

Методические указания для СРС к содержанию курсовой работы

1 Общие положения

Курсовая работа представляет собой учебно-практическое или научно-экспериментальное исследование, предназначенное для систематизации, углубления и закрепления знаний, полученных студентом в процессе изучения конкретной дисциплины в соответствии с учебным планом.

Структурными элементами курсовой работы являются:

- **титульный лист;**
- **задание на работу;**
- **аннотация;**
- **оглавление;**
- **введение;**
- обзор литературы и постановка задачи;
- **основной материал по специальной части;**
- **заключение;**
- **библиографический список;**
- приложения.

Обязательные структурные элементы выделены полужирным шрифтом. Шаблон титульного листа приведен в приложении А. Шаблон задания на работу приведен в приложении Б.

Аннотацию помещают в пояснительной записке после задания. Аннотация включает:

- характеристику основной темы;
- проблемы объекта (если есть);
- цели (и задачи) работы;
- результаты работы.

Оглавление состоит из перечня разделов, глав, подразделов работы и включает: введение, наименование всех разделов и подразделов, заключение, библиографический список и наименование приложений, для каждого из которых указываются номер страниц, с которых начинаются эти элементы курсовой работы. От конца текста до номером страницы дается отточие.

Во введении должна быть раскрыта актуальность темы курсовой работы, приведены цель и задачи работы, объект и предмет работы, а также показана практическая применимость полученных автором результатов.

Заключение должно содержать:

- краткие выводы по результатам выполнения курсовой работы и оценку полноты решений поставленных в работе задач и достижения цели работы;
- рекомендации по конкретному использованию результатов курсовой работы;
- оценку результативности или эффективности предлагаемого решения.

В основной части курсовой работы приводятся данные, отражающие сущность, методику и основные результаты выполненной работы.

Содержание основной части определяется задачами работы, приведенными во введении. Основная часть должна содержать:

- постановку задачи;
- описание программы;
- инструкцию по установке и требования к компьютеру;
- руководство пользователя

В приложения рекомендуется включать материалы, связанные с выполненной курсовой работой, которые по каким-либо причинам не были включены в основную часть.

2 Содержание основной части

2.1 Постановка задачи

Краткое описание задачи и предметной области (объекты и их поведение, взаимодействие).

Требования к программе (текст/графика, мышь/клавиатура).

2.2 Описание программы

Используемые библиотеки и программы других разработчиков (в т.ч. транслятор) *файлы .h - это не библиотеки!*

Разбиение на модули, для каждого модуля указывается:

Классы и их интерфейсы (назначение переменных, назначение и параметры методов на 1-2 строки).

Функции (назначение и параметры) и глобальные переменные.

Иерархия классов.

Пояснения по алгоритму и особенностям реализации, возможностям модификации и развития программы (т.е. какие возможности есть в программе (например, виртуальные методы, константы-параметры, файлы с настройкой), которые позволяют развить/изменить поведение, внешний вид, язык общения с пользователем, не изменяя основного алгоритма и основных частей программы)

Используемые внешние файлы и их форматы

файлы .h - это не внешние файлы!

Описываются файлы, необходимые для работы программы, в т.ч. файлы, создаваемые во время работы программы:

имя файла - содержание файла

Для нестандартных (придуманных программистом) файлов описывается их формат (количество байтов на поле, содержание поля), для баз данных - структура, для картинок - требуемые размеры и количество цветов.

2.3 Инструкция по установке и требования к компьютеру

Требования к компьютеру:

операционная система, доп. программы и драйверы,
необходимое место на жестком диске,
минимальные требования к быстродействию процессора,
ОЗУ, видеокарте,
доп. устройства: звуковая карта, мышь и т.д.

Как установить программу с дискеты на компьютер.

(в простейшем случае "скопировать следующие файлы
на диск: program.exe, image.bmp, ...")

Как настроить программу на текущую конфигурацию
компьютера (тип звуковой карты, видеорежим и т.п.)

Возможные сообщения об ошибках при отсутствии необходимых
компонент (например, мыши).

2.4 Руководство пользователя

Описывается работа пользователя от запуска программы до
выхода. (Например, правила игры, смысл появляющихся сообщений,
управляющие клавиши. Не должно быть: "интерфейс является интуитивно
понятным и не нуждается в описаниях" или "как работать с программой
написано в подсказке, появляющейся по клавише F1, поэтому здесь не
описывается".)

Для библиотек: описание использования классов и функций
библиотеки с небольшими примерами (пользователь - это программист).

3. Приложения

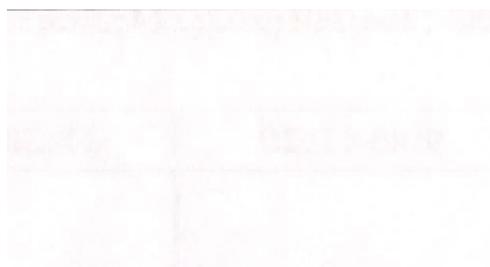
Приложения должны быть представлены для предварительной оценки и на
защиту в электронном виде.

Приложение 1. Исходные тексты программы в электронном виде

Приложение 2. Архив с программой или инсталляционная программа,
подготовленная для установки в соответствии с п.2.3.

4. Проведение защиты курсовой работы

Процедуры проведения и оценивания	Критерии оценивания
<p>Техническое задание выдается в первую неделю семестра. За две недели до окончания семестра студент демонстрирует и сдает преподавателю программный продукт. В процессе демонстрации программного продукта проверяется: соответствие программы техническому заданию; работоспособность в различных режимах. Преподаватель выставляет предварительную оценку и допускает студента к защите.</p> <p>В последнюю неделю семестра проводится защита КР.</p> <p>На защиту студент предоставляет:</p> <ol style="list-style-type: none"> 1. Развернутое техническое задание. 2. Программный продукт. 3. Пояснительную записку на 20-25 страниц в отпечатанном виде, содержащую описание разработки и соответствующие иллюстрации. 4. Программную документацию, указанную в разделе «Требования к программной документации» технического задания. <p>Защита курсовой работы выполняется в комиссии, состоящей не менее, чем из двух преподавателей.</p> <p>На защите студент коротко (3-5 мин.) докладывает об основных проектных решениях, принятых в процессе разработки, и отвечает на вопросы членов комиссии.</p>	<p>Оценка «Отлично» выставляется за курсовую работу, которая полностью соответствует техническому заданию, работоспособна во всех режимах, пояснительная записка имеет логичное, последовательное изложение материала с соответствующими выводами и обоснованными положениями. При защите студент показывает глубокое знание вопросов темы, свободно оперирует данными исследования, вносит обоснованные предложения, легко отвечает на поставленные вопросы.</p> <p>Оценка «Хорошо» выставляется за курсовую работу, которая полностью соответствует техническому заданию, работоспособна в подавляющем большинстве режимов, пояснительная записка имеет грамотно изложенную теоретическую главу, в ней представлены достаточно подробный анализ и критический разбор практической деятельности, последовательное изложение материала с соответствующими выводами, однако с не вполне обоснованными положениями. При ее защите студент показывает знание вопросов темы, оперирует данными исследования, вносит предложения по теме исследования, без особых затруднений отвечает на поставленные вопросы.</p> <p>Оценка «Удовлетворительно» выставляется за курсовую работу, которая не полностью соответствует техническому заданию, работоспособна только в части режимов, пояснительная записка имеет теоретическую главу, базируется на практическом материале, но имеет поверхностный анализ, в ней просматривается непоследовательность изложения материала, представлены необоснованные положения. При ее защите студент проявляет неуверенность, показывает слабое знание вопросов темы, не всегда дает исчерпывающие аргументированные ответы на заданные вопросы.</p> <p>Оценка «Неудовлетворительно» выставляется за курсовую работу, которая не соответствует техническому заданию, не работоспособна или работоспособна только в малой части режимов, пояснительная записка не имеет анализа, не отвечает требо-</p>



ваниям, изложенным в методических рекомендациях кафедры. В работе нет выводов либо они носят декларативный характер. При защите работы студент затрудняется отвечать на поставленные вопросы по ее теме, не знает теории вопроса, при ответе допускает существенные ошибки

5 Варианты для курсовых работ и рекомендации по проектированию

5.1 Генератор арифметических примеров

Программа для генерации примеров для контрольных работ по арифметике. На экране настройки необходимо ввести следующие опции:

Количество примеров: [1]

Количество действий: [5]

Диапазон исп. чисел: [-1000]..[1000] (в т.ч. промежуточные результаты!)

