Министерство образования Республики Беларусь
Учреждение образования
“Белорусский государственный университет
информатики и радиоэлектроники”

Кафедра “Вычислительные методы и программирование”

ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ В СРЕДЕ BUILDER C++

Лабораторный практикум
по курсам «Программирование»
и «Основы алгоритмизации и программирование»
для студентов 1 – 2-го курсов всех специальностей БГУИР
дневной и вечерней форм обучения
В 2-х частях
Часть 1

Под общей редакцией А. К. Синицына

Минск 2004

УДК 681.3.06 (075.8)
ББК 32.973 я 73
П 78

Авторы:
А.К. Синицын, А.А. Навроцкий, А.В. Щербаков,
Т. М. Кривоносова, В. Т. Карцев

П 78 **Программирование** алгоритмов в среде Builder C++**:** Лаб. практикум по курсам «Программирование» и «Основы алгоритмизации и программирование» для студ. 1–2-го курсов всех спец. БГУИР дневной и вечерней форм обуч: В 2 ч. Ч. 1 / А.К. Синицын, А.А. Навроцкий, А.В. Щербаков и др. ; Под общ. ред. А.К. Синицына. – Мн.: БГУИР, 2004. – 92 с.: ил.
ISBN 985-444-584-4 (ч. 1)

Практикум содержит 10 тем, в которых рассмотрены краткие теоретические сведения по основам программирования в среде Builder C++, а также языку программирования C++. Каждой теме соответствует лабораторная работа и индивидуальные задания.

УДК 681.3.06 (075.8)
ББК 32.973 я 73

ISBN 985–444–584-4 (ч. 1)
ISBN 985-444-583-6

СОДЕРЖАНИЕ

ТЕМА 1. ПРОГРАММИРОВАНИЕ ЛИНЕЙНЫХ АЛГОРИТМОВ

Цель лабораторной работы: научиться составлять каркас простейшей программы в среде C++ Builder. Написать и отладить программу линейного алгоритма.

1.1. Интегрированная среда разработчика C++ Builder

Среда C++ Builder визуально реализуется в виде нескольких окон, одновременно раскрытых на экране монитора. Количество, расположение, размер и вид окон может меняться программистом в зависимости от его текущих нужд, что значительно повышает производительность работы. При запуске C++ Builder вы можете увидеть на экране картинку, подобную представленной на рис. 1.1.

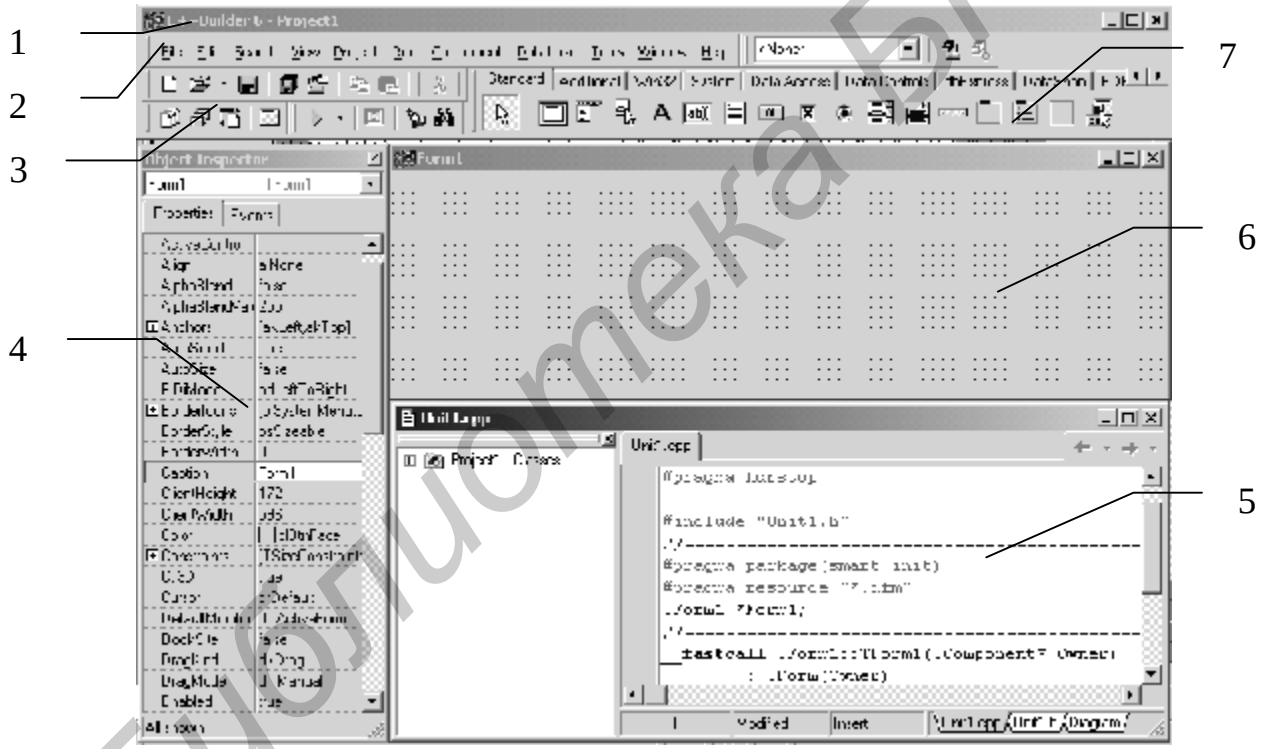


Рис.1.1:

1- главное окно; 2 – основное меню; 3 – пиктограммы основного меню;
4 - окно инспектора объектов; 5 – окно текста программы;
6 - окно пустой формы; 7 – меню компонентов

Главное окно всегда присутствует на экране и предназначено для управления процессом создания программы. Основное меню содержит все необходимые средства для управления проектом. Пиктограммы облегчают доступ к наиболее часто применяемым командам основного меню. Через меню

компонентов осуществляется доступ к набору стандартных сервисных программ среды C++ Builder, которые описывают некоторый визуальный элемент (компонент), помещенный программистом в окно формы. Каждый компонент имеет определенный набор свойств (параметров), которые программист может задавать. Например, цвет, заголовок окна, надпись на кнопке, размер и тип шрифта и др.

Окно инспектора объектов (вызывается с помощью клавиши F11) предназначено для изменения свойств выбранных компонентов и состоит из двух страниц. Страница *Properties* (Свойства) предназначена для изменения необходимых свойств компонента, страница *Events* (События) – для определения реакции компонента на то или иное событие (например, нажатие определенной клавиши или щелчок по кнопке мыши).

Окно формы представляет собой проект Windows-окна программы. В это окно в процессе написания программы помещаются необходимые компоненты. Причем при выполнении программы помещенные компоненты будут иметь тот же вид, что и на этапе проектирования.

Окно текста программы предназначено для просмотра, написания и редактирования текста программы. В системе C++ Builder используется язык программирования C++. При первоначальной загрузке в окне текста программы находится текст, содержащий минимальный набор операторов для нормального функционирования пустой формы в качестве Windows-окна.

Программа в среде C++ Builder составляется как описание алгоритмов, которые необходимо выполнить, если возникает определенное событие, связанное с формой (например, щелчок по кнопке мыши – событие OnClick, создание формы – OnCreate). Для каждого обрабатываемого в форме события с помощью страницы Events инспектора объектов в тексте программы организуется функция, между символами { и }, в которой программист записывает на языке C++ требуемый алгоритм.

Переключение между окном формы и окном текста программы осуществляется с помощью клавиши F12.

1.2. Структура программ C++ Builder

Программа в C++ Builder состоит из множества моделей, которые объединяются в один проект с помощью файла проекта (файл с расширением *.bpr*). Файл проекта автоматически создается и обрабатывается средой C++ Builder и не предназначен для редактирования. Объявления классов, функций и переменных находятся в заголовочном файле (расширение *.h*), текст программы, написанный на языке C++, – в файле исходного текста (расширение *.cpp*). Описание окона формы находится в файле с расширением *.dfm*. Файл проекта может быть только один, файлов с другими расширениями может быть несколько.

Внимание! Для того чтобы перенести проект на другой компьютер, необходимо переписать все файлы с расширениями: **bpr, h, cpp, dfm**.

При запуске программы на выполнение сначала препроцессор преобразует текст в соответствии с имеющимися директивами. После этого компилятор

переводит все тексты в машинный код (расширение .obj), а компоновщик объединяет все файлы в единый модуль (расширение .exe), который может быть запущен на выполнение.

Файл проекта имеет следующую структуру:

```
// Директивы препроцессора
#include <vcl.h>
#pragma hdrstop
  // Подключение файлов форм и файлов ресурсов
USEFORM("Unit1.cpp", Form1);
USEFORM("Unit2.cpp", Form2);
  //  Главная программа
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
                    Application->Initialize();  // Инициализация
// Создание объектов форм
                Application->CreateForm(__classid(TForm1), &Form1);
                Application->CreateForm(__classid(TForm2), &Form2);

                Application->Run();    // Выполнение программы
}
```

Заголовочный файл имеет следующую структуру:

```
 // Директивы препроцессора
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
// Объявление класса формы
class TForm1 : public TForm
{
// Компоненты, размещенные на форме
__published:       // IDE-managed Components
   TLabel *Label1;
   TEdit *Edit1;
   TMemo *Memo1;
  TButton *Button1;
private:       // User declarations
  // Объявления функций, типов переменных, доступных
 //  только в данном модуле
public:                // User declarations
```

```
    // Объявления функций, типов, переменных, доступных
    //   в данном и в других модулях

        __fastcall TForm1(TComponent* Owner);
};
extern PACKAGE TForm1 *Form1;
    // Объявления функций, типов, переменных, которые
    //  не включаются в данный класс
#endif
```

Файл текста программы имеет следующую структуру:

```
    // Директивы препроцессора
#include <vcl.h>    // Подключение заголовочного файла библиотеки VCL
#pragma hdrstop  // Установки компилятора
#include "Unit1.h"  // Подключение заголовочного файла
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;   // Объявление объекта формы
//---------------------------------------------------------------------------
// Вызов конструктора формы
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
```

1.3. Пример написания программы

Задание: составить программу вычисления арифметического выражения для заданных значений x, y, z :

$$u = tg^2(x + y) - e^{y-z}\sqrt{\cos x^2 + \sin z^2}\ .$$

Панель диалога программы организовать в виде, представленном на рис.1.2.

1.3.1. Настройка формы

Пустая форма в правом верхнем углу имеет кнопки управления, которые предназначены для свертывания формы в пиктограмму  , разворачивания формы на весь экран  , возвращения к исходному размеру  и для закрытия формы  . С помощью мыши, «захватывая» одну из кромок формы или выделенную строку заголовка, отрегулируйте нужные размеры формы и ее положение на экране.

1.3.2. Изменение заголовка формы

Новая форма имеет одинаковые имя (Name) и заголовок (Caption) - FORM1. Имя формы менять не рекомендуется, т.к. оно входит в текст программы.

