Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Воронежский государственный лесотехнический университет

им. Г.Ф. Морозова»

**ИНФОРМАЦИОННЫЕ СИСТЕМЫ: ЭТАПЫ РАЗВИТИЯ**

**И ВВЕДЕНИЕ В СПЕЦИАЛЬНОСТЬ**

Лабораторный практикум

Часть II

Воронеж 2018

УДК 004.432

Информационные системы: этапы развития и введение в специальность [Текст]: лабораторный практикум. Часть II / Т.В. Скворцова, И.С. Кущева; М-во образования и науки РФ, ФГБОУ ВО «ВГЛТУ им. Г.Ф. Морозова». – Воронеж, 2018. – 132 с.

**Рецензенты:** кафедра электротехники и автоматики ФГБОУ ВПО

«Воронежский государственный аграрный университет

имени императора Петра I», заведующий кафедрой,

доктор технических наук, профессор Афоничев Д.Н.

начальник лаборатории ОАО «НИИЭТ» к.т.н. А. И. Яньков

Печатается по решению учебно-методического совета ФГБОУ ВО «ВГЛТУ им. Г.Ф. Морозова» (протокол № \_\_ от \_\_\_\_\_\_\_ 2018 г.)

В лабораторном практикуме приведены методические рекомендации к проведению лабораторных работ в системе визуального объектно-ориентированного программирования С++ Builder.

Лабораторный практикум предназначен для студентов направления подготовки 09.03.02 – Информационные системы и технологии

© Скворцова Т.В., Кущева И.С., 2018

ISBN © ФГБОУ ВО «Воронежский государственный

лесотехнический университет имени

Г.Ф. Морозова», 2018

Оглавление

[ВВЕДЕНИЕ 4](#_Toc521578326)

[1.1. C++ Builder – система визуального объектно-ориентированного программирования 5](#_Toc521578327)

[1.2. Общие сведения о программах на C++ Builder 6](#_Toc521578328)

[1.3. Области видимости переменных и функций в программах на C++ Builder 10](#_Toc521578329)

[ЛАБОРАТОРНАЯ РАБОТА № 1 12](#_Toc521578330)

[Тема: «Изучение среды программирования C++ Builder. Программирование и отладка простейших программ». 12](#_Toc521578331)

[Задания для самостоятельной работы 29](#_Toc521578332)

[Контрольные вопросы 30](#_Toc521578333)

[ЛАБОРАТОРНАЯ РАБОТА № 2 31](#_Toc521578334)

[Тема: «Разработка алгоритмов и программ с использованием управляющих операторов». 31](#_Toc521578335)

[Задания для самостоятельной работы 49](#_Toc521578336)

[Контрольные вопросы 51](#_Toc521578337)

[ЛАБОРАТОРНАЯ РАБОТА № 3 53](#_Toc521578338)

[Тема: «Разработка программ с использованием циклических структур». 53](#_Toc521578339)

[Задания для самостоятельной работы 63](#_Toc521578340)

[Контрольные вопросы 64](#_Toc521578341)

[ЛАБОРАТОРНАЯ РАБОТА № 4 66](#_Toc521578342)

[Тема: «Изучение синтаксиса различных вариантов создания статических и динамических массивов на языке С++ Builder». 66](#_Toc521578343)

[Задания для самостоятельной работы 87](#_Toc521578344)

[Контрольные вопросы 88](#_Toc521578345)

[ЛАБОРАТОРНАЯ РАБОТА № 5 90](#_Toc521578346)

[Тема: «Изучение среды C++ Builder. Применение в программах символьных и строковых переменных». 90](#_Toc521578347)

[Задания для самостоятельной работы 113](#_Toc521578348)

[Контрольные вопросы 114](#_Toc521578349)

[ЛАБОРАТОРНАЯ РАБОТА № 6 116](#_Toc521578350)

[Тема: «Изучение среды C++ Builder. Применение в программах графических изображений». 116](#_Toc521578351)

[Задания для самостоятельной работы 128](#_Toc521578352)

[Контрольные вопросы 129](#_Toc521578353)

# ВВЕДЕНИЕ

Лабораторный практикум по учебной дисциплине «Информационные системы: этапы развития и введение в специальность» представляет комплекс работ по изучению системы визуального объектно-ориентированного программирования С++ Builder. Практикум содержит вводныйраздел, посвященный общим правилам выполнения лабораторных работ, в том числе правилам решения задач, и лабораторные работы.

Лабораторные работы включают краткое изложение теоретического материала, примерырешениязадач, рекомендации для выполнения заданий, задания для самостоятельных работ и контрольные вопросы по изучаемой теме.

Теоретический материал, примеры алгоритмов и программ и рекомендации по лабораторным работам студенты должны изучить самостоятельно.

Работывыполняютсястудентами по индивидуальным вариантам заданий, выдаваемых преподавателем. На контрольные вопросы по теоретическому материалу, примерам алгоритмов и программ и рекомендациям по лабораторным работам студент должен ответить перед выполнением лабораторных работ.

Приступая к выполнению лабораторной работы, необходимо освоить основные положения по изучаемой теме, уяснить конструктивные и программируемые возможности свойств элементов управления, внимательно изучить задания на выполнение работы, запрограммировать элементы интерфейса и прикладную задачу. Добиться правильной работы приложения. Выполнить необходимый вариант задания.

## 1.1. C++ Builder – система визуального объектно-ориентированного программирования

C++ Builder – одна из самых мощных систем визуального объектно-ориентированного программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного типа (класса). Программа, построенная по принципу объектной ориентации – это не последовательность операторов, реализующих некий жесткий алгоритм, а совокупность объектов и способов их взаимодействия. *Объект* – это совокупность свойств и методов, а также событий, на которые он может реагировать.

Создание объектов и написание программ функций обработчиков событий является важнейшей частью разработки графического интерфейса пользователя. Важнейшее значение при объектно-ориентированном программировании имеют принципы инкапсуляции, сущность которой заключается в том, что внешним объектам и пользователям прямой доступ к кодам объекта запрещен. Для пользователей программным продуктом доступ к методам и свойствам объектов и данным доступен только через пользовательский интерфейс в объеме, предусмотренном проектировщиком программного продукта.

Программирование вручную графического интерфейса программы (разработка привычных пользователю окон для ввода и отображения информации, кнопок, меню, обработка событий мыши и клавиатуры, включение в программы изображений и звука) требует больших затрат времени программиста. Для решения этой задачи существуют два подхода. Первый – стандартизация системных функций и появление пользовательских интерфейсов API. В них описаны функции, переменные, константы, к которым разработчик может обратиться при разработке программы. Второй – это появление визуального программирования, возникшего в Visual Basic и нашедшего воплощение в C++Builder. Это позволило свести разработку графического пользовательского интерфейса к простым и наглядным процедурам размещения имеющихся классов объектов (компонентов) из библиотеки C++ Builder на оконной форме.

Основное достоинство визуального проектирования заключается в том, что во время проектирования формы и размещения на ней компонентов C++ Builder автоматически формирует коды программы, включая в нее соответствующие фрагменты, описывающие свойства компонент.

## 1.2. Общие сведения о программах на C++ Builder

Программа на C++ Builder состоит из объявлений переменных, констант, типов, классов, функций и описания функций. Среди функций всегда имеется главная *WinMain.* Программа начинает работать именно с этой главной функции. Программа строится по модульному принципу и состоит из множества модулей. Все объекты компонентов размещаются на объектах – формах; для каждой формы создается отдельный модуль. Именно в модулях и размещается программа.

В обработчиках событий объектов – форм и компонентов пишутся коды программы. В процессе проектирования приложения C++ Builder автоматически создает коды головного файла, имеющего расширение (*.cpp*) и содержащего главную функцию проекта *WinMain*, коды заголовочных файлов интерфейса отдельных модулей и коды их файлов реализации. Чтобы увидеть код головного файла проекта надо в среде разработки С++ Builder в меню *Project* выбрать команду *View* *Sourse*. Типичный файл проекта имеет следующий вид:

Головной файл проекта:

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

//*директивы* *препроцессора*

#include <vcl.h>

#pragma hdrstop

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

*//макросы,* *подключающие* *файлы* *ресурсов* *и* *форм*

USEFORM("Unit1.cpp", Form1);

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

//*главная* *функция* *WinMain*

WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)

{

try

{

//*тело* *главной* *функции,* *инициализирующее* *объекты* *компонентов* *данного* *приложения*

Application->Initialize();

Application->CreateForm(\_\_classid(TForm1), &Form1);

Application->Run();//*начало* *выполнения* *программы*

}//*обработка* *исключений (аварийных* *ситуаций)*

catch (Exception &exception)

{

Application->ShowException(&exception);

}

catch (...)

{

try

{

throw Exception("");

}

catch (Exception &exception)

{

Application->ShowException(&exception);

}

}

return 0; //*завершение* *программы*

}

Комментарии, написанные курсивом, характеризуют назначение каждого раздела. Все описанные в головном файле операторы заносятся в него автоматически в процессе проектирования приложения.

Ниже приведены тексты заголовочного файла (расширение .*h*), содержащего описание класса формы, и файла реализации (расширение .*cpp*).

Комментарии в этих текстах поясняют, что и куда в этот код можно добавлять.

Заголовочный файл модуля формы

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

#ifndef Unit1H

#define Unit1H

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

// *сюда* *могут* *помещаться дополнительные* *директивы* *препроцессора (* *в* *частности,* *include),* *не* *включаемые* *в* *файл* *автоматически*

#include <Classes.hpp>

#include <Controls.hpp>

#include <StdCtrls.hpp>

#include <Forms.hpp>

#include <Buttons.hpp>

#include <ExtCtrls.hpp>

#include <math.h>

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

class TForm1 :public TForm {

\_\_published: // IDE-managed Components

//*приводятся,* *размещенные* *на* *форме* *компоненты*

TImage \*Image1;

TBitBtn \*BitBtn1;

TImage \*Image2;

TLabel\*Label1;

TLabel \*Label2;

TLabel\*Label3;

TButton \*Button1;

TButton \*Button2;

//*функции* *обработки* *событий*

void \_\_fastcallBitBtn1Click(TObject \*Sender);

void \_\_fastcallButton1Click(TObject \*Sender);

void \_\_fastcallButton2Click(TObject \*Sender);

private: // User declarations

// *закрытый* *раздел* *класса,* *сюда* *могут* *помещаться* *объявления*

*//типов,* *переменных,* *функций,* *включаемых* *в* *класс* *форм* *и*

*//не* *доступных* *для* *других* *модулей*

public: // User declarations

*//открытый* *раздел* *класса,* *сюда* *могут* *помещаться* *объявления* *//типов,* *переменных,* *функций,* *включаемых* *в* *класс* *форм* *и*

*//* *доступных* *для* *других* *модулей*

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

};

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

extern PACKAGE TForm1 \*Form1;

*//* *сюда* *могут* *помещаться* *объявления* *типов,* *переменных,*

*//* *функций, которые не* *включаются* *вкласс* *форы;* *доступ* *к* *ним* *из*

*//* *других* *блоков* *возможен* *только* *при* *соблюдении* *некоторых*

*//дополнительных* *условий*

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

#endif

Файл реализации модуля

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

#include <vcl.h>

#pragma hdrstop

#include "UnRis1.h"

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

#pragma package (smart\_init)

#pragma resource "\*.dfm"

//*сюда* *могут* *помещаться* *дополнительные* *директивы*

*//* *препроцессора* *(в* *частности,* *include),*

*//* *не* *включаемые* *в* *файл* *автоматически*

*//* *объявление* *объекта* *формы* *Form1* TForm1 \*Form1;

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

// вызов конструктора формы

\_\_fastcallTForm1::TForm1(TComponent\* Owner)

: TForm(Owner) {

// *сюда* *могут* *помещаться* *операторы,* *которые* *должны* // *выполняться* *при* *создании* *формы*

}

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

// *сюда* *могут* *помещаться* *объявления* *типов* *и* *переменных,*

// *доступ* *к* *которым* *из* *других* *модулей* *возможен* *только*

// *при* *соблюдении* *некоторых* *дополнительных* *условий;* *тут*

// *же* *должны* *быть* *реализации* *всех* *функций,* *объявленных* *в*

// *заголовочном* *файле,* *а* *также* *могут* *быть* *реализации*

// *любых* *дополнительных* *функций,* *не* *объявленных* *ранее* *функций* *обработки* *событий*

void \_\_fastcallTForm1::BitBtn1Click(TObject \*Sender)

{

const float Pi= 3.14159;

float x, y; //координаты точки функции

int px,py; // Координаты пиксела, соответствующие

//точки с координатами x,y

for (px = 0; px <= Image1->Width; px++)

{

//x – координата, соответствующая пикселу с координатой px

x = px\*4 \* Pi/Image1->Width;

y = sin(x);

// y - координата пиксела, соответствующая координате y

py = Image1 -> Height - (y + 1)\* Image1 -> Height/2;

// устанавливается черный цвет выбранного пиксела

Image1->Canvas-> Pixels[px][py] = clBlack;

}

}

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

void \_\_fastcallTForm1::Button2Click(TObject \*Sender)

{

Form1->Close();

}

Имена файлам модулей C++ Builder дает по умолчанию: для первого модуля оно равно *Unit1*, для второго *Unit2* и т. д.

## 1.3. Области видимости переменных и функций в программах на C++ Builder

1. Переменные, объявленные в заголовочном файле модуля или в файле его реализации вне описания класса и функций, являются глобальными. Они доступны везде внутри данного модуля. Для доступа к ним из внешних модулей в этих модулях должно быть повторено их объявление (без инициализации) с добавлением спецификации *extern*.

2. Функции, объявленные в *заголовочном* *файле* модуля вне описания класса, являются глобальными. Они доступны везде внутри данного модуля. Для доступа к ним из внешних модулей в этих модулях или надо повторить их объявление, или включить директивой *#include* в заголовочный файл того модуля, в котором функции описаны.

3. Функции, объявленные в *файле реализации* модуля, являются глобальными. Они доступны везде внутри данного модуля. Для доступа к ним из внешних модулей в этих модулях надо повторить их объявление.

4. Элементы (переменные и функции), объявленные в классе в разделе *private*, видимы и доступны только внутри данного модуля. При этом из функций, объявленных внутри класса, к ним можно обращаться непосредственно по имени, а из других функций – только со ссылкой на объект данного класса. Если в модуле описано несколько классов, то объекты этих классов взаимно видят элементы, описанные в их разделах *private*.

5. Элементы, объявленные в классе в разделе *public*, видимы и доступны для объектов любых классов и для других модулей, в которых директивой *#include* включен заголовочный файл данного модуля. При этом из объектов того же класса к ним можно обращаться непосредственно по имени, а из других объектов и процедур – только со ссылкой на объект данного класса.

6. В классах могут быть еще разделы *protected* – защищенные. Элементы, объявленные в классе в разделе *prоtected* ви-димы и доступны для любых объектов внутри данного модуля, а также для объ-ектов классов – наследников данного класса в других модулях. Объекты из других модулей, классы которых не являются наследниками данного класса, защищенных элементов не видят.

7. Элементы, объявленные внутри функции или блока, являются локальными, т. е. они видимы и доступны только внутри данной функции или данного блока. При этом время жизни переменных, объявленных внутри функции или блока, определяется временем активности данного блока. Сделать локальную переменную существующей постоянно можно с помощью спецификации *static*.

8. Переменные и функции, объявленные в головном файле проекта, являются глобальными для этого файла. Если требуется доступ к ним из других модулей, то для функций в них должны быть повторены их объявления, а для переменных – повторено объявление (без инициализации) со спецификацией *extern*.

9. Если во внутреннем блоке, обозначаемом как {}, объявлена переменная с тем же именем, что во внешнем блоке, или с тем же именем, что и глобальная переменная, то соответствующая внешняя или глобальная переменная в блоке не видна. В этом случае получить доступ к одноименной глобальной переменной можно только с помощью унарной операции разрешения области действия (::).

# ЛАБОРАТОРНАЯ РАБОТА № 1

# Тема: «Изучение среды программирования C++ Builder. Программирование и отладка простейших программ».

**Цель** **работы:** Изучение среды программирования C++ Builder: основные возможности, стандартные типы данных, выражения и операции, стандартные компоненты среды программирования и их применение для визуальной разработки программ.

**Упражнение 1.1.** Пусть в результате математической формализации поставленной задачи на разработку программного средства определено, что необходимо вычислить координаты точки, делящей отрезок [a1, a2] в отношении n1:n2 по формулам:

x = (x1 + vx2) / (1 + v); y = (y1 + vy2) / (1 + v),

где

v = n1 / n2;

x1, y1 – координаты точки a1;

x2, y2 – координаты точки a2.

**Исходными данными** для решения задачи являются:

– x1, y1 – координаты начальной точки и x2, y2 – координаты конечной точки, которые могут принимать как положительные, так и отрицательные целые и вещественные числа;

– значения величин n1 и n2, которые могут быть только положительными целыми либо вещественными числами.

**Выходными данными** являются координаты x, y точки, делящей отрезок в отношении n1/n2. Так как в формулах присутствует операция деления, то значения x и y будут вещественными числами.

**Алгоритм** можно считать оптимальным, если степень детализации обеспечивает приемлемый объём и наглядность, а также простоту отладки при решении и удобство анализа результатов.

Для данной задачи определения координат x, y точки, делящей отрезок в заданном отношении, расчеты могут проводиться в следующей последовательности:

1. Рассчитываем коэффициент v: v = n1 / n2.

2. Рассчитываем значение величины b = (1 + v).

3. Рассчитываем значение координаты x по формуле: x = (x1 + vx2) / b.

4. Рассчитываем значение координаты y по формуле: y = (y1 + vy2) / b.

Графическое изображение алгоритма представлено на рисунке 1.1.

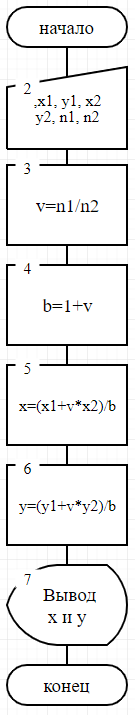


Рисунок 1.1 – Алгоритм решения задачи

**Разработка** **проекта** **на** **языке** **С++** **Builder**

Начать новый проект можно несколькими способами:

1. Команда File | New | VCL Forms Application C++Builder.

2. Команда File | New | Other. При выборе команды открывается окно депозитария New Items, хранящее образцы компонентов, форм и проектов (рисунок 1.2).

3. Кнопка быстрого вызова на панели инструментов.

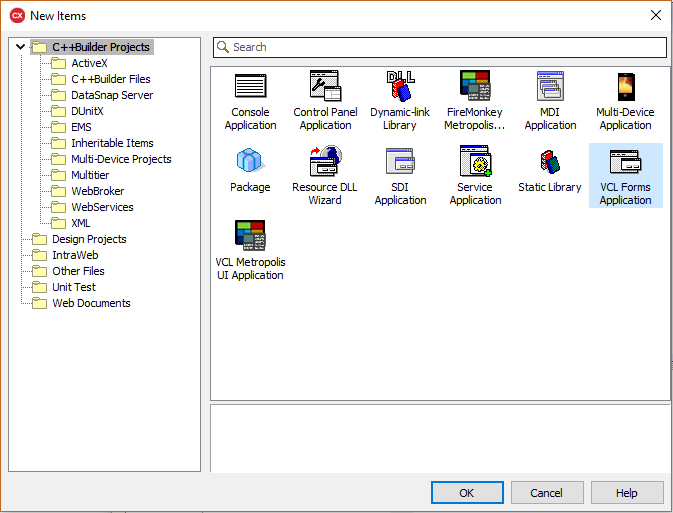


Рисунок 1.2 – Страница New окна депозитария New Items

**Разработка** **интерфейса** **пользователя**

Запустите С++ Builder, выбрав в меню Embarcadero RAD Studio команду С++Builder, и в меню File выберите команду New | VCL Forms Application C++Builder. На экране появится главное окно среды программирования Embarcadero RAD Studio, вид которого представлен на рисунке 1.3.

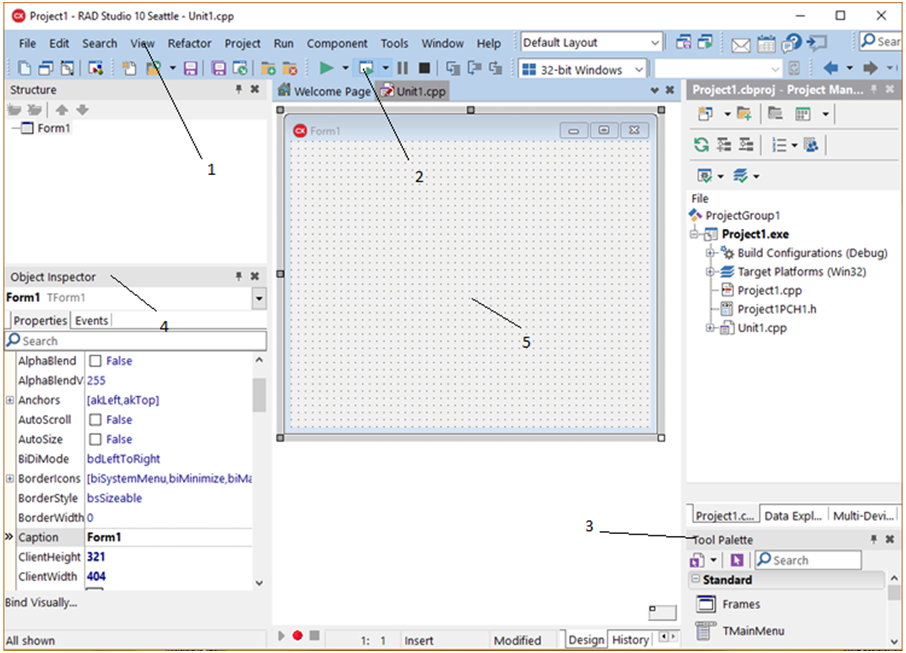


Рисунок 1.3 – Главное окно среды программирования Embarcadero RAD Studio

В верхней части главного окна размещены: меню (1), настраиваемая панель инструментов (2) и палитра компонентов (3). В средней части окна размещены панель Инспектора объектов (4) и форма нового приложения (5). Форма почти полностью закрывает окно редактора кода. На палитре компонентов, выполненной в виде многостраничного блокнота, размещены объекты (компоненты), используемые при визуальном проектировании программы. Формы являются основой приложений C++ Builder и представляют собой заготовку окна интерфейса пользователя разрабатываемой программы. Создание пользовательского интерфейса программы заключается в добавлении в окно формы объектов из панели компонентов. Каждому объекту присущи определенные свойства, характеризующие его вид, положение и поведение. Для редактирования свойств объектов предназначена панель Инспектора объектов.