Допустимые арифметические действия

☒ 2+2

☒ 2-2

☒ 2*2

☒ 2:2

☒ -(-2) (смена знака)

Имя файла для вывода: [kontr1.txt]

Здесь ☒ - флажки (checkbox); [...] - поля ввода.

Сгенерированные примеры вместе с ответами вывести в указанный файл, разделяя пустой строкой в виде:

4 : (1 + 1) =

Ответ: 2

2 + 3 =

Ответ: 5

Перед и после знака арифметического действия выводить 1 пробел.

Генерацию можно производить следующим образом:

Задать случайным образом результат в ук. диапазоне.

1. Выбрать арифметическое действие
2. Выбрать число в ук. диапазоне (это 1 операнд)
3. Рассчитать 2 операнд. Если 2 операнд вне диапазона к шагу 1.

4. Выбрать N ($0 \leq \text{общЧислоДействий} - 1$) - число действий для разложения 1-го операнда, а число действий для разложения 2-го операнда $M = \text{общЧислоДействий} - N - 1$
5. Если $N > 0$ разложить 1 операнд с пом. рек. применения этого алгоритма.
6. Если $M > 0$ разложить 2 операнд с пом. рек. применения этого алгоритма.

Вывести пример

1. Узнать ширину примера
2. Создать строку необходимого размера
3. Вывести пример в строку
4. Вывести строку
5. Вывести ответ

Для представления примера использовать следующий класс:

```

конструктор(целое число)
конструктор(*левая часть, действие, *правая часть)
ширина (вычисляется рекурсивно)
действие
приоритет действия (для уменьшения количества выводимых скобок)
вывод в строку(char*)
    (размеры массива определяются шириной)
  
```

5.2 Текстовый редактор

Написать класс-редактор текста с возможностями NCEDIT (без использования меню, все управление с клавиатуры).

Возможности редактора: перемещение по тексту; режим вставки и замены; удаление и вставка символов и строк; выделение строк.

Оформить редактор в виде класса:

```

конструктор(x1,y1,x2,y2,цвет текста=0x7,цвет выделения=0x70)
выполнять редактирование()
разрешить/запретить изменения(bool) // режим просмотра/редактирования
количество строк()
номер текущей строки() // нумерация с 0
номер текущего символа в строке()
перейти на строку, символ(r, c=0)
считать строку(r=-1) r - номер строки, если -1, то текущую
заменить строку(char *, r=-1)
удалить строку(r=-1)
добавить строку(char *, r=-1)
добавить символы(char *) в текущее местоположение курсора
удалить символы(int=1) начиная с текущего
выделить строки(n1,n2) с n1-й по n2-ю,
    если оба аргумента = -1, убрать выделение
установить обработчик нераспознанных клавиш(void (*f)(int))
    например, функциональные клавиши, Ctrl-, Alt- и т.п.
  
```

для клавиш F5 и F6 выполнять копирование и перемещение блока текста

Для вывода на экран использовать функции из `conio.h`. Рисование рамок, работа с окнами в этот класс не должны входить (и в данной работе не нужны).

В качестве примера сделать программу-редактор для вызова из командной строки:

```
edit text.txt
```

Добавить подсказку по функциональным клавишам (не в классе) как в редакторе Far Manager: F1-помощь F2-сохранить F10-выход и т.п. Помощь загружать из файла, используя новый созданный объект-редактор в режиме просмотра.

5.3 Редактор для кодирования программы методом "сверху-вниз"

Написать редактор для кодирования программы методом "сверху-вниз". Этот метод заключается в следующем:

Пишется программа на языке сверхвысокого уровня (естественном языке). Затем каждый оператор такой программы постепенно расшифровывается, заменяется алгоритмом на более простом языке, пока программа не окажется записанной на каком-либо языке программирования.

Например, программу решения квадратного уравнения можно написать так:

```
* ОБЪЯВИТЬ ПЕРЕМЕННЫЕ
* ВВЕСТИ КОЭФФИЦИЕНТЫ УРАВНЕНИЯ
* ВЫЧИСЛИТЬ
* ВЫВЕСТИ КОРНИ УРАВНЕНИЯ
```

Затем провалиться в оператор 2 и ввести его определение:

```
ВВЕСТИ КОЭФФИЦИЕНТЫ УРАВНЕНИЯ
printf("Введите коэф. уравнения a, b и c:");
scanf(&a,&b,&c);
```

Также можно определить оператор 3:

```
ВЫЧИСЛИТЬ
* ВЫЧИСЛИТЬ d
  if(d>=0)
* ВЫЧИСЛИТЬ КОРНИ УРАВНЕНИЯ
  else
* КОРНЕЙ НЕТ
```

Таким образом, должна существовать возможность вставить как обычный оператор языка программирования, так и оператор, который в

дальнейшем будет расшифровываться (в который можно провалиться для расшифровки). Вложенность может быть достаточно большой.

Первым символом в строке с оператором суперязыка стоит признак расшифрован оператор или нет (* или !).

Рекомендуется следующий формат файла для хранения программы:

*РАСШИФРОВАННЫЙ ОПЕРАТОР

!НЕРАСШИФРОВАННЫЙ ОПЕРАТОР

[пробел]оператор на языке программирования

(конец расшифровки оператора)

Возможности редактора:

Перемещение по тексту (в т.ч. вглубь)

Изменение строки, удаление строки, вставка новой строки,

Размеры строки можно ограничить 75-80 символов.

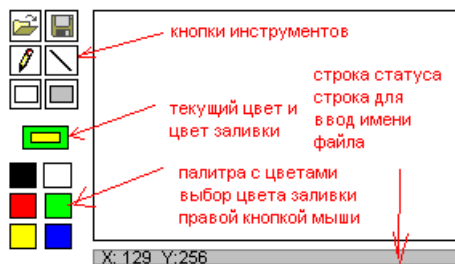
5.4 Растровый графический редактор

Разработать упрощенный графический редактор Paint со следующими возможностями:

- рисование точек (свободное рисование)
- рисование линий
- рисование прямоугольников (заполненных и нет)
- выбор цветов рисования и заполнения из 16 или более
- чтение и запись рисунка в стандартном (.BMP,.PCX,.JPG) или собственном формате

При движении мыши высвечивать координаты и размеры.

Примерный формат экрана:



Развитие темы: добавить копирование и поворот части рисунка, при создании задание размеров рисунка, скроллинг рисунка.

