,\omega)\right| d\omega \right)^{\frac{-1}{q}}.$$

Із властивості 2) функції $\varphi\left(x\right)$ витікає, що

$$\left|\varphi\left(x\right)\right| = (r(|\ a\ |)+|\ a\ |)^{\theta} \left(\left|B_{|a|+r(|a|)}(0)\right|^{-1} \int\limits_{B|a|+r(|a|)(0)} \left|\varphi\left(x\right)\right|^q dy \right)^{\frac{1}{q}}, \quad (5)$$

а з властивості 1) отримуємо

$$\int\limits_{B|a|+r(|a|)(0)} \left|\varphi\left(x\right)\right|^q dy = \int\limits_{Br(|a|)(a)} \left|\varphi\left(x\right)\right|^q dy, \quad (6).$$

Також, неважко бачити, що

$$\left[\varphi\left(x\right)\right] \geq (r(|a|)+|a|)^{\theta} \left(\left|B_{r(|a|)}(a)\right|^{-1} \int\limits_{B_{r(|a|)}(a)} \left|\varphi\left(x\right)\right|^q dx \right)^{\frac{1}{q}}.$$

Таким чином, із останньої нерівності та співвідношень (5) і (6) маємо

$$\frac{\left[\varphi(x)\right]}{\varphi(x)} \geq \frac{(r(|a|)+|a|)^{\theta} \left(\left|B_{r(|a|)}(a)\right|^{-1} \int\limits_{B_{r(|a|)}(a)} \left|\varphi\left(x\right)\right|^q dx \right)^{\frac{1}{q}}}{(r(|\ a\ |)+|\ a\ |)^{\theta} \left(\left|B_{|a|+r(|a|)}(0)\right|^{-1} \int\limits_{B|a|+r(|a|)(0)} \left|\varphi\left(x\right)\right|^q dy \right)^{\frac{1}{q}}} =$$

$$\frac{\left|B_{|a|+r(|a|)}(0)\right|^{\frac{1}{q}}}{\left|B_{r(|a|)}(a)\right|^{\frac{1}{q}}} = \left(\frac{|\ a\ |+r(|\ a\ |)}{r(|\ a\ |)} \right)^{\frac{N}{q}} = C^*.$$

Література

1. Andreucci, D. and Di Benedetto, E. ‘On the Cauchy problem and initial traces for a class of evolution equations with strongly nonlinear sources’, *Annali della Scuola Normale Superiore di Pisa*, Classe di Scienze 4 serie, tome 18 (1991), pp. 363-441. **2. Мартыненко А. В.** , Тедеев А. Ф. ,

Шраменко В. Н. Задача Коши для вырождающегося параболического уравнения с неоднородной плотностью в классе медленно убывающих начальных функций // прийнята до публікації у журналі «Известия РАН. Серия математическая».

Шопін П.Ю. Інтегральні норми функцій зі ступеневою поведінкою на нескінченності.

У статті представленно дослідження еквівалентності інтегральних норм, які широко використовуються для дослідження функції зі ступеневою поведінкою на нескінченності.

Ключові слова: інтегральна норма, ступенева поведінка на нескінченності.

Шопин П.Ю. Інтегральные нормы функций со степенным поведением на бесконечности.

В статье представленно исследование эквивалентности интегральных норм, которые широкко используются для исследования функций со степенным поведением на бесконечности

Ключевые слова: интегральная норма, степенное поведение на бесконечности.

Shopin P.Y. Integral norms of functions with power-law behavior at infinity.

The article presents the study of equivalence of integral norms, that are widely used to study the functions with power-law behavior at infinity.

Keywords: integral norm, power-law behavior at infinity.

УДК 517.53:004.021:378.147.227:372.851

І. О. Хотіна

ВИКОРИСТАННЯ ПРОБЛЕМНИХ СИТУАЦІЙ НА УРОКАХ МАТЕМАТИКИ В ПРОЦЕСІ РОЗВИТКУ ТВОРЧОГО МИСЛЕННЯ МОЛОДШИХ ШКОЛЯРІВ

Основною цінністю гуманістичного особистісно-орієнтованого навчання виступає творчість як спосіб розвитку людини. Саме тому в даний час існує гостра соціальна потреба у творчості і творчих особистостях. Розвиток у школярів творчого мислення - одна з найважливіших задач сучасної школи.

Математика починається зовсім не з лічби, що здається звичайним, а із загадки, проблеми. Щоб у молодшого школяра розвивалося творче мислення, необхідно, щоб він відчув подив і цікавість, повторив шлях людства в пізнанні, задовольнив з апетитом

потреби в записах обчислень. Тільки через подолання труднощів, рішення проблем, дитина може ввійти у світ творчості.

