Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Иркутский национальный исследовательский технический

университет»

Институт *Информационных технологий и анализа данных*

**О Т Ч Ё Т**

о прохождении *учебной практики*

(вид практики: учебная/производственная)

в  *ИРНИТУ, Институт информационных технологий и анализа данных*

(наименование профильной организации)

6.001.00.00

Обучающегося

*Арбаковой Анастасии АСУб-20-2*

(ФИО, группа, подпись)

Руководитель практики от института

*Бахвалов С.В., доцент*

(ФИО, должность, подпись)

Зам. директора института

*Китаева О.И., зам. директора института,* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО, должность, подпись)

Оценка по практике

*Бахвалов С.В.*

(ФИО, подпись, дата)

Содержание отчета на \_\_\_\_\_\_ стр.

Приложение к отчету на \_\_\_\_\_\_ стр.

Иркутск 2021

**Индивидуальное задание на прохождение**

*учебной* **практики**

(вид (тип) практики)

для *Арбаковой Анастасии Вячеславовны*

(ФИО обучающегося полностью)

обучающегося *1*курса группы АСУб-20-2

по направлению подготовки *09.03.01 Информатика и вычислительная техника*

специализация/ профиль/программа *Автоматизированные системы обработки информации и управления*

Место прохождения практики: *ИРНИТУ, Институт информационных технологий и анализа данных*

Сроки прохождения практики с «*14*» *июня* *2021г.* по «*26*» *июня* *2021г.*

Цели и задачи прохождения практики: *Работа с графикой в С++ Bulder/ Embarcadero RAD Studio.*

Содержание практики, вопросы подлежащие изучению: *научиться работать с графикой; научиться строить графики простейших функций с возможностью масштабирования (расчета размера одного деления); научиться выводить графики функций с использованием стандартной компоненты Chart; научиться строить графики функций, заданных в параметрическом виде; научиться воспроизводить эффект анимации.*

Планируемые результаты практики: разработка приложений с использованием *С++ Bulder/ Embarcadero RAD Studio.*

Руководитель практики от кафедры

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / *Бахвалов С.В.*/

(подпись) (ФИО)

Зам. директора института

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / *Китаева О.И.* /

(подпись) (ФИО)

С настоящим индивидуальным заданием и с программой практики ознакомлены, задание принято к исполнению

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_«*14*» *июня* *2021г.*

(подпись обучающегося)

**Задание 1**

**Работа с командной строкой**

Вариант 1

**Цель:**

Получить практические навыки по созданию консольных

приложений.

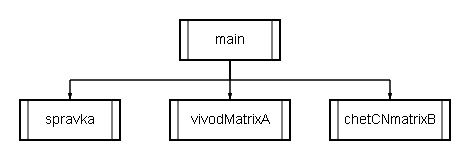
**Задачи:**

Освоение на практике приемов управления консольным

приложением из командной строки.

**Задание:**

Сформировать из квадратной матрицы А порядка 2\*n матрицу В, помещая нечетные элементы исходной матрицы А в нечетные столбцы, четные элементы – в четные. Предварительно учесть, что количество четных и нечетных элементов матрицы А должно быть одинаково.

**Иерархия модулей **

**Таблица спецификаций модулей**

|  |  |  |
| --- | --- | --- |
| **№** | **Имя** | **Параметры** |
| **1** | main | int argc – количество параметров, char\* argv – массив параметров, argv[1]=n – Задает размерность матрицы  argv[2] – Выбор метода заполнение элементов матрицы r – Автоматическое заполнение, m – Ввод элементов матрицы через консоль |
| **2** | spravka | - |
| **3** | vivodMatrixA | int \*A – матрица A, int n – количество элементов в матрице |
| **4** | chetCNmatrixB | int \*A – матрица A, int n – количество элементов в матрице |

**Таблица спецификаций модуля main**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя** | **Назначение** | **Тип** |
| **1** | n | размерность матриц | int |
| **2** | i | счетчик итерации в цикле for | int |
| **3** | j | счетчик итерации в цикле for | int |
| **4** | A | матрица A | int |

**Таблица спецификаций модуля vivodMatrixA**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя** | **Назначение** | **Тип** |
| **1** | i | счетчик итерации в цикле for | int |
| **2** | j | счетчик итерации в цикле for | int |
| **3** | n | размерность матрицы | int |
| **4** | A | матрица A | int |

**Таблица спецификаций модуля chetCNmatrixB**

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя** | **Назначение** | **Тип** |
| **1** | i | счетчик итерации в цикле for | int |
| **2** | j | счетчик итерации в цикле for | int |
| **3** | n | размерность матрицы | int |
| **4** | A | матрица A | int |
| **5** | B | матрица B | int |
| **6** | chet | счетчик четных элементов | int |
| **7** | nechet | счетчик нечетных элементов | int |
| **8** | ch | массив четных элементов | int |
| **9** | nech | массив нечетных элементов | int |
| **10** | q | счетчик итерации в цикле for | int |
| **11** | p | счетчик итерации в цикле for | int |