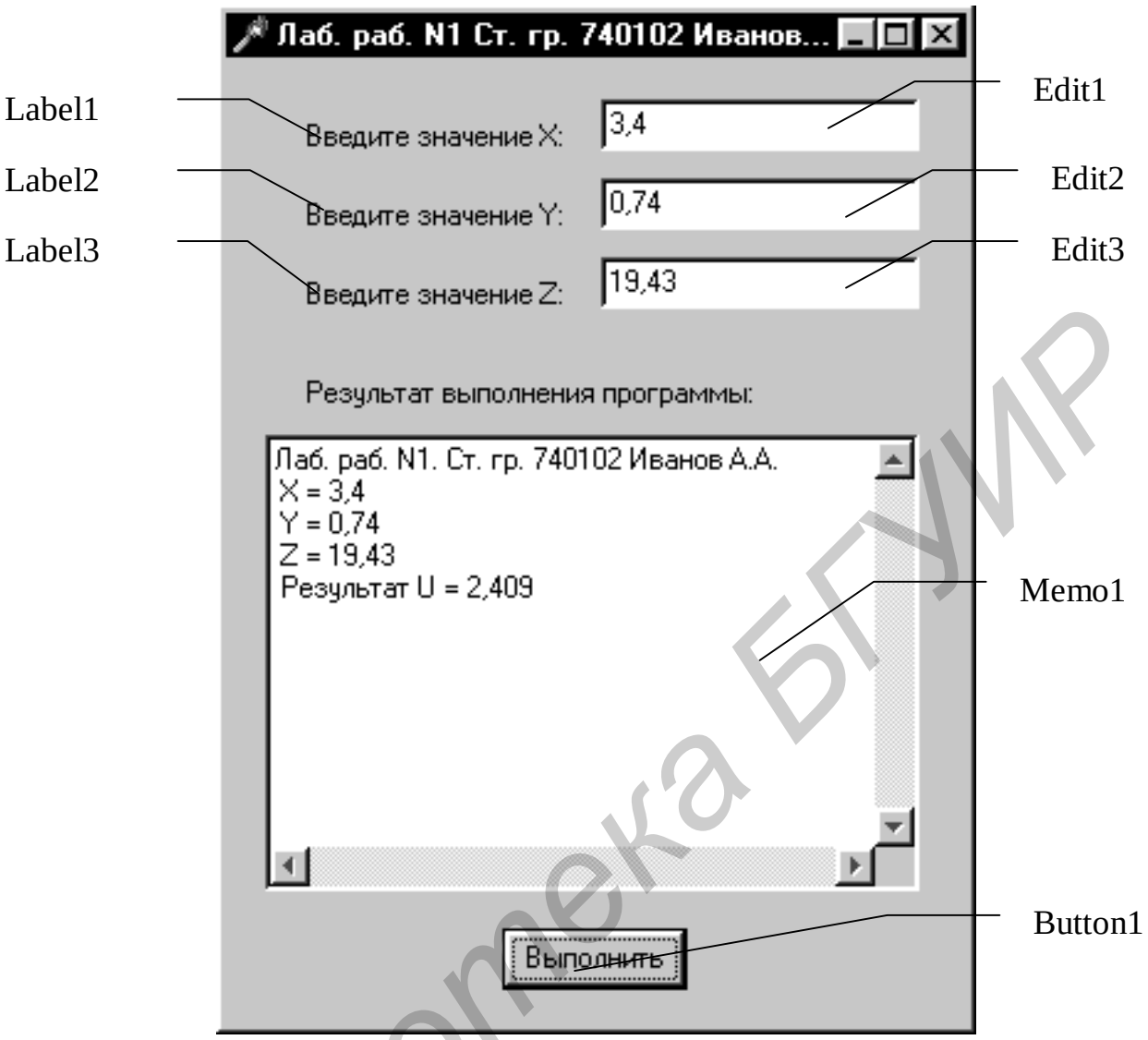


Рис. 1.2

Для изменения заголовка вызовите окно инспектора объектов (F11) и щелкните кнопкой мыши по форме. В форме инспектора объектов найдите и щелкните мышью по Properties – Caption . В выделенном окне наберите номер лабаораторной работы, номер группы, Ф.И.О., например, “Лаб. раб. N1. Ст. гр. 740102 Иванов А.А.”.

1.3.3. Размещение строки ввода (TEdit)

Если необходимо ввести из формы в программу или вывести на форму информацию, которая вмещается в одну строку, используют окно однострочного редактора текста, представляемого компонентом TEdit.

В данной программе с помощью однострочного редактора будут вводиться переменные x,y,z типа double или int.

Выберите в меню компонентов Standard пиктограмму  , щелкните мышью в том месте формы, где вы хотите ее поставить. Вставьте три компонента TEdit в форму. Захватывая их мышью, отрегулируйте размеры окон и их

положение (см. рис. 1.2). Обратите внимание на то, что в тексте программы появились три новые однотипные переменные Edit1, Edit2, Edit3. В каждой из этих переменных с расширением Text будет содержаться строка символов (тип String) и отображаться в соответствующем окне Edit.

Так как численные значения переменных x,y,z имеют действительный тип, для преобразования строковой записи числа, находящегося в переменной Edit1->Text, в действительное, используется стандартная функция X=StrToFloat(Edit1->Text).

Если исходные данные имеют целочисленный тип, например int, то используется стандартная функция X=StrToInt(Edit1->Text).

При этом в записи числа не должно быть пробелов.

С помощью инспектора объектов установите шрифт и размер символов, отражаемых в строке Edit (свойство Font).

1.3.4. Размещение надписей (TLabel)

На форме рис. 1.2 имеются четыре пояснительные надписи. Для нанесения таких надписей на форму используется компонент TLabel. Выберите в меню компонентов Standard пиктограмму  , щелкните по ней мышью. После этого в нужном месте формы щелкните мышью, появится надпись Label1. Проделайте это для четырех надписей. Для каждой надписи, щелкнув по ней, отрегулируйте размер и, изменив свойство Caption инспектора объектов, введите строку, например “Введите значение X:”, а также выберите размер символов (свойство Font).

Обратите внимание, что в тексте программы автоматически появились четыре новые переменные типа TLabel. В них хранятся пояснительные строки, которые можно изменять в процессе работы программы.

1.3.5. Размещение многострочного окна вывода (TMemo)

Для вывода результатов работы программы обычно используется текстовое окно, которое представлено компонентом (TMemo). Выберите в меню компонентов пиктограмму  и поместите компонент TMemo на форму. С помощью мыши отрегулируйте его размеры и местоположение. После установки с помощью инспектора свойства ScrollBars - SSBoth в окне появятся вертикальная и горизонтальная полосы прокрутки.

В тексте программы появилась переменная Memo1 типа TMemo. Информация, которая отображается построчно в окне типа TMemo, находится в массиве строк Memo1->Lines. Каждая строка имеет тип String.

Для очистки окна используется метод Memo1->Clear. Для того чтобы добавить новую строку в окно, используется метод Memo1->Lines->Add (переменная типа String).

Если нужно вывести число, находящееся в переменной действительного или целого типа, то его надо предварительно преобразовать к типу String и добавить в массив Memo1->Lines.

Например, если переменная u=100 целого типа, то метод Memo1->Lines-> Add(“Значение u=”+IntToStr(u)) сделает это и в окне появится строка “Значение u=100”. Если переменная u=-256,38666 действительная, то при использовании метода Memo1->Lines->Add (“Значение u=”+FloatToStrF(u,fffixed,8,2)) будет выведена строка “Значение u= -256.39”. При этом под все число отводится восемь позиций, из которых две позиции занимает его дробная часть.

Если число строк в массиве Memo1 превышает размер окна, то для просмотра всех строк используется вертикальная полоса прокрутки. Если длина строки Memo1 превосходит количество символов в строке окна, то в окне отображается только начало строки. Для просмотра всей строки используется горизонтальная полоса прокрутки.

1.3.6. Написание программы обработки события создания формы (FormCreate)

При запуске программы возникает событие «создание формы» (OnCreate). Создадим программу для обработки этого события, которая заносит начальные значения переменных x, y, z в соответствующие окна TEdit, а в окне TMemo помещает строку с указанием номера группы и фамилии студента. Для этого дважды щелкнем мышью на любом свободном месте формы. На экране появится текст, в который автоматически внесен заголовок функции - обработчика события создания формы: **void __fastcall TForm1::Button1Click(TObject *Sender).** Между {…} вставим текст программы (смотрите пример, п. 1.3.8).

1.3.7. Написание программы обработки события нажатия кнопки (ButtonClick)

Поместите на форму кнопку, которая описывается компонентом TButton, для чего выберите в меню компонентов Standard пиктограмму 

. С помощью инспектора объектов измените заголовок (Caption) – Button1 на слово “Выполнить” или другое по вашему желанию. Отрегулируйте положение и размер кнопки.

После этого два раза щелкните мышью по кнопке, появится текст программы, дополненной заголовком процедуры обработчика события - нажатия кнопки (**void __fastcall TForm2::FormCreate(TObject *Sender)**).

Наберите текст этой процедуры, приведенный в примере.

1.3.8. Запуск и работа с программой

Запустить программу можно, нажав Run в главном меню Run, или клавишу F9, или пиктограмму  . При этом происходит трансляция и, если нет ошибок, компоновка программы и создание единого загружаемого файла с расширением .exe. На экране появляется активная форма программы (см. рис.1.2).

Работа с программой происходит следующим образом. Нажмите (щелкните мышью) по кнопке “Выполнить”. В окне Memo1 появится результат. Измените исходные значения x, y, z в окнах Edit и снова нажмите кнопку ”Выполнить” - появятся новые результаты. Завершить работу программы можно, нажав или

ProgramReset в главном меню Run, или кнопку  на форме. После отладки программы следует сохранить программу, для чего нужно выбрать в меню File пункт Save Project As.

Текст программы:

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Tema1.h"
#include "math.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
Edit1->Text="3,4";   // Начальное значение X
Edit2->Text="0,74";  // Начальное значение Y
Edit3->Text="19,43"; // Начальное значение Z
Memo1->Clear();// Очистка окна редактора Memo1
// Вывод строки в многострочный редактор Memo1
Memo1->Lines->Add("Лаб. раб. N1. Ст. гр. 740102 Иванов А.А.");
}
void __fastcall TForm1::Button1Click(TObject *Sender)
{
double x,y,z;
x=StrToFloat(Edit1->Text);                  // Считывается значение X
Memo1->Lines->Add("x="+Edit1->Text);  // Вывод X в окно Memo1
y=StrToFloat(Edit2->Text);                  // Считывается значение Y
Memo1->Lines->Add("y="+Edit2->Text);  // Вывод Y в окно Memo1
z=StrToFloat(Edit3->Text);                  // Считывается значение Z
Memo1->Lines->Add("z="+Edit3->Text);  // Вывод Z в окно Memo1
// Вычисляем арифметическое выражение
double a=pow(tan(x+y),2);
double b=exp(y-z);
double c=sqrt(cos(x*x)+sin(z*z));
double u=a-b*c;
    // Выводим результат в окно Memo1
Memo1->Lines->Add("Результат U = '+FloatToStrF(u,ffFixed,8,3));
}
//---------------------------------------------------------------------------
```

1.4. Выполнение индивидуального задания

Получите индивидуальное задание у преподователя. Уточните условие задания, количество, наименование, типы исходных данных. В соответствии с этим установите необходимое количество окон Edit, тексты заголовков на форме, размеры шрифтов, а также типы переменных и функции преобразования при вводе и выводе результатов.

С помощью инспектора объектов измените цвет формы, шрифт выводимых символов.

Индивидуальные задания

1. $t = \dfrac{2\cos\left(x - \dfrac{p}{6}\right)}{0.5 + \sin^2 y}\left(1 + \dfrac{z^2}{3 - z^2/5}\right)$.

При x=14.26; y=-1.22; z=3.5×10^{-2} t=0.564849.

2. $u = \dfrac{\sqrt[3]{8 + |x-y|^2 + 1}}{x^2 + y^2 + 2} - e^{|x-y|}\left(tg^2 z + 1\right)^x$.

При x=-4.5; y=0.75×10^{-4}; z=0.845×10^{2} u=-55.6848.

3. $v = \dfrac{1 + \sin^2(x+y)}{\left|x - \dfrac{2y}{1 + x^2 y^2}\right|} x^{|y|} + \cos^2\left(arctg\,\dfrac{1}{z}\right)$.

При x=3.74×10^{-2}; y=-0.825; z=0.16×10^{2} v=1.0553.

4. $w = \left|\cos x - \cos y\right|^{\left(1 + 2\sin^2 y\right)}\left(1 + z + \dfrac{z^2}{2} + \dfrac{z^3}{3} + \dfrac{z^4}{4}\right)$.

При x=0.4×10^{4}; y=-0.875; z=-0.475×10^{-3} w=1.9873.

5. $a = \ln\left(y^{-\sqrt{|x|}}\right)\left(x - \dfrac{y}{2}\right) + \sin^2 arctg(z)$.

При x=-15.246; y=4.642×10^{-2}; z=20.001×10^{2} a =-182.036.

6. $b = \sqrt{10\left(\sqrt[3]{x} + x^{y+2}\right)}\left(\arcsin^2 z - |x-y|\right)$.

При x=16.55×10^{-3}; y=-2.75; z=0.15 b =-40.63069.

7. $g = 5arctg(x) - \dfrac{1}{4}\arccos(x)\dfrac{x + 3|x-y| + x^2}{|x-y|z + x^2}$.

При x=0.1722; y=6.33; z=3.25×10^{-4} g =-205.305.

8. $j = \dfrac{e^{|x-y|}|x-y|^{x+y}}{arctg(x) + arctg(z)} + \sqrt[3]{x^6 + \ln^2 y}$.

При x=-2.235×10^{-2}; y=2.23; z=15.221 j =39.374.

9. $y = \left| x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right| + (y-x)\frac{\cos y - \frac{z}{(y-x)}}{1+(y-x)^2}$.

При x=1.825×10^{2}; y=18.225; z=-3.298×10^{-2} y =1.2131.

10. $a = 2^{-x}\sqrt{x+\sqrt[4]{|y|}}\sqrt[3]{e^{x-1/\sin z}}$.

При x=3.981×10^{-2}; y=-1.625×10^{3}; z=0.512 a=1.26185.

11. $b = y^{\sqrt[3]{|x|}} + \cos^3(y)\frac{|x-y|\left(1+\frac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{|x-y|}+\frac{x}{2}}$.

При x=6.251; y=0.827; z=25.001 b=0.7121.

12. $c = 2^{(y^x)} + (3^x)^y - \frac{y\left(arctgz - \frac{p}{6}\right)}{|x| + \frac{1}{y^2+1}}$.

При x=3.251; y=0.325; z=0.466×10^{-4} c=4.025.

13. $f = \frac{\sqrt[4]{y+\sqrt[3]{x-1}}}{|x-y|\left(\sin^2 z + tgz\right)}$.

При x=17.421; y=10.365×10^{-3}; z=0.828×10^{5} f=0.33056.

14. $g = \frac{y^{x+1}}{\sqrt[3]{|y-2|}+3} + \frac{x+\frac{y}{2}}{2|x+y|}(x+1)^{-1/\sin z}$.

При x=12.3×10^{-1}; y=15.4; z=0.252×10^{3} g=82.8257.

15. $h = \frac{x^{y+1}+e^{y-1}}{1+x|y-tgz|}(1+|y-x|) + \frac{|y-x|^2}{2} - \frac{|y-x|^3}{3}$.

При x=2.444; y=0.869×10^{-2}; z=-0.13×10^{3} h=-0.49871.

В заданиях 16-30 можно использовать только линейные вычисления (нельзя использовать операторы выбора и циклические операторы).

16. Определить количество грузовиков, необходимое для перевозки N ящиков, если каждый грузовик перевозит по M ящиков.

17. Определить время окончания рабочего дня (в часах и минутах), если известны время его начала (в часах и минутах) и продолжительность (вместе с обедом) (в часах и минутах).

18. Перевести белорусское время (в часах) в московское. (Учесть, что 23 часа по белорусскому времени – это 0 часов по московскому).

19. Вывести на экран 0, если заданное число четное, или 1, если оно нечетное.

20. Найти сумму цифр заданного четырехзначного числа.

21. Определить число, полученное выписыванием в обратном порядке цифр заданного трехзначного числа.

22. Присвоить целой переменной k третью от конца цифру в записи положительного целого числа n.

23. Присвоить целой переменной k первую цифру из дробной части положительного вещественного числа.

24. Целой переменной S присвоить сумму цифр трехзначного целого числа k.

25. Идет k-я секунда суток. Определить, сколько полных часов (h) и полных минут (m) прошло к этому моменту.

26. Определить f – угол (в градусах) между положением часовой стрелки в начале суток и ее положением в h часов, m минут и s секунд ($0<=h<=11$; $0<=m$; $s<=59$).

27. Определить h – полное количество часов и m – полное количество минут, прошедших от начала суток до того момента (в первой половине дня), когда часовая стрелка повернулась на f градусов($0<=f<360$; f – *вещественное число*).

28. Пусть k – целое от 1 до 365. Присвоить целой переменной n значение 1,2,…,6 или 7 в зависимости от того, на какой день недели (понедельник, вторник,…, суббота или воскресенье) приходится k-й день невисокосного года, в котором 1 января - понедельник.

29. Поменять местами значения целых переменных x и y, не используя дополнительные переменные.

30. По номеру n ($n>0$) некоторого года определить c – номер его столетия (учесть, что, к примеру, началом XX столетия был 1901, а не 1900 год!).

ТЕМА 2. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

Цель лабораторной работы: научиться пользоваться простейшими компонентами организации переключений (TCheckBox, TRadioGroup). Написать и отладить программу разветвляющегося алгоритма.

2.1. Операторы **if** *и switch языка C++*

Для программирования разветвляющихся алгоритмов в языке C++ используются переменные типа **bool**, которые могут принимать только два значения - **true** и **false** (да, нет), а также операторы **if** и **switch**. Оператор **if** проверяет результат логического выражения или значение переменной типа **int** лабо **bool** и организует разветвление вычислений.

Например, если **bool** bl; double x,y,u; то фрагмент программы с оператором if может быть таким:

```
bl=x>y;
   if (bl) u=x-y;
           else
           u=x+y;
```

Оператор выбора **swith** организует разветвления в зависимости от значения некоторой переменной перечисляемого типа.

Например, если int in, то после выполнения

```
switch (in )
{
case 0: u=x+y; break;
case 1: u=x-y; break;
case 2: u=x*y; break;
default u=0;
}
```

В соответствии со значением **in** вычисляется u. Если in=0, то u=x+y, если in=1, то u=x-y, если in=2, то u=x*y и, наконец, u=0 при любых значениях **in** отличных от 0, 1 или 2. Ветвь **default** можно при необходимости опустить.

2.2. Перечисляемые типы данных

Перечисляемые типы определяют упорядоченное множество идентификаторов, представляющих собой возможные значения переменных этого типа. Вводятся эти типы для того, чтобы сделать код более понятным.

Определяются перечисляемые переменные следующим образом:

enum {константа 1, ... , константа n} <имена переменных>;

Например :

enum {mRed, mYellow, mGreen} mm;

Перечисляемые переменные можно проверять и сравнивать с возможными значениями.

Например:

```
If (mm > mRed)  …

switch (mm) {
case mRed  …
        break;
…

        }
```

По умолчанию перечисляемые значения интерпретируются как целые числа, начиная с -1.

2.3. Кнопки-переключатели в C++ Builder

При создании программ в C++ Builder для организации разветвлений часто используются компоненты в виде кнопок-переключателей. Состояние такой кнопки (включено - выключено) визуально отражается на форме. На форме (рис.2.1) представлены кнопки-переключатели двух типов (TCheckBox, TRadioGroup).

Компонент **TCheckBox** создает кнопку независимого переключателя, с помощью которой пользователь может указать свое решение типа да/нет. В программе состояние кнопки связано со значением булевой переменной, которая проверяется с помощью оператора if.

Компонент **TRadiogroup** создает группу кнопок - зависимых переключателей. При нажатии одной из кнопок группы все остальные кнопки отключаются. В программу передается номер включенной кнопки (0,1,2,...), который анализируется с помощью оператора **switch**.

2.4. Пример написания программы

Задание: ввести три числа - x,y,z. Вычислить по усмотрению u=sin(x), u=cos(x) или u=tg(x). Найти по выбору максимальное из трех чисел: max(u,y,z), или max (|u|,|y|,|z|). Создать форму, представленную на рис. 2.1, и написать соответствующую программу.

2.4.1. Создание формы

Создайте форму, такую же, как в первом задании, скорректировав текст надписей и положение окон TEdit.

2.4.2. Работа с компонентом TCheckBox

Выберите в меню компонентов Standard пиктограмму  и поместите ее в нужное место формы. С помощью инспектора объектов измените заголовок (Caption) на *maxabs*. В тексте программы появилась переменная CheckBox1 типа

TCheckBox. Теперь в зависимости от того, нажата или нет кнопка, булева переменная **CheckBox1.Checked** будет принимать значение true или false.

2.4.3. Работа с компонентом TRadioGroup

Выберите в меню компонентов Standard пиктограмму  и поместите ее в нужное место формы. На форме появится окаймленный линией чистый прямоугольник с заголовком RadioGroup1. Замените заголовок (Caption) на U(x). Для того чтобы разместить на компоненте кнопки, необходимо свойство Columns установить равным единице (кнопки размещаются в одном столбце). Дважды щелкните по правой части свойства Items мышью, появится строчный редактор списка заголовков кнопок. Наберите три строки с именами: в первой строке - cos(x), во второй - sin(x), в третьей - tg(x), нажмите ОК.

После этого на форме внутри окаймления появится три кнопки-переключателя с введенными надписями.

Обратите внимание на то, что в тексте программы появится переменная RadioGroup1 типа TRadioGroup. Теперь при нажатии одной из кнопок группы в переменной целого типа **RadioGroup1->ItemIndex** будет находиться номер нажатой клавиши (отсчитывается от нуля), что используется в тексте приведенной программы.

2.4.4. Создание обработчиков событий FormCreate и Botton1Click

Функции для обработки событий FormCreate и Botton1Click создаются аналогично тому, как и в первой теме. Текст процедур приведен ниже.

Запустите программу и убедитесь в том, что все ветви алгоритма выполняются правильно. Форма приведена на рис.2.1.

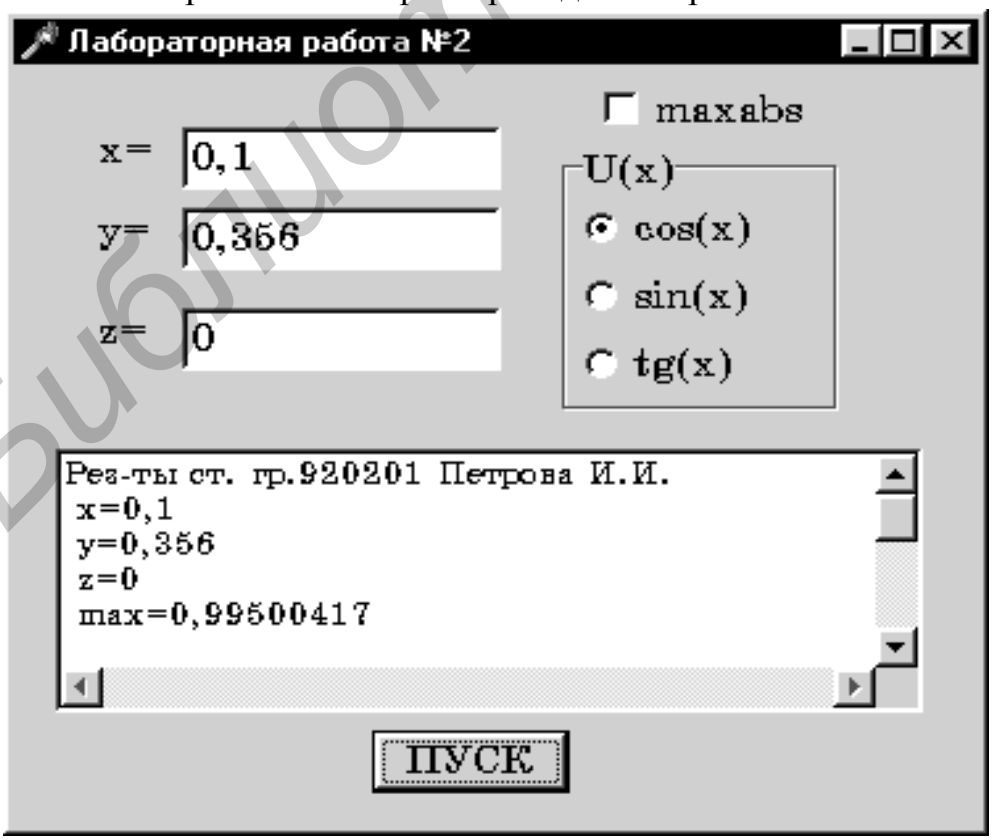


Рис. 2.1

Текст программы:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "math.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
Edit1->Text="0,1";
Edit2->Text="0,356";
Edit3->Text="0";
Memo1->Clear();
Memo1->Lines->Add("Рез-ты ст. гр.920201 Петрова И.И.");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 double x,y,z,u,ma;
        // Ввод исходных данных и вывод в окно Memo1
x=StrToFloat(Edit1->Text);
Memo1->Lines->Add("x="+Edit1->Text);
y=StrToFloat(Edit2->Text);
Memo1->Lines->Add("y="+Edit2->Text);
z=StrToFloat(Edit3->Text);
Memo1->Lines->Add("z="+Edit3->Text);
          // Проверка номера нажатой кнопки и выбор соответствующей ей
функции
switch(RadioGroup1->ItemIndex)
{
case 0: u=cos(x);  break;
case 1: u=sin(x);  break;
case 2: u=tan(x);  break;
}
if (CheckBox1->Checked)  // Проверка состояния кнопки CheckBox1
```

```
{
u=fabs(u);
y=fabs(y);
z=fabs(z);
}
                // Нахождение максимального из трех чисел
if (u>y) ma=u; else ma=y;
if (z>ma) ma=z;
if (CheckBox1->Checked) Memo1->Lines->Add("maxabc="+
                                          FloatToStrF(ma,ffFixed,8,6));
                    else  Memo1->Lines->Add("max="+
                                          FloatToStrF(ma,ffFixed,8,6));
}
```