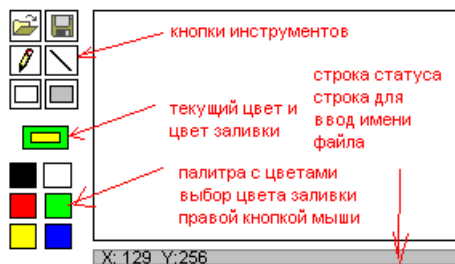
5.5 Векторный графический редактор

Разработать упрощенный графический редактор Corel Draw v0.0 со следующими возможностями:

- рисование линий
- рисование прямоугольников (заполненных и нет)
- выбор цветов рисования и заполнения из 16 или более
- чтение и запись рисунка в собственном формате
- удаление объектов

При движении мыши высвечивать координаты и размеры вставляемого объекта.

Примерный формат экрана:



Развитие темы: добавить выделение, перемещение и копирование объектов, изменение размеров и др. свойств.

5.6 Редактор электрических схем

Написать программу для редактирования электрических схем. Схема составляется из набора конфигурируемых элементов. Элементы можно поворачивать, соединять проводами.

В файле конфигурации содержится список имен файлов с элементами (от 8 до 20), что позволяет настраиваться на конкретную предметную область. Файл с элементом содержит следующую информацию:

LAMP.ELM:

LAMP.ICO // имя картинки с элементом для кнопки

21 21 // размеры элемента

C 10 10 10 0 15 // составляющие элемента -

окружность с центром (10,10) радиусом 10,
незаполненная, цвет белый

L 3 3 17 17 15 // линии крест-накрест, цвет белый

L 17 3 3 17 15

I 0 10 // точки подключения проводов

I 20 10

Допустимы след. команды для рисования элемента:

В - прямоугольник

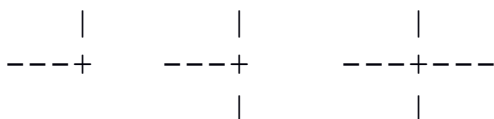
А - дуга

Редактор позволяет вставить в схему любой из указанных в схеме элементов,

соединить элементы проводами (соединение идет по прямой, при соединении достаточно попасть в небольшую окрестность точки подключения, к одной точке можно подключить только один провод), убирать (разрывать) провода, перемещать элементы (провода при этом двигаются в соответствии с точками подключения), удалять элементы (подключенные провода при этом убираются) сохранять схему в файл и восстанавливать из файла.

Для сгиба и соединения проводов в определенных координатах

рекомендуется задать два-три вспомогательных элемента:



Элементы схемы можно представить классом-прототипом:

- наименование файла с элементом
- размеры элемента
- количество точек подключения
- координаты i точки подключения
- нарисовать элемент в точке (x,y) и поворотом

и классом-конкретным элементом:

- указатель на класс-прототип
- координаты элемента
- угол поворота
- координаты i точки подключения с учетом размещения и поворота
- элемент* , подключенный к i точке
- подключить элемент* к i точке
- разорвать i -ю связь
- нарисовать

Библиотека для работы с матрицами

Библиотека должна включать класс Matrix

Из матрицы можно выделить прямоугольную часть - подматрицу `.sub(i,j,n,m)`, строку `.row(i)` и столбец `.col(i)`. С частью матрицы должны быть допустимы те же операции, что и с матрицей; модификация подматрицы изменяет соответствующие элементы в базовой матрице.

Основные операции `+`, `-`, `*`, `=`, `+=`, `-=`, `*=`, `==`, `!=`. В качестве второго операнда может выступать матрица или число.

Операции `/`, `/=` определить только для операнда-числа справа.

Дополнительные полезные функции: транспонирование, максимум, минимум,

поиск элемента, ввод-вывод, единичная и нулевая матрица размером n и др.

Некоторые функции и операции должны создавать новую матрицу.

Для доступа к элементам матрицы переопределить операцию `()` : `a(i,j)`.

Эта операция возвращает ссылку на элемент, т.е. может использоваться и в левой части оператора присваивания.

Для создания подматрицы (строки, столбца) рекомендуется определить приватный

конструктор, который не выделяет память, а получает указатель на матрицу и как

расположена подматрица в ней. Только обычный конструктор матрицы выполняет

реальное выделение памяти. В классе должен быть флажок, определяющий

выделена ли была память в конструкторе или используется память другого объекта.

Рекомендуется определить метод, который создает копию матрицы с выделением собственной памяти. Его можно использовать для перегрузки тех операций, в которых необходимо разорвать связь между подматрицей и матрицей, например:

```
Matrix operator+(const Matrix &a, const Matrix &b)
{ return a.copy()+=b;
}
```

Проверить работу программы на методе Гаусса:

```
cin>>n;
Matrix a(n,n+1);
cin>>a;
// Метод Гаусса решения системы линейных уравнений
for(i=0;i<a.rows();i++)
{ double u,v;
  Matrix c(a.sub(i,i,a.rows()-i,1));
  u=c.max(); // найти номер максимального
  v=c.min(); // элемента по модулю в столбце
  if(fabs(v)>fabs(u)) u=v;
  k=c.find(u)+i;
  if(k!=i) // поменять местами строки
  {
    Matrix t(1,a.cols());
    t=a.row(k);
    a.row(k)=a.row(i);
    a.row(i)=t;
  }
  a.row(i)/=a(i,i); // ведущий элемент привести к 1
  for(j=0;j<a.rows();j++) // вычесть i-строку из всех других
    if(j!=i) // строк
    {
      a.row(j)-=a.row(i)*a(j,i);
    }
}
// в последней колонке получаем решение уравнения
cout<<a.col(n);
```

5.8 Библиотека классов для работы с полиномами

Реализовать класс для работы с полиномами. В качестве элементов используются латинские буквы и числа. Пример полинома:

$$0.1*a*b^3+x*y^4$$

Реализовать операции +, -, *, = для полиномов-операндов
и операции +, -, *, /, где 1 или 2 операнд является числом double
(для деления только 2-ой операнд)

Методы: вычисление полинома при указанных значениях неизвестных,
подстановка вместо буквы другого полинома, дифференцирование по
ук.букве, ввод (>>), вывод (<<) в виде одной строки,
получить набор неизвестных.

конструктор(double=0)

конструктор(char *полином в виде строки)

5.9 Библиотека классов для работы с базами данных объектов (ООБД)

Определить классы для хранения объектов в файле.

Класс для хранимого объекта необходимо произвести от некоторого
базового абстрактного класса и переопределить виртуальные методы

int ReadData(FILE *f); // чтение полей из файла

int WriteData(FILE *f); // запись полей в файл

int Size(); // размер записи (все записи для простоты одинак. размера)

void Recalc(); // вычислить незаписываемые в файл поля

после чтения записи из файла

Можно от созданного класса произвести новый класс,
в методах нового класса в первую очередь нужно вызвать
методы базового класса, а затем произвести действия с
новыми полями этого класса

В базовом классе должны быть определены методы

конструктор(имяфайла)

int Open()

int Close()

long Id() // идентификатор записи (смещение в файле)

int Goto(long id) // перейти на запись с идентификатором id

int First() // перейти на первую запись в файле

int Next() // перейти на следующую запись в файле

int Prev() // перейти на предыдущую запись в файле

int Last() // перейти на последнюю запись в файле

int Post() // записать изменения

int Cancel() // отменить изменения, перечитать запись

int Insert() // добавить новую запись, войти в режим изменения

int Edit() // войти в режим изменений

int Delete() // удалить запись

int Eof() // файл пуст или обнаружен конец файла при выполнении Next()

int Bof() // файл пуст или обнаружено начало файла при выполнении Prev()

long Count() // количество записей

методы возвращают код ошибки или 0

Для доступа к полям в производном классе рекомендуется определить методы SetИмя(значение) и GetИмя()