**Алгоритмизация функции main**

1. Подключение русского языка в консоли на ввод и вывод.
2. Перевод аргумента argv[1] в n - размерность матрицы из строки в число
3. Инициализация матрицы A.
4. Если argv[1] равен символу h, то выводится справка.
5. Если аргументов argc равно трем и argv[1] - число из символов от 0 до 9.
   1. Если argv[1] равен символу m, то вводится матрица A вручную.
      1. Вызов функции vivodMatrixA.
      2. Вызов функции chetCNmatrixB.
   2. Иначе если argv[1] равен символу r, то матрица A заполняется рандомными числами.
      1. Вызов функции vivodMatrixA.
      2. Вызов функции chetCNmatrixB.
   3. Иначе выводится справка.
6. Иначе выводится справка.

**Алгоритмизация функции spravka**

1. Вывод справки.

**Алгоритмизация функции vivodMatrixA**

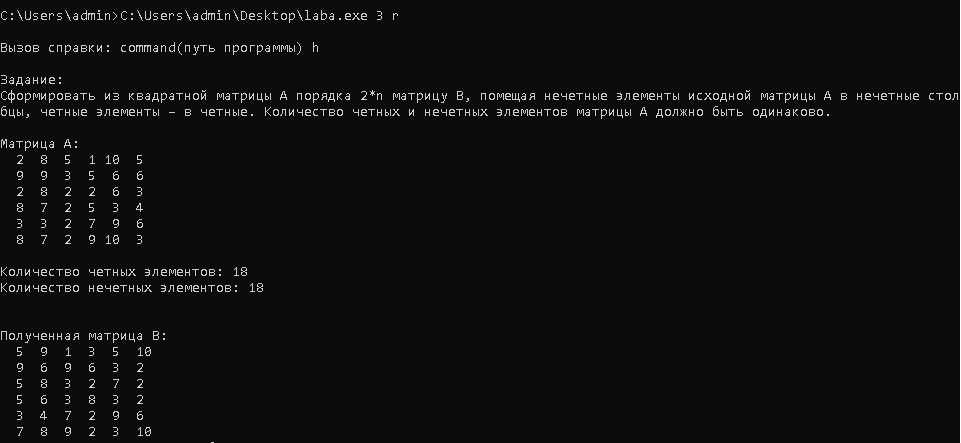
1. Вывод заполненной матрицы A на экран.

**Алгоритмизация функции chetCNmatrixB**

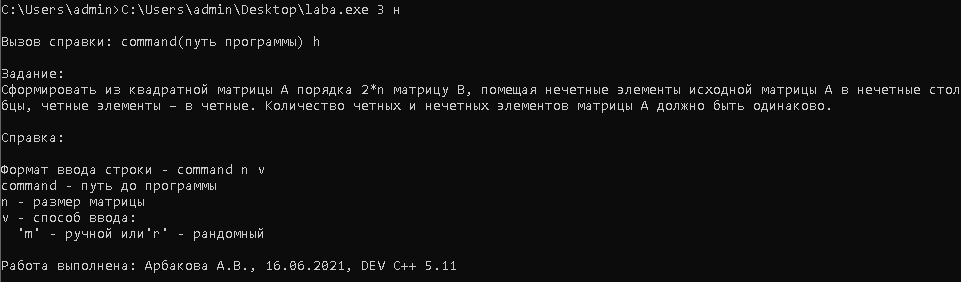
1. Инициализация переменных chet и nechet, массивов ch и nech, матрицы B.
2. Определение четности элемента в циклах и запись его в соответствующий массив четности.
3. Инициализация переменных q и p.
4. Проверка на равность количества четных и нечетных элементов по условию задачи.
   1. Поочередная запись нечетных и четных элементов в матрицу B.
5. Вывод матрицы B.

**Тесты:**

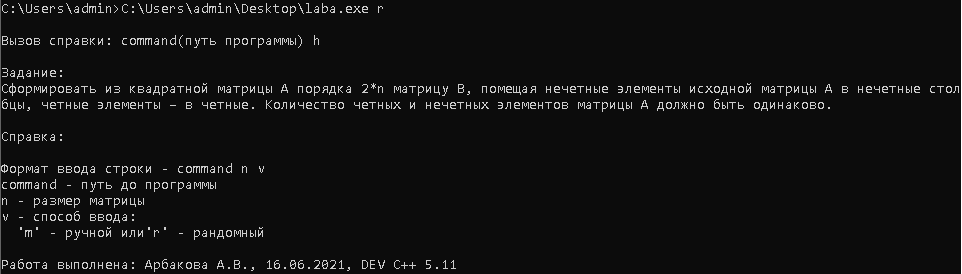
**1 тест:** Проверка работы программы при корректном вводе данных