Работа над новым проектом начинается с создания стартовой формы. Стартовая форма создается путем изменения значений свойств формы и размещения на форме необходимых объектов (полей ввода исходных данных, полей вывода результатов расчетов и текстовой информации для пользователя).

Разрабатываемый проект, входящие в его состав формы и объекты должны иметь уникальное имя, поэтому после открытия окна проекта необходимо присвоить проекту и стартовой форме имена. Для этого выполните следующие действия.

В меню File выберите команду Save As…. В открывшемся окне создайте папку, в которой будут сохраняться все файлы проекта.

**Выбор** **визуальных** **объектов** **для** **стартовой** **формы**

Для конструирования стартовой формы разместим на ней следующие объекты из палитры компонентов:

– метки (Label) для вывода информации для пользователей;

– текстовые окна (Edit) для ввода исходных данных;

–командные кнопки (Button) для передачи в программу команд.

Назначение объектов:

Label1 (метка) – для размещения содержания решаемой задачи;

Edit1– (текстовое окно) для ввода значения координаты x1 точки a1;

Edit2 – для ввода значения координаты y1 точки a1;

Edit3 –для ввода значения координаты x2 точки a2;

Edit4 –для ввода значения координаты y2 точки a2;

Edit5 –для ввода значения n1;

Edit6 –для ввода значения n2;

Label2 – поясняющая надпись для Edit1;

Label3 – поясняющая надпись для Edit2;

Label4 – поясняющая надпись для Edit3;

Label5 – поясняющая надпись для Edit4;

Label6 – поясняющая надпись для Edit5;

Label7 – поясняющая надпись для Edit6;

Label8-10 – поясняющие надписи для вывода результатов;

Label11 – для вывода значения x;

Label12 – для вывода значения y.

Для размещения на форме объектов в палитре компонентов необходимо на вкладке Standard щелкнуть на значке выбранного объекта (рисунок 1.4), установить курсор в ту точку формы, в которой должен быть левый верхний угол объекта, и еще раз щелкнуть кнопкой мыши. В результате на форме появится выбранный компонент. Объект можно перемещать, изменять его размеры и другие характеристики.

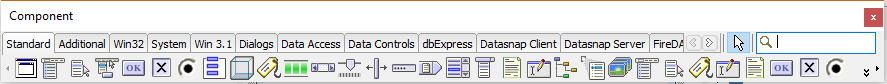
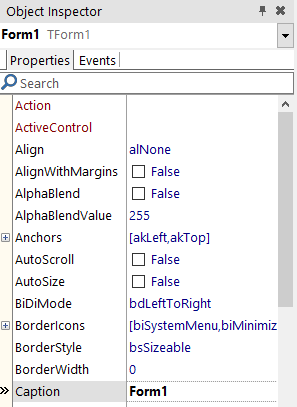
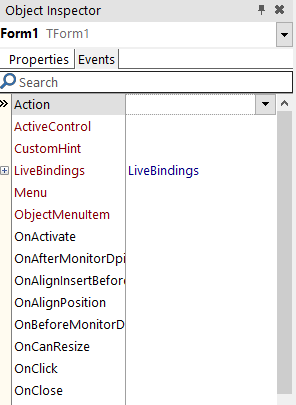


Рисунок 1.4 – Выбор объектов в палитре компонентов

**Задание** **свойств** **объектов**

Каждый объект C++ Builder имеет три разновидности характеристик: свойства, события и методы. Если выбрать компонент из палитры и добавить его к форме либо выделить его на форме щелчком мыши, инспектор объектов автоматически покажет в окне инспектора объектов свойства и события (рисунок 1.5), которые могут быть использованы этим объектом. В верхней части инспектора объектов имеется выпадающий список (селектор объектов), позволяющий выбирать нужный объект из имеющихся на форме.

Свойства являются атрибутами компонента, определяющими его внешний вид и поведение. Многие свойства компонента в колонке свойств имеют значение, устанавливаемое по умолчанию. Свойства компонента отображаются на странице свойств (Properties) Инспектора объектов (Object Inspector), представленного на рисунке 1.5 и используются для установки свойств во время проектирования. Можно определить свойства во время проектирования или написать код для видоизменения свойств компонента во время выполнения приложения.

Страница свойств объектов Страница событий объектов

Рисунок 1.5 – Инспектор объектов (Object Inspector)

Для задания свойств объекта необходимо выбрать объект, размещенный на форме, открыть страницу свойств в инспекторе объектов, выбрать определяемое свойство и изменить его с помощью редактора свойств.

Последовательность действий при размещении объектов на форме и задании их свойств:

1. Щелкните левой клавишей мыши по значку Label (рисунок 1.4) на панели компонентов. Переместите курсор на верхнюю часть формы и нажмите на левую клавишу мыши и, перемещая курсор, растяните прямоугольник метки нужного размера.

2. Откройте Object Inspector, если его нет на рабочем столе, выбрав в меню View опцию Object Inspector. В верхней части окна должно быть имя (по умолчанию) созданного объекта Label1.

3. Откройте страницу свойств, щелкнув по закладке Properties, и в открывшемся списке выберите свойство Caption. Измените значение свойства объекта, присвоенного ему по умолчанию, на *Определить координаты точки, делящей отрезок а1а2 в отношении n1:n2*.

4. Выделите объект Form1, выберите свойство Font, в поле справа щелкните мышкой по стрелке и в открывшемся окне Шрифт установите имя шрифта – MS Sans Serif, начертание – обычный, размер – 11. Установите AutoSize (авторазмер) в значение – true.

5. Аналогичным способом создайте метки Label2 – Label12 и текстовые окна Edit1 – Edit6, задайте указанным объектам значения свойств согласно таблице 1.1.

Таблица 1.1 – Создание меток и текстовых окон с заданием указанным объектам значений свойств

|  |  |  |  |
| --- | --- | --- | --- |
| **Объекты** | **Свойства** | | |
| **Name** | **Caption** | **Font** |
| 1 | 2 | 3 | 4 |
| Form1 | Form1 | Лабораторная работа 1 | Имя – MS Sans Serif, Начертание – Обычный, Размер – 11 |
| Label1 | Label1 | Определить координаты точки, делящей отрезок а1а2 в отношении n1:n2 | Цвет – красный |
| Label2 | Label2 | Координата x1 |  |

Окончание табл.1.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Label3 | Label3 | Координата y1 |  |
| Label4 | Label4 | Координата x2 |  |
| Label5 | Label5 | Координата y2 |  |
| Label6 | Label6 | Значение n1 |  |
| Label7 | Label7 | Значение n2 |  |
| Label8 | LabRezult | Результаты расчета |  |
| Label9 | LabX | Координата x |  |
| Label10 | LabY | Координата y |  |
| Label11 | LabZnx | пусто |  |
| Label12 | LabZny | пусто |  |
| Edit1 | Editx1 | Нет такого свойства |  |
| Edit2 | Edity1 | – |  |
| Edit3 | Editx2 | – |  |
| Edit4 | Edity2 | – |  |
| Edit5 | Editn1 | – |  |
| Edit6 | Editn2 | – |  |
| Button1 | Button1 | Вычислить |  |
| Button2 | Button2 | Выход |  |

На рисунке 1.6 представлен окончательный вид стартовой формы разрабатываемого проекта.

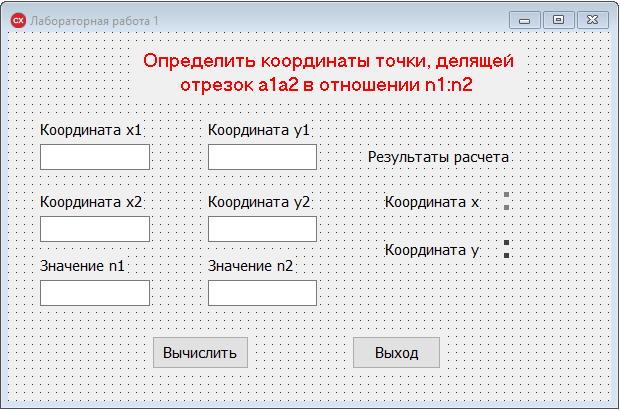


Рисунок 1.6 – Вид стартовой формы разрабатываемого проекта

**Написание** **кода** **программы**

Для запуска программы, разработанной на объектно-ориентированных языках необходима инициализация какого-либо события, которое распознается объектом. Это может быть щелчок мышью по объекту, наведение курсора на объект и др.

Каждый компонент имеет свой собственный набор обработчиков событий. Страница событий (Events) инспектора объектов показывает список событий, распознаваемых тем или иным объектом.

В C++ Builder следует писать функции программы, связывать их с событиями объектов. Создавая функцию для того или иного события, Вы поручаете программе выполнить написанную функцию, если это событие произойдет.

Для того чтобы написать функцию для события, нужно выбрать на форме с помощью мыши объект, который будет распознавать событие, затем открыть страницу событий инспектора объектов и дважды щелкнуть левой клавишей мыши на колонке значений рядом с событием, чтобы заставить C++ Builder сгенерировать заготовку (шаблон) функции и показать его в редакторе кода. При этом автоматически генерируется текст пустой функции, и редактор открывается в том месте, где следует вводить код. В верхней части заготовки указывается имя функции, содержащее имя объекта и имя события, при возникновении которого начнет выполняться функция. Функция события может иметь параметры, которые указываются после имени функции в круглых скобках.

Курсор позиционируется внутри операторных скобок { ... }, где пишется код функции (рисунок 1.7). Далее нужно ввести код, который должен выполняться при наступлении события.

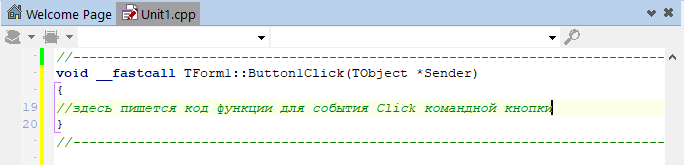


Рисунок 1.7 – Заготовка для написания функции для события «Щелчок мышью по командной кнопке»

Последовательность написания кода функции аналогична написанию функций в языке С++ и состоит из следующих этапов:

1. Выделите командную кнопку «Вычислить», в диспетчере объектов откройте закладку Events и дважды щелкните мышью по строке с событием Click. Откроется окно редактирования кода с заготовкой функции, и курсор будет мигать в том месте, где необходимо писать код функции.

2. Объявите переменные с указанием их типов. Синтаксис объявления: [определение типа] [список идентификаторов переменных];

Список идентификаторов может состоять из нескольких идентификаторов переменных, разделенных запятыми.

Например:

int x1, x2, b;

unsigned char c1, b2;

Именованные константы объявляются так же, как переменные, но с добавлением слова const. Например:

const float Pi = 3.14159;

Для рассматриваемого примера объявление переменных может быть записано в виде:

float x1, y1; // координаты точки a1

float x2, y2; // координаты точки a2

int n1, n2; // переменные определяющие соотношение,

//в котором делится отрезок a1 a2

float x, y; // Координаты точки, делящей отрезок на две части.

float v, b;

// вспомогательная переменная

3. Напишем код для считывания исходных данных из текстовых окон и присваивания их объявленным переменным.

Для ввода исходных данных в окно Edit используется свойство Text. Поэтому для считывания исходных данных надо написать выражение обращения к свойству Text. Синтаксис обращения к свойству объекта имеет вид: [имя объекта] -> [ имя свойства].

Данные, введенные в текстовые окна, принимают строковый тип, поэтому при их считывании в переменную числового типа необходимо их преобразовывать в числовой тип, а при программной записи в текстовое окно числовых данных их необходимо преобразовывать в строковый тип. Для такого преобразования используются функции:

StrToFloat (…) – преобразование строкового типа в числовой.

FloatToStr(…) – преобразование числового типа в строковый.

Для рассматриваемой задачи это будут следующие выражения:

x1 = StrToFloat(Editx1->Text);

y1 = StrToFloat(Edity1->Text);

x2 = StrToFloat(Editx2->Text);

y2 = StrToFloat(Edity2->Text);

n1 = StrToInt(Editn1->Text);

n2 = StrToInt(Editn2->Text);

4. Решение рассматриваемой задачи осуществляется с использованием арифметических операций. Арифметические операции применяются к действительным числам, целым числам и указателям. В языке С++ определены следующие бинарные операции (таблица 1.2).

Таблица 1.2 – Бинарные операции в языке С++

|  |  |  |  |
| --- | --- | --- | --- |
| **Обозначение** | **Операция** | **Типы** **операндов** **и** **результата** | **Пример** |
| + | Сложение | арифметические | a + b |
| – | Вычитание | арифметические | a – b |
| \* | Умножение | арифметические | a \* b |
| **/** | Деление | арифметические | a / b |
| % | Остаток целочисленного деления | целые | a % b |
| + | Унарный плюс | арифметические | +a |

Для арифметических операций действуют следующие правила.

1. В процессе вычисления автоматически осуществляется преобразование типов по принципу: если операция имеет операнды разных типов, то тип операнда «младшего» типа приводится к «операнду» старшего типа. Иначе говоря, менее точный тип приводится к более точному типу. Например, если в операции участвуют короткое целое и длинное целое, то короткое приводится к длинному; если участвуют целое и действительное (вещественное), то целый операнд приводится к действительному.

2. Бинарные операции сложения и вычитания применимы к целым и действительным числам, а также к указателям.

3. В операциях умножения и деления операнды могут быть любых арифметических типов.

4. В операции вычисления остатка отделения оба операнда должны быть целыми числами.

5. В операциях деления и вычисления остатка второй операнд не может быть равен нулю.

6. Если оба операнда в этих операциях целые, а результат деления является не целым числом, то знак результата вычисления остатка совпадает со знаком первого операнда, а для операции деления используются следующие правила:

a. Если первый и второй операнды имеют одинаковые знаки, то результат операции деления – наибольшее целое, меньшее истинного результата деления.

b. Если первый и второй операнды имеют разные знаки, то результат операции деления – наименьшее целое, большее истинного результата деления.

с. Округление всегда осуществляется по направлению к нулю.

Все это относится к арифметическим операциям, но не относится к операциям присваивания. При присваивании осуществляется автоматическое приведение типа результата арифметической операции к типу левого операнда. Это можно исправить, применив операцию явного приведения типов. Она записывается в виде (тип операции):

Например:

Int m = 1, n = 2;

double a = m / n; // результат равен 0, не верно, но:

double a = double m / n; //результат равен 0,5, верно.

Код алгоритма решаемой задачи может быть записан в виде:

v = (float) n1 / n2; // так как при делении целых чисел

//результат округляется до целого, то применимо

//принудительное преобразование //переменных целого типа в действительный тип

b = 1 + v;

x = (x1 + v\*x2) / b;

y = (y1 + v\*y2) / b;

5. Напишем код выражений для вывода полученных результатов в заранее подготовленные поля меток LabZnx и LabZny, используя поле свойства Caption.

LabZnx -> Caption = FloatToStr(x);

LabZny -> Caption = FloatToStr(y);

6. Напишем функцию для события «Щелчок мышью по командной кнопке «Выход», позволяющей выйти из программы.

Для этого выделим командную кнопку Выход, в окне диспетчера задач откроем закладку Events, щелкнув дважды по строке Click, и в открывшемся окне редактора кода напишем следующее выражение:

Form1–> Close();

В данном выражении происходит обращение к методу Close (закрывать), который может реализовываться формой.

Метод является функцией, которая связана с объектом, и которая объявляется как часть объекта. Создавая обработчики событий, можно вызывать методы, используя следующую нотацию: ->, например:

Edit1->Show(); // показывать.

**Сохранение** **проекта**

Проект – это набор файлов, используя которые компилятор создает выполняемый файл программы (.exe-файл). Чтобы сохранить проект, нужно в меню File выбрать команду Save / Save As. Если проект еще ни разу не был сохранен, то C++ сначала предложит сохранить модуль (содержимое окна редактирования кода). В результате щелчка на кнопке Save будут созданы файлы с расширениями: .cpp, .h, .dfm, .cpproj.

После отладки программы необходимо провести тестирование программы, используя контрольные исходные данные и полученные для них результаты расчетов, проведенных другими способами.

Если программа показывает результаты расчета, совпадающие с контрольными расчетами, можно считать, что программа составлена верно (рисунок 1.8).

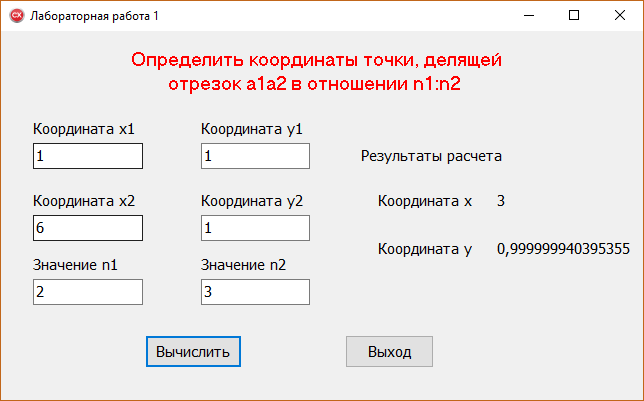


Рисунок 1.8 – Результат расчета

**Упражнение 1.2.** В треугольнике заданы две стороны а, b и угол напротив одной из них А. Определить сторону с, углы В и С и площадь треугольника S (Исходные данные вводить в текстовые окна, результаты выводить в метки, линейные размеры вводить и выводить в см, площадь – в см2, углы – в градусах).

1. Исходные данные: a, b – стороны треугольника, вещественные положительные числа; A – угол, вещественное положительное число в диапазоне 0 < A < 180 град.

Результаты решения задачи:

B – угол, вещественное положительное число в диапазоне 0<B<180 град. C – угол, вещественное положительное число в диапазоне 0 < C < 180 град.; с – сторона треугольника, вещественное положительное число; S – площадь треугольника, вещественное число.

2. Определение требования к интерфейсу пользователя:

Интерфейс должен содержать:

* три текстовых окна для ввода исходных данных;
* три метки, поясняющие назначение этих окон;
* четыре текстовых окна для вывода результатов решения задачи;
* четыре метки, поясняющие назначение меток вывода результатов;
* командную кнопку для запуска программы на решение. Командную кнопку для выхода из программы.

3. Выбор метода решения задачи:

Решение задачи основано на использовании тригонометрических формул: из теоремы синусов: В = arcsin ((b / a) sin A);

сумма углов треугольника:180° – С = 180 – (A + B);

из теоремы синусов: c = a (sin C / sin A);

S = 0.5 ab sin С.

4. Алгоритм решения задачи линейный.

Графическое изображение алгоритма представлено на рисунке 1.9.

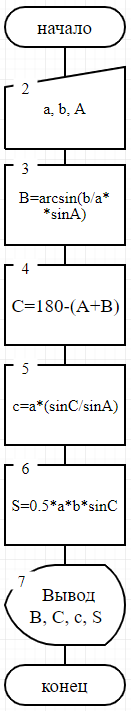


Рисунок 1.9 – Алгоритм решения задачи

5. Проектирование интерфейса.

Создадим новый проект, разместим на форме требуемые объекты, как показано на рисунке 1.10. Размер шрифта для всех объектов можно установить в свойстве Font формы. В этом случае все объекты формы будут наследовать это свойство. Для текстовых окон удалить значение свойства Техt.

6. Написание кода программы.

В формулах используются тригонометрические функции:

y = sin (x), x = arcsin (y)

где х – величина угла в радианах, 0 < y < 1.

Для пересчета рад. в град. и обратно используются формулы:

x (рад). = 180 \* x (град) / π,

x (град). = π \* x (рад) / 180.

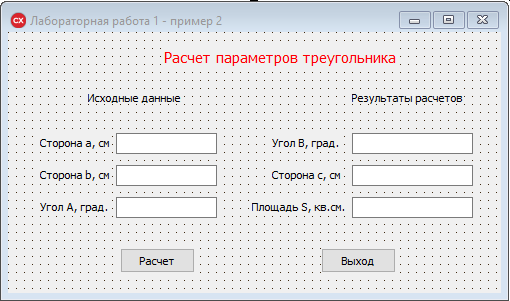
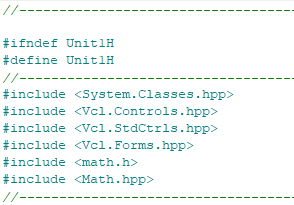


Рисунок 1.10 – Интерфейс программы вычисления элементов треугольника

Следовательно, потребуется подключить файлы с математическими и тригонометрическими функциями. Для их включения в проект откройте заголовочный файл модуля формы, щелкнув мышкой по закладке Unit1.h внизу окна редактирования кодов и запишите директиву подключения файлов math.h и Math.hpp:



Выберите для командной кнопки событие Button1Click и напишите приведенный ниже код обработки события:

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

float a, b, c, A,B,C,S;

const float e = 180;

float A1, B1, C1, d;

a =StrToFloat(Edit1->Text);

b=StrToFloat(Edit2->Text);

A= StrToFloat(Edit3->Text); A1 = M\_PI\*A/180;

d = (b/a)\* sin(A1); B1 = ArcSin(d);

B = 180\* B1/M\_PI; C =e-(A+B);

C1 = M\_PI\*C/180;

c=a\*sin(C1)/sin(A1);

S= 0.5\*a\*b\*sin(C1);

Edit4->Text = FloatToStr(B);

Edit5->Text = FloatToStr(C);

Edit6->Text = FloatToStr(c);

Edit7->Text = FloatToStr(S);

}

Для командной кнопки EXIT напишите следующую функцию обработки события для выхода из программы:

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

{

Form1-> Close();

}

Введите контрольные исходные данные и проверьте работоспособность программы (рисунок 1.11).