Творчість є предметом аналізу багатьох учених, що працювали над дослідженням природи цього феномену. У роботах В.К. Андрєєва, Л.С. Виготського, В.І. Давидова, В.О. Крутецького, В.А. Моляко, М.П. Морозова, Я.І. Пономарьова, В.В. Рибалки, С.Р. Рубінштейна, О.О. Савченко, С.В. Сисоєвої та інших висвітлюються психолого-педагогічні фактори, які сприяють формуванню творчих здібностей особистості, мотивів її творчої діяльності, розвитку творчої активності.

Технологія проблемного навчання теоретично обґрунтована такими видатними вченими, як В. Оконь, І.Я. Лернер, М.І. Махмутов, Т.В. Кудрявцев та ін. Дослідження вітчизняних і закордонних психологів і педагогів над поліпшенням основ цієї технології продовжують розвиватися. Велика увага приділяється виявленню механізмів творчої діяльності і природи творчого мислення, що і є одним із напрямків мого дослідження.

Математика, починаючи з першого класу, сприяє розвитку мислення і творчих можливостей учнів. Але чому ж саме на проблемне навчання покладено визначну роль у досягненні мети - розвитку творчого мислення учнів? Які існують можливості використання проблемних ситуацій на уроках математики? Ці питання і зумовили вибір мети нашого дослідження: розкрити проблему розвитку творчого мислення молодших школярів засобами використання проблемних ситуацій на уроках математики.

Чим вище рівень творчого розвитку учня, тим вище рівень його загального розумового розвитку, тим вище його працездатність. Саме тому одне із завдань, які ставить перед собою сучасний педагог - це розвиток творчих здібностей учня.

В початкових класах, діти стикаються з багаточисельними проблемними ситуаціями, які спонукають їх до математичного мислення. Вже простий розподіл зошитів, підручників може стати для учнів першого класу проблемою, якщо ми їх запитаємо, чи вистачить учбового приладдя для всього класу. Діти дивляться на відносно невелику пачку зошитів і, ймовірно, думають, що їх не вистачить, бо мають на увазі величину тих і інших елементів. Перевіркою правильності припущення дітей буде роздача зошитів. Вказана проблема є прикладом порівняння однієї множини з іншою і оцінки кількості елементів множин.

Творче мислення краще всього формувати в ході вирішення завдань нестандартного типу, коли учень натрапляє на посильні для нього проблеми і питання, формулює їх. Роль вчителя – поставити дитину, що вчиться в положення першовідкривача, дослідника. Успішність вирішення таких завдань залежить від рівня співпраці учня і вчителя, від опанування учнем системою розумових дій (порівняння, аналіз, синтез і т. д.). Спеціально підібрані завдання мають бути орієнтовані на:

- поступове ускладнення матеріалу;
- поетапне збільшення об'єму роботи;
- підвищення рівня самостійності учня;
- інтеграцію завдань і способів діяльності;
- залучення елементів теорії для вирішення завдань;
- навчання способам міркування з врахуванням принципу варіативності завдань;
- формування побіжності думок, оригінальність, гнучкість розуму, допитливість, уміння висувати і розробляти гіпотези [1, с. 46].

Проблемність при навчанні математиці виникає абсолютно природно, не вимагаючи жодних спеціальних вправ, штучно підібраних ситуацій. По суті, не лише кожна текстова задача, але і добра половина інших вправ, представлених в підручниках математики і дидактичних матеріалах, і є свого роду проблемами, над вирішенням яких учень повинен задуматися, якщо не перетворювати їх виконання на чисто тренувальну роботу, пов'язану з рішенням по готовому, даному вчителем зразку.

Введення математичних понять представляє також багато можливостей для організації проблемних ситуацій в класі. Наприклад, учень отримав завдання: “До 2 додай 5 і помнож на 3”. І інше: “До 2 додай 5, помножене на 3”. Можна скласти два вирази і обчислити таким чином:

$2+5\cdot3=21$

$2+5\cdot3=17$.

Такий запис викликає здивування у дітей. Після аналізу дій учні приходять до висновку, що два різні результати можуть бути правильними і залежать від того, в якій черговості виконувати додавання і множення. Виникає проблемне питання: як записати розв’язання, щоб отримати правильну відповідь? Питання спонукає дітей до пошуків, внаслідок чого вони приходять до поняття дужок. Після вписування дужок, розв’язання набирає вигляд:

$(2+5)\cdot3=21$

$2+5\cdot3=17$.