2.5. Выполнение индивидуального задания

Получите индивидуальное задание у преподавателя. В качестве f(x) использовать по выбору: sh(x), x^2, e^x. Отредактируйте вид формы и текст программы в соответствии с полученным заданием. Предусмотрите вывод информации, показывающий, по какой ветви производились вычисления.

1. $$a=\begin{cases}(f(x)+y)^2-\sqrt{|f(x)*y|}, & xy\rangle 0\\ (f(x)+y)^2+\sqrt{|f(x)+y|}, & xy\langle 0\\ (f(x)+y)^2+1, & xy=0.\end{cases}$$

2. $$b=\begin{cases}\ln(f(x))+(f(x)^2+y)^3, & x/y\rangle 0\\ \ln|f(x)/y|+(f(x)+y)^3, & x/y\langle 0\\ (f(x)^2+y)^3, & x=0\\ 0, & y=0.\end{cases}$$

3. $$c=\begin{cases}f(x)^2+y^2+\sin(y), & x-y=0\\ (f(x)-y)^2+\cos(y), & x-y\rangle 0\\ (y-f(x))^2+tg(y), & x-y\langle 0.\end{cases}$$

4. $$d=\begin{cases}(f(x)-y)^3+arctg(f(x)), & x\rangle y\\ (y-f(x))^3+arctg(f(x)), & y\rangle x\\ (y+f(x))^3+0.5, & y=x.\end{cases}$$

5. $$e=\begin{cases}i\sqrt{f(x)}, & i-\text{нечетное}, \ x\rangle 0\\ i/2\sqrt{|f(x)|}, & i-\text{четное}, \ x\langle 0\\ \sqrt{|if(x)|}, & \text{иначе.}\end{cases}$$

6. $$g=\begin{cases}e^{f(x)-|b|}, & 0.5\langle xb\langle 10\\ \sqrt{|f(x)+b|}, & 0.1\langle xb\langle 0.5\\ 2f(x)^2, & \text{иначе.}\end{cases}$$

7. $$s=\begin{cases}e^{f(x)}, & 1\langle xb\langle 10\\ \sqrt{|f(x)+4*b|}, & 12\langle xb\langle 40\\ bf(x)^2, & \text{иначе.}\end{cases}$$

8. $$j=\begin{cases}\sin(5f(x)+3m|f(x)|), & -1\langle m\langle x\\ \cos(3f(x)+5m|f(x)|), & x<m\\ (f(x)+m)^2, & x=m.\end{cases}$$

9. $$l=\begin{cases}2f(x)^3+3p^2, & x\rangle|p|\\ |f(x)-p|, & 3\langle x\langle|p|\\ (f(x)-p)^2, & x=|p|.\end{cases}$$

10. $$k=\begin{cases}\ln(|f(x)|+|q|), & |xq|\rangle 10\\ e^{f(x)+q}, & |xq|\langle 10\\ f(x)+q, & |xq|=10\end{cases}$$

11. $$m=\frac{\max(f(x),y,z)}{\min(f(x),y)}+5.$$

12. $$n=\frac{\min(f(x)+y,y-z)}{\max(f(x),y)}.$$

13. $$p=\frac{|\min(f(x),y)-\max(y,z)|}{2}.$$

14. $$q=\frac{\max(f(x)+y+z,xyz)}{\min(f(x)+y+z,xyz)}.$$

15. $r=\max(\min(f(x),y),z)$.

16. Известно, что из четырех чисел a_1, a_2, a_3 и a_4 одно отлично от трех других, равных между собой. Присвоить номер этого числа переменной *n*.

17. По заданному номеру месяца определить сезон (весна, лето, осень или зима).

18. Определить, поместится ли прямоугольная коробка размером a x b в прямоугольный ящик размером c x d (учесть возможность поворота коробки).

19. Определить, является ли заданный год високосным.

20. Дано целое *k* от 1 до 180. Определить, какая цифра находится в *k*-й позиции последовательности 10111213…9899, в которой выписаны подряд все двузначные числа.

21. В старояпонском календаре был принят 60-летний цикл, состоявший из пяти 12-летних подциклов. Подциклы обозначались цветами: green (зеленый), red (красный), yellow (желтый), white (белый) и black (черный). Внутри каждого подцикла годы носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи, например: 1984 год – *год зеленой крысы* – был началом очередного цикла.

Разработать программу, которая вводит номер некоторого года нашей эры и выводит его название по старояпонскому календарю.

22. Если сумма трех попарно различных действительных чисел *x*, *y*, *z* меньше единицы, то наименьшее из этих трех чисел заменить полусуммой двух других; в противном случае заменить меньшее из *x* и *y* полусуммой двух оставшихся значений.

23. Для целого числа *k* от 1 до 99 вывести фразу “мне *k* лет”, учитывая при этом, что при некоторых значениях *k* слово “лет” надо заменить на слово “год” или “года”.

24. Для натурального числа *k* вывести фразу “мы выпили *k* бутылок фанты”, согласовав окончание слова “бутылка” с числом *k*.

25. Вывести на экран *1* или *0* в зависимости от того, имеют три заданных целых числа одинаковую четность или нет.

26. Вывести на экран *1* или *0* в зависимости от того, равна ли сумма двух первых цифр заданного четырехзначного числа сумме двух его последних цифр.

27. Вывести на экран *1* или *0* в зависимости от того, равен ли квадрат заданного трехзначного числа кубу суммы цифр этого числа.

28. Вывести на экран *1* или *0* в зависимости от того, есть ли среди первых трех цифр дробной части заданного положительного вещественного числа цифра ноль.

29. Вывести на экран *1* или *0* в зависимости от того, есть ли среди цифр заданного трехзначного числа одинаковые.

30. Значения переменных *a, b* и *c* поменять местами так, чтобы оказалось a<=b<=c.

ТЕМА 3. ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ АЛГОРИТМОВ

Цель лабораторной работы: изучить простейшие средства отладки программ в среде C++ Builder. Составить и отладить программу циклического алгоритма.

3.1. Операторы организации циклов do..while, while, for *языка C++*

Под циклом понимается многократное выполнение одних и тех же операторов при различных значениях промежуточных данных. Число повторений может быть задано в явной или неявной форме.

Для организации повторений в языке C++ предусмотрены три различных оператора цикла.

Оператор

```
do
< операторы >
 while ( < условие > );
```

организует повторение операторов, помещенных между ключевыми словами **do** и **while**, до тех пор, пока выполнится <условие>=true, после чего управление передается следующему за циклом оператору.

Оператор

```
while ( < условие > ) {
                        < операторы >
                         }
```

организует повторение операторов, помещенных между **{** и **}**, до тех пор, пока не выполнится <условие>=false. Заметим, что если <условие>=false при первом входе, то <операторы> не выполнятся ни разу, в отличие от **do..while**, в котором они выполнятся хотя бы один раз.

Оператор

```
for (i=i1; i<=i2 ; i++ )
{
  < операторы >
}
```

организует повторение операторов при нарастающем изменении переменной цикла i от начального значения i1 до конечного i2 с шагом “единица”. Заметим, что если i2>i1, то <операторы> не выполнятся ни разу.

3.2. Средства отладки программ в C++ Builder

Практически в каждой вновь написанной программе после запуска обнаруживаются ошибки.

Ошибки первого уровня (ошибки компиляции) связаны с неправильной записью операторов (орфографические, синтаксические). При обнаружении ошибки компилятор C++ Builder останавливается напротив первого оператора, в котором обнаружена ошибка. В нижней части экрана появляется текстовое окно, содержащее сведения обо всех ошибках, найденных в проекте. Каждая строка этого окна содержит имя файла, в котором найдена ошибка, номер строки с

ошибкой и характер ошибки. Для быстрого перехода к интересующей ошибке необходимо дважды щелкнуть по строке с ее описанием. Для получения более полной информации о характере ошибки необходимо обратиться к HELP нажатием клавиши F1. Следует обратить внимание на то, что одна ошибка может повлечь за собой другие, которые исчезнут при ее исправлении. Поэтому следует исправлять ошибки последовательно, сверху вниз и, после исправления каждой ошибки компилировать программу снова.

Ошибки второго уровня (ошибки выполнения) связаны с ошибками выбранного алгоритма решения или с неправильной программной реализацией алгоритма. Эти ошибки проявляются в том, что результат расчета оказывается неверным либо происходит переполнение, деление на ноль и др. Поэтому перед использованием отлаженной программы ее надо протестировать, т.е. сделать просчеты при таких комбинациях исходных данных, для которых заранее известен результат. Если тестовые расчеты указывают на ошибку, то для ее поиска следует использовать встроенные средства отладки среды C++ Builder.

В простейшем случае для локализации места ошибки рекомендуется поступать следующим образом. В окне редактирования текста установить курсор в строке перед подозрительным участком и нажать клавишу F4 (выполнение до курсора). Выполнение программы будет остановлено на строке, содержащей курсор. Теперь можно увидеть, чему равно значение интересующей переменной. Для этого можно поместить на нужную переменную курсор (на экране будет высвечено ее значение) либо нажать Ctrl-F7 и в появившемся диалоговом окне указать интересующую переменную (с помощью данного окна можно также изменить значение переменной во время выполнения программы). Нажимая клавишу F7 (пошаговое выполнение), можно построчно выполнять программу, контролируя изменение тех или иных переменных и правильность вычислений. Если курсор находится внутри цикла, то после нажатия F4 расчет останавливается после одного выполнения тела цикла. Для продолжения расчетов следует нажать <Run> меню Run.

3.3. Порядок выполнения задания

Задание: написать и отладить программу, которая выводит таблицу значений функции S(x) для x, изменяющегося в интервале от X1 до X2 с шагом h.

$$S(x) = \sum_{k=0}^{N} (-1)^k \frac{x^k}{k!}$$

Панель диалога представлена на рис. 3.1.