5.10 Текстовая оконная библиотека

Написать библиотеку для оконного интерфейса в текстовом режиме. Библиотека должна работать в режиме 80x25 и 80x50. Должны определяться следующие классы:

- окно
 - координаты
 - размеры
 - видимость
 - цвет фона
 - текст заголовка
 - процедура, вызываемая для нераспознанных клавиш окна
 - верхнее окно // свойство класса
 - добавить(элемент) // метод
- метка (надпись)
 - координаты
 - видимость
 - цвет
 - текст метки
- кнопка
 - координаты
 - размеры
 - видимость
 - цвет кнопки
 - текст кнопки
 - цвет текста
 - процедура, вызываемая при нажатии кнопки
- ввод строки
 - координаты
 - ширина
 - видимость
 - цвет поля
 - цвет текста
 - текст

Рекомендуется использовать активные свойства, или определить пару методов setСвойство/getСвойство. Работа от клавиатуры. Для примера рассмотреть интерфейс Far Manager (Windows)/Midnight Commander (Linux).

5.11 Графическая оконная библиотека

Написать библиотеку для оконного интерфейса в графическом режиме. Библиотека должна работать для произвольного графического режима, обеспечиваемого выбранной графической библиотекой. Должны определяться следующие классы:

- окно

- координаты
- размеры
- видимость
- цвет фона
- текст заголовка
- процедура, вызываемая для нераспознанных клавиш окна
- верхнее окно // свойство класса
- добавить(элемент) // метод
- метка (надпись)
 - координаты
 - видимость
 - цвет
 - текст метки
- кнопка
 - координаты
 - размеры
 - видимость
 - цвет кнопки
 - текст кнопки
 - цвет текста
 - процедура, вызываемая при нажатии кнопки
- ввод строки
 - координаты
 - ширина
 - видимость
 - цвет поля
 - цвет текста
 - текст

Максимальные размеры окон ограничены размером экрана, поэтому запрещается использовать `getImage` для сохранения части экрана.

Рекомендуется использовать активные свойства (см. `Property.cpp`), или определить пару методов `setСвойство/getСвойство`. Работа от клавиатуры. Для примера рассмотреть интерфейс Windows.

5.12 Анимационная библиотека

Написать библиотеку для создания игр и различных мультипликаций. Объекты характеризуются свойствами:

- координаты на экране x, y и уровень z
($z=0$ уровень фона, объект с максимальным z находится над всеми другими объектами)
- видимость
- имя файла с картинкой
- размеры объекта (по умолчанию размеры картинки;
- игнорируемый (фоновый) цвет
(по умолчанию цвет пиксела в верхем левом углу)
- получить цвет пиксела (i, j) в тек. состоянии

свойства можно читать (getимя) и изменять (setимя), изменения на экран выводятся после команды "показать изменения"

К экрану можно применить следующие методы
показать изменения

Определить очередь событий:

```
typedef int Event;
class TQueue
{
public:
    TQueue(int maxsize=100);
    ~TQueue();
    void add(int dt, Event obj); // добавить событие в момент (сейчас+dt)
    Event get(int &dt);         // получить событие (dt- превышение интервала)
    bool isReady();             // подошло время обработать к-л события
    bool isEmpty();             // очередь пуста
    bool isFull();              // очередь полна
};
```

Для получения текущего времени в миллисекундах использовать
timeGetTime()

Работу библиотеки продемонстрировать на простой игре типа тира:
прицел, управляемый мышью;
выбегающие время от времени слева или справа враги;
несколько предметов, за которые враги могут прятаться.

Если щелкнул центром прицела на пиксел врага, отличный от
фонового, значит враг убит.

5.13 Библиотека классов для работы с целыми числами большой точности

Реализовать класс для работы с целыми десятичными числами с заданным
в аргументе конструктора числом цифр.

Реализовать операции +, -, *, +=, -=, *=, =, сравнения (6 шт.),
ввода-вывода (<<, >>) для чисел указанного класса,
а также операции +, -, *, /, +=, -=, *=, /=, =, сравнения (6 шт.),
где вторым операндом является число long (все) или double (кроме /).
double используется для записи чисел с большим числом нулей типа 2e100

Уэзерел Ч. Этюды для программистов

Кнут Д. Искусство программирования для ЭВМ Т.2

5.14 Программа для обучения основам программирования

С помощью этой программы учащиеся смогут познакомиться с
основными понятиями программирования:

Программа должна позволять составлять программу для некоторого

робота (манипулятора). Программа является _скелетом_ для написания различных роботов, т.е. обеспечивает интерфейс, загрузку заданий из файла, набор классов и взаимодействие между ними.

Робот действует на некотором поле с размерами 10 на 10, Задача состоит в приведении поля в некоторое конечное состояние после выполнения программы, написанной учащимся.

Робот определяется в некотором классе программы, что позволяет изменять внешний вид и некоторые особенности поведения робота (муравей, трактор, уборщик, газонокосильщик), не изменяя логику основной программы. Это же касается и объектов на поле.

Рекомендуются следующие варианты классов:

--Робот

количество команд

наименование i-ой команды

текущие координаты

установить координаты и поле для работы

выполнить i-ю команду, результат=1, если команда успешно выполнена, иначе 0

нарисовать(x,y)

текущее направление

сообщение о причине неудачи

--Задание

инициализировать(поле, робот)

иниц. перед выполнением программы(поле) // случайное размещение препятствий

задание выполнено(поле, робот)?

нарисовать конечное состояние поля(x,y) // для объяснения задания

текст задания

перейти на следующий уровень (если требуется

решить набор заданий)

--Поле

цвет клетки(i,j)

установить цвет клетки(i,j)

какой объект на клетке(i,j)

установить объект на клетку(i,j)

нарисовать(x,y)

--Объект

тип: препятствие, ловушка и т.д.

разновидность: стена, мина, яма и т.д.

наименование

нарисовать(x,y)

Программа составляется из команд робота

Программа должна составляться с помощью выбора из меню (списка вариантов). Можно вставлять и удалять операторы программы

Выполнение программы заканчивается, если робот не может выполнить команду и выводится сообщение

Варианты заданий/роботов (для каждого варианта составляется своя модификация программы)

- черепашка-художник (раскрасить пол в определенный узор)
- муравей-грузчик (сокобан: передвинуть коробки на место)
- урожай (скушать яблоки, появляющиеся случайно до написания программы)

Задания усложняются введением препятствий и/или ловушек.

5.15 Программа для размещения мебели "Home design"

Мебель, стены, двери и др. элементы представляются в виде набора прямоугольных блоков, которые можно перемещать на виде сверху. Можно указать высоту размещения над полом. Объекты не должны накладываться друг на друга. Объекты делятся на несколько типов:

1. стены, которым можно задать длину и высоту
2. двери, окна и др. объекты, которые обычно накладываются на стену
толщина определяет место, где не могут быть размещены другие объекты
3. мебель, столы, стулья, люстры, горшки с цветами

Основные возможности программы: вставка объекта, который нужно выбирать из нескольких списков (определяется файлами конфигурации), поворот объекта на угол кратный 90 градусов, передвижение объекта в координатах XY и изменение высоты Z, т.е. курсовая работа сводится к передвижению параллелепипедов у которых задано направление (угол поворота 0,90,180,270)

Управление можно сделать только клавиатурой:

просто стрелка - перемещение по 10 см

Shift-стрелка по 1 см

По умолчанию объект вставляется рядом с предыдущим размещенным объектом (сдвинутым на ширину предыдущего объекта и повернутым на тот же угол).