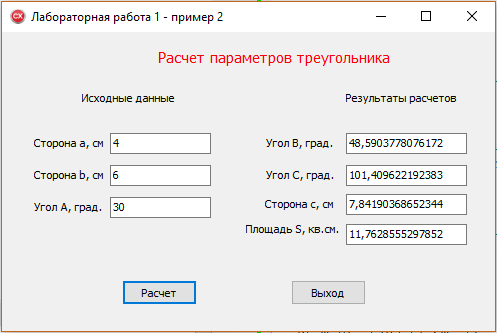


Рисунок 1.11 – Результат расчета

## Задания для самостоятельной работы

Разработайте алгоритм, интерфейс пользователя, программу решения задачи:

1. Площади трапеции, где *а* и *b* – длины оснований; *h* – высота трапеции:

*S*=0,5(*a*+*b*)*h*.

2. Площадь треугольника, если известны координаты вершин его углов: *a* – *x*1, *y*1; *b* – *x*2, *y*2; *c* – *x*3, *y*3.

3. Величинутокачерезцепьиздвухпоследовательно соединенных сопротивлений, *R*1, *R*2 и напряжении сети – *v*, В.

4. Пересчет веса из фунтов в килограммы (один фунт – это 405,9 грамма).

5. Программу пересчета расстояния из километров в версты (одна верста – это 1066,8 м).

6. В треугольнике заданы две стороны *а,* *b* и площадь *S.* Вычислить сторону *c*, углы *A,* *B,* *C* треугольника.

7. В треугольнике заданы стороны *a,* *b* и половина периметра *р.* Вычислить сторону *c* и углы *A,* *B,* *C* треугольника.

8. . При x=14.26, y=-1.22, z=3.5×10-3.

9. . При x=0.4×104, y=-0.875, z=-0.475×10-3.

10. В треугольнике заданы углы *А,* *В* и радиус описанной окружности *R.* Вычислить стороны a, b, c и угол C треугольника.

11. В треугольнике заданы стороны *a,* *b* и радиус описанной окружности *R.* Вычислить сторону *c*, и углы *A,* *B,* *C* треугольника.

12. В треугольнике заданы углы *А,* *С* и высота *hb.* Вычислить стороны *a,* *b,* *c* и угол *B* треугольника.

13. В треугольнике заданы стороны *а,* *с* и высота *hb.* Вычислить сторону b и углы *A,* *B,* *C* треугольника.

14. В треугольнике заданы угол *А,* сторона *с* и высота *hb.* Вычислить стороны *a,* *b* и углы *B,* *C* треугольника.

15. В ромбе заданы координаты вершин *ха,* *уа,* *хь,* *уь,* *хс,* *ус* и *xd,* *yd* . Вычислить стороны и диагонали ромба треугольника.

## Контрольные вопросы

1. Перечислите состав программного проекта в среде C++ Builder.

2. Охарактеризуйте структуру головного файла проекта.

3. Охарактеризуйте структуру заголовочного файла формы.

4. Охарактеризуйте структуру файла реализации формы.

5. Охарактеризуйте области видимости переменных и функций в проекте.

6. Охарактеризуйте состав главного окна интегрированной среды разработки C++ Builder.

7. Охарактеризуйте состав и назначение панели инструментов управления проектом.

8. Состав панели компонентов, назначение и вызов объектов с панели компонентов и их размещение на форме.

9. Окно редактирования кода, его состав, назначение, работа с ним.

10. Охарактеризуйте назначение и состав Инспектора объектов.

11. Свойства объекта и назначение их объектам. Обязательные свойства объекта.

12. Задание свойств объектам при проектировании интерфейса.

13. Задание и изменение свойств объектов при написании программы.

14. События, их роль в программе на языке C++ Builder.

15. Функции обработки событий и их написание в программе.

16. Как записать передачу данных из текстового окна в программу.

17. Что такое комментарий, для чего он служит и как оформляется?

18. Что представляет собой оператор, какие функции выполняют операторы в программе?

# ЛАБОРАТОРНАЯ РАБОТА № 2

# Тема: «Разработка алгоритмов и программ с использованием управляющих операторов».

**Цель** **работы:** Ознакомиться с технологией использования управляющих операторов при разработке ветвлений в алгоритмах решения задач.

**Теоретические** **сведения**

Даны переменные a и b (a ≠ b). Разработать алгоритм, позволяющий переменной max присвоить численное значение наибольшей переменной.

Математическая формулировка задачи:

Анализ задачи показывает:

1. Существуют два возможных пути решения задачи: max = a ( ветвь 1) и max = b ( ветвь 2).

2. Задача о том, какое численное значение какой из переменных, a или b, следует присвоить переменной max, может быть решен только после проверки одного из условий: a > b или a < b. Проверке может быть подвергнуто только одно из двух условий, поскольку они являются взаимоисключающими.

3. Проверка выбранного условия производится в процессе выполнения программы, следовательно, разработчику программы результат этой проверки заранее не известен. Поэтому в алгоритме нужно предусмотреть возможность вычислений по каждой из двух ветвей. Схема алгоритма представлена на рисунке 2.1.

Алгоритм состоит из линейной структуры, в которую вложен оператор проверки условия a > b и в зависимости от выполнения условия осуществляется выбор, по какой ветви продолжать вычисления.

Алгоритмы, в которых выбор процесса вычисления зависит от выполнения каких-либо условий, относятся к алгоритмам с управляющими структурами.

Оператор условной передачи управления в программе используют для программирования ветвлений, т. е. таких вычислений, когда при определенных условиях возникает необходимость выбора одного из возможных вариантов вычислений.

В зависимости от количества проверяемых условий операторы условной передачи управления делятся на *простые* и *сложные*.



Рисунок 2.1 – Схема алгоритма с управляющими структурами

*Простыми* называются операторы условной передачи управления с проверкой одного условия. *Сложными* – операторы с проверкой двух (нескольких) условий.

Простые операторы условной передачи, в свою очередь, по сложности проверяемого условия можно разделить на элементарные и составные.

Структура оператора условной передачи управления предполагает наличие в своем составе:

1. Условия L, которое записывается в виде логического выражения и может принимать значение true – истина или false – ложь.

2. Двух или нескольких альтернативных (взаимоисключающих) путей решения задачи, называемых ветвями и представляющих собой типовые линейные структуры, причем математическая формулировка задачи каждой ветви ставит в соответствие вполне определенное значение условия L.

В качестве условия может выступать, например, операция сравнения двух чисел. Если, допустим, условие записано в виде a > b, то результатом проверки такого условия может быть либо утверждение «true» (истина), либо «false» (ложь). При этом каждому результату должна соответствовать одна из двух возможных ветвей.

Существует три разновидности ветвящейся структуры: ответвление, разветвление и условный оператор множественного выбора (переключение) (рисунок 2.2).

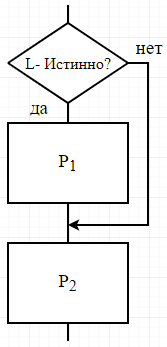
  

Рисунок 2.2 – Разновидности структуры оператора условной передачи управления: а) ответвление; б) разветвление; в) переключение

**Оператор ответвления.** Для ответвления характерно наличие операций только в одной из двух ветвей, а для разветвления – в обеих. Отличительной особенностью переключения является то, что условие L может принимать несколько значений, в соответствии с каждым из которых будет выполняться одна из нескольких ветвей P1, P2, ..., Pn.

При реализации структур оператора условной передачи управления на языке С++ Builder используются операции отношения и эквивалентности двух операндов (таблица 2.1). Они возвращают true – истина, если указанные соотношения операндов выполняются, и false – ложь, если соотношение не выполняется.

Таблица 2.1 – Операции отношения и эквивалентности

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение | Операция | Типы операндов | Пример записи |
| = = | Равно | Арифметический, указатели | I == Max |
| ! = | Не равно | Арифметический, указатели | X != Y |
| < | Меньше чем | Арифметический, указатели | X < Y |
| <= | Меньше или равно | Арифметический, указатели | X <= Y |
| > | Больше чем | Арифметический, указатели | X > Y |
| >= | Больше или равно | Арифметический, указатели | X >= Y |

Операнды должны быть совместимых типов, за исключением целых и действительных типов, которые могут сравниваться друг с другом.

Простой оператор условной передачи управления может быть записан в виде:

if (условие) оператор;

Скобки, обрамляющие условие, обязательны.

Например, в результате выполнения операторов:

a = 10;

b =20;

if (b > a) c = b;

переменная с станет равной максимальному из чисел a и b, поскольку оператор c = b будет выполнен только при b **>** a.

Условием может быть арифметическое выражение, которое может быть преобразовано в булев тип. Поскольку в С++ арифметическое (целое и действительное) значение может преобразовываться к булеву типу (любое ненулевое значение воспринимается как true, а нулевое – как false, то условие может иметь целый тип. Например:

int a, b, c, d;

…………

if ( a – b/c) d = c;

………….

В результате выполнения указанных операторов, если выражение (a – b**/**c) будет иметь ненулевое значение (положительное или отрицательное), то переменной d будет присвоено значение c, если выражение будет иметь нулевое значение, будет выполняться оператор, следующий за условным оператором.

Не рекомендуется в условии использовать арифметическое выражение с действительными числами, так как в результате ошибок округления результат вычисления может не быть равен нулю, даже когда чисто теоретически результат дает нуль.

**Оператор разветвления**. Форма записи структуры разветвления имеет вид:

if (условие) оператор1;

else оператор2;

Если условие возвращает true, то выполняется первый из указанных операторов, а в противном случае выполняется второй оператор. Обратите внимание, что в конце первого оператора перед ключевым словом else ставится точка с запятой.

Например:

if (j == 0)

ShowMessage (“Деление на нуль”);

else

Result = i/j;

Смысловое содержание приведенных операторов состоит в следующем: если значение переменной j = 0, то выводится предупреждение, что в следующем операторе будет выполняться деление на нуль и выполнение этого оператора пропускается, в противном случае выполняется оператор Result = i/j.

В качестве первого и второго оператора могут использоваться составные операторы:

if (j == 0)

{

ShowMessage (“Деление на нуль”);

Result = 0;

}

else

{

Result = i/j;

B = Result\*d;

}

**Вложенные конструкции операторов.** При написании программы могут использоваться условные операторы, имеющие вложенные конструкции, когда внутри одного условия проверяются другие условия. Схемы вложенных конструкций оператора условной передачи управления представлены на рис 2.3.

Рисунок 2.3 – Схемы вложенных условных операторов

Запись оператора условной передачи управления с вложенными конструкциями имеет вид:

If ( условие1)

If ( условие2)

Оператор1;

Else оператор2;

Else оператор 3;

При написании операторов условной передачи управления с вложенными конструкциями необходимо проверять соответствие if и else. Компилятор всегда считает, что else относится к последней из конструкций if, в которой не было раздела else.

Например в конструкции:

if ( условие1)

if ( условие2)

Оператор1;

else оператор2;

else будет отнесено компилятором ко второй конструкции if , т. е. опера-тор2 будет выполняться, если первое условие истинно, а второе – ложно. Данная запись оператора тождественна следующей записи:

if ( условие1)

{

if ( условие2)

оператор1;

else оператор2;

}

Если необходимо отнести к первому, то выражение надо записать в явном виде с помощью фигурных скобок:

if ( условие1)

{

if ( условие2) оператор1;

}

else оператор2;

**Оператор с составным условием.** При составлении конструкции оператора условной передачи управления могут использоваться составные условия, в которых условия соединены логическими операциями

Таблица 2.2 – Основные логические операции

|  |  |  |
| --- | --- | --- |
| Обозначение | Операции | Пример |
| ! | Отрицание НЕ | !A |
| && | Логическое И | A && B |
| | | | Логическое ИЛИ | A | | B |

Оператор условной передачи управления с составной конструкцией условия записывается в виде:

if ((условие1 && условие2) | | условие3) оператор1;

else оператор2;

В данном операторе условной передачи управления при выполнении условия1 и условия2 или условия3 выполняется оператор1, в противном случае оператор2.

Имеется ещё одна форма записи оператора условной передачи управления – условная операция. Синтаксис ее записи имеет вид:

условие ? оператор1 :оператор2;

Результатом выполнения условной операции является выполнение оператора1, если условие истинно, либо оператор2, если условие ложно.

Например, оператор:

max = ( x > y) ? х : у;

присваивает переменной max наибольшее из значений переменных х и у.

**Оператор множественного выбора (переключатель).** Оператор множественного выбора (иногда называют переключателем) позволяет провести анализ значения некоторого выражения и в зависимости от его значения выполнить те или иные действия. В общем случае формат записи оператора switch следующий:

switch (выражение выбора)

{

case значение 1:оператор 1

break;

case значение 2: оператор 2

break;

…………………………………..

case значение k: оператор k

break;

[ default: оператор k+1]

}

где оператор default не является обязательным.

В этой конструкции выражение выбора (условия, указанного в круглых скобках) должно иметь порядковый тип – целый, перечисляемый и т. д. Поэтому, например, нельзя использовать выражения, возвращающие действительные числа или строки.

Значения, указываемые в метках case, должны быть константными выражениями, соответствующие возможным значениям выбора. После значения ставится двоеточие «:», а затем пишется оператор (можно писать составной оператор), который должен выполняться, если выражение приняло указанное в метке значение.

При выполнении оператора switch реализуется последовательность операторов, соответствующая текущему значению выражения выбора. Если значение выбора совпало со значением, указанным в метке, то выполняется оператор, записанный после этой метки, после чего, если не принять соответствующих мер, будут выполняться все последующие операторы остальных меток. Поскольку это обычно нежелательно, то, как правило, после оператора, который должен выполняться, записывают оператор:

break;

Если это значение не совпадает ни с одним из перечисленных значений и оператор default присутствует, выполняется следующий за ним оператор. Если не нашлось соответствующего значения выражения выбора и оператор default отсутствует, то действие оператора switch аналогично пустому оператору.

**Упражнение 2.1.** Разработать алгоритм, интерфейс пользователя и программу решения задачи ветвящейся структуры с разветвлением.

Даны переменные a и b (a≠b). Разработать программу, позволяющую определить максимальное или минимальное из этих двух чисел и вывести на форму по желанию пользователя максимальное либо минимальное число:

При разработке интерфейса необходимо предоставить возможность пользователю изменять шрифт и цвет шрифта.

Так как необходимо найти максимальное либо минимальное число, то из результата проверки условий следует: если число a имеет максимальное значение, то число b имеет минимальное значение и наоборот. Следовательно, для решения поставленной задачи достаточно проверки только двух указанных условий.

Для организации вывода максимального либо минимального числа можно использовать стандартные объекты панели компонентов: RadioButton (радиокнопки), RadioGroup (группы радиокнопок) и др.

Алгоритм состоит из линейной структуры, в которую вложена типовая ветвящаяся структура – разветвление (рисунок 2.4).



Рисунок 2.4 – Схема алгоритма решения задачи

**Разработка интерфейса пользователя.**

Запустим C++ Bilder и создадим новый проект. На открывшейся форме разместим объекты: Label1, Label2, Label3, Label4, Label5, Edit1, Edit2, RadioButton1, RadioButton2. Для управления переключателями используется свойство Checked. Которое принимает значение true, если переключатель отмечен, и false, если переключатель не отмечен. В качестве исходного состояния переключателей установим для одного из них значение свойства Checked, равное true, для другого – false.

Разместив на форме указанные объекты, сделаем соответствующие надписи, как показано на рисунке 2.5.

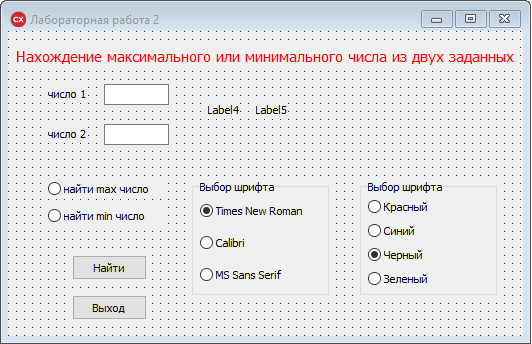


Рисунок 2.5 – Форма с интерфейсом пользователя

Для выбора шрифта и цвета шрифта объекты – RadioGroup1и RadioGroup2, в каждом из которых разместим по три переключателя, работающих согласованно. Управление переключателями осуществляется через свойство ItemIndex, которое в зависимости от числа переключателей может принимать значение по числу переключателей. В данном примере используются три переключателя, поэтому свойство ItemIndex для включенного переключателя может принимать значение {1, 2, 3} соответственно.

Написание кода программы будем производить в следующей последовательности:

1. Напишем код функции обработки события Click для кнопки Выход.

2. Напишем код функции обработки события Click для кнопки Найти, включив в неё операторы выбора варианта решения задачи в зависимости от отмеченного переключателя RadioButton.

3. Напишем код функции обработки события Click для объекта RadioGroup1 и RadioGroup2. Листинг программы приведен ниже.

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

{

//поиск максимального и минимального числа

int a,b,c,d;

a = StrToInt(Edit1->Text);

b = StrToInt(Edit2->Text);

if (a >= b)

{

c = a;

d = b;

}

else

{ c = b;

d = a;

}

// выбор выводимого числа в метку в зависимости от включенной радио-кнопки

if (RadioButton1->Checked)

{

Label5->Caption = "";

Label4->Caption = "max";

Label5->Caption = IntToStr(c);

}

else

{Label4->Caption ="min";

Label5->Caption = IntToStr(d);

}

}

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

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

{

//поиск максимального и минимального числа

int a,b,c,d;

a = StrToInt(Edit1->Text);

b = StrToInt(Edit2->Text);

if (a >= b)

{

c = a;

d = b;

}

else

{ c = b;

d = a;

}

// выбор выводимого числа в метку в зависимости от включенной радио-кнопки

if (RadioButton1->Checked)

{

Label5->Caption = "";

Label4->Caption = "max";

Label5->Caption = IntToStr(c);

}

else

{Label4->Caption ="min";

Label5->Caption = IntToStr(d);

}

}

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

}

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

void \_\_fastcall TForm1::RadioGroup1Click(TObject \*Sender)

{

//изменения шрифта текста в метке Label5

if (RadioGroup1->ItemIndex == 0) Label5->Font->Name = "Times New Roman";

if (RadioGroup1->ItemIndex ==1) Label5->Font->Name ="Calibri";

if (RadioGroup1->ItemIndex == 2) Label5->Font->Name = "Tahoma";

}

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

void \_\_fastcall TForm1::RadioGroup2Click(TObject \*Sender)

{

// изменение цвета текста в метке Label6

if (RadioGroup2->ItemIndex ==0) Label5->Font->Color =clRed;

if (RadioGroup2->ItemIndex ==1) Label5->Font->Color =clBlue;

if (RadioGroup2->ItemIndex ==2) Label5->Font->Color =clBlack;

if (RadioGroup2->ItemIndex ==3) Label5->Font->Color =clGreen;

}

Проверьте работоспособность программы.

**Упражнение 2.2.** Разработать алгоритм, интерфейс пользователя и программу для решения задачи с глубиной вложения ветвящейся структуры, равной двум, с использованием логических выражений.

Составить программу решения следующей задачи:

При разработке интерфейса предусмотреть ввод исходных данных и вывод результата расчетов на форму.

Внешняя спецификация задачи:

a – положительное целое;

x – положительное целое;

Z – вещественное.

При решении задачи используются стандартные математические функции: возведение в степень и извлечение корней 2, 3 и 5 степени.

Следовательно, требуется подключить файлы с математическими и тригонометрическими функциями math.h и Math.hpp в заголовочный файл модуля формы, как показано в лабораторной работе 1.

В отличие от предыдущего примера, где в качестве условий в операторах if использовались выражения отношения, здесь условия представляют собой более сложные логические выражения, содержащие логические операции «и» и «или».

Алгоритм решения задачи представлен на рисунке 2.6.



Рисунок 2.6 – Алгоритм решения задачи

Создайте проект, на форме разместите метки и текстовые окна для информирования пользователя и ввода исходных данных. Вывод результатов расчета будет осуществляться в окно редактирования списков Memo.

Для вывода строки данных в окно Memo используется выражение:

Memo1->Lines->Add(FloatToStr(z));

Вид формы проекта после размещения указанных объектов представлен на рисунке 2.7.

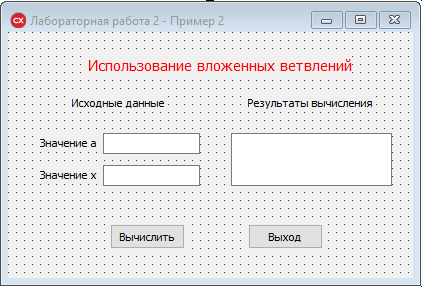


Рисунок 2.7 – Интерфейс формы

Выделите командную кнопку «Вычислить», в Инспекторе объектов перейдите на вкладку Events, выделите событие Click и в окне редактора кодов в секции для события Click командной кнопки напишите код обработки события:

void\_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

unsigned short a,x;

float z,z1,b;

const float b1 =1;

b = b1 / 3;

a = StrToInt(Edit1->Text);

x = StrToFloat(Edit2->Text);

if(a==2 && (x>0 && x<=4))

{

z =sqrt(float (a\*x));

//вывод результата в окно Memo

Memo1->Lines->Add(FloatToStr(z));

}

else {if (a==3 && (x>4 && x<=16))

{

z1 =float (a\*a\*x);

z = pow(z1,b); .// функция возведения в степень (Z1)b

Memo1->Lines->Add(FloatToStr(z));

}

else

{

b =b1/5;

z1 =float (a\*a\*a\*x);

z = pow(z1,b);

Memo1->Lines->Add(FloatToStr(z));

}

}

}

Проверьте работу программы.

**Упражнение 2.3.** Разработать программу, позволяющую определить результаты тестирования. В зависимости от количества правильных ответов интерфейс программы должен позволять вводить число правильных ответов и выводить результат тестирования с комментариями.