Інший приклад завдання пов'язаний з геометричним матеріалом. Вчитель пропонує увазі першокласників плакат, на якому змальовано декілька чотирикутників і п'ятикутників. Всі ці фігури на плакаті ніяк не згруповані, але чотирикутники забарвлені в червоний колір, а п'ятикутники – в зелений. Вчитель повідомляє, що всі червоні фігури можна назвати чотирикутниками, а зелені – п'ятикутниками. Після цього перед класом ставитися проблемне питання: “Як ви думаєте, чому червоні фігури можна назвати чотирикутниками, а зелені – п'ятикутниками?”. Для вирішення даної проблеми діти повинні провести ряд спостережень, співставлень, порівнянь.

Вони повинні порівнювати в думках терміни “чотирикутник” і “п’ятикутник”. Аналізуючи ці слова, вони можуть розчленувати їх,

виділивши в них знайомі їм слова, що є частинами нових термінів, – “чотири” і “кут”, “п'ять” і “кут”. Такий аналіз вже може направити їх думку в певному напрямі. Перевірити правильність отриманих припущень вони зможуть, звернувшись до уважного розгляду запропонованих вчителем фігур. Тут знову доведеться провести ряд спостережень, зіставлень, порівнянь, в результаті яких вони повинні переконатися, що дійсно всі червоні фігури містять по чотири кути, а зелені – по п'ять кутів. Помітивши цю особливість, зіставивши її з особливостями термінів – назв даних фігур, діти повинні прийти до висновку, який і буде відповіддю на поставлене проблемне питання.

Проблемна ситуація спеціально створюється вчителем шляхом застосування особливих методичних прийомів:

- учитель підводить школярів до протиріччя і пропонує їм самим знайти спосіб його розв'язку;
- зіштовхує протиріччя практичною діяльністю;
- викладає різні точки зору на одне питання;
- пропонує класові розглянути явище з різних позицій;
- спонукує тих, кого навчають, робити порівняння, узагальнення, висновки із ситуації, зіставляти факти;
- ставить конкретні питання (на узагальнення, обґрунтування, конкретизацію, логіку міркування;
- визначає проблемні теоретичні і практичні завдання;
- ставить проблемні задачі (з недостатніми або надлишковими вихідними даними; з невизначеністю в постановці питання; із суперечливими даними; зі свідомо допущеними помилками; з обмеженим часом вирішення; на подолання психічної інерції й іншим).

Проблемне навчання не може бути однаково ефективним у будь-яких умовах. Практика показує, що процес проблемного навчання породжує різні рівні як інтелектуальних утруднень учнів, так і їхній пізнавальній активності і самостійності при засвоєнні нових знань або застосуванні вже отриманих раніше знань у іншій ситуації.

Для реалізації проблемної технології навчання необхідно:

- добір самих актуальних, сутнісних задач;
- визначення особливостей проблемного навчання в різних видах навчальної роботи;
- побудова оптимальної системи проблемного навчання, створення навчальних і методичних посібників;
- особистісний підхід і майстерність учителя, здатних викликати активну пізнавальну діяльність дитини.

Головна цінність проблемного навчання в тому, що діти в черговий раз дістають можливість порівнювати, спостерігати, робити виводи; переконуються в тому, що не на кожне питання є готова відповідь, що відповідь може бути неоднозначною, що кожен з них має повне право шукати і знаходити свою відповідь, відстоювати свою думку. Зміни, що відбуваються в дітях, вказують на те, що учбові

проблеми створюють сприятливі умови для загального розвитку кожної дитини. Учні повинні під керівництвом вчителя спостерігати, порівнювати, описувати, обговорювати факти і явища, робити висновки.

Говорячи про проблему творчих здібностей, я згодна з думкою великого вченого Л.С. Виготського, який написав: «Творчество на деле существует не только там, где создают великие творческие произведения, но и везде там, где человек воображает, комбинирует, изменяет и создает что-либо новое, какой бы крупицей ни казалось это новое по сравнению с созданиями гениев» [2, с. 21].

Література

1. Брушлінський А.В. Психологія мислення і проблемне навчання. / А.В. Брушлінський. – М. : Знання, 1983. – 96 с. **2. Виготський Л.С.** Уява і творчість в дитячому віці: Психологічний нарис: Книга для вчителя. 3 видавництва. / Л.С. Виготський. – М. : Освіта, 1991. – 93 с.

Хотіна І. О. Використання проблемних ситуацій на уроках математики в процесі розвитку творчого мислення молодших школярів

В цій статті визначається і характеризується роль проблемного навчання на уроках математики в початковій школі як одного з чинників формування творчої особистості. Автор розкриває методичні прийоми, що допоможуть вчителеві створювати на уроках математики проблемні ситуації, спонукаючи учнів до творчого пошуку.