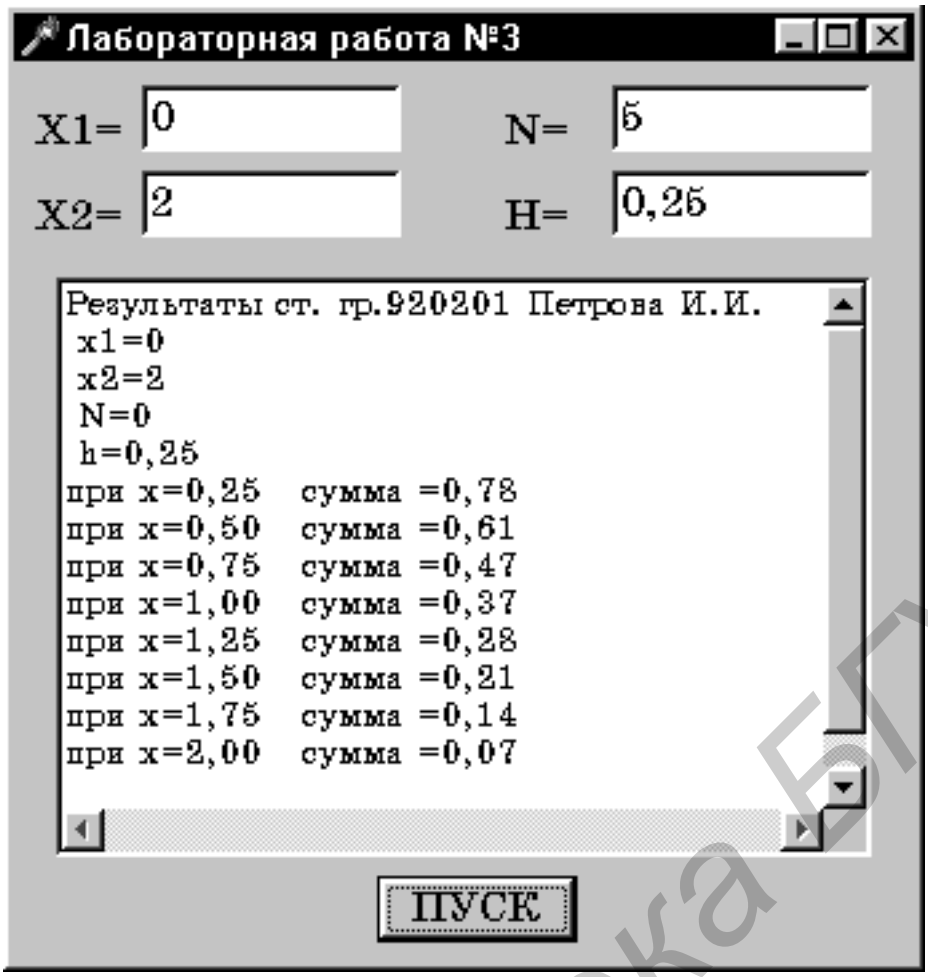


Рис. 3.1

Текст программы:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
Edit1->Text="0";
```

```
Edit2->Text="2";
Edit3->Text="5";
Edit4->Text="0,25";
Memo1->Clear();
Memo1->Lines->Add("'Результаты ст. гр.920201 Петрова И.И");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
double x1,x2,x,h,a,s;
int N;
x=x1=StrToFloat(Edit1->Text);
Memo1->Lines->Add("x1="+Edit1->Text);
x2=StrToFloat(Edit2->Text);
Memo1->Lines->Add("x2="+Edit2->Text);
N=StrToInt(Edit3->Text);
Memo1->Lines->Add("N="+Edit3->Text);
h=StrToFloat(Edit4->Text);
Memo1->Lines->Add("h="+Edit4->Text);
int c=-1;
do
{ x+=h;  // Запись эквивалентна x=x+h;
double a=1,s=1;
for(int k=1;k<=N;k++)
{
a=c*a*x/k;
s+=a;
}
Memo1->Lines->Add("при x”+FloatToStrF(x,ffFixed,8,2)
+"сумма="+FloatToStrF(s,ffFixed,8,2));
}
while(x<x2);
}
//---------------------------------------------------------------------------
```

3.4. Выполнение индивидуального задания

Получите индивидуальное задание у преподавателя. Откорректируйте панель диалога и текст программы.

Индивидуальные задания

В заданиях с 1 по 15 (табл. 3.1) необходимо вывести на экран таблицу значений функции Y(x) и ее разложения в ряд S(x) для x, изменяющегося от a до b с шагом h=(b-a)/10. Близость значений S(x) и Y(x) во всем диапазоне значений x указывает на правильность вычисления S(x) и Y(x).

Таблица 3.1

№ п.п.	a	b	S(x)	n	Y(x)
1	0.1	1	$x-\frac{x^3}{3!}+\ldots+(-1)^n\frac{x^{2n+1}}{(2n+1)!}$	160	$\sin x$
2	0.1	1	$1+\frac{x^2}{2!}+\ldots+\frac{x^{2n}}{(2n)!}$	100	$\frac{e^x+e^{-x}}{2}$
3	0.1	1	$1+\frac{\cos\frac{p}{4}}{1!}x+\ldots+\frac{\cos n\frac{p}{4}}{n!}x^n$	120	$e^{x\cos\frac{p}{4}}\cos(x\sin\frac{p}{4})$
4	0.1	1	$1-\frac{x^2}{2!}+\ldots+(-1)^n\frac{x^{2n}}{(2n)!}$	80	$\cos x$
5	0.1	1	$1+3x^2+\ldots+\frac{2n+1}{n!}x^{2n}$	140	$(1+2x^2)e^{x^2}$
6	0.1	1	$x+\frac{x^3}{3!}+\ldots+\frac{x^{2n+1}}{(2n+1)!}$	80	$\frac{e^x-e^{-x}}{2}$
7	0.1	1	$\frac{x^3}{3}-\frac{x^5}{15}+\ldots+(-1)^{n+1}\frac{x^{2n+1}}{4n^2-1}$	120	$\frac{1+x^2}{2}arctgx-\frac{x}{2}$
8	0.1	1	$1+\frac{2x}{1!}+\ldots+\frac{(2x)^n}{n!}$	100	e^{2x}
9	0.1	1	$1+2\frac{x}{2}+\ldots+\frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n$	140	$\left(\frac{x^2}{4}+\frac{x}{2}+1\right)e^{\frac{x}{2}}$
10	0.1	0.5	$x-\frac{x^3}{3}+\ldots+(-1)^n\frac{x^{2n+1}}{2n+1}$	150	$arctgx$
11	0.1	1	$1-\frac{3}{2}x^2+..+(-1)^n\frac{2n^2+1}{(2n)!}x^{2n}$	100	$\left(1-\frac{x^2}{2}\right)\cos x-\frac{x}{2}\sin x$
12	0.1	1	$-\frac{(2x)^2}{2}+\frac{(2x)^4}{24}-\ldots+(-1)^n\frac{(2x)^{2n}}{(2n)!}$	80	$2(\cos^2 x-1)$
13	-2	-0.1	$-(1+x)^2+\frac{(1+x)^4}{2}+\ldots+(-1)^n\frac{(1+x)^{2n}}{n}$	160	$\ln\frac{1}{2+2x+x^2}$
14	0.2	0.8	$\frac{x}{3!}+\frac{4x^2}{5!}+\ldots+\frac{n^2}{(2n+1)!}x^n$	120	$\frac{1}{4}\left(\frac{x+1}{\sqrt{x}}sh\sqrt{x}-ch\sqrt{x}\right)$
15	0.1	0.8	$\frac{x^2}{2}-\frac{x^4}{12}+\ldots+(-1)^{n+1}\frac{x^{2n}}{2n(2n-1)}$	180	$xarctgx-\ln\sqrt{1+x^2}$

В заданиях с № 16 по 30 (табл. 3.2.) необходимо вывести на экран таблицу значений функции Y(x) и ее разложения в ряд S(x) с точностью ε. Близость значений S(x) и Y(x) во всем диапазоне значений x указывает на правильность вычисления S(x) и Y(x). Вывести число итераций, необходимое, для достижения заданной точности.

Таблица 3.2

№ п.п.	a	b	S(x)	ε	Y(x)
1	2	3	4	5	6
16	-0,9	0,9	$x+\frac{x^3}{3}+\ldots+\frac{x^{2k+1}}{2k+1}$	10^{-4}	$\frac{1}{2}\ln\frac{1+x}{1-x}$
17	0,1	0,9	$1-\frac{x^2}{3!}+\ldots+(-1)^k\frac{x^{2k}}{(2k+1)!}$	10^{-5}	$\frac{\sin x}{x}$
18	-0,9	0,9	$1+\frac{x}{3}+\sum_{k=2}^{\infty}(-1)^{k-1}\frac{1\cdot2\cdot5\cdot8\cdot\ldots(3k-4)}{3^k k!}x^k$	10^{-3}	$\sqrt[3]{1+x}$
19	-3	3	$1+\frac{\ln 9}{1}x+\ldots+\frac{(\ln 9)^k}{k!}x^k$	10^{-4}	9^x
20	-1	1	$\frac{x^3}{2\cdot3}+\frac{1\cdot3\cdot x^5}{2\cdot4\cdot5}+\ldots+\frac{(2k-1)!!}{(2k)!!}\cdot\frac{x^{2k+1}}{(2k+1)}$	10^{-3}	$-x+\arcsin x$
21	-0,9	0,9	$-\frac{x^2}{2\cdot4}+\frac{3\cdot x^3}{2\cdot4\cdot6}+\ldots+(-1)^{k-1}\frac{(2k-3)!!}{(2k)!!}x^k$	10^{-3}	$\sqrt{1+x}-1-\frac{x}{2}$
22	-0,5	0,5	$\frac{x^3}{3}+\frac{x^7}{7}+\ldots+\frac{x^{4k-1}}{4k-1}$	10^{-5}	$\frac{1}{4}\ln\frac{1+x}{1-x}-\frac{1}{2}arctgx$
23	-0,3	0,4	$\frac{1}{1+x}+\frac{2x}{1+x^2}+\ldots+\frac{2^{k-1}x^{(2^{k-1}-1)}}{1+x^{(2^{k-1})}}$	10^{-4}	$\frac{1}{1-x}$
24	-2	2	$\frac{\cos x}{1}+\frac{\cos 3x}{9}+\ldots+\frac{\cos(2k-1)x}{(2k-1)^2}$	10^{-4}	$\frac{p\left(p-2\lvert x\rvert\right)}{8}$
25	-0,5	0,5	$\sum_{k=1}^{\infty}\left(\frac{(-1)^{k+1}}{k}+\frac{(-1)^k\cdot6}{k^3p^2}\right)\sin kp\,x$	10^{-3}	$\frac{p\,x^3}{2}$
26	-1	1,3	$\sum_{k=2}^{\infty}(-1)^k\frac{\cos kx}{k^2-1}$	10^{-5}	$\frac{2x\sin x-2+\cos x}{4}$
27	1	2,5	$\sum_{k=1}^{\infty}\frac{\cos kx}{k^2}$	10^{-5}	$\frac{3x^2-6p\,x+2p^2}{12}$
28	-1,5	1,5	$\sum_{k=1}^{\infty}(-1)^{k-1}\frac{\cos kx}{k}$	10^{-4}	$\ln(2\cos\frac{x}{2})$

Окончание табл. 3.2

1	2	3	4	5	6
29	-0,85	0,95	$\sum_{k=2}^{\infty} \frac{(-1)^{k-1}(4k-5)!!}{(4k)!!} x^k$	10^{-4}	$\sqrt[4]{x+1} - \frac{4-x}{4}$
30	-2,5	1,3	$\sum_{k=1}^{\infty} \frac{\sin(2k-1)x}{2k-1}$	10^{-4}	$\frac{p \cdot signx}{4}$

31. Подсчитать k - количество цифр в десятичной записи целого неотрицательного числа n.

32. Переменной t присвоить значение 1 или 0 в зависимости от того, является ли натуральное число k степенью 3.

33. Дано n вещественных чисел. Вычислить разность между максимальным и минимальным из них.

34. Дана непустая последовательность различных натуральных чисел, за которой следует 0. Определить порядковый номер наименьшего из них.

35. Даны целое n>0 и последовательность из n вещественных чисел, среди которых есть хотя бы одно отрицательное число. Найти величину наибольшего среди отрицательных чисел этой последовательности.

36. Дано n вещественных чисел. Определить, образуют ли они возрастающую последовательность.

37. Дана последовательность из n целых чисел. Определить, со скольких отрицательных чисел она начинается.

38. Определить k – количество трехзначных натуральных чисел, сумма цифр которых равна n(1<=n<=27). Операции деления (/, div и mod) не использовать.

39. Вывести на экран в возрастающем порядке все трехзначные числа, в десятичной записи которых нет одинаковых цифр (операции деления не использовать).

40. Переменной t присвоить значение 1 или 0 в зависимости от того, можно или нет натуральное число n представить в виде трех полных квадратов.

41. Дано натуральное число n. Выяснить, входит ли цифра 3 в запись числа n^2.

42. Дано натуральное число n. Найти сумму его цифр.

43. Дано целое n>0, за которым следует n вещественных чисел. Определить, сколько среди них отрицательных.

44. Дано натуральное число n. Переставить местами первую и последнюю цифры числа n.

45. Дано натуральное число n. Заменить порядок следования цифр числа n на обратный.

46. Дано натуральное число *k*. Определить *k*–ю цифру в последовательности 11010010001000010000 0…, в которой выписаны подряд степени числа 10.

ТЕМА 4. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ

Цель лабораторной работы: изучить свойства компонента TStringGrid. Написать программу с использованием массивов.

4.1. Работа с массивами

Массив - это упорядоченный набор однотипных элементов, объединенных под одним именем. Каждый элемент массива обозначается именем, за которым следует один или несколько индексов, каждый в квадратных скобках, например: a[1], bb[I], c12[I][j*2], q[1][1][I*j-1]. В качестве индекса можно использовать любые порядковые типы. В C++ индексы массива всегда начинаются с нуля.

Примеры описания массивов:

```
const Nmax=10;                    // Задание максимального значения индекса;
typedef double mas1[Nmax];        // Описание типа одномерного массива;
typedef double mas2[Nmax][Nmax];  // Описание типа двухмерного массива;
mas1 A;                           // B – массив типа mas1;
mas2 B;                           //A – массив типа mas2;
int Ss[10];                       // Ss – массив из десяти целых чисел;
char Y[5][10];                    // Y – двухмерный массив символьного типа.
```

Элементы массивов могут использоваться в выражениях так же, как и обычные переменные, например:

```
F=2*A[3]+A[Ss[I]+1]*3;
A[n]=1+sqrt(fabs(A[n-1]));.
```

4.2. Компонент TStringGrid

При работе с массивами ввод и вывод информации на экран удобно организовывать в виде таблиц. Компонент TStringGrid предназначен для отображения информации в виде двухмерной таблицы, каждая ячейка которой представляет собой окно однострочного редактора (аналогично окну TEdit). Доступ к информации осуществляется с помощью свойства Cells[ACol, ARow: Integer]: string, где ACol, ARow - индекс элемента двухмерного массива (ACol – номер столбца, ARow – номер строки). Свойства ColCount и RowCount устанавливают количество столбцов и строк в таблице, а свойства FixedCols и FixedRows задают количество столбцов и строк фиксированной зоны. Фиксированная зона выделена другим цветом, и в нее запрещен ввод информации с клавиатуры.

4.3. Порядок выполнения задания

Задание: создать программу расчета значений вектора $\vec{Y} = A * \vec{B}$, где A - квадратная матрица размерностью NxN, а Y, B – одномерные массивы размерностью N. Элементы вектора Y определяются по формуле $Y_i = \sum_{j=0}^{N-1} A_{ij} \cdot B_i$.

Значение N вводить в компонент TEdit, A и B - в компонент TStringGrid. Результат, после нажатия кнопки типа TButton, вывести в компонент TStringGrid.

Панель диалога приведена на рис. 4.1.

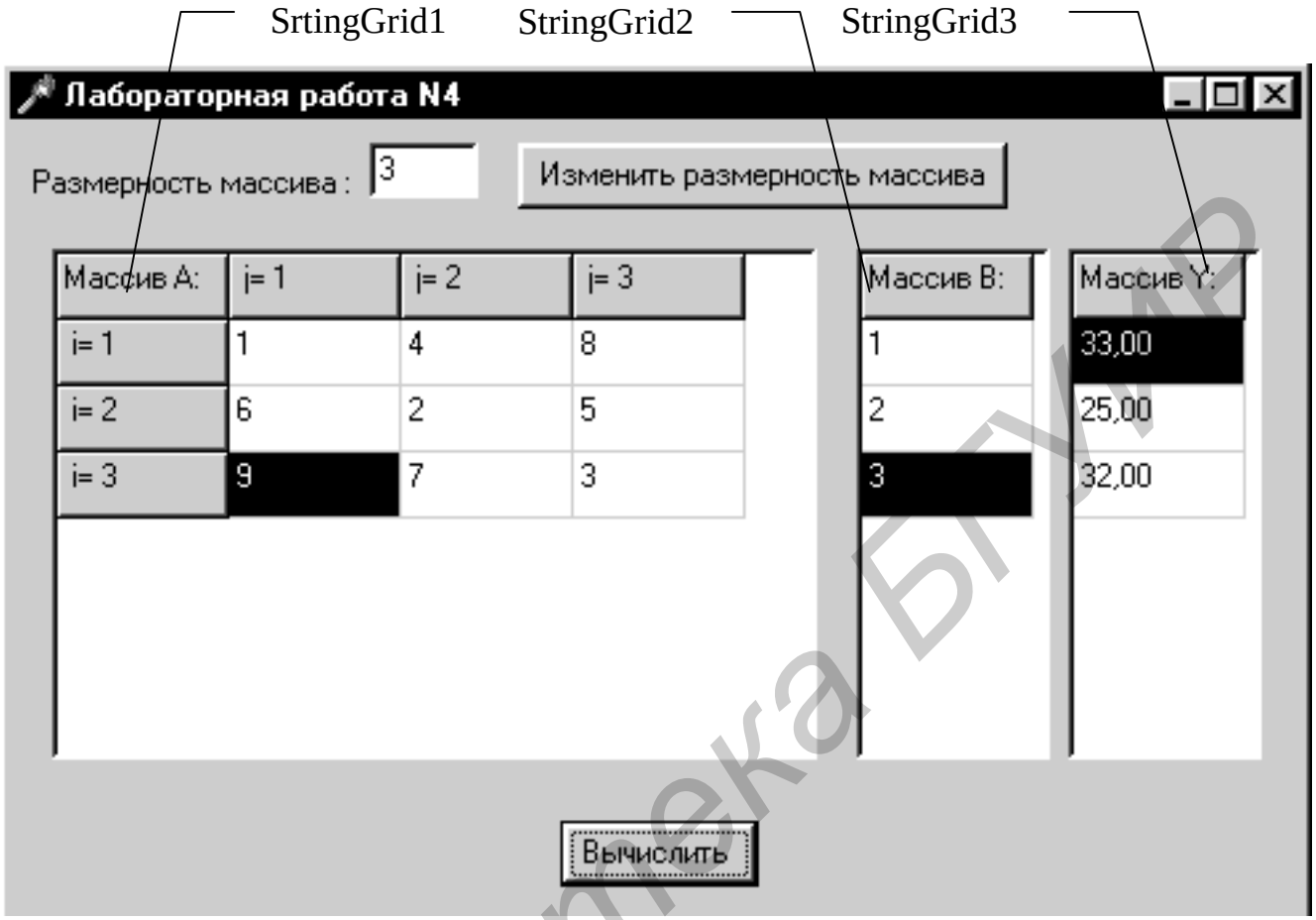


Рис. 4.1

Настройка компонента TStringGrid

Для установки компонента TStringGrid на форму необходимо на странице Additional меню компонентов щелкнуть мышью по пиктограмме  . После этого щелкнуть мышью в нужном месте формы. Захватывая кромки компонента, отрегулируйте его размер. В инспекторе объектов значения свойств ColCount и RowCount установить 2 (два столбца и две строки), а FixedCols и FixedRows установить 1 (один столбец и одна строка с фиксированной зоной). Так как компоненты StringGrid2 и StringGrid3 имеют только один столбец, то у них: ColCount=1, RowCount=2, FixedCols=0 и FixedRows=1. По умолчанию в компонент TStringGrid запрещен ввод информации с клавиатуры, поэтому необходимо свойство Options goEditing для компонентов StringGrid1 и StringGrid2 установить в положение True.

Текст программы:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
const Nmax=10;    // Максимальная размерность массива
typedef double mas2[Nmax][Nmax]; // Объявление типа двухмерного
массива
typedef double mas1[Nmax];  // Объявление типа одномерного массива
int N=3;
void __fastcall TForm1::FormCreate(TObject *Sender)
{
//Задание числа строк и столбцов в таблицах
Edit1->Text=FloatToStr(N);
StringGrid1->ColCount=N+1;
StringGrid1->RowCount=N+1;
StringGrid2->RowCount=N+1;
StringGrid3->RowCount=N+1;
//Ввод в левую верхнюю ячейку таблицы названия массива
StringGrid1->Cells[0][0]="Массив A";
StringGrid2->Cells[0][0]="Массив  B";
StringGrid3->Cells[0][0]="Массив  Y";
for(int i=1; i<=N;i++)
{
StringGrid1->Cells[0][i]="i="+IntToStr(i);
StringGrid1->Cells[i][0]="j="+IntToStr(i);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
N=StrToInt(Edit1->Text);
StringGrid1->ColCount=N+1;
StringGrid1->RowCount=N+1;
StringGrid2->RowCount=N+1;
StringGrid3->RowCount=N+1;
StringGrid1->Cells[0][0]="Массив A";
StringGrid2->Cells[0][0]="Массив  B";
StringGrid3->Cells[0][0]="Массив  Y";
for(int i=1; i<=N;i++)
```

```
{
StringGrid1->Cells[0][i]="i="+IntToStr(i);
StringGrid1->Cells[i][0]="j="+IntToStr(i);
}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
mas2 a; // Объявление двухмерного массива
mas1 b,y; // Объявление одномерных массивов
int i,j;
//Заполнение массива A элементами из таблицы StringGrid1
for(i=0; i<N;i++)
for(j=0; j<N;j++)
a[i][j]=StrToFloat(StringGrid1->Cells[i+1][j+1]);
//Заполнение массива B элементами из таблицы StringGrid2
for( i=0; i<N;i++)
b[i]=StrToFloat(StringGrid2->Cells[0][i+1]);
//Умножение массива A на массив B
for( i=0; i<N;i++)
{
double s=0;
for( j=0; j<N;j++)
s+=a[j][i]*b[j];
y[i]=s;
//Вывод результата в таблицу StringGrid3
StringGrid3->Cells[0][i+1]=FloatToStrF(y[i],ffFixed,8,2);
}
}
//---------------------------------------------------------------------------
```

4.4. Индивидуальные задания

Во всех заданиях по теме «Массивы» скалярные переменные вводить с помощью компонента TEdit с соответствующим пояснением в виде компонента TLabel. Скалярный результат выводить в виде компонента TLabel. Массивы вводить с формы и выводить на форму, используя компонент TStringGrid, в котором 0-й столбец и 0-ю строку использовать для отображения индексов массивов. Вычисления выполнять после нажатия кнопки типа TButton.

1. Задана матрица размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение 0, если все элементы *k*-го столбца матрицы нулевые, и значение 1 – в противном случае.

2. Задана матрица размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение 1, если элементы *k*–й строки матрицы упорядочены по убыванию, и значение 0 – в противном случае.

3. Задана матрица размером *NxM*. Получить массив *B*, присвоив его *k*-му элементу значение 1, если *k*-я строка матрицы симметрична, и значение 0 – в противном случае.

4. Задана матрица размером *NxM*. Определить *k* – количество “особых” элементов матрицы, считая элемент “особым”, если он больше суммы остальных элементов своего столбца.

5. Задана матрица размером *NxM*. Определить *k* – количество “особых” элементов матрицы, считая элемент “особым”, если в его строке слева от него находятся элементы, меньше его, а справа – больше.

6. Задана символьная матрица размером *NxM*. Определить *k* - количество различных элементов матрицы (т.е. повторяющиеся элементы считать один раз).

7. Дана матрица размером *NxM*. Упорядочить ее строки по возрастанию их первых элементов.

8. Дана матрица размером *NxM*. Упорядочить ее строки по возрастанию суммы их элементов.

9. Дана матрица размером *NxM*. Упорядочить ее строки по возрастанию их наибольших элементов.