Внешняя спецификация программы. Общее число заданных вопросов – положительное целое. Число правильных ответов – положительное целое. Оценка за ответ, соответствующая числу правильных ответов – положительное целое. Комментарии к оценке – строковый тип. В качестве комментариев для результатов будем использовать следующие строковые выражения:

"Поздравляем, вы получили 5";

"Хорошо, у вас 4";

"У вас всего лишь 3";

"Плохо, у вас 2";

"Ужасно! Учите лучше!";

Решение задачи может быть описано выражением:

Алгоритм решения задачи с использованием оператора множественного выбора представлен на рисунке 2.8.



Рисунок 2.8 – Алгоритм решения задачи

Создайте проект, на форме разместите метки и текстовые окна для информирования пользователя, ввода исходных данных и вывода результата тестирования, как показано на рисунке 2.9.

Код программы необходимо написать для функции обработчика события Click командной кнопки Оценка. Для досрочного прекращения сравнения условий используем оператор break, а в конце используем оператор default.

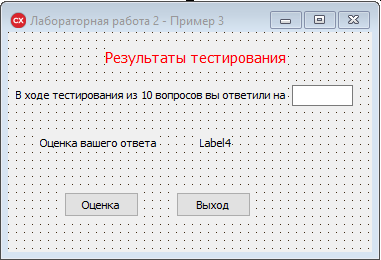


Рисунок 2.9 – Интерфейс программы «Результаты тестирования»

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

const unsigned int A=8, B=6, C=5, D=4;

unsigned int grade;

grade =StrToInt( Edit1->Text);

switch (grade)

{

case 10:

Label4->Caption= "Поздравляем, вы получили 5";

break;

case 9:

Label4->Caption= "Поздравляем, вы получили 5";

break;

case A:

Label4->Caption= "Поздравляем, вы получили 5";

break;

case B:

Label4->Caption ="Хорошо, у вас 4";

break;

case C:

Label4->Caption ="У вас всего лишь 3";

break;

case 'D':

Label4->Caption= "Плохо, у вас 2";

break;

default:

Label4->Caption= "Ужасно! Учите лучше!";

}

}

Протестируйте программу.

## Задания для самостоятельной работы

Разработать алгоритм, интерфейс пользователя и написать программу решения указанного варианта задания. При разработке интерфейса предусмотреть размещение на форме информации для пользователя, ввод исходных данных и вывод результатов вычисления в текстовые окна и метки.

1. Можноли построить треугольник из трех отрезков длиной *a* и *b*, *c*, вывести результат.
2. Вычислить площадь одной из геометрических фигур по выбору пользователя: – треугольника – по заданным основанию и высоте; – прямоугольника – по заданным длине и ширине; – трапеции – по заданным основаниям и высоте; – круга – по заданному радиусу.
3. На каждый день недели запланированы определенные работы. По введенному признаку дня недели вывести запланированное на этот день задание.
4. Известно расписание занятий на день. По введенному времени занятий вывести наименование занятий и номер аудитории.
5. Данаокружностьрадиуса *R* с центром в точке с координатами *x*0, *y*0. Определить, лежит ли точка с координатами *x*1, *y*1 внутри окружности.
6. Генератор случайных чисел выдает целое число в диапазоне от 1 до 100. Определить, является это число четным или нечетным.
7. Генератор случайных чисел выдает целое число в диапазоне от 1 до 100. Определить,являетсяэто число четным, или делится на 3, или не делится ни на 2, ни на 3.
8. Рассчитать значение переменной s. Предусмотреть выбор вида функции f(x): *sh(x)* , или .
9. Рассчитать значение переменной b. Предусмотреть выбор вида функции f(x): *sh(x)* , или .
10. Рассчитать значение переменной c. Предусмотреть выбор вида функции f(x): *sh(x)* , или .
11. Рассчитать значение переменной d. Предусмотреть выбор вида функции f(x): *sh(x)* , или .
12. Рассчитать значение переменной e. Предусмотреть выбор вида функции f(x): *sh(x)* , или .
13. Рассчитать значение переменной b. Предусмотреть выбор вида функции f(x): *sh(x)* , или .
14. Рассчитать значение переменной b. Предусмотреть выбор вида функции f(x): *sh(x)* , или .
15. Рассчитать значение переменной s. Предусмотреть выбор вида функции f(x): *sh(x)* , или .

## Контрольные вопросы

1. Какие функции выполняет условный оператор?

2. Что такое полный и неполный условный оператор? В чем разница между ними? Нарисуйте их схемы.

3. Что такое ветвь?

4. Почему в программе необходимо предусматривать вычисления по всем возможным ветвям?

5. Какие вычислительные процессы относятся к ветвящимся?

6. Что такое вложенный условный оператор? Как определяется соответствие между операторами if и else во вложенных условных операторах?

7. Что такое условная операция и чем она отличается от условного оператора?

8. Какие операции позволяют формировать условие в условном операторе и условной операции?

9. Что такое составной оператор и когда его следует применять внутри условного оператора?

10. Из каких компонент состоит оператор множественного ветвления?

11. Дайте характеристику служебного слова default. Как работает оператор switch при отсутствии оператора default.

12. Как выполняется отладка программ, содержащих ветвления вычислительного процесса?

13. Для чего нужны операторы передачи управления?

14. Приведите структуру оператора перехода.

15. Что представляет собой метка, для чего она используется?

16. Можно ли передать управление одной из альтернатив при помощи оператора goto?

17. Для чего выполняются условно-линейные схемы алгоритмов?

18. Почему разные условно-линейные схемы требуют различных условных операторов?

19. Какие логические операции используются в С++ Builder?

20. Какие объекты в языке С++ Builder могут использоваться для организации выбора пользователем альтернативного варианта?

21. Как записывается код обращения к объектам выбора альтернативных вариантов?

22. Как можно вывести в интерфейс информацию об осуществленном выборе?

23. Дайте характеристику свойств и методов объекта RadioButton.

24. Дайте характеристику свойств и методов объекта RadioGroup.

25. Дайте характеристику свойств и методов объекта GroupBox.

# ЛАБОРАТОРНАЯ РАБОТА № 3

# Тема: «Разработка программ с использованием циклических структур».

**Цель работы:** Изучить технологию решения задач с использованием циклических алгоритмов.

**Теоретические** **сведения**

**Циклическим** называется процесс многократного повторения некоторого участка вычислений при изменении хотя бы одной из входящих в него величин.

**Цикл** – повторяющийся участок вычисления.

**Параметр цикла** – входная величина, изменяющая своё значение от цикла к циклу.

**Тело** **цикла** – совокупность действий, осуществляемых в цикле.

**Закон** **изменения** **параметра** **цикла** – зависимость, связывающая текущее и предыдущее значения параметра цикла.

**Условие** **повторения цикла** – зависимость, предписывающая повторение цикла либо выход из него.

Все циклические структуры по способу определения количества повторений (N) разделяются на арифметические и итерационные.

**Арифметической** называется циклическая структура, число повторений в которой может быть определено заранее, т. е. не зависит от результатов счёта в теле цикла. Такая циклическая структура называется циклом со счетчиком или цикл «for».

**Итерационной** именуется циклическая структура, число повторений в которой зависит от результатов вычислений в теле цикла и не может быть определено заранее. К таким циклическим структурам относятся:

– цикл с предусловием, часто называемым циклом «ПОКА» или циклом «while»;

– цикл с постусловием, часто называемым циклом «ДО» или циклом «do».

Независимо от того, к какому классу относится вычислительный процесс, каждый из них содержит обязательные элементы: вход в цикл (формирование начального значения параметра цикла); вычисления в теле цикла; выход из цикла.

1. Цикл for. Обеспечивает циклическое повторение некоторого оператора заданное число раз. Повторение цикла определяется некоторой управляющей переменной (счетчиком), которая изменяется при каждом выполнения тела цикла. Повторение завершается, когда управляющая переменная достигает заданного значения. Синтаксис структуры for:

for ( выражение1; выражение2; выражение3)

{

оператор;

}

где выражение1 задает начальное значение переменной, управляющей циклом, выражение2 является условием продолжения цикла, а выражение3 изменяет управляющую переменную.

Структура for работает следующим образом. Сначала выполняется вы-ражение1 (оно может состоять из ряда выражений, разделенных запятой т. е. может использоваться операция последования, которая определяет значения любых переменных перед началом цикла). Затем проверяется выражение2 – условие завершения цикла. Если условие истинно (возвращает true – ненулевое значение), то выполняется тело цикла, (оператор, записанный в теле цикла). После завершения тела цикла выполняется выражение3, определяющее изменение переменной цикла. Затем опять проверяется условие, записанное как выражение2, и при истинности этого условия выполнение тела цикла продолжается.

Как только в каком-нибудь цикле выражение2 вернет false (нулевое значение), цикл прерывается и управление передается оператору, расположенному следом за структурой for.

**Упражнение 3.1.** Определить максимальное значение и сумму 100 случайных чисел в диапазоне от 1 до100.

**Внешняя спецификация программы.**

Исходные данные: 100 целых чисел, выбранных случайным образом в диапазоне от 1 до 100: Выходные данные: Максимальное значение целого числа. Сумма всех чисел.

**Выбор метода решения задачи.**

Выбор случайных чисел осуществляется с помощью генератора случайных чисел. Выбранные числа выводятся и сохраняются. Далее осуществляется последовательное сравнение чисел, поиск максимального числа и суммирование всех чисел.

Алгоритм решения задачи приведен на рисунке 3.1.



Рисунок 3.1 – Схема алгоритма использования цикла for

**Разработка интерфейса пользователя.**

Для ввода количества чисел необходимо использовать текстовое окно Edit. Для вывода результатов – текстовые окна Edit, для вывода дополнительной информации – метки Label.

Для генерации случайного числа используйте стандартную функцию random(), для вывода сгенерированных чисел – объект ListBox, его свойство Items, метод Add(k) для добавления элементов в список. Синтаксис выражения:

ListBox1->Items->Add(k).

Для считывания из ListBox элемента используйте свойство Items и метод Strings[i] – считать строку. Синтаксис выражения:

ListBox1-> Items->Strings[i].

Для управления программой создайте три командные кнопки: Генерировать числа, Вычислить и EXIT.

Разработанный интерфейс представлен на рисунке 3.2.

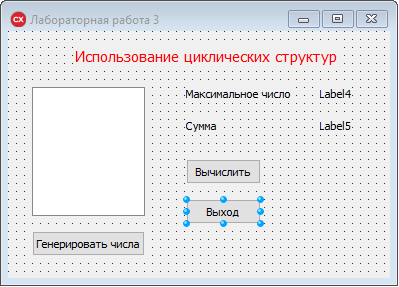


Рисунок 3.2 – Интерфейс программы определения суммы n чисел и нахождения максимального числа

**Разработка кода программы.**

Программа содержит три функции обработки событий Click трех командных кнопок: Генерировать числа, Вычислить и Exit. Программа содержит два цикла. В первом цикле генерируется n чисел и они выводятся в ListBox1. Во втором цикле осуществляется поиск максимального числа и суммирование чисел.

Листинг программы генерации n чисел, нахождения их суммы и максимального числа:

void \_\_fastcallTForm1::Button3Click(TObject \*Sender)

{

Form1->Close();

}

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

// генерация чисел

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

randomize();

int i,k,j,m,n;

k=0;

i=0;

j=0;

m=100;

n=100;

for (i=0;i<m;i++)

{

k= random(n); j=j+1;

ListBox1->Items->Add(k);

}

}

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

//поиск максимального числа и суммирования чисел

void \_\_fastcallTForm1::Button2Click(TObject \*Sender)

{

int i, k, max, sum, m;

max=0;

sum=0;

m=100;

for (i=0;i<m;i++)

{

k=0;

k = StrToInt(ListBox1-> Items->Strings[i]);

if (k > max) max = k;

sum = sum+k;

}

Label4->Caption=max;

Label5->Caption=sum;

}

**Цикл с предусловием:**

Оператор while используется для организации многократного выполнения тела цикла, пока выполняется некоторое условие. Синтаксис структуры оператора while:

**while** (условие) тело цикла.

Здесь условие представляет собой любое выражение, принимающее значение 0 (ложь) или не 0 (истина), тело цикла – простой (или составной) оператор либо блок.

Вначале вычисляется условие. Если оно оказывается истинным, то выполняется тело цикла; в противном случае управление передается оператору, следующему за телом цикла. Поскольку проверка условия осуществляется перед выполнением оператора тела цикла, то, если условие сразу ложно, оператор не будет выполнен ни одного раза.

Иногда может потребоваться досрочный выход из цикла, например при выполнении некоторого условия в теле цикла. Для этого может быть использован оператор break, прерывающий цикл. Блок-схема цикла с предусловием представлена на рисунке 3.3, б.

**Цикл с постусловием.**

Структура do…while используется для организации многократного выполнения оператора или совокупности операторов (блока), составляющих тело цикла, до тех пор пока не окажется нарушенным некоторое условие. Синтаксис цикла do….while:

**do** тело цикла **while** (условие)

Структура работает следующим образом. Вначале выполняется тело цикла, затем вычисляется заданное условие, которое должно возвращать результат, если условие выполняется (true), то повторяется выполнение тела цикла и после этого снова проверяется условие. Если проверяемое условие вернет false, то выполнение цикла прерывается и управление передается следующему за структурой do…while оператору. Поскольку проверка условия осуществляется после выполнения тела цикла, то цикл будет заведомо выполнен хотя бы один раз, даже если условие сразу не выполняется.

Для досрочного выхода из цикла может использоваться оператор break. Блок-схема цикла с постусловием представлена на рисунке 3.3, в.



Рисунок 3.3 – Схемы выполнения операторов циклов а – оператор for; б – оператор while; в – оператор do….while

**Упражнение 3.2.** Решение задачи с использованием циклов while и do….while для вычисления суммы членов числового ряда.

Необходимо разработать алгоритм, интерфейс и программу, позволяющую найти сумму членов бесконечного ряда с точностью до члена ряда, меньшего ε, и число слагаемых в сумме:

**Выбор метода решения задачи.**

При нахождении суммы такого ряда не следует вычислять отдельно каждый элемент. Достаточно установить зависимость, по которой из предыдущего элемента ряда образуется следующий. Обозначим через an n-й член ряда, тогда an+1 = an\*x / (n + 1), a0 = 1. Такой подход позволяет сократить количество вычислительных операций и уменьшить время счета. Сумма будет накапливаться в переменной S, которую предварительно необходимо обнулить. Условие окончания цикла an≤ε. Число повторных вычислений в теле цикла равно числу слагаемых в сумме. На рисунке 3.4 представлены алгоритмы решения поставленной задачи с использованием циклов while и do…while.



Рисунок 3.4 – Алгоритм нахождения суммы членов числового ряда

**Разработка интерфейса пользователя.**

Для ввода исходных данных и вывода результатов расчета используйте текстовые окна Edit. Для вывода дополнительной информации используйте метки Label. Для управления программой разместите три командные кнопки: Вычислить while, Вычислить do…while и EXIT.

Интерфейс формы представлен на рисунке 3.5.

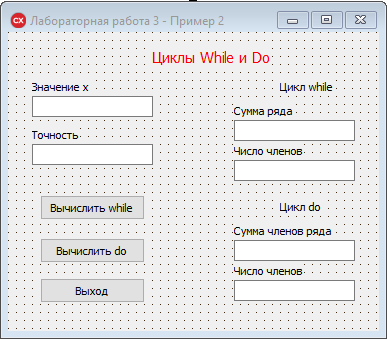


Рисунок 3.5 – Интерфейс программы для вычисления суммы членов числового ряда с заданной точностью

**Разработка кода программы.**

В данной программе операторы циклов вычисления суммы членов ряда размещены в двух функциях обработки событий двух кнопок. Для того чтобы переменные были доступны в этих двух функциях, они должны быть объявлены вне этих функций, как показано в листинге программы:

void \_\_fastcallTForm1::Button3Click(TObject \*Sender)

{

Form1->Close();

}

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

//объявление переменных, видимость которых в пределах

//модуля, производится перед функцией обработки события.

int i;

float a1,a2,x,s,d, d1 ;

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

x = StrToFloat(Edit1->Text);

d = StrToFloat(Edit2->Text);

s=1;

i=1;

d1=1;

a1 = 1;

a2 = 0;

while(d1 >= d) //цикл while

{

i = i+1;

a2=a1\*x/i;

s = s+a2;

d1 = a2;

}

Edit3->Text = s;

Edit4->Text = i;

}

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

void \_\_fastcallTForm1::Button2Click(TObject \*Sender)

{

Do //цикл do

{

i = i+1;

a2=a1\*x/i;

s = s+a2;

d1 = a2;

}

while(d1 >= d);

Edit5->Text = s;

Edit6->Text = i;

}

Проверьте работоспособность программы. Из анализа результатов расчета (рисунок 3.6) видно, что при решении задачи с использованием цикла while количество слагаемых на единицу больше, чем при использовании цикла do…while.

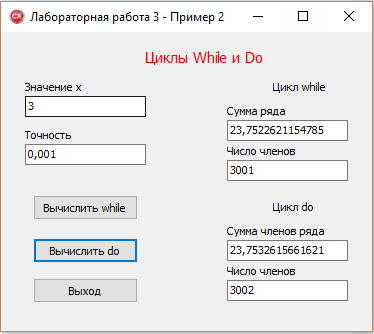


Рисунок 3.6 – Результат выполнения программы

## Задания для самостоятельной работы

Разработать алгоритм, интерфейс пользователя и написать код программы, позволяющей вычислить и вывести на печать таблицу значений функции Y(x) и её разложения в ряд *S(x)* для *x*, изменяющегося от *a* до *b* с шагом *h=(b-a)/10* (таблица 3.1).

Таблица 3.1 – Варианты заданий

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***№*** | ***a*** | ***b*** | ***S(x)*** | ***n*** | ***Y(x)*** |
| **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | 0.1 | 1 |  | 160 | sin x |
| 2 | 0.1 | 1 |  | 100 |  |
| 3 | 0.1 | 1 |  | 120 |  |
| 4 | 0.1 | 1 |  | 80 | cos x |
| 5 | 0.1 | 1 |  | 140 |  |
| 6 | 0.1 | 1 |  | 80 |  |

Окончание табл. 3.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 0.1 | 1 |  | 120 |  |
| 8 | 0.1 | 1 |  | 100 |  |
| 9 | 0.1 | 1 |  | 140 |  |
| 10 | 0.1 | 0.5 |  | 150 | arctg x |
| 11 | 0.1 | 1 |  | 100 |  |
| 12 | 0.1 | 1 |  | 80 |  |
| 13 | –2 | –0.1 |  | 160 |  |
| 14 | 0.1 | 0.8 |  | 180 |  |

## 

## Контрольные вопросы

1. Какой вычислительный процесс называется циклическим?

2. Что такое цикл, тело цикла, параметр цикла?

3. Каковы обязательные элементы цикла?

4. На какие виды подразделяются арифметические циклы?

5. На какие разновидности делятся циклические процессы?

6. Как графически обозначаются структуры циклов?

7. Какова структура оператора цикла с предусловием, как он выполняется?

8. В чем заключаются правила записи и выполнения оператора while?

9. Какова структура оператора цикла с постусловием, как он выполняется?

10. В чем заключаются правила записи и выполнения оператора do ... while?

11. В чем заключаются правила записи и выполнения оператора for?

12. Как принудительно выйти из любого цикла?

13. Чем отличаются циклы с предусловием от циклов с постусловием?

14. Опишите последовательность функционирования циклов while и do while. Какой из этих циклов является циклом с предусловием, а какой – с постусловием?

15. Опишите схему работы оператора цикла for. Является ли цикл for циклом с предусловием или циклом с постусловием?

16. Какие циклы называются вложенными?

# ЛАБОРАТОРНАЯ РАБОТА № 4

# Тема: «Изучение синтаксиса различных вариантов создания статических и динамических массивов на языке С++ Builder».

**Цель** **работы**. Освоить разработку алгоритмов и программ решения задач с использованием массивов в среде С++ Builder.

**Теоретические** **сведения**

Массив представляет собой структуру данных, позволяющую хранить совокупность значений любого типа. Массив характеризуется своим именем, типом хранимых элементов, размером (числом хранимых элементов), нумерацией элементов и размерностью. Массивы могут быть статическими и динамическими, одномерными и многомерными.

Статическим массивом называется такой массив, число элементов в котором постоянно и заранее определено и задается программистом при разработке программы.

Динамическим массивом называется массив, в котором число элементов заранее неизвестно, оно может быть определено в ходе вычисления в программе или задано пользователем. В динамическом массиве число элементов и размерность могут быть изменены при работе программы.

Объявление статического одномерного массива имеет вид:

<тип\_элементов> <имя\_ массива> [константное выражение - число элементов]

Элементы массива могут иметь любой тип. Константное выражение числа элементов должно иметь тип целого без знака.

Например, оператор:

int A[10];

объявляет массив А, содержащий 10 целых чисел.

char S[10];

объявляет массив S, содержащий 10 символов.

Можно объявить и многомерный массив, элементами которых являются массивы. Например, двумерный массив можно объявить таким образом:

Int B [10][5];

Этот оператор описывает двумерный массив, который можно представить как таблицу, состоящую из 10 строк и 5 столбцов.

Доступ к элементам массива осуществляется выражением: A[i], B[i][j],

где i и j индексы, являющиеся для данных примеров целыми числами 0 <= i <= 9 и 0 <= j <= 4.

Следует обратить внимание на то, что индекс массива начинается с 0, и индекс последнего элемента на единицу меньше размера массива.

Перед использованием массива его необходимо инициализировать, т. е. присвоить элементам массива начальные значения. Начальные значения элементам массива можно задавать при объявлении массива

Например:

int A[10] ={2,4,3,6,3,6,1,23,5,25};// массив целых чисел;

char S[10] ={“abcdefghi\0”};// массив символов

По принятому соглашению массивы символов, содержащие строку, после конца строки обязательно должны содержать нулевой символ. По этой причине в массиве символов S[10] присутствуют только девять значащих символов, и последний элемент является нулевым символом.