Ключові слова: творче мислення, математика, проблемне навчання.

Хотина И. А. Использование проблемных ситуаций на уроках математики в процессе развития творческого мышления младших школьников

В этой статье определяется и характеризуется роль проблемного обучения на уроках математики в начальной школе как одного из составляющих формирования творческой личности. Автор раскрывает методические приемы, которые помогут учителю создавать на уроках математики проблемные ситуации, которые ведут учеников к творческому поиску.

Ключевые слова: творческое мышление, математика, проблемное обучение.

Khotina I.A. The use of problem situations in mathematics lessons in the development of creative thinking of primary school children

In this article, is defined and characterized the role of problem-based learning in mathematics lessons in elementary school as one of the components forming a creative personality. The author reveals the

instructional techniques that help teachers create lessons on mathematics problem situations that lead students to creative solutions.

Key words: creative thinking, mathematics, problem-based instruction.

ВІДОМОСТІ ПРО АВТОРІВ

1. **Агаджанов А.Т.** – 4-ПІ. Науковий керівник – Жукова В.М., к.п.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

2. **Винник В.В.** – 4-ПІ. Науковий керівник – Онопченко С.В., к.п.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

3. **Габієв М.А.** – 4-Інф. Науковий керівник – Донченко В.Ю., асистент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

4. **Гашенко П.В.** – 4-ПІ. Науковий керівник – Томилин Д.Ф., асистент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

5. **Гладкий И.А.** – 4-ПІ. Науковий керівник – Томилин Д.Ф., асистент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

6. **Громова Я.И.** Науковий керівник – Тихонов Ю.Л., к.т.н., доцент кафедри інформаційних техннологій та систем, Онопченко С.В., к.п.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

7. **Дудник Е.А.** – 4-ПІ. Науковий керівник – Жукова В.М., к.п.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

8. **Душенко О.С.** – 4-Інф. Науковий керівник – Лущенко О.І., асистент кафедри теоретичної та прикладної інформатики ЛНУ імені Тараса Шевченка

9. **Ефремов Е.С.** – 2-ПН. Науковий керівник – Киричевский Р.В., к.т.н., доцент кафедри інженерно-педагогічних дисциплін ЛНУ імені Тараса Шевченка

10. **Зубков Е.В.** – 4-ПІ. Науковий керівник – Козуб Г.А., к.т.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

11. **Зубков Е.В.** – 4-ПІ. Науковий керівник – Скачко В.В., асистент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

12. **Іщенко Я.С.** – 4-Інф. Науковий керівник – Половінка Д.В., к.т.н., доцент кафедри теоретичної та прикладної інформатики ЛНУ імені Тараса Шевченка

13. **Калиниченко В.А.** – 2-Інф. Науковий керівник – Калініченко Ю.В., асистент кафедри теоретичної та прикладної інформатики ЛНУ імені Тараса Шевченка

14. **Калиниченко Є.А.** – 5-КСМ. Науковий керівник – Калініченко В.Ю., асистент кафедри теоретичної та прикладної інформатики ЛНУ імені Тараса Шевченка

15. **Ковалева В.С.** – 4-ПІ. Науковий керівник – Крамаренко Т.А., асистент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

16. **Кожемякіна Ю.Ю.** – магістрантка спеціальності Інформатика. Науковий керівник – Сквірський В.Д., к.т.н., доцент кафедри теоретичної та прикладної інформатики ЛНУ імені Тараса Шевченка

17. **Козуб В.Ю.** – 1-ПІ. Науковий керівник – Козуб Г.О., к.т.н., доцент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

18. **Коломийцева А.С.** – 2-ПІ. Науковий керівник – Могильный Г.А., к.т.н., доцент кафедри інформаційних технологій та систем, Тихонов Ю.Л., к.т.н., доцент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

19. **Корячкіна О.В.** – 4-ПСт. Науковий керівник – Іє О.М., к.ф.м.н., доцент кафедри математичного аналізу та алгебри ЛНУ імені Тараса Шевченка

20. **Кулешова В.А.** – 3-Інф. Науковий керівник – Тихонов Ю.Л., к.т.н., доцент кафедри інформаційних технологій та систем, Онопченко С.В., к.п.н., доцент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

21. **Лавренченко А.А.** – 4-ПІ. Науковий керівник – Онопченко С.В., к.п.н., доцент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