****

**2 тест:** Проверка работы программы при некорректном вводе параметров

****

**3 тест:** Проверка работы программы при вводе не всех параметров

****

**4 тест:** Проверка работы программы при вводе только полного имени файла

****

**Листинг**

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

#include <Windows.h>

using namespace std;

*//C:\Users\admin\Desktop\laba.exe путь программы*

void spravka() *//справка*

{

cout<<"\nСправка:";

cout<<"\n\nФормат ввода строки - command n v";

cout<<"\ncommand - путь до программы";

cout<<"\nn - размер матрицы";

cout<<"\nv - способ ввода:";

cout<<"\n 'm' - ручной или'r' - рандомный";

cout<<"\n\nРабота выполнена: Арбакова А.В., 16.06.2021, DEV C++ 5.11\n"<<endl;

}

void vivodMatrixA(int \*A, int n) *//вывод матрицы А*

{

cout<<endl<<"Матрица А:"<<endl;

for(int i=0; i<2\*n; ++i)

{

for (int j=0; j<2\*n; ++j)

printf("%3d", A[i\*2\*n+j]);

cout<<endl;

}

}

void chetCNmatrixB(int \*A, int n) *//решение задачи, включающее подсчет четных и нечетных элементов*

{ *//и запись матрицы В по условию задания*

int chet=0, nechet=0;

int ch[4\*n], nech[4\*n];

int B[2\*n][2\*n];

for(int i=0; i<2\*n; ++i)

for (int j=0; j<2\*n; ++j)

if (A[i\*2\*n+j] % 2 == 0) *//определение четности элемента и запись его в соотвественный массив четности*

{

ch[chet]=A[i\*2\*n+j]; *//массив с четными элементами*

chet++; *//количество четных элементов*

}

else

{

nech[nechet]=A[i\*2\*n+j]; *//массив с нечетными элементами*

nechet++; *//количество нечетных элементов*

}

cout<<endl<<"Количество четных элементов: "<<chet<<endl;

cout<<"Количество нечетных элементов: "<<nechet<<endl<<endl;

int q=0, p=0;

if (chet==nechet) *//условие задания - равность количества четных и нечетных элементов*

{

for(int i=0; i<2\*n; i++)

for (int j=0; j<2\*n; j++)

{

B[i][j]=nech[q]; *//поочередная запись нечетных и четных элементов в матрицу В*

B[i][j+1]=ch[p];

q++; p++; j++;

}

cout<<endl<<"Полученная матрица B:"<<endl;

for(int i=0; i<2\*n; i++) *//вывод матрицы В*

{

cout<<" ";

for (int j=0; j<2\*n; j++)

cout<<B[i][j]<<" ";

cout<<endl;

}

}

else

cout<<"Количество четных и нечетных элементов матрицы должны быть равны. Попробуйте еще раз.";

}

int main(int argc, char\* argv[])

{

SetConsoleCP(1251); *//русский язык в консоли на ввод и вывод*

SetConsoleOutputCP(1251);

int n = atoi(argv[1]); *//перевод строк. в числ. для значения размера матрицы - n*

int A[2\*n][2\*n];

cout<<endl<<"Вызов справки: command(путь программы) h"<<endl;

if (\* argv[1] == 'h') //h - help - справка

{

spravka();

return 0;

} *//условие задачи*

cout<<"\nЗадание:\nСформировать из квадратной матрицы А порядка 2\*n матрицу В, помещая нечетные элементы исходной матрицы А в нечетные столбцы, четные элементы – в четные. Количество четных и нечетных элементов матрицы А должно быть одинаково."<<endl;

if ((argc==3) && ((\* argv[1]>='0') && (\* argv[1]<='9'))) *//условие на ввод всех трех аргументов (путь - размер - способ)*

{

if (\* argv[2]=='m') *//m - manual - ручной ввод*

{

cout<<"Введите матрицу А:"<<endl; *//ввод матрицы A - ручной*

for(int i=0; i<2\*n; i++)

for (int j=0; j<2\*n; j++)

{

cout<<"A["<<i+1<<"]["<<j+1<<"] = ";

cin>>A[i][j];

}

vivodMatrixA(&A[0][0], n); *//вызов функции для вывода матрицы А*

chetCNmatrixB(&A[0][0], n); *//вызов функции для решения задания*

}

else if (\* argv[2]=='r') *//r - random - рандомный(случайный)*

{

for(int i=0; i<2\*n; i++) *//ввод матрицы A - рандомный*

for (int j=0; j<2\*n; j++)

A[i][j] = rand() % 10 + 1;

vivodMatrixA(&A[0][0], n); *//вызов функции для вывода матрицы А*

chetCNmatrixB(&A[0][0], n); *//вызов функции для решения задания*

}

else

spravka(); *//вызов справки при неправильном вводе способа*

}

else

spravka(); *//вызов справки при неправильном вводе размера матрицы и количества аргументов*

system("pause");

}

**Задание 2**

**Работа с графикой в С++ Bulder/ Embarcadero RAD Studio**

* 1. Построить графики четырех функций (табл. 1) на PaintBox или Image. Все

графики построить различными цветами (точки/линии графика выводятся в

цикле). Предусмотреть масштабирование графика. Интервал изменения

аргумента, цвет графика должны задаваться пользователем.