Если начальных значений меньше, чем элементов массива, то оставшиеся элементы автоматически инициализируются. Это позволяет инициализировать символьный массив в более удобной форме:

char kaf\_name [ ] = ”кафедра ВТиИС”;

При объявлении со списком инициализации размер массива можно не указывать, тогда количество элементов массива будет равно количеству элементов в списке начальных значений. Например, объявление Int A [] = {1,2,3,4,5}; создает массив из пяти элементов.

Если многомерный массив инициализируется при объявлении, список значений по каждой размерности заключается в фигурные скобки. Например, инициализация при объявлении трехмерного массива записывается в виде:

Int A3[4] [3] [2] = { {{0,1}, {2,3}, {4,5}}, {{6,7}, {8,9}, {3,8}},

{{1.3}, {5,2}, {8,5}}, {{7,3,1}, {4,5}, {2,9}} };

Эта запись создает массив А3, четыре строки которого являются матрицами размерностью (3 х 2).

**Упражнение 4.1.** Необходимо разработать алгоритм, интерфейс пользователя и программу, позволяющую создать одномерный массив из 100 целых случайных чисел в диапазоне от 0 до 100, распределенных по равномерному закону, и найти среднее значение, максимальный элемент, сумму.

**Внешняя спецификация задачи:**

1. Размерность массива константа – целое число, равное 100. 2. Значения чисел – целые в диапазоне от 0 до 100.

3. Числа должны генерироваться датчиком случайных чисел по равномерному закону.

4. Максимальное число–это целое число в диапазоне от 0 до 100.

5. Сумма чисел массива – целое число, максимальное значение которого не превысит 10 000.

6. Среднее значение чисел – действительное число.

**Алгоритм решения задачи.**

Вначале объявите массив с заданным числом элементов. Для генерации элементов массива используйте датчик случайных, для инициализации массива – цикл со счетчиком. Для вывода созданного массива используйте объект ListBox. Для поиска максимального элемента, вычисления суммы элементов и среднего значения элемента массива – цикл со счетчиком, в теле которого разместите выражение для поиска максимального элемента, выражение для суммирования членов массива, после цикла – выражение для вычисления среднего значения элемента массива. Алгоритм решения задачи представлен на рисунке 4.1.

**Разработка интерфейса пользователя.**

Интерфейс пользователя для программы решения задачи приведен на рисунке 4.2. При проектировании интерфейса кроме ранее использованных объектов используется объект ListBox (список), в который можно выводить переменные строкового типа AnsiString. Поэтому значения элементов массива перед их добавлением в ListBox преобразуются в строковый тип. Добавление элементов в ListBox осуществляется через свойство Items методом Add(Ctr) с использованием конструкции:

AnsiString Ctr;//объявление переменной строкового типа

Ctr = IntToStr(N);// присваивание строковой переменной числа

ListBox1->Items->Add(Ctr);//добавление строки в ListBox



Рисунок 4.1 – Алгоритм нахождения среднего значения, максимального элемента, суммы

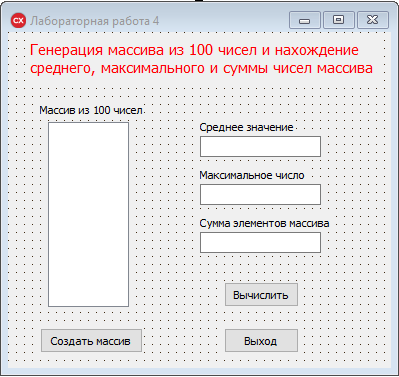


Рисунок 4.2 – Интерфейс программы нахождения параметров статического массива

**Написание кода программы.**

При написании программы следует учитывать, что создание массива и вычисление его параметров управляется двумя командными кнопками, и в функциях обработки событий для этих кнопок используются одни и те же переменные. Переменные объявляются вне функций обработки событий и поэтому имеют локальную в пределах формы область видимости. Для того чтобы датчик случайных чисел к при каждом нажатии на кнопку «Создать» генерировал новую последовательность чисел, необходимо записать в обработчик события void \_\_fastcall TForm1::FormCreate(TObject \*Sender) функцию randomize.

Листинг программы создания статического массива и нахождения значения его максимального элемента, среднего значения элемента и суммы элементов массива.

**//---------------------------------------------------------------------------**

**//**Переменные объявляются вне функции обработчика события и они будут видны во всех процедурах формы

**//--------------------------------------------------------------------------**

int N, i, s1, b;

int Array[100];

float cr;

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

AnsiString Ctr; N =0;

i=0;

for(i =0; i<99;i++)

{

Array[i] =random(100); N = Array[i];

Ctr = IntToStr(N);

ListBox1->Items->Add(Ctr);

}

}

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

void \_\_fastcallTForm1::Button3Click(TObject \*Sender)

{

Form1->Close();

}

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

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

{

cr=0;

b=1;

s1=0;

for( i=0; i<99; i++)

{

s1 = s1+Array[i];

if (Array[i] > b) b = Array[i];

}

cr=(float)s1/100;

LabeledEdit1->Text= FloatToStr(cr);

LabeledEdit2->Text= IntToStr(b);

LabeledEdit3->Text= IntToStr(s1);

}

**Создание многомерного массива, операции с массивами.**

Массив в отличие от других видов данных всегда передается в функцию по ссылке, а не по значению.

Так как массив занимает смежные ячейки памяти, то при использовании имени массива в качестве аргумента язык С++ обеспечивает передачу функции адреса первого элемента этого массива.

Имя массива является константным указателем на первый элемент массива, поэтому оно не может модифицироваться, и к нему не применимы все операции присваивания.

К имени массива можно применить операцию sizeof, которая возвращает значение, равное числу элементов (объему) массива, например: sizeof (A) или sizeof (A[0]).

При передачи массива в функцию в качестве параметра заголовок функции содержит тип и имя массива с последующими пустыми квадратными скобками. например, если функция F должна принимать массив Bs как параметр, ее прототип может иметь вид:

void F(int Bs[ ]);

При вызове функции указывается просто имя массива. Например, обращение к такой функции может быть записано в виде:

const Amax =10;

int Bs[Amax];

F(Bs);

В большинстве случаев только имени массива мало, чтобы провести в функции обработку его элементов, Внутри функции требуется знать размер массива, чтобы можно было организовать его циклическую обработку, поэтому в функцию передается не только массив, но и его размер. При этом заголовок функции имеет вид:

void F (int Bs[ ], int N);

а вызов функции:

F (Bs, Amax);

При использовании некоторых библиотечных функций необходимо в качестве второго параметра использовать не размер массива, а значение его последнего символа, которое на единицу меньше размеров. В этом случае вызов функции может иметь вид:

F (Bs, Amax-1);

Передача массива по ссылке не гарантирует защиты от несанкционированного изменения программой значений элементов массива. Если необходимо защитить массив от подобных изменений, его надо передать в функцию как константный:

void F(const int Bs[ ], int N);

При вызове функции необязательно передавать весь массив. Например, можно передать в функцию параметр размера массива, меньший истинного. Для этого в качестве начала массива передать в функцию указатель на какой-то элемент массива. Например, если задан массив int Bs[10] и при обращении к функции Вы передадите массив как Sum(Bs-2, Bmax-2), то функция будет обращаться к элементам массива с индексами от 2 до 7. То есть сумма будет посчитана по элементам, начиная с третьего.

Если в функцию передается многомерный массив, то в заголовке функции квадратные скобки первой размерности остаются пустыми, а в скобках следующих размерностей должны указываться константами их размеры. Например, если функция F2 должна принимать двумерный массив размером 3х3, то ее заголовок может иметь вид:

void F (const int Bs[ ] [3]);

Вызов этой функции производится передачей в неё имени массива. Например, F(Bs).

**Упражнение 4.2.** Разработать интерфейс и программу, позволяющую сгенерировать матрицу размером 5х5, элементами которой являются целые числа, распределенные по равномерному закону. Вычислить сумму элементов, находящихся ниже главной диагонали, и сумму всех элементов матрицы.

**Внешняя спецификация задачи.**

1. Матрица размерностью 5х5 элементов, каждый из которых является целым числом в диапазоне от 0 до 10. Так как матрица представляет статический двумерный массив, то ее размерность и инициализация должны быть описаны в программе.

1. Для генерации значений элементов матрицы необходимо использовать датчик случайных чисел.

2. Результаты вычисления суммы элементов матрицы, лежащих ниже главной диагонали и суммы всех элементов матрицы, являются целыми числами.

**Разработка алгоритма решения задачи.**

Алгоритм решения задачи включает объявление двухмерной матрицы с заданным числом элементов. Инициализации матрицы случайными числами, генерируемыми датчиком случайных чисел, организации вывода матрицы в объект StrinGrid. Для определения сумм элементов матрицы, лежащих ниже главной диагонали, и суммы всех элементов организуются вложенные циклы.

Внешние циклы осуществляют перебор элементов по строкам, внутренние циклы – перебор по столбцам массива.

Алгоритм решения задачи приведен на рисунке 4.3.



Рисунок 4.3 – Алгоритм вычисления суммы элементов двухмерного массива

**Разработка пользовательского интерфейса программы.**

Информация для пользователя, выводимая на форму, должна содержать размерность матрицы, т. е. число строки столбцов, числовые значения элементов матрицы, результаты вычислений сумм.

Для вывода информации о размерности матрицы будем использовать объекты Label .

Для вывода значений элементов матрицы – объект StringGrid, основными используемыми свойствами которого являются:

RowCount – количество строк;

ColCount – количество столбцов;

FixedRows – количество фиксированных строк;

FixedCols – количество фиксированных столбцов.

Метод Cells[i][j] = Array[i][j] – заполнение ячеек таблицы элементами массива Array программным способом. Для того чтобы можно было заполнять таблицу StringGrid с клавиатуры, необходимо установить в свойстве Options значение goEditing, равным True.

Для вывода результатов вычисления суммы элементов, находящихся ниже главной диагонали матрицы используем объект Label, а для вывода суммы всех элементов диалоговый модуль сообщений, который вызывается оператором, имеющим синтаксис:

ShowMessage ("Сумма элементов массива = ” IntToStr(Sum2(Array)))

Для управления программой используйте четыре командных кнопки, назначение которых указывается в свойстве Caption объектов.

Внешний вид интерфейса пользователя приведен на рисунке 4.4.

**Разработка кода программы.**

Процесс разработки кода программы аналогичен рассмотренному в предыдущем примере. Особенностью является объявление двумерного массива, которое производится в два приема. Вначале объявляется число элементов в каждой размерности массива константами Amax и Bmax, которые должны учитывать, что число строк и столбцов в объекте StrinGrid на единицу больше, чем число значащих элементов в массиве.

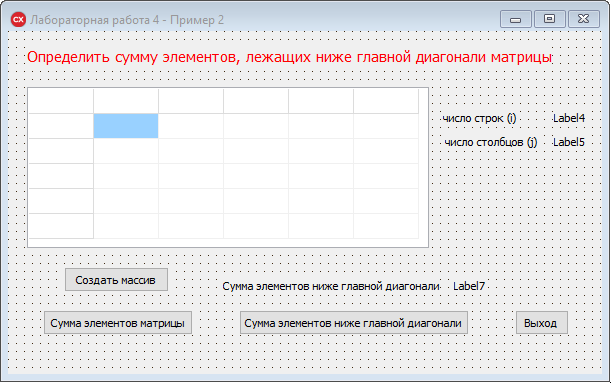


Рисунок 4.4 – Интерфейс программы вычисления элементов двухмерного массива

Листинг программы вычисления суммы элементов двухмерного массива, лежащих ниже главной диагонали и суммы всех элементов:

int i, j;

const Amax=6;

const Bmax=6;

int Array[Amax][Bmax];

float cr;

float S;

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

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

randomize();

Label4->Caption = IntToStr(Amax-1);

Label5 ->Caption = IntToStr(Bmax-1);

AnsiString Ctr;

i=0;

for (i=1;i<6;i++)

{

StringGrid1->Cells[i][0]= "столбец"+IntToStr(i);

}

for (i=1;i<6;i++)

{

StringGrid1->Cells[0][i]= "строка"+IntToStr(i);

}

for(i=1; i<6;i++)

{

for(j=1; j<6;j++)

{

Array[i][j] =random(10);

StringGrid1->Cells[i][j]= IntToStr(Array[i][j]);

}

}

}

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

void \_\_fastcallTForm1::Button4Click(TObject \*Sender)

{

Form1->Close();

}

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

void \_\_fastcallTForm1::Button3Click(TObject \*Sender)

{

S=0;

for(int i=3; i<6;i++)

{

for(int j=1; j<i;j++)

{

S = S + Array[i][j];

}

}

Label7->Caption = FloatToStr(S);

}

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

int Sum2( int A1[][6])

{

int n1 =6; int S2 =0;

for (int i=1;i<n1;i++)

{

for (int j=1;j<n1;j++)

{

S2+=A1[i][j];

}

} return S2;

}

void \_\_fastcallTForm1::Button2Click(TObject \*Sender)

{

ShowMessage("Сумма элементов= "+IntToStr(Sum2(Array)));

}

**Динамические массивы.**

Динамическое распределение памяти во время выполнения является чрезвычайно полезной возможностью.

Многие программы интенсивно используют массивы для хранения множества значений определенного типа. Вместо редактирования и перекомпилирования программ, которые просто запрашивают память с запасом, вам следует создавать свои программы таким образом, чтобы они распределяли требуемую им память динамически во время выполнения. В этом случае ваши программы могут адаптировать использование памяти в соответствии с вашими изменившимися потребностями, избавляя вас от необходимости редактировать и перекомпилировать программу. Распределяя память динамически, программы непрерывно изменяют свои потребности без дополнительного программирования. Если ваши программы запрашивают память во время выполнения, они указывают требуемое количество памяти, а C++ возвращает указатель на эту память.

Язык C++ Builder позволяет распределить память из специальных областей памяти – heap, которые называются свободной памятью.

Динамическое распределение памяти в этой области может производиться несколькими способами: с помощью операций new и delete или с помощью библиотечных функций malloc, calloc, realloc, free. Указанные функции объявлены в файле stdlib.h или alloc.h.

Распределение памяти с помощью операций new и delete.

Следующий фрагмент использует оператор new для получения указателя на N-байтный массив, число элементов которого N вводится пользователем:

int N

N = StrToInt (Edit1->Text) int \*Barray = new int[N];

После выделения памяти под динамический массив необходимо проверить, успешно ли завершилась эта операция. Для этого можно использовать следующую конструкцию.

if (Barray != NULL) ShowMessage ("Память успешно выделена") else

{

ShowMessage ( " Ошибка выделения памяти "); exit (1);

}

Программа сразу проверяет значение, присвоенное оператором new переменной-указателю. Если указатель содержит значение NULL, значит new не смог выделить запрашиваемый объем памяти. Если же указатель содержит не NULL, следовательно, new успешно выделил память и указатель содержит адрес начала блока памяти.

Если new не может удовлетворить запрос на память, он возвратит NULL. Для размещения в памяти двумерного массива необходимо вначале задать количество строк и столбцов. Затем создать массив указателей на массив строк и потом циклом задать число элементов в каждой строке.

Например:

int N = StrToInt( Edit1-Text); // число строк

int M = StrToInt( Edit1-Text);// число столбцов

int\*\* Array = new int \*[N]; // массив указателей на массив строк

for ( int I = 0;

i<N; i++);// массивы указателей на элементы в каждой строке

Array[i] = new int [M];

Созданный приведенным кодом массив можно использовать как обычный массив, обращаясь к элементам массива как Array [i][j].

Для освобождения памяти, занятой двумерным массивом, необходимо удалить все связанные с ним массивы.

Например:

for (int I =0; i<N; i++)

delate[] Array[i];

delate [] A;

При использовании оператора new для выделения памяти может случиться так, что ваш запрос не может быть удовлетворен, поскольку нет достаточного объема свободной памяти. Если оператор new не способен выделить требуемую память, он присваивает указателю значение NULL. Проверяя значение указателя, как показано в предыдущей программе, вы можете определить, был ли удовлетворен запрос на память.

Динамически распределенную память надо освобождать, когда отпадает необходимость в размещенных в ней объектах. В противном случае получится неоправданная утечка памяти.

Удаление памяти осуществляется операцией delete. Например:

delete Barray;

Barray = NULL;

Для того, чтобы изменить число элементов в массиве созданного оператором new, необходимо создать вспомогательный массив с числом элементов, равным исходному. Затем переместить элементы из исходного массива во вспомогательный. Увеличить размер исходного массива и переместить в него элементы из вспомогательного массива и удалить вспомогательный массив.

Распределение памяти с помощью функций malocc, calloc, relloc. Для того, чтобы указанные функции были объявлены в файле stdlib.h alloc.h, которые записываются в заголовочный файл формы, функция malloc выделяет блок памяти размером size байтов. В случае успешного выделения памяти возвращает указатель на выделенный блок, в противном случае возвращается NULL. Объявление функции следующее:

void \*malloc (size\_t size);

Функция calloc выделяет память под несколько объектов, размер каждого из которых равен size. Функция объявляется следующим образом:

void \*calloc (size\_t n\_object, size\_t size);

Общий объем выделенной памяти для n объектов составляет n\_object\* size. В случае успешного выделения памяти возвращает указатель на выделенный блок, в противном случае возвращается NULL.

Функция realloc позволяет изменить размер ранее выделенного блока памяти. Функция объявляется следующим образом:

void \*realloc (void \*name\_block, size\_t size);

Данная функция изменяет размер блока, на который указывает name\_block, до размеров size. При этом предполагается, что block указывает блок памяти, выделенный ранее функциями malloc, calloc или realloc**.** Если блок нового размера не может быть выделен, то функция realloc возвращает NULL. Если же память выделилась успешно, то возвращается адрес выделенного блока. При этом он может отличаться от начального значения name\_block, поскольку функция при необходимости осуществляет копирование содержимого блока в новое место.

Если размер size задан равным нулю, то выделенный ранее блок, на который указывает name\_block, освобождается, а функция возвращает NULL.

Функция free объявляется следующим образом: void free(void \*block);

Она освобождает блок памяти, на который указывает block, выделенный ранее функциями malloc, calloc или realloc.

Примеры использования описанных функций:

1) int \*Carray;

int N;

N = StrToInt(Edit1->Text);

Carray = (int\*) malloc(N);

………………….

Free(Carray);

В данном примере функция malloc выделяет память под массив из N целых чисел, где значение N вводится пользователем. После выполнения с массивом каких-либо операций память освобождается функцией free.

2) В этом примере можно было использовать для выделения памяти функцию calloc;

Carray = (int\*) calloc (N) , sizeof(int));

3) Для изменения выделенной памяти применим функцию realloc. Carray = (int\*) realloc ( Carray, N1);

Или более простой оператор: realloc(Carray, N1);

**Упражнение** **4.3.** Разработать алгоритм, интерфейс пользователя и программу, позволяющую создать одномерный массив целых чисел, распределенных по случайному равномерному закону. Число элементов массива и диапазон значений чисел должны задаваться пользователем. Программа должна осуществлять сортировку массива чисел по убыванию их значений, выводить на форму исходный и отсортированный массивы и продолжительность сортировки.

**Внешняя спецификация задачи.**

1. В исходном массиве число элементов определяется пользователем.

2. Элементами массива являются целые положительные числа.

3. Максимальное значение чисел определяется пользователем.

4. Значения чисел, заполняющих массив, распределены по равномерному случайному закону.

5. На форму должны выводиться исходный и отсортированный массивы.

6. Продолжительность процесса сортировки должна измеряться с точностью до миллисекунды.

**Выбор метода решения задачи.**

Для создания динамического массива необходимо использовать оператор new. Для генерации случайных чисел – датчик целых чисел, распределенных по равномерному закону. Сортировку чисел массива необходимо производить методом прямого перебора, сущность которого состоит в последовательном сравнении чисел, начиная с первого, со всеми последующими и определении его места размещения в соответствии с выбранным условием сортировки (по возрастанию либо по уменьшению). Продолжительность процесса сортировки определяется как разность между началом сортировки и ее окончанием.

**Разработка алгоритма решения задачи.**

Алгоритм решения задачи подобен алгоритмам решения предыдущих задач. Поэтому приведем только основные особенности:

1. Объявляются переменные.

2. Осуществляется ввод исходных данных.

3. Создается динамический массив.

4. Осуществляется вывод исходного массива.

5. Засекается по системному времени начало сортировки.

6. Осуществляется сортировка массива.

7. Засекается по системному времени окончание сортировки.

8. Осуществляется вывод продолжительности сортировки.

9. Осуществляется вывод отсортированного массива.

Алгоритм решения задачи приведен на рисунке 4.5.

**Разработка интерфейса пользователя.**

Интерфейс пользователя содержит: три текстовых окна для ввода числа элементов массива, максимального значения массива и вывода продолжительности процесса сортировки, метку для информирования пользователя. Для вывода исходного и отсортированного массива используйте объект ListBox. Для управления программой используются две командные кнопки.

Внешний вид интерфейса пользователя приведен на рисунке 4.6.



Рисунок 4.5 –. Алгоритм решения задачи сортировки массива

**Разработка кода программы.**

Особенностями программы является создание динамического массива с помощью оператора new. Выделение функции прямого перебора, которая описывается вне функций обработки событий. Функция для определения системного времени в момент начала и завершения сортировки взята библиотечная, синтаксис которой имеет вид:

DecodeTime(Time(), Hour, Min, Sec, MSec),

для ее использования в заголовочный файл формы Unit.h должны быть включены файлы time.h и DateUtils.hpp.