22. **Місюра С.В.** – ІПД. Науковий керівник – Сквірський В.Д., к.т.н., доцент кафедри теоретичної та прикладної інформатики ЛНУ імені Тараса Шевченка

23. **Могильная Ю.Г.** – 1-ПІ. Науковий керівник – Фоменко А.В., к.п.н., доцент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

24. **Мосійчук Ю.Б.** – 4-Інф. Науковий керівник – Лущенко О.І., асистент кафедри теоретичної та прикладної інформатики ЛНУ імені Тараса Шевченка

25. **Ореховський О.В. –** студент 2-го курсу економічного факультету СПбГУСЕ ТФ. Науковий керівник – Никифорова Н.Г., к.т.н., доцент кафедри сервіс та технології Тихвінського філіалу СПбГУСЕ

26. **Понамарев Н.В.** – 3-Інф. Науковий керівник – Тихонов Ю.Л., к.т.н., доцент кафедри інформаційних технологій та систем, Онопченко С.В., к.п.н., доцент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

27. **Рабой В.В.** – 5-Інф. Науковий керівник – Сквірський В.Д., к.т.н., доцент кафедри теоретичної та прикладної інформатики ЛНУ імені Тараса Шевченка

28. **Романченко Т.Ю., Черенкова Э.В.** – 3-Мат. Науковий керівник – Томилин Д.Ф., асистент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

29. **Сєдих В.П.** – студентка 3 курсу спеціальності Початкова освіта ВП Лисичанський педагогічний колледж ЛНУ імені Тараса Шевченка. Науковий керівник – Овчаренко О.П., викладач математики ВП Лисичанський педагогічний колледж ЛНУ імені Тараса Шевченка

30. **Сич Ю.В.** – 4-ПІ. Науковий керівник – Донченко В.Ю., асистент кафедри інформаційних технологій та систем ЛНУ імені Тараса Шевченка

31. **Сичова А.А.** – 4-ПІ. Науковий керівник – Фоменко А.В., к.п.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

32. **Титаренко А.С.** – 3-Інф. Науковий керівник – Тихонов Ю.Л., к.т.н., доцент кафедри інформаційних техннологій та систем, Онопченко С.В., к.п.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

33. **Трофимов Я.В. –** студент 2-го курсу економічного факультету СПбГУСЕ ТФ. Науковий керівник – Никифорова Н.Г., к.т.н., доцент кафедри сервіс та технології Тихвінського філіалу СПбГУСЕ

34. **Фоменко Е.А.** – 1-ПІ. Науковий керівник – Фоменко А.В., к.п.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

35. **Хотіна І.О.** – студентка 4 курсу спеціальності Початкова освіта ВП Лисичанський педагогічний колледж ЛНУ імені Тараса Шевченка. Науковий керівник – Довгопола О.В., викладач математики ВП Лисичанський педагогічний колледж ЛНУ імені Тараса Шевченка

36. **Черевань Е.Н.** – 4-ПІ. Науковий керівник – Донченко В.Ю., асистент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

37. **Черніков А.С.** – 4-Інф. Науковий керівник – Михайлова І.О., к.т.н., доцент кафедри теоретичної та прикладної інформатики ЛНУ імені Тараса Шевченка

38. **Шевченко О.В.** – 4-ДІД. Науковий керівник – Цодікова Н.О., асистент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

39. **Шопін П.Ю.** – 4-Мат Науковий керівник – Ревенко О.В., к.ф-м.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

40. **Юрченко М.А.** – 2-ПІ. Науковий керівник – Могильный Г.А., к.т.н., доцент кафедри інформаційних техннологій та систем, Тихонов Ю.Л., к.т.н., доцент кафедри інформаційних техннологій та систем ЛНУ імені Тараса Шевченка

Наукове видання

НАУКОВИЙ ПОШУК МОЛОДИХ ДОСЛІДНИКІВ (фізико-математичні та технічні науки)

Збірник наукових праць студентів

№ 8, 2014

Відповідальний за випуск:
ст. викл. О. В. Радчук

Здано до склад. 28.04.2014 р. Підп. до друку 28.05.2014 р.
Формат 60x84 1/8. Папір офсет. Гарнітура Times New Roman.
Друк ризографічний. Ум. друк. арк. 26,51. Наклад 100 прим. Зам. № 85.

Видавець і виготовлювач
Видавництво Державного закладу
«Луганський національний університет імені Тараса Шевченка»
вул. Оборонна, 2, м. Луганськ, 91011. Тел./факс: (0642) 58-03-20.
e-mail: alma-mater@list.ru
свідоцтво суб'єкта видавничої справи ДК №3459 від 09.04.2009 р.