В файлах конфигурации может содержаться следующая информация:

GROUPS.CFG:

walls.cfg Стены, двери, окна

tables.cfg Столы

...

WALLS.CFG:

стена, обои в горошек: 1

дверь 90 см, белая, со стеклом: 2 90 200 90

^

^

наименование (до :) тип объекта

^
 ширина
 ^
 высота
 ^
 толщина

холодильник 2-х дверный, зеленый: 3 120 160 60

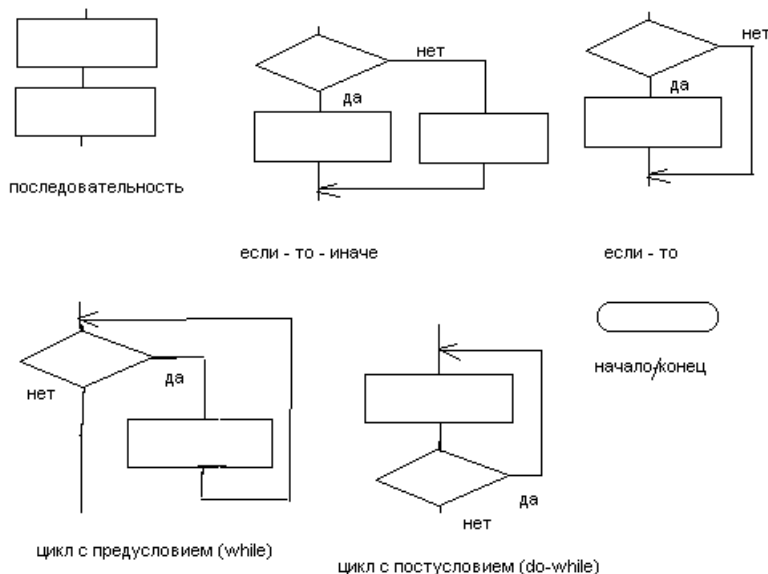
Можно всю конфигурацию хранить в одном файле.

Развитие темы:

Построение вида для любого угла и добавление перспективы.

5.16 Редактор блок-схем программ

В соответствии с принципами структурного программирования для написания программ достаточно 3 типов операторов: последовательность, условный оператор и оператор цикла



Некоторые блоки (ввод-вывод, завершение программы) имеют специальную рамку.

Написать редактор для блок-схем (структурного проектирования программы), который позволял бы редактировать блок-схемы.

Первоначально в схеме есть блоки "начало" и "конец".

Команды редактора: вставить блок после текущего/в левую/в правую ветвь, удалить текущий блок, изменить текст в блоке, сохранить и считать блок-схему в файле.

Для упрощения программы рекомендуется управление с помощью клавиатуры (движение стрелками по блокам: вниз/вверх - следующий/предыдущий блок, влево/вправо - левая/правая ветвь, если есть или следующий блок). Рекомендуется хранить указатель на текущий блок.

Для представления блок-схемы рекомендуется класс со свойствами тип блока

текст, выводимый в блоке
левая ветка // отсутствует у простых блоков
правая ветка // -//-
следующий блок
предыдущий блок
блок, в котором находится ветка с этим блоком
выч. высоту изображения блока с ветками
выч. ширину -//-
нарисовать блок и его ветки в (x,y)

Для упрощения рисования рекомендуется левую ветвь рисовать всегда вниз и принять стандартный размер блока.

5.17 Библиотека для моделирования систем массового обслуживания

Написать библиотеку классов для моделирования включающую следующие классы:

- Генератор объектов
 - тип генерируемого объекта
(объекты в системе представляются номером типа)
 - среднее время появления
 - тип случайного распределения - равномерное
 - включить/выключить
 - кол-во сгенерированных объектов
 - процедура, вызываемая при генерации объекта
- Очередь объектов к устройству обслуживания
 - добавить объект
 - взять объект
 - текущее число объектов в очереди
 - максимальное, минимальное, среднее
времени ожидания в очереди (расчет при моделировании)
 - максимальная длина очереди (расчет при моделировании)
- Устройство обслуживания
 - поставить объект на обработку
 - среднее время обработки
 - тип случайного распределения - равномерное
 - работает/простаивает
 - коэффициент загрузки (расчет при моделировании)
 - кол-во обработанных объектов
 - процедура, вызываемая при завершении обработки объекта
- Система в целом
 - количество объектов в системе
 - добавить объект в систему
 - удалить объект из системы
 - интервал моделирования
 - текущее время
 - процедура, вызываемая при завершении интервала моделирования

-- Очередь событий в системе

Написать модель следующей системы:

На карьере работает 2 экскаватора и М машин. В начальный момент машины поровну делятся между очередями экскаватора. Экскаватор наполняет кузов машины за 6-10 минут (распределение времени равномерное). Машина отвозит песок потребителю за 45 минут (распределение нормальное со стандартным отклонением 15 минут или 30-60 минут).

После возвращения становится под загрузку простаивающего экскаватора или в самую маленькую очередь. Промоделировать систему в течении 8 часов при различных M и выбрать M минимизирующее простой экскаваторов и машин.

Для моделирования поездки машины к потребителю можно использовать генератор, который необходимо включать, а после генерации объекта выключать. А можно сделать М устройств обслуживания . Развитие темы:

Добавить новые типы случайного распределения.

О генерации случайных чисел, отличного от равномерного распределения см.
Кнут Д. Искусство программирования для ЭВМ Т.2

5.18 Графический редактор для детей

Написать редактор, позволяющий составлять рисунок из набора готовых картинок. Для простоты общение с помощью надписей должно быть минимизировано.

Редактор должен позволять:

- выбирать группу картинок;
- выбирать картинку из группы;
- вставлять выбранную картинку в рисунок при этом видно как она будет расположена на рисунке; (при вставке фоновый цвет картинки игнорируется)
- сохранять и загружать рисунок в стандартном формате .BMP, предпочтительно при общении с пользователем вместо имени использовать маленькое изображение рисунка, а имя генерировать автоматически.

Примерный формат экрана:

набор кнопок	+--+	рисунок
с группами	+--+	
картинок	+--+	
и действиями	+--+	
	+--+	

Редактор должен конфигурироваться, в каком-то файле можно указать имена файлов с картинками, разбиение на группы и имена файлов для кнопок с группами.

5.19 Генератор тестов

Написать набор классов для генерации тестовых данных.

Структура представляется в виде совокупности объектов различного класса.

Д.б. классы для выбора, циклов и других элементов.

У всех классов д.б. метод generate, выводящий в файл результат генерации.

Класс последовательность Seq(), в которую можно добавлять (.add())

следующие элементы:

целое число, ссылку на переменную или строку,

указанное значение выводится при генерации

объекты классов

Any(строка)

при генерации выводится случайный символ из строки

Rep(строка)

при генерации выводится случайная перестановка символов строки

Format(число или переменная, строка)

при генерации выводится значение по указанному формату

Rep(целое число или переменная, элемент)

действие повторяется заданное число раз

While(переменная, элемент)

действие повторяется пока значение переменной не нулевое

другие последовательность или

Select().add(элемент [,целое число=1]).add(...