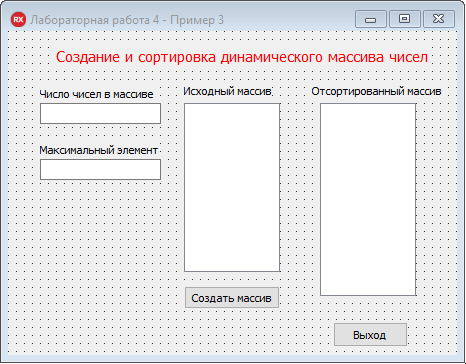


Рисунок 4.6 – Интерфейс пользователя программы создания и сортировки динамического массива

Листинг программы создания и сортировки динамического массива:

#include <time.h>

#include <DateUtils.hpp>

**-----------------------------------------------------------------------**

unsigned int n;

int N,i,u,j,m;

AnsiString Ctr;

int \*Array = new int[n];

unsigned short Hour1, Hour2, Min1, Min2, Sec1, Sec2, MSec1, MSec2;

//функция сортировки прямым перебором

void ASort(int Ar[],int k)

{

int m,i,j;

m = 0;

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

{

for(j=0; j<(n);j++)

{

if (Ar[i] > Ar[j]) {

m = Ar[i];

Ar[i]= Ar[j];

Ar[j] =m;

}

}

}

}

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

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

ListBox1->Clear();

ListBox2->Clear();

n = StrToInt(LabeledEdit1->Text);

u = StrToInt(LabeledEdit2->Text);

int \*Array = new int[n];

if (Array != NULL)

{

ShowMessage("Массив успешно создан");

}

Else

{

ShowMessage("Ошибка выделения памяти");

Abort();

}

N =0;

i=0;

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

{

Array[i] =random(V);

N = Array[i];

Ctr = IntToStr(N);

ListBox1->Items->Add(Ctr);

}

i=0;

j=0;

m = 0;

//Вызов функции сортировки

ASort(Array, n);

N=0;

// вывод отсортированного массива

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

{

N = Array[i];

Ctr = IntToStr(N);

ListBox2->Items->Add(Ctr);

}

delete Array;

}

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

void \_\_fastcallTForm1::Button2Click(TObject \*Sender)

{

delete Array;

Array = NULL;

Form1->Close();

}

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

void \_\_fastcallTForm1::FormCreate(TObject \*Sender)

{

randomize();

}

Протестируйте программу, задавая число элементов в массиве от 10 до 10000 и максимальное значение числа до 1000.

## Задания для самостоятельной работы

1. Разработать алгоритм, интерфейс пользователя и программу, позволяющую в матрице размером 10х10 действительных чисел заменить нулями все элементы, превышающие среднее значение элемента.
2. Разработать алгоритм, интерфейс пользователя и программу, позволяющую упорядочить строки матрицы размером 6х6 по возрастанию значений минимальных элементов этих строк.
3. Разработать алгоритм, интерфейс пользователя и программу, позволяющую в матрице размером 10х10 найти минимальный элемент в каждой строке.
4. Разработать алгоритм, интерфейс пользователя и программу, позволяющую в массиве B (100) найти сумму 10 наибольших чисел.
5. Разработатьалгоритм, интерфейс пользователя и программу, позволяющую определить порядковый номер максимального и минимального по абсолютной величине элемента массива типа double.
6. Разработать алгоритм, интерфейс пользователя и программу, позволяющую определить и вывести в отдельный массив элементы заданного исходного одномерного массива целого типа, а также превышающих среднее арифметическое значение его элементов.
7. Разработать алгоритм, интерфейс пользователя и программу, позволяющую вычислить в массиве целого типа квадратный корень из суммы квадратов значений элементов с нечетными номерами и среднее арифметическое значение элементов с четными номерами и наименьшее из этих двух значений.
8. Разработать алгоритм, интерфейс пользователя и программу, позволяющую создать массив, из n элементов найти максимальный и минимальный элементы и поменять их местами.
9. Разработать алгоритм, интерфейс пользователя и программу, позволяющую задать матрицу 8х8 и вывести в текстовое окно элементы главной диагонали.
10. Разработать алгоритм, интерфейс пользователя и программу, позволяющую задать матрицу N x M и заполнить ее случайными числами. Увеличить число столбцов на 3 и заполнить целыми числами.
11. Разработать алгоритм, интерфейс пользователя и программу, позволяющую создать два одномерных массива N и M элементов N > M. Заполнить первый массив целыми числами и переписать во второй массив M элементов из первого в обратном порядке.
12. Разработать алгоритм, интерфейс пользователя и программу, позволяющую создать одномерный массив N целых чисел, значения которых распределены по нормальному закону и определить математическое ожидание и дисперсию.
13. Разработать алгоритм, интерфейс пользователя и программу, позволяющую создать две матрицы N x M, сложить эти матрицы и вычислить минимальный элемент в полученной матрице.
14. Разработать алгоритм, интерфейс пользователя и программу, позволяющую вычислить значения функции y = (x2 – a) \* b2 на интервале от 2a до 2b при условии b > 2a c шагом (b – a) /10.
15. Разработать алгоритм, интерфейс пользователя и программу, позволяющую создать матрицу NxM целых чисел и вычислить сумму членов каждого ряда.

## Контрольные вопросы

1. Дайте определение массива.

2. Что понимается под размерностью массива?

3. Какие массивы называются статическими?

4. Как описать одномерные массивы заданной размерности?

5. Как инициализировать одномерный массив?

6. Что означает имя массива без имени?

7. Как объявляются многомерные массивы?

8. Как инициализировать многомерный массив?

9. Как обратиться к элементу массива?

10. Как создать массив, число элементов в котором определяется пользователем?

11. Как увеличить число элементов в одномерном массиве?

12. Как увеличить число элементов в многомерном массиве?

13. Как уменьшить число элементов в массиве?

14. Как вывести элементы массива в объект ListBox?

15. Как ввести элементы массива в объект ListBox c клавиатуры?

16. Как вывести элементы массива в объект StringGrid?

17. Как ввести элементы массива в объект StringGrid с клавиатуры?

18. Какие свойства объекта StringGrid определяют количество строк и столбцов?

19. Какие операторы и функции используются для удаления массива из памяти?

# ЛАБОРАТОРНАЯ РАБОТА № 5

# Тема: «Изучение среды C++ Builder. Применение в программах символьных и строковых переменных».

**Цель работы:** изучение методов применения символьных и строковых типов переменных при разработке программ и интерфейса пользователя.

**Теоретический** **материал**

В языке С++ Builder используется один символьный тип данных – char. Под величину символьного типа отводится один байт, что позволяет хранить в нем только один символ из 256-символьного набора ASCII.

В С++ Builder строки рассматриваются как массивы символов, оканчивающиеся нулевым символом (\0). Строка доступна через указатель на первый символ в строке.

Строка может быть объявлена либо как массив символов, либо как переменная char\*. В качестве примера приводится объявление строковых переменных с одновременной их инициализацией.

сhar Str[]= “студент”; //объявление как массив символов

сhar \* St = “студент”; // объявление через указатель

Оба примера идентичны, можно использовать любой из них.

Каждое из приведенных эквивалентных объявлений присваивает строковой переменной значение «строка». Первое объявление создает массив из 8 символов – ‘с’,’т’,’у’,’д’,’е’,’н’,’т’ и символа ‘\0’. Второе объявление создает переменную указателя St, который указывает на строку (массив) с текстом «студент», лежащую где-то в памяти. Но в любом случае число хранимых символов на единицу больше числа значащих символов за счет конечного нулевого символа. Максимальное количество символов, которое можно сохранить в указанной строковой переменной, равно 256, так как для одного символа отводится всего один байт.

Доступ к отдельным символам строки осуществляется по индексам, начинающимся с нуля, например Str[0]и St[0] – первые символы, Str[1] и St[2] – вторые и т. д. В приведенных примерах длина строк определяется автоматически компилятором. Можно объявлять строковые переменные заданной длины, указывая в квадратных скобках требуемое количество символов. Например, char Slovo [100] содержит строку из 99 символов.

В некоторых случаях, когда количество символов в строке заранее неизвестно, нужно использовать динамические массивы с элементами типа char.

Например:

1. Объявляется указатель на массив из элементов типа char:

char \*Sp;

2. Прежде чем заносить по этому указателю какой-то текст, надо выделить в памяти блок соответствующего размера функциями malloc, calloc или оператором new;

Str = (char \*) malloc(100);

Str = (char \*)relloc (Str, 100);

chart \*Srr = new char[N];

Для того чтобы выделить память под строку, записанную в окне Edit, используется конструкция:

Str = (char \*) malloc(strlen (Edit1->Text.c\_str()) +1);

где c\_str() – функция преобразования строки типа AnsiString в строку типа char.

Для того чтобы занести в строку текст из окна, применяется выражение:

ctrcpy(St, Edit1->Text.c\_str());

Можно также воспользоваться функцией StrAlloc, например:

shar \*St = StrAlloc(strlen (Edit1->Text.c\_str()) +1);

выделяется блок памяти, в который заносится содержимое Edit.

В этом выражении функция strlen определяет количество символов в окне и добавляет еще один символ для нулевого символа.

Если требуется изменить (увеличить) размер блока памяти, на который указывает St, можно сделать функцией realloc:

realocc (St, strlen(“дополнительный текст”) +1); и затем в строку добавить новый текст:

strcpy (Sp, “дополнительный текст”);

Для обработки строк имеется ряд библиотечных функций, которые приведены в табл. 5.1.

Таблица 5.1 – Библиотечные функции работы со строками типа char

(для символов кириллицы)

|  |  |  |
| --- | --- | --- |
| **Функция** | **Синтаксис** | **Описание** |
| strcat (StrCat) | char \*strcat (char \*S1, const char \*S2) | Конкатенация. Добавляет строку S2 в конец строки S1 |
| strcpy | сhar \*strcpy (char \*S1, const char \*S2) | Копирует строку S2 в строку S1 |
| An-siStrComp | int AnsiStrComp (char \*S1, char \*S2) | Сравнивает строки S1 и S2; c учетом регистра. Результат <0 при S1<S2, = 0 при S1 = S2 и >0 при S1>.S2 |
| strstr | char \*strstr(const char \*S1, char \*S2) | Возвращает первое вхождение строки S1 в строку S1 |
| strlen | size\_t strlen( const char\*S) | Определяет число символов в строке S, не считая нулевого) |
| AnsiStrUp-per | char\*AnsiStrUpper(char S1) | Преобразование символов строки S1 к верхнему регистру |
| Sprintf |  | Построение строки по заданному шаблону форматирования и списку аргументов |
| strchr | char \*strсhr(const char \*S, int C) | Возвращает указатель на первое вхождение C в S |
| strtok | char \*strtok(char \*S1, const char\*s2) | Ищет первое вхождение разделителей из строки S2 в строке S1 и усекает строку S1. Возможны повторения |
| free | free(S) | Освобождает память, выделенную под строку |
| StrNew | StrNew(const char\*Str) | Создает в памяти копию строки Str и заполняет ее символами исходной строки |
| StrECopy | StrECopy(S, S1) | Копирует строку S1 в строку S и возвращает указатель на последний символ скопированной строки |

Примеры использования библиотечных функций обработки строковых переменных:

1. Пусть объявлен массив символов St [50]; Для того, чтобы занести в него какой-либо текст нужно применить функцию:

strcpy: strcpy (St, “Текст копируемый в St”);

2. Пусть необходимо в конец первой строки St1 прибавить текст, хранящийся в строке St2. Это можно сделать с помощью функции strcat:

char St1[20] = “текст 1, St2[10]= “ Текст2”;

stract (St1, St2);

При записи данного алгоритма размер первой строки выбирается с запасом, чтобы можно было разместить второй текст.

Операции, связанные с выделением памяти подстроку и занесением в неё текста строки, довольно громоздки и часто служат причиной ошибок. Ошибка в выделении памяти не замечается во время компиляции, а во время выполнения программы, если длина строки окажется больше, чем выделено памяти, то последние символы строки сотрут какие-то другие данные в памяти, что может привести к грубым ошибкам и зависанию программы. В этой связи необходимо при выделении памяти ставить обработчик исключений.

**Строковые** **переменные** **типа** **AnsiString.**

В языке С++ Builder реализован тип строк AnsiString как класс, объявленный в файле vc/dstring.h. Данный тип строк аналогичен типу данных строк в Delphi. Это тоже строка с нулевым символом на конце, как и строки char\*. Каждый символ в строке AnsiString занимает 8 бит.

Существенным достоинством типа AnsiString является то, что число символов в строке AnsiString лимитировано только объемом памяти компьютера. Память под строки выделяется динамически, причем программисту не приходится об этом думать. Строки AnsiString позволяют работать с символами кириллицы и русского языка, учитывают принятые способы отображения дат и времени.

Со строками AnsiString намного проще, чем при использовании char\*: осуществляется занесение текста в строку, объединение нескольких строк, работа с компонентами C++ Builder.

Тип AnsiString используется для ряда свойств компонентов C++ Builder, например, свойства Text окна редактирования, свойства Caption меток, разделов меню и т.д. Этот же тип используется для отображения отдельных строк в списках строк типа TStrings.

Поэтому, если нет каких-то иных весомых соображений, всегда целесообразно работать со строками AnsiString.

При объявлении строковых переменных типа AnsiString они инициализируются пустыми строками и имеют вид:

AnsiString S1, S2, S3;

Переменные этого типа можно объявлять и как переменные String – это просто псевдоним AnsiString.

Несмотря на то, что применение строкового типа AnsiString практически всегда удобнее (char\*), иногда требуется осуществить переход к типу char\*. Такие переходы приходится делать при передаче параметров в некоторые функции, чаще всего это связано с вызовом функций API Windows или функций C++ Builder, инкапсулирующих такие функции.

Преобразование строки AnsiString в строку(char\*) осуществляется функцией c\_str() без параметров**,** возвращающей строку с нулевым символом в конце, содержащую текст той строки AnsiString, к которой она применена.

Например, если имеются строки S1и S2 типа AnsiString, которые необходимо передать в функцию Application -> MessageBox в качестве сообщения и заголовка окна, то вызов Application -> MessageBox может иметь вид:

Application -> MessageBox(S1.c\_str(), S2.c\_str(), MB\_OK);

Возможно и обратное преобразование строки (char\*) в строку AnsiString, для этого используется функция:

AnsiString (char \*S);

которая возвращает строку типа AnsiString, содержащую текст, записанный в строке S, являющейся аргументом функции.

Для AnsiString определены операции отношения: ==, >, <, >=, <=. Сравнение производится с учетом регистров.

Для AnsiString определены также операции присваивания =, +=, операция склеивания (конкатенации), а также операция индексации [ ], индексы начинаются с 1.

Класс имеет множество методов и библиотечных функций, которые позволяют эффективно работать с переменными указанного типа. Подробное описание функций можно найти в руководстве по языку C++ Builder. Некоторые из наиболее распространенных функций приведены в табл. 5.2.

Таблица 5.2 – Основные функции работы со строками типа AnsiString

|  |  |  |
| --- | --- | --- |
| **Функция** | **Синтаксис** | **Описание** |
| 1 | 2 | 3 |
| Lenght | S.Length() | Возвращает длину строки S в байтах |
| SetLenght | SetLength (S, int new-Length ) | Позволяет уменьшить длину строки S до newLenght символов |
| Создание копии строки | AnsiString S2 = An-siString S1 | Создает строку S2, являющуюся копией строки S1 |
| StringOfChar | StringOfChar(char ch, int count) | Заполняет строку N (count) символами ch |
| Pos | S.Pos(const AnsiString & subStr) const | Функция ищет в строке S первое вхождение подстроки, заданной ее параметром subStr (S2). Если поиск успешный, функция возвращает индекс первого символа найденного вхождения подстроки |
| SubString | SubString (int index, int count) | Возвращает строку, начинающуюся с символав позиции, заданной параметром count |
| Trim | Trim(S) | Возвращает строку, соответствующую исходной, но без пробельных символов до и после значащих символов |
| MidStr | MidStr (count AnsiString AText, int AStart, int ACount) | Возвращает подстроку AText, начинающуюся с символа в позиции AStart и содержащую число символов, не превышающее ACount |
| LeftStr | LeftStr ( const AnsiString AText, int ACount) | Возвращает фрагмент из ACount символов, с которого начинается строка AText |
| RightStr | RightStr (const An-siString AText, int ACount) | Возвращает фрагмент из ACount символов, которым заканчивается строка AText |

Окончание табл. 5.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Delete | S. Delete (int index, int count) | Удаляет из строки S, начиная с позиции, заданной параметром index, число символов, заданное параметром count |
| Insert | S. Insert (const An-siString& str, int index | Вставляет в строку S подстроку, заданную параметром str, в позицию, заданную параметром index |
| TrimLeft | TrimLeft(S) | Возвращает строку, соответствующую исходной, но без начальных пробельных символов |
| TrimRight | TrimRight(S) | Возвращает строку, соответствующую исходной, но без заключительных пробельных символов |

**Упражнение** **5.1.**

Разработать алгоритм, интерфейс пользователя и программу, позволяющую найти в предложении St1 первое вхождение слова St2, и если это слово присутствует, заменить его на фрагмент, содержащийся в строке St3. При создании программы использовать как строки типа char\*, так и AnsiString.

**Выбор метода решения.**

Одним из возможных методов решения задачи является просмотр последовательно всех символов в предложении, выделение слов, находящихся между двух пробелов и сравнение этих слов с требуемым. В случае совпадения произвести последовательную замену символов найденного слова на символы требуемого слова.

Однако использование библиотечных функций позволяет существенно упростить решение поставленной задачи.

Для решения задачи с применением строковых переменных типа char применим три функции (таблица 5.1):

– функцию strstr (S1, S2), которая ищет в строке S1 первое вхождение строки S2 и если поиск прошел удачно, то возвращает указатель на первый символ этого вхождения;

–strcpy(S, S3), которая копирует в строку S строку, накоторую указывает S3;

– strcat (S, S2), позволяющую добавить в конец строки S строку S2;

– strlen (S) – определяет число символов в строке S, не считая нулевого.

Для решения задачи с применением строковых переменных типа AnsiString применим функции (таблица 5.2):

- c\_str() – преобразование строки AnsiString в строку char\*;

- S.Pos( S2) – функция ищет в строке S первое вхождение подстроки, за-данной ее параметром subStr (S2);

- SubString(k,n) – возвращаетстроку, начинающуюся с символа в позиции k, c числом символов n;

- Length() – возвращает длину строки S в байтах.

**Разработка интерфейса пользователя.**

Создайте проект и разместите на форме две командные кнопки для управления программой, три текстовых окна с метками, надписи в которы хуказывают их назначение для ввода исходных данных, два переключателя RadioButton для выбора типа используемых типов строк. Результаты преобразования строк будем выводить в метки. Внешний вид формы проекта приведен на рисунке 5.1.

**Разработка кода программы.**

Программа состоит из двух подпрограмм. В первой подпрограмме выполняется замена слова в предложении с использованием строк типа char, во второй – с использованием строк типа AnsiString, Переключение между подпрограммами осуществляется с использованием переключателей.

В первой подпрограмме помимо строк St1, St2, St3 объявлена строка St, являющаяся буфером, в который будет помещаться результирующий текст. Объявлен также указатель на строку Sb, которая требуется в качестве вспомогательной переменной. Ввод исходного предложения, заменяемого слова и заменяющего слова вводится в окна Edit. Так как текст, введенный в окна Edit имеет тип AnsiString, то для преобразование их к типу char применяется функция c\_str( ). Первый выполняемый оператор кода ищет с помощью функции strstr вхождение строки St2 в строку St1 и присваивает результат поиска переменной Sb. Если поиск не дал результата, то функция strstr возвращает нуль и печатается сообщение «Текст не найден».

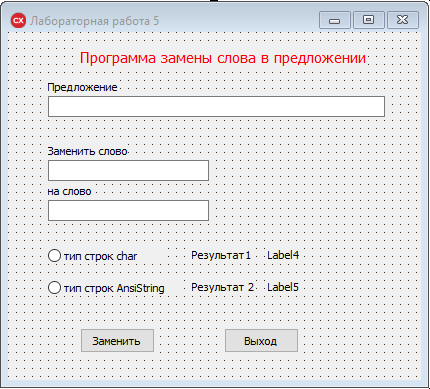


Рисунок 5.1 – Форма для программы замены слова в предложении

Если же поиск прошел успешно, то выполняются следующие операции. Сначала в символ, на который указывает Sb, засылается 0 – это эквивалентно нулевому символу. Таким образом, выделяется первая часть строки St1, расположенная дозаменяющего текста. Затем указатель Sb сдвигается на длину заменяемого текста, которая определяется функцией strlen. После этой операции Sb начинает указывать на первый символ строки S1 после заменяемого текста. Следующий оператор формирует в буфере St текст с заменой и отображает его в метке Label1. Формирование текста осуществляется вложенными вызовами функций strcat и strcpy. Последний внешний вызов strcat добавляет к сформированному тексту часть строки St1, расположенную после замененного фрагмента. Именно на эту часть строки указывает Sb. Результат выводится в метку Label.

**Во** **второй** **подпрограмме** **для** **строк** **типа** **AnsiString** использован ряд функций класса AnsiServer; Pos, SubString, Length. Обратите внимание на то, что обращение к указанным функциям осуществляется операцией (.) – точка вместо более привычной в C++ Builder операции доступа к методам и свойствам объектов (–>). Первый выполняемый оператор использует функцию Pos(S2) для поиска в строке S1 первое вхождение фрагмента S2. Если поиск успешный, функция возвращает индекс первого символа найденного фрагмента, в противном случае – нуль. Индексы начинаются с нуля. Если Индекс не равен нулю, то производится формирование строки с заменой найденной подстроки. Строка формируется склеиванием трех строк: начальной части строки S1, для получения которой используется функция SubString. Аналогично выражение S1.SubString(i+S2.Length(),225) – частьстроки S1, расположенная после найденного вхождения фрагмента S2. В приведенном выражении в качестве второго параметра функции SubString задано число 255, которое, как ожидается, превышает длину подстроки. Результат выводится в метку Label.