* 1. Построить графики тех же функций с использованием компонент Chart с

несколькими сериями Series и ColorBox для выбора цвета линий).

**Вариант 2:**

1. sin(x), 2sin(x), sin(2x), sin(x/2)

**2)** Построить график функции, заданной в параметрическом виде. Цвет,

значение параметра (-ов), диапазон, и шаг изменения угла задаются

пользователем.

**Вариант 1:**

Улитка Паскаля: , , , , . Рассмотреть случаи, когда , , .

**3)** Анимация графических объектов. На канве главной формы обеспечить

движение объекта.

**Вариант 1:**

Изобразить на экране точку, движущуюся по окружности с постоянной угловой

скоростью.

**4)** Используя компоненты Image и Timer (событие OnTimer, возникающее через

заданный интервал Interval компоненты Timer) организовать смену картинок,

загружаемых из созданных графических файлов, сформировать анимационный

ролик (например, летящую птицу, движущиеся облака, летающую тарелку,

растущий цветок и т.п.).

**Математическая модель**

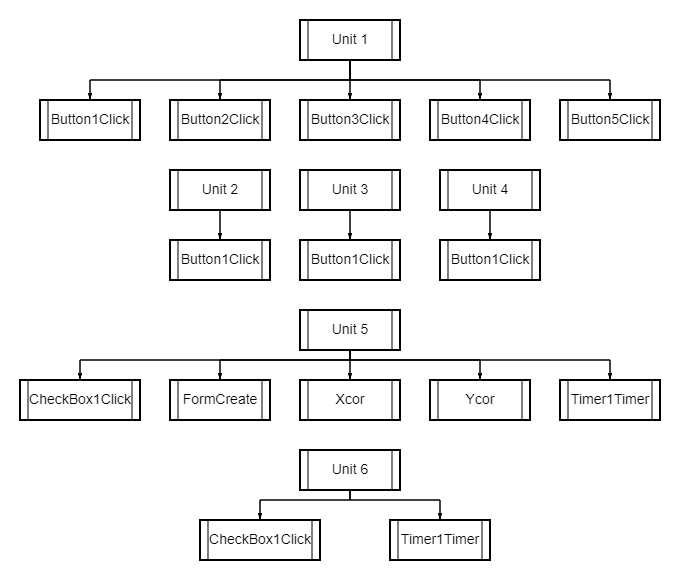
Графики в заданиях 1.1 и 1.2 вычисляются по формулам:

sin(x), 2sin(x), sin(2x), sin(x/2)

График функции в параметрическом виде задается по формуле:

, , , , . Рассмотреть случаи, когда , , .

**Иерархия модулей**

****

**Таблица спецификаций модулей**

**Unit 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя модуля** | **Назначение** | **Тип** | **Параметры** |
| 1 | Button1Click | Кнопка для открытия Form2 (Задание 1.1). | void | \*Sender – указатель на объект |
| 2 | Button2Click | Кнопка для открытия Form3 (Задание 1.2). | void | \*Sender – указатель на объект |
| 3 | Button3Click | Кнопка для открытия Form4 (Задание 2). | void | \*Sender – указатель на объект |
| 4 | Button4Click | Кнопка для открытия Form5 (Задание 3). | void | \*Sender – указатель на объект |
| 5 | Button5Click | Кнопка для открытия Form6 (Задание 4). | void | \*Sender – указатель на объект |

**Unit 2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Button1Click | Кнопка для запуска построения графика. | void | \*Sender – указатель на объект |

**Unit 3:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Button1Click | Кнопка для запуска построения графика. | void | \*Sender – указатель на объект |

**Unit 4:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Button1Click | Кнопка для запуска построения графика. | void | \*Sender – указатель на объект |

**Unit 5:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | CheckBox1Click | Флажок для включения таймера. | void | \*Sender – указатель на объект |
| 2 | FormCreate | Создание формы. | void | \*Sender – указатель на объект |
| 3 | Xcor | Функция для расчета координаты Х для движущейся точки. | float | float xc, float r, int t |
| 4 | Ycor | Функция для расчета координаты Y для движущейся точки | float | float xc, float r, int t |
| 5 | Timer1Timer | Функция таймера. | void | \*Sender – указатель на объект |

**Unit 6:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | CheckBox1Click | Флажок для включения таймера. | void | \*Sender – указатель на объект |
| 2 | Timer1Timer | Функция таймера. | void | \*Sender – указатель на объект |

**Словесное описание алгоритмов**

**Unit 1:**

1. Button1Click

При нажатии кнопки вызывается Form2.