группа выбора, элемент из группы выбирается с вероятностью

прямо пропорциональной числу

Assign(переменная, выражение)

при генерации переменной присваивается выражение

выражение строится из объектов

Random(целое число или переменная, целое число или переменная)

случайное значение в указанном диапазоне

целых чисел и переменных,

соединяемых знаками операций +, -, *, /, %

```
Var N;
```

```
Seq s1;
```

```
s1.add(Assign(N,Random(1,100))).add(N).add("\n");
```

```
// присвоить N случайное число от 1 до 100, вывести N, вывести "\n"
```

```
s1.add(Rep(N,Seq().add(Rep(10,Any("ABC")))).add("\n"));
```

```
// выполнить N раз вывод строки из 10 случайных символов A,B или C
```

```
s1.generate();
```

```
// сгенерировать тест, результат выводится в cout
```

Пример сгенерированного теста:

3

АВВАССAAAB

АСВВВАСААС

ВВАССССAAA

5.20 Обработка XML

Написать класс для работы с XML-документами.

Каждый XML-документ содержит 1 или более элементов в форме:

<ИМЯ_ЭЛЕМЕНТА АТРИБУТЫ>

СОДЕРЖАНИЕ

</ИМЯ_ЭЛЕМЕНТА>

или

<ИМЯ_ЭЛЕМЕНТА АТРИБУТЫ />

АТРИБУТЫ разделяются символами пробела, табуляции, перехода на новую строку

и выглядят так

ИМЯ_АТРИБУТА="ЗНАЧЕНИЕ" -- ЗНАЧЕНИЕ не содержит символов "

и <

или

ИМЯ_АТРИБУТА='ЗНАЧЕНИЕ' -- ЗНАЧЕНИЕ не содержит символа ' и

<

ИМЯ_ЭЛЕМЕНТА или ИМЯ_АТРИБУТА начинается с латинской буквы или ':' или '_'

затем следует 0 или более букв, цифр, символов '-', '!', '_', '!'.

СОДЕРЖАНИЕ это произвольный текст, не содержащий символа < вперемешку 0 или более элементов.

Текст вне элементов объединяется

Для некоторых символов в тексте и значениях атрибутов используются специальные обозначения:

& &

< <

" "

> >

&#число; символ с указанным кодом, например, ' это '

Основные методы класса XML:

сохранить_в_файл(char *file_name)

int количество_элементов_содержания()

XML *элемент_содержания(i)

char *имя_элемента()

```
char *получить_текст()  
установить_текст(char *)
```

```
int количество_атрибутов()  
char *имя_атрибута(i)  
char *значение_атрибута(i)
```

```
XML *найти_элемент_содержания(char *name, char *attr, char *value)  
    // по имени и значению одного из атрибутов  
char *найти_значение_атрибута(char *name)  
int номер_атрибута(char *name)
```

```
добавить_атрибут(char *attr, char *value)  
    // или заменить, если существует  
удалить_атрибут(i)  
XML *добавить_элемент_содержания(char *name)  
удалить_элемент_содержания(i)
```

ПРИЛОЖЕНИЕ А

Министерство образования и науки РФ
ФГАОУ ВО ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИУ)
Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования

«Растровый графический редактор»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ
по дисциплине «Объектно-ориентированное программирование»
ЮУрГУ–220501.2008.313.ПЗ КР

Руководитель,

_____ *Оленчикова Т.Ю.*

«___» _____ 2017г.

Автор работы:

Студентка группы: ЕТ – 201

_____ *Иванова И.И.*

«___» _____ 2017г.

Работа защищена с оценкой

«___» _____ 2017 г.

Челябинск – 2017

Министерство образования и науки РФ
 ФГАОУ ВО ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИУ)
 Институт естественных и точных наук

Факультет математики, механики и компьютерных технологий
 Кафедра прикладной математики и программирования
 Направление 01.03.02 «Прикладная математика и информатика»

УТВЕРЖДАЮ

Заведующий кафедрой ПРИМА

_____ А.А. Замышляева
 _____ 2017 г.

ЗАДАНИЕ

на курсовую работу студента

_____ Иванова Ивана Ивановича

Группа ЕТ-201

1. Дисциплина Объектно-ориентированное программирование
 2. Тема работы Библиотека для работы с матрицами

3. Срок сдачи студентом законченной работы 25 мая 2017 г.

4. Перечень вопросов, подлежащих разработке

- 1) объектно-ориентированный анализ предметной области;
- 2) разработка интерфейса классов на языке C++;
- 3) реализация методов классов на языке C++;
- 4) тестирование программы (библиотеки);
- 5) оформление отчета по курсовой работе.

5. Календарный план

Наименование разделов курсовой работы	Срок выполнения разделов работы	Отметка о выполнении руководителя
Введение. Постановка задачи.	01.03.2017-10.03.2017	
Описание программы: разбиение на модули; интерфейсы классов; иерархия классов; разработка алгоритмов.	10.03.2017-01.05.2017	
Инструкция по установке. Разработка тестов для проверки программы.	01.05.2017-10.05.2017	
Руководство пользователя.	10.05.2017-20.05.2017	
Заключение.	20.05.2017-25.05.2017	

Руководитель работы _____ / _____

Студент _____ / _____
 (подпись)

3 ПРИМЕР ОФОРМЛЕНИЯ АННОТАЦИИ НА КУРСОВУЮ РАБОТУ

АННОТАЦИЯ

Иванов Ю.П. Учет основных средств. – Челябинск: ЮУрГУ, ЕТ-314, 2017. – 48 с., 3 ил., 5 табл., библиографический список – 24 наим., 8 прил.

Объектом исследования является порядок бухгалтерского учета основных средств в коммерческой организации.

Цель работы – освоить методику формирования в бухгалтерском учете достоверной информации о наличии и движении объектов основных средств.

В работе рассмотрен учет объектов основных средств в соответствии с действующими на дату написания курсовой работы законодательно-нормативными документами. Приведены правила документального оформления операций по движению объектов основных средств. Рассмотрены порядок синтетического и аналитического учета операций по движению объектов основных средств, особенности формирования информации об основных средствах в учетных регистрах и бухгалтерской отчетности. Выявлены особенности формирования учетной политики применительно к учету объектов основных средств. Раскрыт порядок проведения, оформления и отражения в учете результатов инвентаризации основных средств

На примере условного предприятия показана методика учета основных средств в коммерческой организации.

ПРИМЕР ОФОРМЛЕНИЯ ВВЕДЕНИЯ РАБОТЫ

ВВЕДЕНИЕ

Актуальность темы. Объектно-ориентированный подход является наиболее прогрессивной технологией разработки программных систем, позволяет разрабатывать более сложные системы.

Цель работы – Разработать растровый графический редактор Paint

Задачи работы:

- изучить приемы объектно-ориентированного анализа;
- научиться разрабатывать программы в объектно-ориентированном стиле;
- овладеть технологиями объектно-ориентированного анализа и проектирования;
- изучить концепции объектно-ориентированного программирования; изучить особенности объектной модели языка программирования C++
- научиться самостоятельно и творчески использовать знания и полученные практические навыки;
- овладеть навыками самостоятельного получения новых знаний по теории и практике объектного подхода в программировании.

Объект работы – программа для рисования

Предмет работы – применение объектно-ориентированного подхода для разработки программы.