Листинг программы замены слова в предложении

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

{

if (RadioButton1->Checked)

//Программа для строки типа char\*

{

char St[100];

char \*Sb;

AnsiString S,S1,S2; //вспомогательная переменная

S = LabeledEdit1 ->Text; // предложение

char \*St1 =S.c\_str();

// преобразование строки AnsiString в строку char\*

S1 = LabeledEdit2 ->Text; // слово, которое надо заменить

char\* St2 = S1.c\_str();

S2 = LabeledEdit3 ->Text;// слово, которое надо вставить

char\* St3 = S2.c\_str();

Sb = strstr(St1, St2);

if(Sb)

{

\*Sb = 0;

Sb += strlen(St2);

Label4 -> Caption = strcat(strcat(strcpy(St,St1),St3),Sb); }

else Label4 -> Caption = "Текст не найден";

}

else

//Программа для строки типа AnsiString

{

AnsiString S1,S2, S3;

int i;

S1 = LabeledEdit1->Text;

S2 = LabeledEdit2->Text;

S3 = LabeledEdit3->Text;

i=S1.Pos(S2);

if(i)

Label5->Caption=S1.SubString(1,i-1)+S3 +S1.SubString(i+S2.Length(),255);

else

Label5->Caption ="Текст не найден";

}

}

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

void \_\_fastcallTForm1::Button2Click(TObject \*Sender)

{

Form1 -> Close();

}

Проверьте результат выполнения программы (рисунок 5.2).

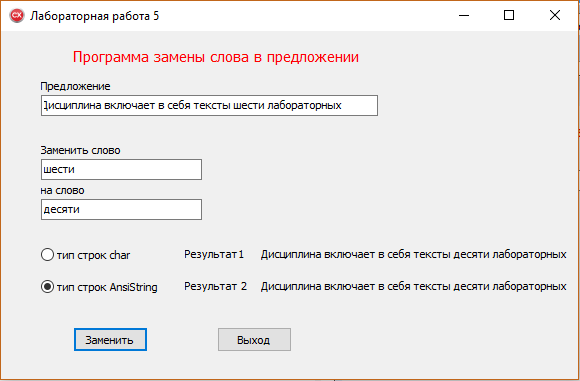


Рисунок 5.2 – Результат выполнения программы по замене слова в предложении

**Упражнение** **5.2**. Разработать алгоритм, интерфейс пользователя и программу, позволяющую определить введенное пользователем слово на русском языке строчными буквами.

Алгоритм решения поставленной задачи состоит в следующем:

1. Определяем количество букв в слове (длину слова).

2. Слово в языке С++ является массивом символов, поэтому выбирая элемент массива, можно сравнить его в цикле с буквами алфавита, который также задается словом (массивом).

3. Определив совпадение первого выбранного элемента с элементом массива алфавита, переходим к следующему элементу. Таким образом, алгоритм включает вложенные циклы со счетчиком.

Внешний цикл – выбор последовательности элементов в слове.

Внутренний – проверка совпадения элемента слова с буквой алфавита. Интерфейс пользователя приведен на рисунке 5.2.

Вводимое слово не маскируется для того, чтобы можно было проверить работу программы.

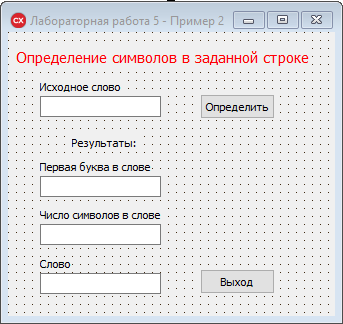


Рисунок 5.2 – Интерфейс пользователя

Листинг программы определения введенного слова:

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

void \_\_fastcallTForm1::Button2Click(TObject \*Sender)

{

Form1->Close();

}

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

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

int i,j;

AnsiString Old,Nov,st1;

AnsiString d,c;

Old = LabeledEdit1->Text;

Nov = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

int n = Old.Length();

LabeledEdit2->Text = Old[1];

LabeledEdit3->Text= n;

for(i=1; i<=Old.Length();i++)

{

for(j=1;j<=Nov.Length(); j++)

{

if (Old[i]==Nov[j]) st1+=Nov[j];

}

}

LabeledEdit4->Text = st1;

}

**Упражнение** **5.3.** Напишите код функции, позволяющий заменить первую строчную букву слова на прописную (верхнего регистра).

Содержание алгоритма состоит в следующем:

1. Выделить первую букву слова.

2. Заменить первую букву слова на прописную и сохранить.

3. Удалить первую букву из слова.

4. Вставить на ее место сохраненную прописную букву.

5. Интерфейс пользователя приведен на рисунке 5.3.

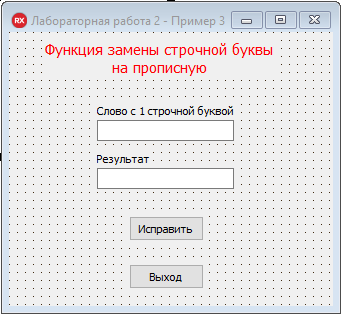


Рисунок 5.3 – Интерфейс пользователя

Код функции, написанный для события Button1Click, приведен ниже. При разработке кода была создана функция AnsiString UCas(AnsiString R, AnsiString R1), которая объявляется и определяется в файле реализации формы в разделе конструктора формы. В обработчике события используется вызов указанной функции.

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

: TForm(Owner)

{

//объявление функции

AnsiString UCas(AnsiString R, AnsiString R1);

}

//Определение функции

AnsiString UCas(AnsiString R, AnsiString R1)

{

AnsiString b = AnsiUpperCase(R[1]); //первая буква слова делается про-писной и сохраняется в переменной b

AnsiString C = R.Delete(1,1); // удаляется первая буква из слова

R1 = b+C;// к слову присоединяется прописная первая буква

return R1;

}

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

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

{

AnsiString S1, b,C,R1;

S1= LabeledEdit1->Text;

LabeledEdit2->Text = UCas(S1,R1);// вызов функции

}

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

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

{

Form1->Close();

}

Проверьте правильность работы программы.

**Структуры** **в** **языке** **C++** **Builder**

Структуры – это составные типы данных, построенные с использованием других типов. Они представляют собой объединенный общим именем набор данных различных типов. Именно тем, что в них могут храниться данные разных типов, они и отличаются от массивов, хранящих данные одного типа.

Отдельные данные структуры называются элементами или полями. Все это напоминает запись в базе данных, только хранящуюся в оперативной памяти компьютера.

Простейший вариант объявления структуры может выглядеть следующим образом:

struct TPers

{

AnsiString Fam,Nam,Par;

unsigned Year;

bool Yes;

AnsiString Dep;

};

Ключевое слово struct начинает определение структуры. Идентификатор TPers – тег (обозначение, имя-этикетка) структуры. Тег структуры используется при объявлении переменных структур данного типа. В этом примере имя нового типа – TPers. Имена, объявленные в фигурных скобках описания структуры, – это элементы структуры. Элементы одной и той же структуры должны иметь уникальные имена, но две разные структуры могут содержать не конфликтующие элементы с одинаковыми именами. Каждое определение структуры должно заканчиваться точкой с запятой.

Определение TPers содержит шесть элементов. Предполагается, что такая структура может хранить данные о сотруднике некоего учреждения. Типы данных разные: элементы Fam, Nam, Par и Dep – строки, хранящие соответственно фамилию, имя, отчество сотрудника и название отдела, в котором он работает. Элемент Year целого типа хранит год рождения, элемент Yes булева типа хранит сведения о поле. Элементы структуры могут быть любого типа, но структура не может содержать экземпляры самой себя.

Например, элемент типа TPers не может быть объявлен в определении структуры TPers. Однако может быть включен указатель на другую структуру типа TPers. Структура, содержащая элемент, который является указателем на такой же структурный тип, называется структурой с самоадресацией. Такие структуры очень полезны для формирования различных списков.

Объявление структуры не резервирует никакого пространства в памяти; оно только создает новый тип данных, который может использоваться для объявления переменных. Переменные структуры объявляются так же, как переменные других типов.

Объявление:

TPers Pers, PersArray[10], \*Ppers;

объявляет переменную Pers типа TPers,

массив PersArray – с 10 элементами типа TPers и указатель Ppers на объект типа TPers.

Переменные структуры могут объявляться и непосредственно в объявлении самой структуры после закрывающейся фигурной скобки. В этом случае указание тега не обязательно:

Struct

{

AnsiString Fam,Nam,Par;

unsigned Year;

bool Yes;

AnsiString Dep;

}

Pers, PersArray[10], \*Ppers;

Для доступа к элементам структуры используются операции доступа к элементам: операция точка (.) и операция стрелка (->). Операция точка обращается к элементу структуры по имени объекта или по ссылке на объект.

Например:

Pers.Fam = "Иванов";

Pers.Nam = "Иван";

Pers.Par = "Иванович";

Pers.Year = 1960;

Pers.Yes = true;

Pers.Dep = "Бухгалтерия";

Операция «стрелка» обеспечивает доступ к элементу структуры через указатель на объект. Допустим, что выполнен оператор:

Ppers = SPers;

который присвоил указателю Ppers адрес объекта Pers. Тогда указанные выше присваивания элементам структуры можно выполнить так:

Ppers->Fam = "Иванов";

Ppers->Nam = "Иван";

Ppers->Par = "Иванович";

Ppers->Year = 1960;

Ppers->Yes = true;

Ppers->Dep = "Бухгалтерия";

**Упражнение** **5.4.** Разработать алгоритм, интерфейс пользователя и программу, позволяющую вести ведомость успеваемости студентов и позволяющую находить данные о студенте по введенной фамилии.

**Внешняя спецификация программы.**

Исходные данные:

Фамилия студента содержит не более 15 букв.

Инициалы студента содержат две прописные буквы, точка между ними и после второй буквы всего четыре символа.

Номер учебной группы (ИС2-151-ОБ) содержит десять символов.

Оценка по дисциплине 1 содержит один символ.

Оценка по дисциплине 2 содержит один символ.

Ведомость успеваемости должна быть представлена в табличной форме, программа должна позволять вводить дополнительные записи и в случае ошибки в записи, должна позволять корректировать записи. Поиск данных о студенте должен производиться по фамилии, результаты должны выводиться в таблицу.

**Выбор способа решения задачи.**

Данные о студенте (запись) представим как переменную типа структура (struct). Для обеспечения ввода данных будем использовать диалоговое окно InputBox.Данные о студентах и результаты поиска будем размещать в объектах StringGrid.

**Разработка алгоритма решения задачи.**

Применительно к поставленной задаче приведем словесное описание алгоритма, так как основные трудности решения данной задачи заключаются в правильном написании кода программы и использовании свойств объектов и стандартных функций.

Алгоритм решения задачи включает следующую последовательность действий.

1. Формируем базу данных (в данной задаче примитивную) об успеваемости студентов.

2. Заполняем базу данных сведениями об успеваемости студентов.

3. Выводим таблицу с данными об успеваемости студентов на экран для визуального контроля.

4. Если при внесении данных допущена ошибка, открываем доступ к таблице для непосредственной корректировки записей.

5. Создаем файл для сохранения базы данных и открываем его для записи.

6. Сохраняем таблицу с данными об успеваемости студентов в созданном файле.

7. Открываем файл для чтения.

8. Считываем данные из файла в таблицу.

9. Вводим фамилию студента, успеваемость которого необходимо просмотреть.

10. Осуществляем поиск данных об успеваемости студента в таблице. 11. Выводим данные об успеваемости студента для просмотра.

**Разработка интерфейса пользователя.**

Интерфейс включает два объекта StringGrid, у которых верхняя строка является фиксированной, в которую выводятся заголовки столбцов. Для этого значению свойства FixedCols объекта присвоим 0, а свойству FixedRows – 1. Для запрета доступа к таблице cвойству goEditing свойства Options объекта присвоим значение false (запрет на доступ к полям таблицы). Для управления доступом в обработчике события для командной кнопки «Исправить» запишем выражение, переводящее значение этого свойства в True (разрешающее доступ к полям таблицы). Свойству Visible присвоим значение false, что делает таблицу 2 невидимой, и в обработчике события для кнопки «Найти» свойству Visible присвоим значение true, что сделает таблицу 2 видимой при осуществлении поиска. Интерфейс пользователя приведен на рисунке 5.4.

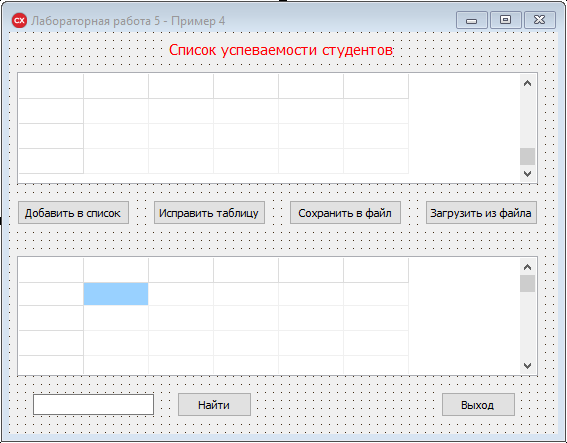


Рисунок 5.4 – Интерфейс формы «Ведомость успеваемости»

Листинг программы

//Объявление функции считывания записи

int GetLine(int f, AnsiString \*st);

//объявление структуры

struct TStudent

{

AnsiString Fam,Inic,Grup,D1,D2;

};

void \_\_fastcallTForm1::Button5Click(TObject \*Sender)

{

Form1->Close(); //завершение работы программы

}

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

int i;

//присваивание наименований столбцам таблицы 1

void \_\_fastcallTForm1::FormCreate(TObject \*Sender)

{

StringGrid1->Cells[0][0]= "№ п/п";

StringGrid1->Cells[1][0]= "Фамилия";

StringGrid1->Cells[2][0]= "Инициалы";

StringGrid1->Cells[3][0]= "Группа";

StringGrid1->Cells[4][0]= "ИС";

StringGrid1->Cells[5][0]= "ИТ";

StringGrid1->Options << goEditing;// разрешение непосредственного доступа к полям таблицы

StringGrid1->Options << goTabs; // Перемещение по полям таблицы с помощью клавиши Tab

//присваивание наименований столбцам таблицы 2

StringGrid2->Cells[0][0]= "№ п/п";

StringGrid2->Cells[1][0]= " Фамилия ";

StringGrid2->Cells[2][0]= " Инициалы ";

StringGrid2->Cells[3][0]= " Группа ";

StringGrid2->Cells[4][0]= " ИС";

StringGrid1->Cells[5][0]= " ИТ";

}

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

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

int i,j;

int index,index0;

int num;

int Lstr;

AnsiString Lst;

TStudent Student1;

Student1.Fam = InputBox("Ввод данных о студенте", "Введите фамилию",Name);

Student1.Inic = InputBox("Ввод данных о студенте", "Введите инициа-лы", "Пусто");

Student1.Grup = InputBox("Ввод данных о студенте", "Введите груп-пу","Пусто");

Student1.D1 = InputBox("Ввод данных о студенте", "ИС", "Пусто");

Student1.D2 = InputBox("Ввод данных о студенте", "Введите оценку по ИТ", "Пусто");

//определение числа строк в таблице

index0 = StringGrid1->RowCount;

// определение номера последней записи for (i=0; i <index0;i++)

{

Lst= StringGrid1->Cells[1][i];

Lstr =Lst.Length(); //определение длины записи в 1-м столбце i-ой строки

if (Lstr==0) break;

index = i; // индекс последней записи

}

//проверка необходимости увеличения числа строк на 1

if(index <(index0 -1)) StringGrid1->RowCount=(index0+1);

num=index +1; // индекс следующей записи

StringGrid1->Cells[0][num]= num;// запись номера по порядку //добавление следующей записи в таблицу

StringGrid1->Cells[1][num]=Student1.Fam;

StringGrid1->Cells[2][num]=Student1.Inic;

StringGrid1->Cells[3][num]=Student1.Grup;

StringGrid1->Cells[4][num]=Student1.D1;

StringGrid1->Cells[5][num]=Student1.D2;

}

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

//функция для обработки события командной кнопки «Исправить таблицу»

void \_\_fastcallTFSpisok::Button2Click(TObject \*Sender)

{

StringGrid1->Enabled=true; //открытие доступа к таблице

}

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

//Функция обработки события командной кнопки «Найти»

void \_\_fastcallTFSpisok::Button3Click(TObject \*Sender)

{

StringGrid2->Visible = true;

AnsiString s1,s2;

s1 = Edit1->Text;

for (i=1; i<=StringGrid1->RowCount; i++)

{

s2= StringGrid1->Cells[1][i];

if(s1==s2) break;

}

StringGrid2->Rows[1]=StringGrid1->Rows[i];

}

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

//Функция обработки события командной кнопки «Сохранить в файл»

void \_\_fastcallTFSpisok::Button4Click(TObject \*Sender)

{

int f; //дискриптор файла

//открыть существующий файл, если его нет, создать новый

if (FileExists("tab1.grd"))

f= FileOpen("tab1.grd",fmOpenWrite); // Открыть файл на запись

else

f = FileCreate("tab1.grd");

if (f!=-1)

{

for (int i =1; i<StringGrid1->RowCount; i++)

{

// формирование строки, оканчивающейся символами конец строки и перевод каретки

AnsiString st = StringGrid1->Rows[i]->DelimitedText+"\r\n";

FileWrite(f,st.c\_str(), st.Length()); //запись строки в файл

}

FileClose(f);//закрытие файла

}

else

ShowMessage("ошибка доступа к файлу");

}

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

void \_\_fastcallTForm1::Button5Click(TObject \*Sender) {

// функция обработки события командной кнопки «Загрузить из файла»

int f;

AnsiString st;

bool f1=true;

if((f = FileOpen("tab1.grd",fmOpenRead))==-1)

return;

while (GetLine(f,&st) !=0)

{

if (f1)

{

StringGrid1->Rows[StringGrid1->Row] -> DelimitedText= st;

f1= false;

} else

{

StringGrid1->RowCount++;

StringGrid1->Row = StringGrid1->RowCount -1;

StringGrid1->Rows[StringGrid1->Row]-> DelimitedText=st;

}

}

FileClose(f);

}

//читает из файла строку символов

//от текущей позиции до символа конец строки

//значение функции количество прочитанных символов

int GetLine (int f, AnsiString \*st)

{

unsigned char buf[256]; //задаем величину буфера для считывания символов строки

unsigned char \*p = buf;// указатель на строку

int n;//количество прочитанных байт

int len = 0;// длина строки

n = FileRead(f,p,1); // стандартная функция считывания байт из файла while (n!= 0)

{

if(\*p =='\r')

{

n = FileRead(f,p,1); //считывание символов строки

break;

}

len++;

p++;

n = FileRead(f,p,1);

}

\*p = '\0';

if (len !=0)// если длина записи не равна 0. Считанные байты из буфера записываются в переменную st

st -> printf("%s",buf);

return len;

}

Запустите программу, отладьте и протестируйте.

## Задания для самостоятельной работы

1. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую определить число слов в предложении.
2. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую определить число букв в предложении.
3. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую определить число гласных букв в предложении.
4. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую определить частоту гласных букв в слове.
5. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую преобразовать введенный текст в прописные буквы.
6. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую определить число согласных букв в предложении.
7. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую определить число слов в двух предложениях.
8. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую произвести инвертирование слова.
9. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую упорядочить по начальной букве фамилии список сотрудников, содержащий фамилию и инициалы.
10. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую упорядочить по начальной букве, фамилии и инициалам список сотрудников, содержащий фамилию и инициалы.
11. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую упорядочить по начальной букве фамилии записи в ведомости успеваемости студентов.
12. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую распознать слово, содержащее прописные и строчные буквы.
13. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую предупредить пользователя о том, что он вводит данные английским шрифтом.
14. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую заменить в слове из прописных букв все буквы кроме первой на строчные.
15. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую поменять в предложении первое и последнее слово местами.

## 

## Контрольные вопросы

1. Переменная символьного типа, её объявление и инициализация. Область использования переменного типа.

2. Строковая переменная типа Char. В чем ее отличие от переменной символьного типа?

3. Способы объявления строковой переменной типа Char.

4. Как осуществляется доступ к отдельным символам строковой переменной.

5. Сколько байт содержится в строковой переменной и сколько значащих символов?

6. Каким элементом заканчивается строковая переменная типа Char?

7. Какая функция используется для склеивания двух строк типа Char?

8. Поясните особенности склеивания строковых переменных типа Char.

9. Как скопировать в строку типа Char какой-либо текст.

10. Какая функция преобразует символы строки типа Char к верхнему регистру.

11. Как определить длину строки типа Char?

12. Объясните в чем особенность строковых переменных типа AnsiString?

13. Какие операции определены для строк типа AnsiString?

14. Как объявляется и инициализируется строка типа AnsiString?

15. Как преобразовать строку типа AnsiString встроку типа Char?

16. В каких случаях необходимо использовать строки типа AnsiString и типа Char?

17. Как преобразовать строку типа Char в строку типа AnsiString?

18. Как склеить несколько строк типа AnsiString?

19. Как можно осуществить доступ к отдельным символам строки типа AnsiString?

20. Что такое структуры в языке C++ Builder?

21. Как объявляются, определяются и используются структуры?

22. Чем структуры отличаются от классов.

23. Какие свойства объекта StringGrid определяют число столбцов и строк?

24. Какое свойство используется для открытия доступа к редактированию таблицы?

# ЛАБОРАТОРНАЯ РАБОТА № 6

# Тема: «Изучение среды C++ Builder. Применение в программах графических изображений».

**Цель** **работы:** Изучение использования графических примитивов для создания графических изображений.

Некоторые компоненты в C++ Builder имеют свойство Canvas (канва, холст для рисования), представляющие собой область компонента, на котором можно рисовать или отображать готовые изображения. Это свойство имеют формы, графические компоненты Image, PaintPoint, Bitmap и многие другие. Холст состоит из отдельных точек – пикселей. Пиксель – это наименьший элемент поверхности рисунка, с которым можно манипулировать. Положение пикселя на поверхности холста имеет координаты X и Y. В системе координат на холсте значение X возрастает при перемещении слева направо, а значение координаты Y – при перемещении сверху вниз. Левая верхняя точка системы координат имеет координаты (0, 0), правая нижняя – координаты (ClientWidth, ClientHeight). Доступ к отдельному пикселю осуществляется через свойство Pixels, представляющую собой двумерный массив, элементы которого содержат информацию о цвете (свойство TColor) точек холста.