1. Button2Click

При нажатии кнопки вызывается Form3.

1. Button3Click

При нажатии кнопки вызывается Form4.

1. Button4Click

При нажатии кнопки вызывается Form5.

1. Button5Click

При нажатии кнопки вызывается Form6.

**Unit 2:**

1. Button1Click

Запускается процесс построение графика через Image: Обновляется Image. Строится координатная сетка с масштабом, введенным ранее через ScrollBar. Вычисление и дальнейшее изображение координат функций sin(x), sin(2x), 2sin(x) и sin(x/2), согласно выбранного диапазона, цвета и масштаба.

**Unit 3:**

1. Button1Click

Запускается процесс построение графика через TChart: Очищение Series. Строится изображение функций sin(x), sin(2x), 2sin(x) и sin(x/2), согласно выбранного диапазона и цвета, в компоненте TChart.

**Unit 4:**

1. Button1Click

Запускается процесс построение Улитки Паскаля через Image: Обновляется Image. Строится координатная сетка с масштабом, введенным ранее через ScrollBar. Вычисление и дальнейшее изображение координат функции, согласно выбранного диапазона, цвета и масштаба.

**Unit 5:**

1. CheckBox1Click

Включение таймера при активном флажке.

1. FormCreate

Отключенный таймер, при открытии формы.

1. Xcor

Вычисление X координаты для движущейся точки.

1. Ycor

Вычисление Y координаты для движущейся точки.

1. Timer1Timer

Очищение формы и последующее создание круга и движущейся точки с вычисленными координатами.

**Unit 6:**

1. CheckBox1Click

Включение таймера при активном флажке.

1. Timer1Timer

Посменная загрузка картинок, образующие гиф-изображение.

### **Таблица тестов**

|  |  |  |
| --- | --- | --- |
| **№** | **Назначение** | **Результат** |
| 1 | Проверка работы программы в задании 1.1 |  |
| 2 | Проверка работы программы в задании 1.2 |  |
| 3 | Проверка работы программы в задании 2 |  |
| 4 | Проверка работы программы в задании 3 |  |
| 5 | Проверка работы программы в задании 4 |  |

**Листинг программы:**

**Unit 1:**

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit2.h"

#include "Unit3.h"

#include "Unit4.h"

#include "Unit5.h"

#include "Unit6.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

Form5->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

Form2->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

Form3->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

Form4->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button5Click(TObject \*Sender)

{

Form6->Show();

}

//---------------------------------------------------------------------------

**Unit 2:**

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit2.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm2 \*Form2;

//int s1, s2;

//---------------------------------------------------------------------------

\_\_fastcall TForm2::TForm2(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::Button1Click(TObject \*Sender)

{

Image1->Picture->Assign(0);

double x=0;

double z;

double y=0;

float n=0;

int m=0;

//zoom

if (ScrollBar1->Position==1) {

z = 10;

}

else if (ScrollBar1->Position==2) {

z = 20;

}

else if (ScrollBar1->Position==3) {

z = 30;

}

else if (ScrollBar1->Position==4) {

z = 40;

}

else if (ScrollBar1->Position==5) {

z = 80;

}

Image1->Canvas->Pen->Color=clBlack;

n=round(Image1->Width/z)+1;

m=round(Image1->Height/z)+1;

//толщина ручки

Image1->Canvas->Pen->Width=2;

//у ордината

Image1->Canvas->MoveTo(Image1->Width / 2, 0);

Image1->Canvas->LineTo(Image1->Width / 2,Image1->Height);

Image1->Canvas->MoveTo(Image1->Width / 2,0);

Image1->Canvas->LineTo(Image1->Width / 2-5,15);

Image1->Canvas->MoveTo(Image1->Width / 2,0);

Image1->Canvas->LineTo(Image1->Width / 2+5,15);

//х ордината

Image1->Canvas->MoveTo(0,Image1->Height / 2);

Image1->Canvas->LineTo(Image1->Width,Image1->Height/2);

Image1->Canvas->MoveTo(Image1->Width,Image1->Height / 2);

Image1->Canvas->LineTo(Image1->Width-15,Image1->Height / 2-5);

Image1->Canvas->MoveTo(Image1->Width,Image1->Height / 2);

Image1->Canvas->LineTo(Image1->Width-15,Image1->Height / 2+5);

//подпись осей

Image1->Canvas->Pen->Width=1;