Результаты работы можно использовать в процессе последующего обучения в соответствии с учебным планом подготовки бакалавров по направлению «Прикладная математика и информатика»

ПРИМЕР ОФОРМЛЕНИЯ ЗАКЛЮЧЕНИЯ РАБОТЫ

ЗАКЛЮЧЕНИЕ

В результате проведенной работы определены этапы создания СМК в организации применительно к высшему учебному заведению. Изучен опыт разработки СМК в вузе и установлены особенности создания СМК в учебном заведении в отличие от разработки СМК на предприятии. Выявлены проблемы создания СМК вуза.

Определены порядок и содержание этапов работ по созданию СМК вуза. Описан и визуализирован процесс «Разработка СМК вуза» при помощи IDEF-моделирования. Определены показатели и количественные критерии эффективности и результативности процесса «Разработка СМК вуза». Таким образом, цель работы достигнута, задачи – решены.

Результаты работы рекомендуется использовать при разработке СМК факультетов ЮУрГУ. Выполненная работа имеет практическую ценность и рекомендована для апробации на МТ-факультете.

ПРИМЕРЫ ОФОРМЛЕНИЯ ОСНОВНОЙ ЧАСТИ

1 Постановка задачи

Необходимо разработать растровый графический редактор Paint со следующими возможностями:

- рисование точек (свободное рисование);
- рисование линий;
- рисование прямоугольников (заполненных и нет);
- выбор цветов рисования и заполнения из 16 или более;
- чтение и запись рисунка в стандартном (.BMP, .PCX, .JPG) или собственном формате.

При движении мыши высвечивать координаты.

Анализ предметной области выявляет следующие объекты:

- кнопки управления, имеющие квадратную форму и отличающиеся изображением или цветом;
- объект, отображающий текущие цвета рисования;
- объект, отображающий текущие координаты мыши и обеспечивающий ввод имени файла при загрузке и сохранении;
- объект — поле для рисования, реакция которого на движение мыши зависит от текущего выбранного цвета и инструмента.

Кнопки управления по поведению можно разделить на 3 группы, различающиеся по поведению:

- 16 кнопок выбора цвета, изменяющие состояние объекта, отображающего текущие цвета рисования
- 4 кнопки выбора инструмента;
- 2 кнопки загрузки и сохранения рисунка.

Требования к программе:

- Графический интерфейс;
- Работа с мышью

2. Описание программы

2.1 Для разработки программы были использованы:

- компилятор MinGW GNU C/C++ 5.1
- библиотека winbgim

2.2 Программа состоит из 4 модулей:

1 Модуль buttons (интерфейсная часть в файле buttons.h, реализация в файле buttons.cpp) содержит следующие классы:

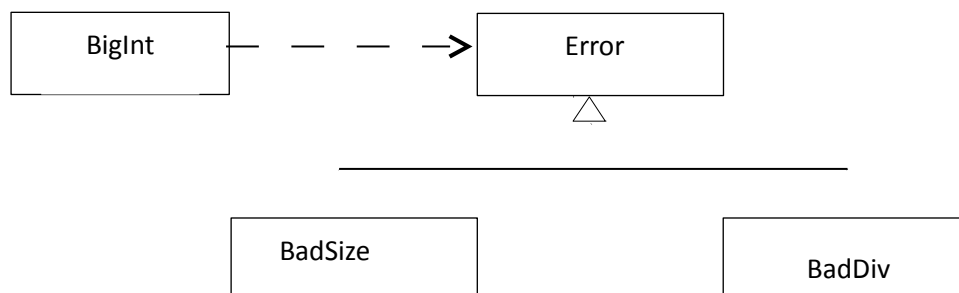
```
class Base_Object //базовый класс для всех объектов-инструментов программы
{
protected:
```

```

    int x, y; //координаты базовой точки(верхняя левая точка объекта)
    int w, h; //размеры объекта
public:
    virtual void draw() = 0; //рисует объект
    virtual void Left_Button() = 0; //обработка нажатия левой кнопки мыши на объект
    virtual void Right_Button() = 0; //обработка нажатия правой кнопки мыши на объект
    Base_Object(int x, int y, int w, int h):x(x), y(y), w(w), h(h) {} // конструктор,
    передаются координаты левого верхнего угла (x,y) и размеры (w,h)
    virtual ~Base_Object() {} // деструктор
    bool Check_Place(int x, int y); //попадают ли координаты (x,y) на объект?
};
class Colour_Palette:public Base_Object //палитра цветов для выбора при рисовании
{
protected:
    int colour; //номер цвета из списка цветов
public:
    Colour_Palette(int x, int y, int w, int h, int colour):
        Base_Object(x, y, w, h), colour(colour) {} // конструктор, передаются
координаты левого верхнего угла (x,y), размеры (w,h) и цвет кнопки colour
    void draw(); //прорисовка кнопки
    void Left_Button(); //выбор текущего цвета
    void Right_Button(); //выбор цвета заливки
    void set_colour(int c) {colour = c;} //устанавливаем цвет c для кнопки
};
...

```

2.3 Иерархия классов



2.4 Пояснения по реализации

Класс хранит целое число большой точности в виде массива, где i -ый элемент соответствует коэффициенту числа при $BASE^i$. В данной работе $BASE=10000$, в случае, если на `int` отводится 2 байта и $BASE=1000000000$, если `int` занимает 4 байта. Это позволяет повысить эффективность работы программы. Каждый элемент массива, представляет собой четырёхзначное число или восьмизначное число. При этом элементы хранятся в обратном порядке, что позволяет придать реализации алгоритмов естественный вид.

Например, число $20! = 243,2902,0081,7664,0000$ представляется по основанию 10000 как

$20! = 0 + 7664 \cdot BASE + 81 \cdot BASE^2 + 2902 \cdot BASE^3 + 243 \cdot BASE^4$,
соответствующий массив коэффициентов:

Номер элемента массива	0	1	2	3	4
Значение	0	7664	81	2902	243

...

2.5 Используемые внешние файлы и их форматы

Программа использует файлы в формате BMP:

xload.bmp – картинка для рисования кнопки загрузки из файла

xsave.bmp – картинка для рисования кнопки сохранения в файл

pen.bmp – картинка для рисования кнопки свободного рисования

line.bmp – картинка для рисования кнопки линий

rect.bmp – картинка для рисования кнопки незакрашенного прямоугольника

fillrect.bmp – картинка для рисования кнопки закрашенного прямоугольника

Все картинки имеют размер 40x40 и 24-битный цвет и должны находиться в подпапке buttons текущей папки.

Также программа может в процессе работы сохранять, а потом загружать файлы в BMP формате, размером 640x480, цветовое разрешение которых зависит от текущих настроек Windows.

3. Инструкция по установке и требования к программе

Требования к компьютеру:

Процессор: Intel Pentium 4 1ГГц или выше

Память: 512+ МБ

Видеокарта: разрешение 1024x768 или выше, 24-битный цвет

Свободное место на диске: 15 Мб

Операционная система: Windows XP и выше

Дополнительное устройство: мышь

Для установки скопировать следующие файлы на диск:

- paint.exe;
- папку buttons со всеми файлами.