Для того чтобы на поверхности формы или компонента нарисовать линию, окружность, прямоугольник или другой графический элемент, необходимо к свойству Canvas применить соответствующий метод. Рисовать можно разными способами: первый вариант – рисование по пикселям, второй вариант – рисование с помощью свойства Pen (перо).

**Рисование** **по** **пикселям**

**Упражнение** **6.1.** Построить график функции y = sin (x). Для данной функции xmin = 0, xmax = 4π, ymin = –1, ymax = 1.

Создайте проект и разместите на форме метки, командные кнопки и два объекта Image1 и Image2.

Выделите оба окна Image и, вызвав контекстное меню, командой Size установите размеры окон одинаковыми.

Напишите код программы для события Click командной кнопки Нарисовать 1:

void \_\_fastcallTForm1::BitBtn1Click(TObject \*Sender)

{

const float Pi= 3.14159;

float x, y; //координата функции

int px,py; // координата пиксела

for (px = 0; px <= Image1->Width; px++)

{

//x – координата точки функции, соответствующая пикселю с координатой px

x = px\*4 \* Pi/Image1->Width;

y = sin(x);

//py – координата пикселя, соответствующая координате y точки

py = Image1 -> Height - (y + 1)\* Image1 -> Height/2;

// устанавливается цвет пикселя для отображения на холсте

Image1->Canvas-> Pixels[px][py] = clBlack;

}

}

В этом коде вводятся переменные x и y, являющиеся значениями аргумента и функции, а также переменные px и py, являющиеся координатами пикселов, соответствующими x и y. Сама процедура состоит из цикла по всем значениям горизонтальной координаты пикселов px компонента Image1. Сначала выбранное значение px пересчитывается в соответствующее значение x. Затем производится вызов функции и определяется ее значение y. Это значение пересчитываетсяв вертикальную координату py. И в заключение цвет пикселя с координатами (px, py) устанавливается черным.

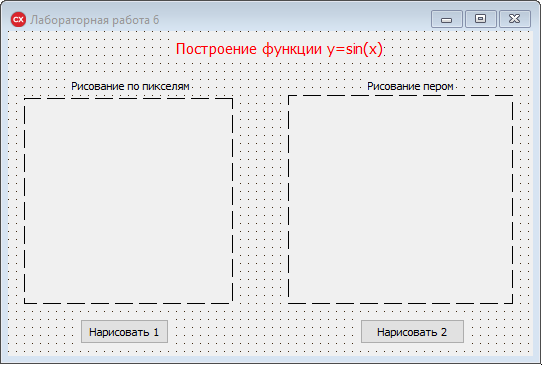


Рисунок 6.1 – Форма программы построения графика sin (x)

**Рисование** **с** **помощью** **пера**

У канвы имеется объекты Pen – перо и Brush –кисть. В свою очередь эти объекты имеют ряд свойств и методов. Свойства объекта Pen (таблица 6.1) задают цвет, толщину, тип линии и границы геометрической фигуры. Свойства объекта Brush (таблица 6.2) задают способ закраски внутри замкнутой области (внутри прямоугольника, внутри окружности, сектора). Методы вычерчивания графических примитивов с помощью пера приведены в таблице 6.3.

Таблица 6.1 – Свойства объекта Pen

|  |  |  |
| --- | --- | --- |
| **Свойство** | **Что** **определяет** | **Значения** |
| Color | Цвет линии | clRed – красный, clGreen – зеленый, clBlue – голубой, clBlack – черный и др., либо цифровой код, определяющий цвет. |
| Width | Толщина линии | Задается в пикселях |
| Style | Вид линии | Ps Solid – сплошная линия, psDash – штриховая линия, psDot – пунктирная линия, psClear – нет линии, psInsideFrame – сплошная линия, но при Width > 1, допускающая цвета, отличные от палитры Windows. |

Таблица 6.2 – Свойства объекта Brush

|  |  |  |
| --- | --- | --- |
| **Свойство** | **Что** **определяет** | **Значения** |
| Color | Цвет закрашивания замкнутой области | clRed – красный, clGreen – зеленый, clBlue – голубой, clBlack – черный и др. либо цифровой код, определяющий цвет. |
| Width | Толщина линии | Задается в пикселях |
| Style | Стиль закрашивания области | bsSolid – сплошная заливка. Штриховка:  bsGorizontal – горизонтальная; bsVertical – вертикальная; bsFDiogonal – диагональная с наклоном линий влево;  bsBDiogonal – диагональная с наклоном линий вправо; bsCross – в клетку;  bsDiagCross – диагональная клетка. |

Таблица 6.3 – Методы вычерчивания графических примитивов с помощью пера

|  |  |
| --- | --- |
| **Метод** | **Действия** |
| LineTo (x, y) | Рисует линию из текущей точки в точку с установленными координатами. |
| Rectangle (x1,y1,x2,y2) | Рисует прямоугольник. (x1, y1, x2, y2) – координаты левого верхнего и нижнего правого углов прямоугольника. |
| FillRect (x1,y1,x2,y2) | Рисует закрашенный прямоугольник. (x1, y1, x2, y2) – координаты диагональных углов прямоугольника. |
| FrameRect(x1,y1,x2,y2) | Рисует контур прямоугольника. Определяет координаты диагональных углов (x1, y1, x2, y2). |
| RounRect(x1,y1,x2,y2) | Рисует прямоугольник с округленными углами. |
| Ellipse(x1,y1,x2,y2) | Рисует окружность или эллипс. (x1, y1, x2, y2) – координаты диагональных углов квадрата либо прямоугольника, в котором рисуется окружность, эллипс соответственно. |
| Are(x1,y1,x2,y2,x3,y3,x4,y4) | Рисует дугу окружности или эллипса. x1, y1, x2, y2 – определяют эллипс (круг). x3, y3, x4, y4 – задают начальную и конечную точки дуги. |
| Chord(x1,y1,x2,y2,x3,y3,x4,y4) | Рисует замкнутую фигуру (сегмент), ограниченную дугой окружности или эллипса и хордой. x1, y1, x2, y2 – определяют эллипс (круг). x3, y3, x4, y4 – задают начальную и конечную точки дуги сегмента. |
| Pie(x1,y1,x2,y2,x3,y3,x4,y4) | Рисует сектор окружности или эллипса. x1, y1, x2, y2 – определяют эллипс (круг). x3,y3,x4,y4 – задают начальную и конечную точки дуги сектора. |
| Polyline (points, n) | Рисует ломаную линию. points – массив типа TPoint. Каждый элемент массива представляет собой запись поля x и y, который содержит координаты точки перегиба ломанной; n – количество звеньев ломанной. |
| Poligon (p, n) | Рисует замкнутую фигуру с кусочно-линейной границей, p – массив записей типа TPoint, который содержит координаты вершин многоугольника, n – количество вершин. |

У канвы имеется свойство PenPos.Это свойство определяет в координатах канвы текущую позицию пера. Перемещение пера без прорисовки линии, т. е. изменение PenPos производится методом канвы MoveTo(x,y). Здесь x, y координаты точки, в которую перемещается перо.Эта текущая точка является исходной, от которой методом LineTo(x, y) можно провести линию в точкус координатами (x, y). При этом текущая точка перемещается в конечную точку линии, и новый вызов метода LineTo будет проводить точку из этой новой текущей точки.

**Упражнение 6.2.** Программы рисования синусоиды методом Pen (Пера). Для формы созданной в первом примере, напишите код программы, указанный в листинге для командной кнопки Нарисовать2:

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

const float Pi= 3.14159;

float x, y; //координаты точки функции

int px,py;// Координаты пикселя, соответствующие

Image2->Canvas->MoveTo(0,Image2->Height/2); //точке с координатами x,y

for (px = 0; px <= Image2->Width; px++)

{

//x – координата, соответствующая пикселю

//с координатой px

x = px\*4 \* Pi/Image2->Width;

y = sin(x);

// y – координата пикселя, соответствующая координате y

py = Image2 -> Height - (y + 1)\* Image2 -> Height/2;

// проводится линия на втором графике

Image2->Canvas->LineTo(px,py);

}

}

Проверьте результат выполнения программы (рисунок 6.2).

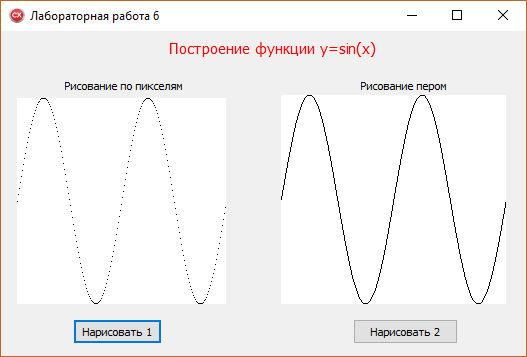


Рисунок 6.2 – Результат выполнения программы

**Рисование** **геометрических** **фигур**

**Упражнение 6.3.** Нарисуйте и закрасьте программным способом фигуры, приведенные в таблице 6.3.

Для выполнения поставленной задачи, создайте проект и разместите на форме объект Image и девять командных кнопок. Сделайте на них надписи, как показано на рисунке 6.3.

Для каждой командной кнопки напишите функцию обработки события Click,как показано на листинге 3. Для построения фигур необходимо определить значения координат основных точек фигур в соответствии с обозначениями, приведенными в табл. 6.3.

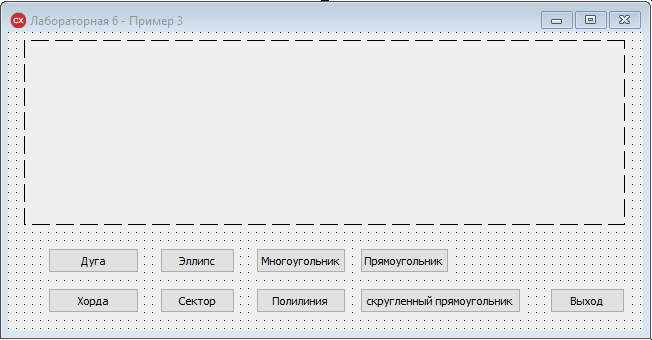


Рисунок 6.3 – Интерфейс формы для построения геометрических фигур

Листинг программы построения примитивных геометрических фигур:

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

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

#pragma package(smart\_init)

#pragma resource"\*.dfm"

TForm1\*Form1;

TPoint points[4];

//---------------------------------------------------------------------------\_\_fastcallTForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

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

void \_\_fastcallTForm1::Button1Click(TObject \*Sender)

{

Image1->Canvas->Font->Style<<fsBold;

Image1->Canvas->Arc(10,10,90,90,90,50,10,50);

Image1->Canvas->TextOut(40,60,"Arc");

}

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

void \_\_fastcallTForm1::Button2Click(TObject \*Sender)

{

Form1->Close();

}

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

void \_\_fastcallTForm1::Button3Click(TObject \*Sender)

{

Image1->Canvas->Brush->Color= clRed;

Image1->Canvas->Chord(110,10,190,90,190,50,110,50);

Image1->Canvas->TextOut(135,60,"Chord");

}

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

void \_\_fastcallTForm1::Button4Click(TObject \*Sender)

{

Image1->Canvas->Brush->Color= clBlue;

Image1->Canvas->Ellipse(210,10,290,50);

Image1->Canvas->TextOut(230,60,"Ellipse");

}

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

void \_\_fastcallTForm1::Button5Click(TObject \*Sender)

{

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

Image1->Canvas->Pie(310,10,390,90,390,30,310,35);

Image1->Canvas->TextOut(340,60,"Pie");

}

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

void \_\_fastcallTForm1::Button6Click(TObject \*Sender)

{

Image1->Canvas->Brush->Color=clAqua ;

points[0].x =30;

points[0].y =150;

points[1].x =40;

points[1].y =130;

points[2].x =40;

points[2].y =140;

points[3].x =60;

points[3].y =130;

points[4].x =170;

points[4].y =150;

Image1->Canvas->Polygon(points,4);

Image1->Canvas->TextOut(30,170,"Polygon"); }

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

void \_\_fastcallTForm1::Button7Click(TObject \*Sender)

{

points[0].x = 200;

points[0].y = 100;

points[1].x = 220;

points[1].y = 150;

points[2].x = 250;

points[2].y = 120;

points[3].x = 280;

points[3].y = 180;

points[4].x = 300;

points[4].y = 150;

Image1->Canvas->Polyline(points,4);

Image1->Canvas->TextOut(250,170,"Polyline");

}

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

void \_\_fastcallTForm1::Button8Click(TObject \*Sender)

{

Image1->Canvas->Brush->Color= clLime;

Image1->Canvas->Rectangle(330,170,480,100);

Image1->Canvas->TextOut(340,170,"Rectangle");

}

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

void \_\_fastcallTForm1::Button9Click(TObject \*Sender)

{

Image1->Canvas->Brush->Color= clMaroon;

Image1->Canvas->RoundRect(500,170,580,100,20,20);

Image1->Canvas->TextOut(540,170,"RoundRect");

}

Проверьте правильность работы программы в соответствии с рисунком 6.4.

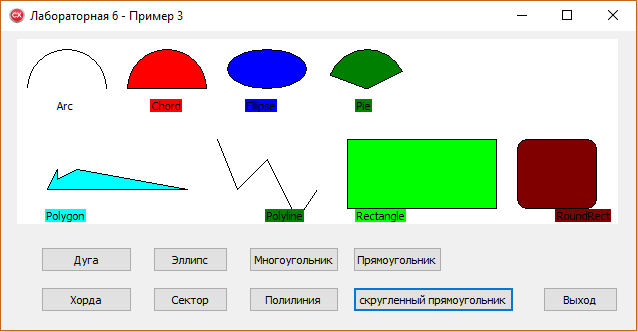


Рисунок 6.4 – Результат рисования геометрических фигур

**Упражнение** **6.4.** Создадим программу, рисующую олимпийскую символику. Целью этого примера является освоение использования функций построения геометрических примитивов для построения более сложного рисунка.

Алгоритм решения данной задачи состоит в следующем:

1. Необходимо рассчитать, на каком расстоянии должны располагаться центры колец и каковы должны быть диаметры колец, чтобы они пересекались.

2. Необходимо определить размеры окна (полотна), на котором будут рисоваться кольца.

3. Определить место и размеры размещения надписи. Все эти расчеты можно провести с использованием линейки, карандаша и циркуля, нарисовав макет. В качестве единиц измерения используются пиксели. Готовая программа должна выводить на печать олимпийскую символику.

Листинг программы с подробными комментариями:

#pragma package(smart\_init)

#pragma resource"\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------\_\_fastcallTForm1::TForm1(TComponent\*Owner)

: TForm(Owner)

{

Canvas->Font->Name="Tahoma";

Form1->Canvas->Font->Size=25;

}

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

void \_\_fastcallTForm1::Button2Click(TObject \*Sender)

{

Form1->Close();

}

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

void \_\_fastcallTForm1::FormPaint(TObject \*Sender)

{

#define WB 140 //ширина полотнища

#define HB 160 //высотаполотнища

#define D 100 //диаметр колец

int x,y;

AnsiString st="Citius, Altius, Fortius!";

// определим координаты левого верхнего угла флага

x =(ClientWidth-WB)/2;

y = (ClientHeight- HB)/2 - Canvas->Font->Size; // полотнище

// выбираем цвет полотнища – белый

Canvas ->Brush->Color= (TColor)RGB(255,255,255); //Рисуем прямоугольник полотнища

Canvas -> Rectangle(x-WB,y-HB/2,x+1.8\*WB,y+HB);

int x1 = (ClientWidth- Canvas -> TextWidth(st))/2;

/\*Чтобы область вывода текста не была закрашена цветом фона, а также чтобы метод Ellipse рисовал окружность, а не круг, значение свойства Brush-> Syle должно быть равно bsClear \*/

Canvas -> Brush->Style= clWhite;

bsClear;

//вывод девиза

Canvas -> TextOutW(x1,y+HB+6,st); // ширина колец – 2 пикселя

Canvas -> Pen->Width=5;

//первый ряд колец

//\*3,2\*D –ширина области, занимаемая кольцами первого ряда\*/

x = x + (WB-3.2\*D)/2;

y =y +(HB-1.8\*D)/2;

Canvas -> Pen->Color= (TColor)RGB(0,0,225); //синий

Canvas -> Ellipse(x,y,x+D,y+D);

x= x +1.1\*D;

Canvas -> Pen->Color= clBlack; //черный

Canvas -> Ellipse(x,y,x+D,y+D);

x= x +1.1\*D;

Canvas -> Pen->Color= (TColor)RGB(225,0,0);

//красный

Canvas -> Ellipse(x,y,x+D,y+D);

// рисуем второй ряд колец

x= x -D\*0.55;

y = y + 0.6\*D;

Canvas->Pen->Color=(TColor)RGB(0,128,0); //зеленый

Canvas -> Ellipse(x,y,x+D,y+D);

x= x-D\*1.1;

Canvas -> Pen->Color= (TColor)RGB(250,217,25);//желтый

Canvas -> Ellipse(x,y,x+D,y+D);

}

//---------------------------------------------------------------------------void \_\_fastcallTForm1::FormResize(TObject \*Sender)

{

Form1->Refresh();

}

Результат выполнения программы представлен на рисунке 6.5.

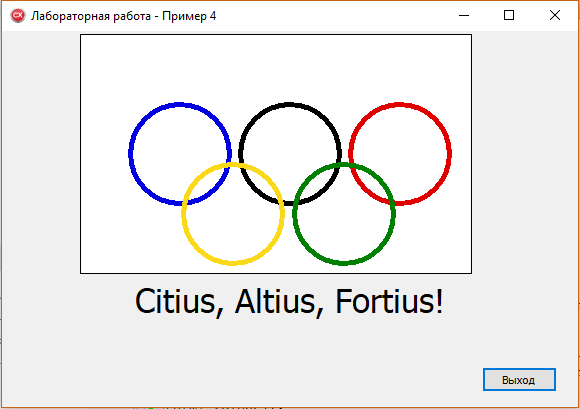


Рисунок 6.5 – Результат выполнения программы

## Задания для самостоятельной работы

1. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме график экспоненциальной функции.

2. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме график параболической функции.

3. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме график гиперболической функции.

4. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме пирамиду из квадрата, круга и треугольника.

5. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме вращающийся круг.

6. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме вертикально расположенную цепочку из 5 колец.

7. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме график функции.

8. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме движущийся круг.

9. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме вращающийся квадрат.

10. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме круг, сектора которого раскрашены в различные цвета.

11. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме график функции.

12. Разработайте алгоритм, интерфейс пользователяи программу, позволяющую отобразить на форме цепочку из разноцветных овалов (эллипсов).

13. Разработайтеалгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме заполнение формы небольшими разноцветными кружочками (конфетти).

14. Разработайтеалгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме прыгающий шарик.

15. Разработайте алгоритм, интерфейс пользователя и программу, позволяющую отобразить на форме падающий квадрат.

## Контрольные вопросы

1. Какие типы графических файлов поддерживает С++ Builder?

2. В чем различия при хранении информации в битовых матрицах и метафайлах?

3. Какие классы объектов определены для хранения графической информации?

4. Какой класс объектов может использоваться для хранения графической информации любого формата?

5. Какие задачи по подготовке графических изображений можно решать с помощью редактора изображений Image Editor?

6. Что представляет собой свойство Canvas, какие объекты имеют это свойство?

7. В чем состоит сущность способа рисования по пикселам (Pixel)?

8. В чем состоит сущность способа рисования пером (Pen)?

9. Какие два метода используются для рисования пером и в чем их сущность?

10. Какой метод при рисовании пером используется для вывода текста?

11. Для чего используется свойство Brush (кисть) у свойства Canvas?

12. В чем сущность стирания изображения?

13. Какой метод используется для закрашивания замкнутой области?

14. Какие свойства формы определяют ее размер по ширине и по длине?

**Библиографический список**

**Основная литература**

1. Голицына, О. Л. Информационные системы [Электронный ресурс]: Учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - 2-e изд. - М.: Форум: НИЦ ИНФРА-М, 2014. - 448 с. – ЭБС "Знаниум".

**Дополнительная литература**

1. Федотова, Е.Л. Информационные технологии и системы [Электронный ресурс]: Учебное пособие / Е.Л. Федотова. - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 352 с.–ЭБС "Знаниум".
2. Скворцова, Т. В. Информационные системы: этапы развития и введение в специальность [Электронный ресурс]: лабораторный практикум. Ч.1 / Т. В. Скворцова, Е. В. Кондрашова ; Т. В. Скворцова, Е. В. Кондрашова; М-во образования и науки Рос. Федерации, Фед. гос. бюджет. образоват. учреждение высш. образования "Воронеж. гос. лесотехн. ун-т им. Г. Ф. Морозова". - Воронеж, 2016. - 115 с.
3. Скворцова, Т. В. Информационные системы: этапы развития и введение в специальность [Электронный ресурс]: методические указания для самостоятельной работы студентов / Т.В. Скворцова; М-во образования и науки РФ, ФГБОУ ВО «ВГЛТУ им. Г.Ф. Морозова». – Воронеж, 2017. – 31 с.

Учебное издание

**Скворцова** Татьяна Владимировна

**Кущева** Ирина Сергеевна

ИНФОРМАЦИОННЫЕ СИСТЕМЫ: ЭТАПЫ РАЗВИТИЯ И ВВЕДЕНИЕ В СПЕЦИАЛЬНОСТЬ

Лабораторный практикум

Часть II

Редактор

Подписано в печать . Формат 60×90 /16. Объем п. л.

Усл. печ. л. . Уч.-изд. л. . Тираж экз. Заказ

ФГБОУ ВО «Воронежский государственный лесотехнический университет имени Г.Ф. Морозова»

РИО ФГБОУ ВО «ВГЛТУ». 394087, г. Воронеж, ул. Тимирязева, 8

Отпечатано в УОП ФГБОУ ВО «ВГЛТУ»

394087, г. Воронеж, ул. Докучаева, 10