Image1->Canvas->Font->Color = clGreen;

Image1->Canvas->TextOut(Image1->Width - 15,Image1->Height/2+5,'x');

Image1->Canvas->TextOut(Image1->Width/2-15,0,'y');

Image1->Canvas->TextOut(Image1->Width/2-10,Image1->Height/2+2,'0');

//разлиновывание вдоль оси у

x=round(Image1->Width/2);

y=0;

for (int i = 0; i < n; i++)

{

Image1->Canvas->MoveTo(x,y);

Image1->Canvas->LineTo(x,Image1->Height);

x=(x+z);

}

x=Image1->Width/2;

for (int i = 0; i < n; i++)

{

Image1->Canvas->MoveTo(x,y);

Image1->Canvas->LineTo(x,Image1->Height);

x=(x-z);

}

//разлиновывание вдоль оси х

x=0;

y=Image1->Height/2;

for (int i = 0; i < n; i++)

{

Image1->Canvas->MoveTo(x,y);

Image1->Canvas->LineTo(Image1->Width ,y);

y=(y+z);

}

y=Image1->Height/2;

for (int i = 0; i < n; i++)

{

Image1->Canvas->MoveTo(x,y);

Image1->Canvas->LineTo(Image1->Width ,y);

y=(y-z);

}

Image1->Canvas->MoveTo(Image1->Width / 2,Image1->Height / 2);

int shirina, visota, s2, v2;

double sbx, sbx1;

shirina=Image1->Width;

visota=Image1->Height;

s2=Image1->Width / 2;

v2=Image1->Height / 2;

//SIN(X)

sbx=SpinEdit1->Value;

sbx1=SpinEdit5->Value;

Image1->Canvas->Pen->Color=ColorBox1->Selected;

Image1->Canvas->Pen->Width=3;

Image1->Canvas->MoveTo(shirina/2+sbx\*z,v2-z\*sin((sbx)));

for (x = 0+sbx\*z; x < sbx1\*z; x++) {

y=(z\*sin(x/z));

Image1->Canvas->LineTo(x+shirina/2,v2-y);

Image1->Canvas->MoveTo(x+shirina/2,v2-y);

}

//SIN(2X)

sbx=SpinEdit2->Value;

sbx1=SpinEdit6->Value;

Image1->Canvas->Pen->Color=ColorBox2->Selected;

Image1->Canvas->Pen->Width=3;

Image1->Canvas->MoveTo(shirina/2+sbx\*z,v2-z\*sin((2\*sbx)));

for (x = 0+sbx\*z; x < sbx1\*z; x++) {

y=(z\*sin(2\*x/z));

Image1->Canvas->LineTo(x+shirina/2,v2-y);

Image1->Canvas->MoveTo(x+shirina/2,v2-y);

}

//2SIN(X)

sbx=SpinEdit3->Value;

sbx1=SpinEdit7->Value;

Image1->Canvas->Pen->Color=ColorBox3->Selected;

Image1->Canvas->Pen->Width=3;

Image1->Canvas->MoveTo(shirina/2+sbx\*z,v2-2\*z\*sin((sbx)));

for (x = 0+sbx\*z; x < sbx1\*z; x++) {

y=(2\*z\*sin(x/z));

Image1->Canvas->LineTo(x+shirina/2,v2-y);

Image1->Canvas->MoveTo(x+shirina/2,v2-y);

}

//SIN(X/2)

sbx=SpinEdit4->Value;

sbx1=SpinEdit8->Value;

Image1->Canvas->Pen->Color=ColorBox4->Selected;

Image1->Canvas->Pen->Width=3;

Image1->Canvas->MoveTo(shirina/2+sbx\*z,v2-z\*sin((sbx/2)));

for (x = 0+sbx\*z; x < sbx1\*z; x++) {

y=(z\*sin((x/2)/z));

Image1->Canvas->LineTo(x+shirina/2,v2-y);

Image1->Canvas->MoveTo(x+shirina/2,v2-y);

}

}

//---------------------------------------------------------------------------

**Unit 3:**

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit3.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm3 \*Form3;

//---------------------------------------------------------------------------

\_\_fastcall TForm3::TForm3(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::Button1Click(TObject \*Sender)

{

Series1->Clear();

Series2->Clear();

Series3->Clear();

Series4->Clear();

double x;

int sbx, sbx1,sbx2, sbx3, sbx4, sbx5, sbx6, sbx7;

sbx=SpinEdit1->Value;

sbx1=SpinEdit5->Value;

sbx2=SpinEdit2->Value;

sbx3=SpinEdit6->Value;

sbx4=SpinEdit3->Value;

sbx5=SpinEdit7->Value;

sbx6=SpinEdit4->Value;

sbx7=SpinEdit8->Value;

for( x=0+sbx; x<=sbx1; x=x+0.1){

Series1->AddXY(x, sin(x), "", ColorBox1->Selected);