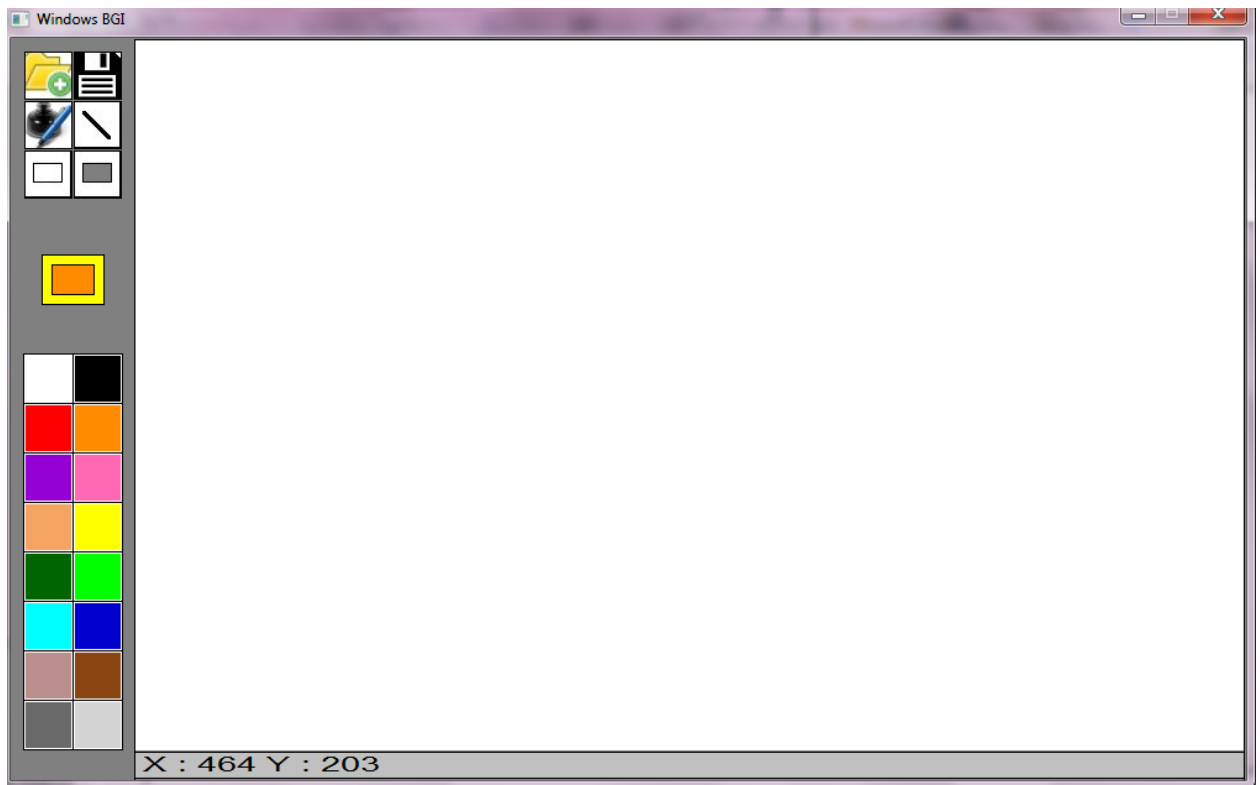
Вариант для библиотек:

Для сборки библиотеки необходим компилятор, поддерживающий C++11.

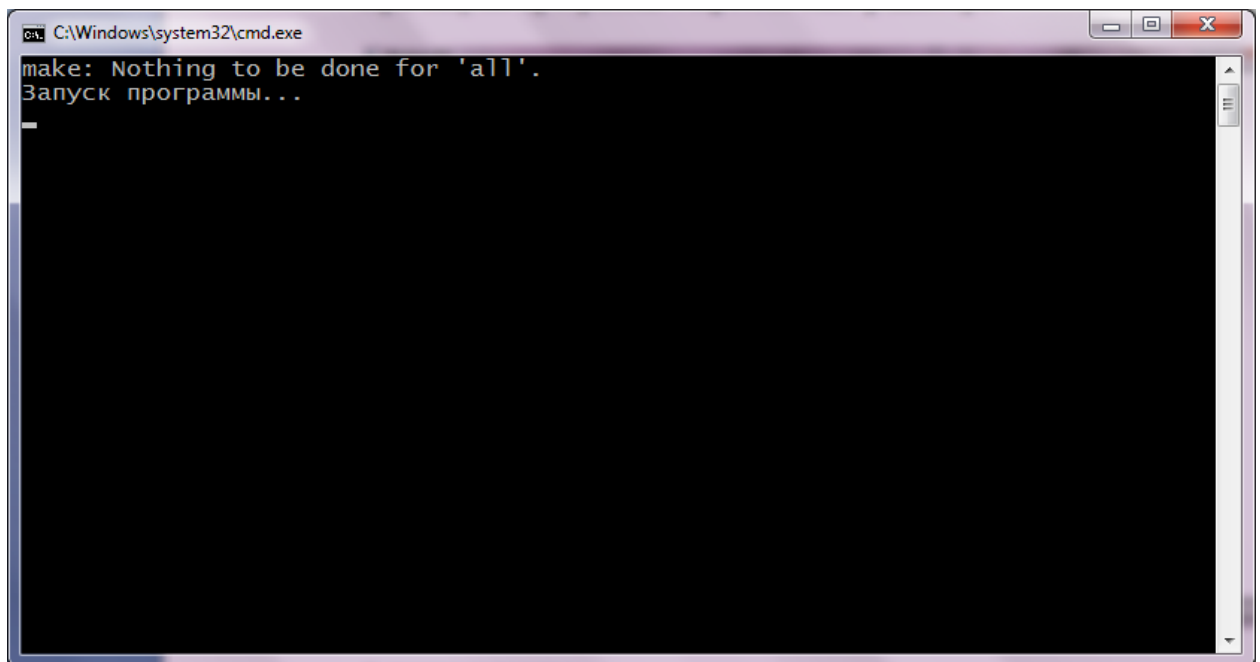
Установка и настройки не требуются. Для работы необходимо добавить файлы BigInt.h, BigInt.cpp в проект, использующий длинные целые числа.

4. Руководство пользователя

При запуске программы откроется окно редактора:



А также консоль:



Белое поле в центре – поле для рисования. При перемещении мыши по полю в левом нижнем углу в строке состояния высвечиваются текущие координаты курсора в системе координат: ось Ох направлена из верхнего левого угла белого поля вправо, ось Оу вниз. При покидании белого поля координаты не отображаются.

Слева пользователь видит палитру из 16 цветов. Нажатием левой кнопки мыши выбирается цвет линий, правой – цвет заливки. Эти цвета отображаются в прямоугольнике над палитрой.

Рассмотрим по порядку кнопки инструментов слева сверху слева направо сверху вниз:

...

Вариант для библиотек:

Конструктор выделяет память под количество цифр, заданных в аргументе конструктора

Пример:

```
BigInt A(10);
```

```
BigInt B(10);
```

Выделяется память под 10 цифр для чисел A и B.

Деструктор освобождает используемые объектом ресурсы. Вызывается компилятором автоматически.

Оператор присваивания, позволяет одному объекту BigInt присвоить значения другого объекта.

Пример:

```
A=B;
```

Если текущее количество цифр в объекте B превышает количество цифр, заданных аргументе конструктора объекта A, то возникает исключительная ситуация BadSize.

Конструктор преобразователь long в BigInt

Пример:

```
A=BigInt(345L);
```

Преобразует число 345 типа long к числу типа BigInt

Конструктор преобразователь double в BigInt

Пример:

```
A=BigInt(2e100);
```

Преобразует число 2e100 типа double к числу типа BigInt.

Сложение с присваиванием.

Пример:

```
A+=B;
```

Число A складывается с числом B, полученное значение присваивается объекту A. Если текущее количество цифр в результате превышает количество цифр, заданных аргументе конструктора объекта A, то возникает исключительная ситуация BadSize.

...