}

for( x=0+sbx2; x<=sbx3; x=x+0.1){

Series2->AddXY(x, sin(2\*x), "", ColorBox2->Selected);

}

for( x=0+sbx4; x<=sbx5; x=x+0.1){

Series3->AddXY(x, 2\*sin(x), "", ColorBox3->Selected);

}

for( x=0+sbx6; x<=sbx7; x=x+0.1){

Series4->AddXY(x, sin(x/2), "", ColorBox4->Selected);

}

//---------------------------------------------------------------------------

}

**Unit 4:**

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit4.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm4 \*Form4;

//---------------------------------------------------------------------------

\_\_fastcall TForm4::TForm4(TComponent\* Owner)

: TForm(Owner)

{ //x=a\*cos^2t + b\*cost y=a\*cost\*sint + b\*sint

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm4::Button1Click(TObject \*Sender)

{

Image1->Picture->Assign(0);

double x=0;

double y=0;

int n=0;

int m=0;

Image1->Canvas->Pen->Color=clBlack;

Image1->Canvas->Pen->Width=2;

n=round(Image1->Width)+1;

m=round(Image1->Height)+1;

//у ордината

Image1->Canvas->MoveTo(Image1->Width / 2, 0);

Image1->Canvas->LineTo(Image1->Width / 2,Image1->Height);

Image1->Canvas->MoveTo(Image1->Width / 2,0);

Image1->Canvas->LineTo(Image1->Width / 2-5,15);

Image1->Canvas->MoveTo(Image1->Width / 2,0);

Image1->Canvas->LineTo(Image1->Width / 2+5,15);

//х ордината

Image1->Canvas->MoveTo(0,Image1->Height / 2);

Image1->Canvas->LineTo(Image1->Width,Image1->Height/2);

Image1->Canvas->MoveTo(Image1->Width,Image1->Height / 2);

Image1->Canvas->LineTo(Image1->Width-15,Image1->Height / 2-5);

Image1->Canvas->MoveTo(Image1->Width,Image1->Height / 2);

Image1->Canvas->LineTo(Image1->Width-15,Image1->Height / 2+5);

//подпись осей

Image1->Canvas->Pen->Width=1;

Image1->Canvas->Font->Color = clGreen;

Image1->Canvas->TextOut(Image1->Width - 15,Image1->Height/2+5,'x');

Image1->Canvas->TextOut(Image1->Width/2-15,0,'y');

Image1->Canvas->TextOut(Image1->Width/2-10,Image1->Height/2+2,'0');

//разлиновывание вдоль оси у

x=round(Image1->Width/2);

y=0;

for (int i = 0; i < n; i++)

{

Image1->Canvas->MoveTo(x,y);

Image1->Canvas->LineTo(x,Image1->Height);

x=(x+20);

}

x=Image1->Width/2;

for (int i = 0; i < n; i++)

{

Image1->Canvas->MoveTo(x,y);

Image1->Canvas->LineTo(x,Image1->Height);

x=(x-20);

}

//разлиновывание вдоль оси х

x=0;

y=Image1->Height/2;

for (int i = 0; i < n; i++)

{

Image1->Canvas->MoveTo(x,y);

Image1->Canvas->LineTo(Image1->Width ,y);

y=(y+20);

}

y=Image1->Height/2;

for (int i = 0; i < n; i++)

{

Image1->Canvas->MoveTo(x,y);

Image1->Canvas->LineTo(Image1->Width ,y);

y=(y-20);

}

Image1->Canvas->MoveTo(Image1->Width / 2,Image1->Height / 2);

double sbx1, sbx2, sbx3, sbx4, sbx5, sbx6;

sbx1=SpinEdit1->Value;

sbx2=SpinEdit2->Value;

sbx3=SpinEdit3->Value;

sbx4=SpinEdit4->Value;

sbx5=SpinEdit5->Value;

sbx6=SpinEdit6->Value;

int shirina, visota;

shirina=Image1->Width;

visota=Image1->Height;

if (2\*sbx1<=sbx2) {

double t1;

t1=StrToFloat(Edit1->Text);

Image1->Canvas->Pen->Color=ColorBox1->Selected;

Image1->Canvas->Pen->Width=3;

double x1, y1;

x1=sbx1\*cos(-3.14)\*cos(-3.14)+sbx2\*cos(-3.14);

y1=sbx1\*cos(-3.14)\*sin(-3.14)+sbx2\*sin(-3.14);

Image1->Canvas->MoveTo(shirina/2+x1,visota/2+y1);

for (double t = -3.14; t <= 3.2; t=t+t1) {

x1=sbx1\*cos(t)\*cos(t)+sbx2\*cos(t);

y1=sbx1\*cos(t)\*sin(t)+sbx2\*sin(t);

Image1->Canvas->LineTo(shirina/2+x1,visota/2+y1);

Image1->Canvas->MoveTo(shirina/2+x1,visota/2+y1);

}

} else {

ShowMessage("Условие 1 не соблюдено");

}

if ((sbx3<sbx4) && (sbx4<=2\*sbx3)) {

double t2;

t2=StrToFloat(Edit2->Text);

Image1->Canvas->Pen->Color=ColorBox2->Selected;

Image1->Canvas->Pen->Width=3;

double x2, y2;

x2=sbx3\*cos(-3.14)\*cos(-3.14)+sbx4\*cos(-3.14);

y2=sbx3\*cos(-3.14)\*sin(-3.14)+sbx4\*sin(-3.14);

Image1->Canvas->MoveTo(shirina/2+x2,visota/2+y2);

for (double t = -3.14; t <= 3.2; t=t+t2) {

x2=sbx3\*cos(t)\*cos(t)+sbx4\*cos(t);

y2=sbx3\*cos(t)\*sin(t)+sbx4\*sin(t);

Image1->Canvas->LineTo(shirina/2+x2,visota/2+y2);

Image1->Canvas->MoveTo(shirina/2+x2,visota/2+y2);

}

} else {

ShowMessage("Условие 2 не соблюдено");

}

if (sbx5>sbx6) {

double t3;

t3=StrToFloat(Edit3->Text);

Image1->Canvas->Pen->Color=ColorBox3->Selected;

Image1->Canvas->Pen->Width=3;

double x3, y3;

x3=sbx5\*cos(-3.14)\*cos(-3.14)+sbx6\*cos(-3.14);

y3=sbx5\*cos(-3.14)\*sin(-3.14)+sbx6\*sin(-3.14);

Image1->Canvas->MoveTo(shirina/2+x3,visota/2+y3);

for (double t = -3.14; t <= 3.2; t=t+t3) {

x3=sbx5\*cos(t)\*cos(t)+sbx6\*cos(t);

y3=sbx5\*cos(t)\*sin(t)+sbx6\*sin(t);

Image1->Canvas->LineTo(shirina/2+x3,visota/2+y3);

Image1->Canvas->MoveTo(shirina/2+x3,visota/2+y3);

}

} else {

ShowMessage("Условие 3 не соблюдено");

}

}

//---------------------------------------------------------------------------

**Unit 5:**

#include <vcl.h>

#pragma hdrstop

#include "Unit5.h"

#include <math.h>

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm5 \*Form5;

//---------------------------------------------------------------------------

float xc, yc, r; //xc координата x и yc координата y

int t1=0; //для таймера, отсчёт

\_\_fastcall TForm5::TForm5(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm5::CheckBox1Click(TObject \*Sender)

{

Timer1->Enabled=CheckBox1->Checked; //таймер включен

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm5::FormCreate(TObject \*Sender)

{

Timer1->Enabled=false; //таймер выключен Timer задаёт интервалы времени в приложении, по истечении которого запускается запрограммированное событие

xc=Form5->Width/2; //ширина точка по середине

yc=Form5->Height/2; //длина точка по середине

r=Width/2-300; //радиус (от ширины точки по середине вниз 300 пикселей)

}

//---------------------------------------------------------------------------

float Xcor(float xc, float r, int t){ //функция для расчета координаты Х для движущейся точки

float m;

m=round(xc+r\*sin(t\*M\_PI/180)-1);

return m;

}

float Ycor(float xc, float r, int t){ //функция для расчета координаты Y для движущейся точки

float m;

m=round(yc+r\*cos(t\*M\_PI/180)-1);

return m;

}

void \_\_fastcall TForm5::Timer1Timer(TObject \*Sender)

{

Form5->Repaint(); //очистка

Canvas->Brush->Style=bsClear;

Form5->Canvas->Ellipse(xc-100,yc-100,xc+100,yc+100); //круг

Canvas->Brush->Color=clRed; //заливка красным

Form5->Canvas->Ellipse(Xcor(xc,100,t1)-2,Ycor(xc,100,t1)-2,Xcor(xc,100,t1)+2,Ycor(xc,100,t1)+2); //точка

t1++;

}

//---------------------------------------------------------------------------

**Unit 6:**

#pragma hdrstop

#include "Unit6.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm6 \*Form6;

int i=0;

//---------------------------------------------------------------------------

\_\_fastcall TForm6::TForm6(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm6::CheckBox1Click(TObject \*Sender)

{

Timer1->Enabled=CheckBox1->Checked; //таймер включен

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm6::Timer1Timer(TObject \*Sender)

{

i++;

Image1->Picture->LoadFromFile("gif/"+IntToStr(i)+ ".jpg");

if(i==8)i=0;

}

//---------------------------------------------------------------------------