10. Определить, является ли заданная квадратная матрица *n*-го порядка симметричной относительно побочной диагонали.

11. Для матрицы размером *NxM* вывести на экран все ее седловые точки. Элемент матрицы называется седловой точкой, если он является наименьшим в своей строке и одновременно наибольшим в своем столбце, или наоборот.

12. В матрице *n*-го порядка переставить строки так, чтобы на главной диагонали матрицы были расположены элементы, наибольшие по абсолютной величине.

13. В матрице *n*-го порядка найти максимальный среди элементов, лежащих ниже побочной диагонали, и минимальный среди элементов, лежащих выше главной диагонали.

14. В матрице размером *NxM* поменять местами строку, содержащую элемент с наибольшим значением со строкой, содержащей элемент с наименьшим значением.

15. Из матрицы *n*-го порядка получить матрицу порядка *n-1* путем удаления из исходной матрицы строки и столбца, на пересечении которых расположен элемент с наибольшим по модулю значением.

16. Дан массив из *k* символов. Вывести на экран сначала все цифры, входящие в него, а затем все остальные символы, сохраняя при этом взаимное расположение символов в каждой из этих двух групп.

17. Дан массив, содержащий от 1 до *k* символов, за которым следует точка. Вывести этот текст в обратном порядке.

18. Дан непустой массив из цифр. Вывести на экран цифру, наиболее часто встречающуюся в этом массиве.

19. Отсортировать элементы массива *X* по возрастанию.

20. Элементы массива *X* расположить в обратном порядке.

21. Элементы массива *X* циклически сдвинуть на *k* позиций влево.

22. Элементы массива *X* циклически сдвинуть на *n* позиций вправо.

23. Преобразовать массив *X* по следующему правилу: все отрицательные элементы массива перенести в начало, а все остальные – в конец, сохраняя

исходное взаимное расположение как среди отрицательных, так и среди остальных элементов.

24. Элементы каждого из массивов *X* и *Y* упорядочены по неубыванию. Объединить элементы этих двух массивов в один массив *Z* так, чтобы они снова оказались упорядоченными по неубыванию.

25. Дан массив из *k* символов. Определить, симметричен ли он, т.е. читается ли он одинаково слева направо и справа налево.

26. Даны два массива. Найти наименьшие среди тех элементов первого массива, которые не входят во второй массив.

27. Определить количество инверсий в этом массиве X (т.е. таких пар элементов, в которых большее число находится слева от меньшего: $x_i > x_j$ при $i < j$).

28. Дан массив из строчных латинских букв. Вывести на экран в алфавитном порядке все буквы, которые входят в этот текст по одному разу.

29. Вывести на экран заданный массив из *k* символов, удалив из него повторные вхождения каждого символа.

30. Определить, сколько различных символов входит в заданный текст, содержащий не более *k* символов и оканчивающийся точкой (в сам текст точка не входит).

ТЕМА 5. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТРОК

Цель лабораторной работы: изучить правила работы с компонентами TListBox и TComboBox. Написать программу работы со строками.

5.1. Типы данных для работы со строками

5.1.1. Нуль-терминальная строка char[]

В языке C++ нет специального типа для объявления строк, поэтому они представляются в виде одномерных массивов символов. Последний элемент такого массива – нулевой байт ‘\0’ – (нуль-терминатор). Длина строки равна количеству символов плюс нуль-терминатор. При наборе нуль-терминатор в конце строки ставить не нужно, его автоматически ставит компилятор языка C++ (надо не забывать выделять под него место).

Строки могут быть описаны следующим образом:

```
char st1[10]=”123456789”;
chat st2[]=”1234”;
```

Если размер строки не объявлен явно, то он будет установлен автоматически и будет равен количеству введенных символов +1.

Для работы с такими строками существует набор функций, который расположен в файле string.h. Наиболее часто применяются следующие функции:

strcpy(st1, st2) - копирует содержимое строки st2 в строку st1.

strcat(st1, st2) - приписывает содержимое строки st2 к строке st1.

strcmp(st1, st2) – сравнивает содержимое строк st2 и st1. Если st1>st2, то результат отрицательный, если st1=st2 - результат равен нулю, если st1<st2 – результат положительный. Функция **strcmpi(st1, st2)** делает то же самое, но без учета регистра и только для латинского алфавита.

strstr(st1, st2) – указывает первое появление подстроки st2 в строке st1.

strlen() – возвращает длину строки (нуль-терминатор ‘\0’ не учитывается)

Функции преобразования строк в числа:

atoi(st) – преобразует строку st в число целого типа (int).

atol(st) – преобразует строку st в число длинного целого типа (long).

atof(st) – преобразует строку st в число действительного типа (double).

Функции преобразования чисел в строки:

itoa(a,st,kod) – преобразует числа целого типа (int) в строку st.

ltoa(a,st,kod) – преобразует число длинного целого типа (long) в строку st.

5.1.2. Тип AnsiString

Основной строковый тип в C++ Builder имеет большой набор функций, позволяющих производить операции присваивания переменных различных типов. Основные методы:

c_str() - возвращает указатель на нуль-терминальную строку, содержащую ту же информацию, что и исходная строка.

FloatToStrF() - форматированный перевод из действительного числа в строку.

IntToHex() - перевод шестнадцатеричного числа. Второй параметр - минимальное число цифр.

StrToInt(), **StrToDouble()** – перевод строк в числа, соответственно **int**, **double**.

Delete(), **SubString()** - удаление символов и копирование подстроки.

Pos() - позиция строки-аргумента, начиная с 1.

Length() - длина строки.

5.1.3. Широкая строка типа WideString

Этот класс взят из **Delphi**. Основным его отличием от **AnsiString** является хранение массива расширенных символов, называемых **wide characters**, тип **wchar_t***. Поэтому он в основном используется в **COM**.

5.2. Компонент TListBox

Компонент TListBox представляет собой список, элементы которого выбираются при помощи клавиатуры или мыши. Список элементов задается свойством Items, методы Add, Delete и Insert которого используются для добавления, удаления и вставки строк. Объект Items хранит строки, находящиеся в списке. Для определения номера выделенного элемента используется свойство ItemIndex.

5.3. Компонент TComboBox

Комбинированный список TComboBox представляет собой комбинацию списка TListBox и редактора TEdit, поэтому практически все свойства заимствованы у этих компонентов. Для работы с окном редактирования используется свойство Text, как в TEdit, а для работы со списком выбора - свойство Items, как в TListBox. Существует пять модификаций компонента, определяемых его свойством Style. В модификации csSimple список всегда раскрыт, в остальных он раскрывается после нажатия кнопки справа от редактора.

5.4. Компонент TBitBtn

Компонент TBitBtn расположен на странице Additonal палитры компонентов и представляет собой разновидность стандартной кнопки TBotton. Его отличительная особенность – наличие растрового изображения на поверхности кнопки, которое определяется свойством Clyph. Кроме того, имеется свойство Kind, которое задает одну из 11 стандартных разновидностей кнопок. Нажатие любой из них, кроме bkCustom и bkHelp, закрывает модальное окно и

возвращает в программу результат mr*** (например bkOk - mrOk). Кнопка bkClose закрывает главное окно и завершает работу программы.

5.5. Обработка событий

Обо всех происходящих в системе событиях, таких, как создание формы, нажатие кнопки мыши или клавиатуры и т.д., ядро Windows информирует окна путем посылки соответствующих сообщений. Среда C++ Builder позволяет принимать и обрабатывать большинство таких сообщений. Каждый компонент содержит обработчики сообщений на странице Events инспектора объектов.

Для создания обработчика события необходимо раскрыть список компонентов в верхней части окна инспектора объектов и выбрать необходимый компонент. Затем на странице Events нажатием левой клавиши мыши выбрать обработчик и дважды щелкнуть по его левой (белой) части. В ответ C++ Builder активизирует окно текста программы и покажет заготовку процедуры обработки выбранного события.

Каждый компонент имеет свой набор обработчиков событий, однако некоторые из них присущи большинству компонентов. Наиболее часто применяемые события представлены в табл. 5.1.

Таблица 5.1

Событие	Описание события
OnActivate	Форма получает это событие при активации
OnCreate	Возникает при создании формы (компонент TForm). В обработчике данного события следует задавать действия, которые должны происходить в момент создания формы, например: установка начальных значений
OnKeyPress	Возникает при нажатии кнопки на клавиатуре. Параметр Key имеет тип WORD и содержит ASCII-код нажатой клавиши (клавиша Enter клавиатуры имеет код 13, клавиша Esc - 27 и т.д.). Обычно это событие используется в том случае, когда необходима реакция на нажатие одной из клавиш
OnKeyDown	Возникает при нажатии клавиши на клавиатуре. Обработчик этого события получает информацию о нажатой клавише и состоянии клавиш Shift, Alt и Ctlr, а также о нажатой кнопке мыши. Информация о клавише передается параметром Key, который имеет тип Word
OnKeyUp	Является парным событием для OnKeyDown и возникает при отпускании ранее нажатой клавиши
OnClick	Возникает при нажатии кнопки мыши в области компонента
OnDblClick	Возникает при двойном нажатии кнопки мыши в области компонента

5.6. Порядок выполнения индивидуального задания

Задание: написать программу подсчета числа слов в произвольной строке. В качестве разделителя может быть любое число пробелов. Для ввода строк и работы с ними использовать TComboBox. Ввод строки заканчивать нажатием клавиши Enter. Для выхода из программы использовать кнопку Close.

Панель диалога будет иметь вид (рис. 5.1).

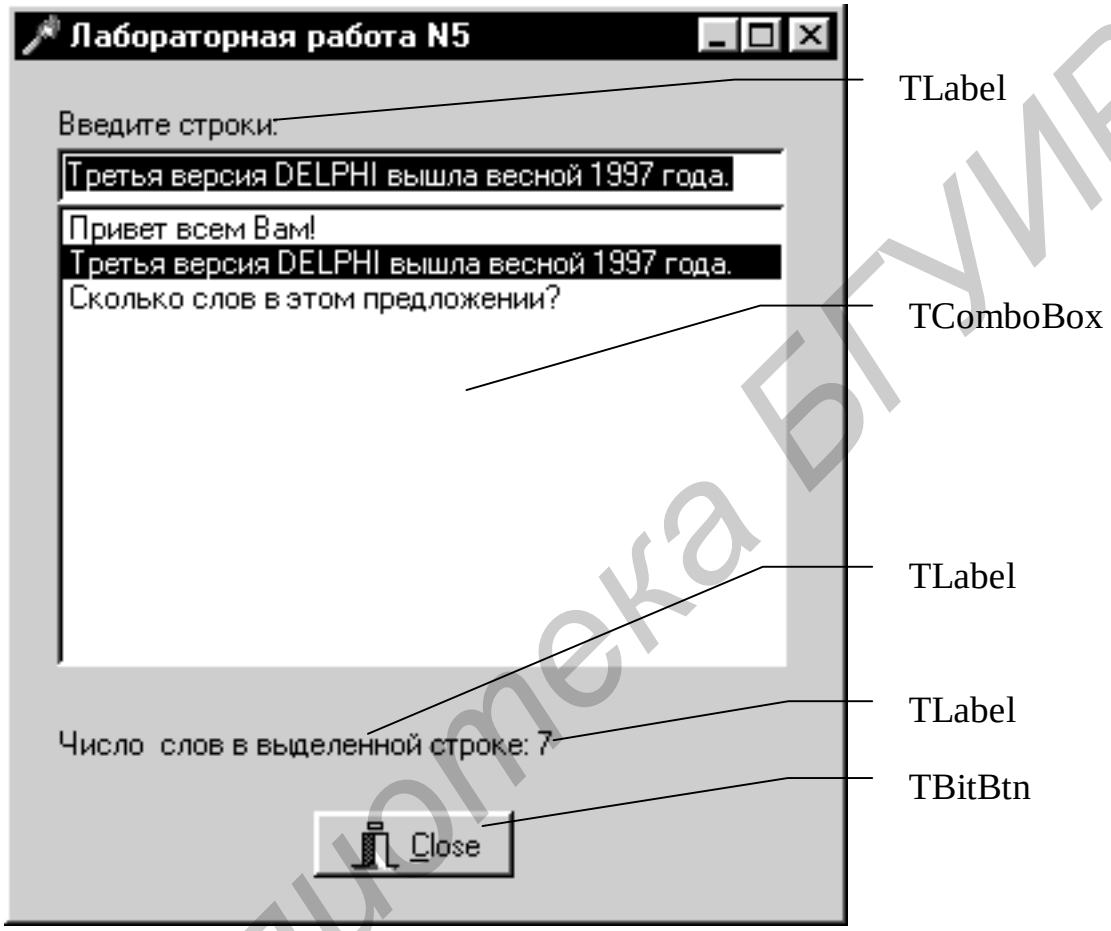


Рис. 5.1

Текст программы:

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormActivate(TObject *Sender)
{
Form1->ComboBox1->SetFocus();      // Передача фокуса ComboBox1
}
//---------------------------------------------------------------------------
   // Обработка события нажатия клавиши на клавиатуре
void __fastcall TForm1::ComboBox1KeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
   if (Key==13)
{
ComboBox1->Items->Add(ComboBox1->Text);  // Строка из окна
                                         //  редактирования  заносится
                                         // в ComboBox1
  ComboBox1->Text=""; // Очистка окна редактирования
}
}
   // Обработка события нажатия левой клавиши мыши
void __fastcall TForm1::ComboBox1Click(TObject *Sender)
{
int n,i,nst;
nst=ComboBox1->ItemIndex;         // Определение номера выбранной строки
String st=ComboBox1->Items->Strings[nst];    // Занесение выбранной строки в
                                              //  переменную st
if (st[1]!=' ') n=1; else n=0;

for(i=1;i<st.Length();i++)      // Просмотр всех символов строки st
if(st[i]==' '&&st[i+1]!=' ') n++;

Form1->Label3->Caption=IntToStr(n);   // Вывод числа слов в Label3
}
//---------------------------------------------------------------------------
```

5.7. Индивидуальные задания

Во всех заданиях исходные данные вводить с помощью компонента TEdit в компонент TListBox либо с помощью свойства Text в свойство Items компонента TComboBox. Скалярный результат выводить с помощью компонента TLabel. Ввод строки заканчивать нажатием клавиши Enter. Для выхода из программы использовать кнопку Close. Для расчетов вводить несколько различных строк.

1. Дана строка, состоящая из групп нулей и единиц. Каждая группа отделяется от другой одним или несколькими пробелами. Найти количество групп с пятью символами.

2. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран самую короткую группу.

3. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество символов в самой длинной группе.

4. Дана строка, состоящая из групп нулей и единиц. Найти и вывести на экран группы с четным количеством символов.

5. Дана строка, состоящая из групп нулей и единиц. Подсчитать количество единиц в группах с нечетным количеством символов.

6. Дана строка, состоящая из букв, цифр, запятых, точек, знаков “+” и “-“. Выделить подстроку, которая соответствует записи целого числа (т.е. начинается со знака “+” или “-“ и внутри которой нет букв, запятых и точек).

7. Дана строка символов, состоящая из букв, цифр, запятых, точек, знаков “+” и “-“. Выделить подстроку, которая соответствует записи вещественного числа с фиксированной точкой.

8. Дана строка символов, состоящая из букв, цифр, запятых, точек, знаков “+” и “-“. Выделить подстроку, которая соответствует записи вещественного числа с плавающей точкой.

9. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести на экран числа этой строки в порядке возрастания их значений.

10. Дана строка символов, состоящая из произвольных десятичных цифр, разделенных пробелами. Вывести четные числа этой строки.

11. Дана строка символов, состоящая из произвольного текста на английском языке, слова разделены пробелами. Вывести на экран слова этого текста в порядке, соответствующем латинскому алфавиту.

12. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Вывести на экран порядковый номер слова, накрывающего *k*-ю позицию (если на *k*-ю позицию попадает пробел, то номер предыдущего слова).

13. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Разбить исходную строку на две подстроки, причем первая длиной *k*-символов (если на *k*-ю позицию попадает слово, то его следует отнести ко второй строке, дополнив первую пробелами до *k*-позиции).

14. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Вывести на экран порядковый номер слова максимальной длины и номер позиции строки, с которой оно начинается.

15. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Вывести на экран порядковый номер слова минимальной длины и количество символов в этом слове.

16. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. В каждом слове заменить первую букву на прописную.

17. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Удалить первые *k* слов из строки, сдвинув на их место последующие.

18. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Поменять местами *i*-е и *j*-е слова.

19. Дана строка символов, состоящая из текста, слова которого разделены пробелами. Поменять местами первую и последнюю буквы каждого слова.

20. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Заменить буквы латинского алфавита на соответствующие им буквы русского алфавита.

21. Дана строка символов $S_1S_2 \ldots S_m$, в которой могут встречаться цифры, пробелы, буквы “E”, “e” и знаки “+”, “-“. Известно, что первый символ S_1 является цифрой. Из данной строки выделить подстроки, разделенные пробелами. Определить, является ли первая подстрока числом. Если да, то выяснить: целое или вещественное это число, положительное или отрицательное.

22. Дана строка символов, содержащая некоторый текст. Разработать программу форматирования этого текста, т.е. его разбиения на отдельные строки (по *k* символов в каждой строке) и выравнивания по правой границе путем вставки между отдельными словами необходимого количества пробелов.

23. Дана строка символов, содержащая некоторый текст на русском языке. Заменить буквы русского алфавита на соответствующие им буквы латинского алфавита.

24. Дана строка символов, содержащая некоторый текст. Разработать программу, которая определяет, является ли данный текст палиндромом, т.е. читается ли он справа налево так же, как и слева направо (например, «А роза упала на лапу Азора»).

25. Составить программу, которая читает построчно текст другой программы (ввести с клавиатуры) на языке *Pascal,* обнаруживает комментарии и выводит их на экран.

26. Составить программу, которая читает построчно текст другой программы (ввести с клавиатуры) на языке *Pascal*, подсчитывает количество ключевых слов «*begin*» и «*end*» и выводит на экран соответствующее сообщение.

27. Разработать программу, которая заданное целое число от 1 до 1999 выводит на экран римскими цифрами.

28. Дан текст из заглавных латинских букв, за которым следует пробел. Определить, является ли этот текст правильной записью римскими цифрами целого числа от 1 до 999, и, если является, вывести на экран это число арабскими цифрами(в десятичной системе).

29. Дан текст из *k* символов. Вывести на экран только строчные русские буквы, входящие в этот текст.

30. Дан текст из *k* символов. Вывести на экран в алфавитном порядке все различные прописные русские буквы, входящие в этот текст.

ТЕМА 6. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ СТРУКТУР

Цель лабораторной работы: Написать программу с использованием данных типа “структура”.

6.1. Программирование с использованием переменных типа “структура”

Структура – это тип данных, объединяющий элементы одикаковых или различных типов, называемых полями. Структуры удобны для создания баз данных с элементами разных типов, например:

```
typedef struct
 {
        char FIO[40];       // Поле ф.и.о.
        Byte otc[3];        // Поле массива оценок
       double sball;        // Поле среднего балла
        } TStudent;
    TStudent Stud;        //Объявление переменной типа  структура
```

Доступ к каждому полю может осуществляться посредством операции принадлежности, т.е. указанием имени структуры и поля, разделенных точкой, например:

```
Stud.FIO= “Иванов А.И.”;     //Внесение данных в поля записи
Stud.sball =5;
```

Организация доступа с использованием косвенной адресации выглядит следующим образом:

```
Stud->FIO= “Иванов А.И.”;   //Внесение данных в поля записи
Stud->sball =5;
```

6.2. Порядок выполнения задания

Задание: Дана ведомость абитуриентов, сдавших вступительные экзамены. Каждая запись содержит фамилию, оценки по физике, математике и сочинению. Вывести список абитуриентов, имеющих средний балл выше 4,5 в порядке уменьшения их среднего балла.

Форма приведена на рис. 6.1.

Текст программы:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
```

```
typedef struct {
    AnsiString FIO;
    int otc[3];
    double srb;
       } TStudent;
TStudent Stud[100], st;
int nzap=-1;

__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
Edit1->Text="";
Edit2->Text="";
Edit3->Text="";
Edit4->Text="";
Memo1->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
nzap++;
 Stud[nzap].FIO = Edit1->Text;
 Stud[nzap].otc[0]=StrToInt(Edit2->Text);
 Stud[nzap].otc[1]=StrToInt(Edit3->Text);
 Stud[nzap].otc[2]=StrToInt(Edit4->Text);
 Stud[nzap].srb=(Stud[nzap].otc[0]+Stud[nzap].otc[1]+Stud[nzap].otc[2])/3.0;
Memo1->Lines->Add(Stud[nzap].FIO+" "+IntToStr(Stud[nzap].otc[0])+" "
          +IntToStr(Stud[nzap].otc[1])+" "+IntToStr(Stud[nzap].otc[2]));
Edit1->Text="";
Edit2->Text="";
Edit3->Text="";
Edit4->Text="";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn2Click(TObject *Sender)
{
Memo1->Lines->Add("Result");
for (int i=0; i<=nzap-1; i++)
for (int j=i+1; j<=nzap; j++)
if (Stud[i].srb>Stud[j].srb)
```

```
                {
        st=Stud[i];
        Stud[i]=Stud[j];
        Stud[j]=st;
                }
for (int i=0; i<=nzap; i++)
 if (Stud[i].srb>4.5)
    Memo1->Lines->Add(Stud[i].FIO+" "+IntToStr(Stud[i].otc[0])+" "
                +IntToStr(Stud[i].otc[1])+" "+IntToStr(Stud[i].otc[2]));
}
```

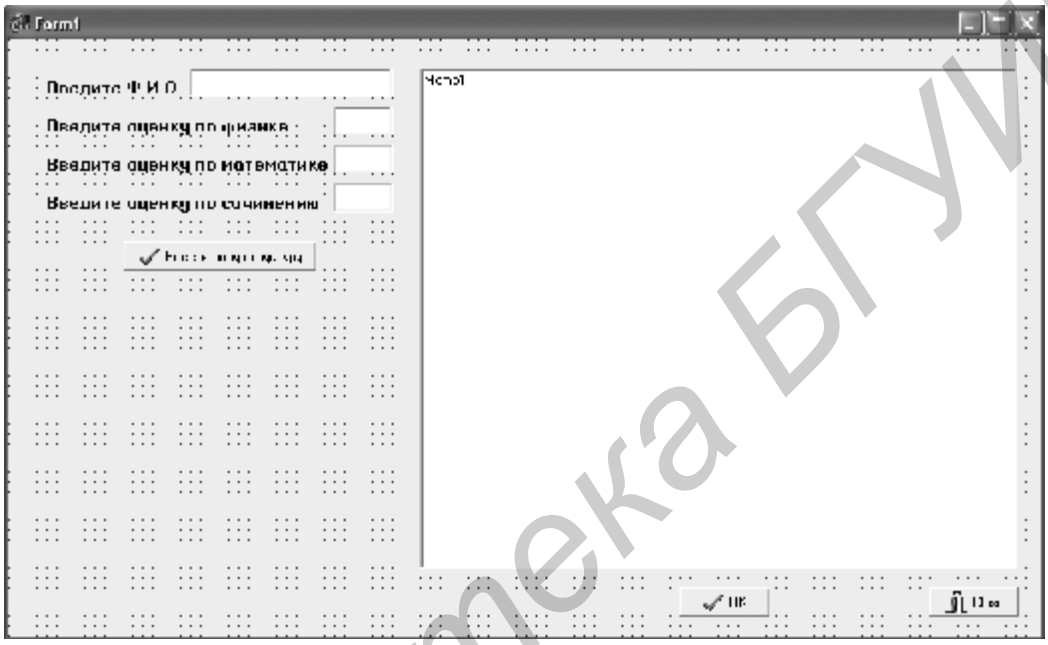


Рис. 6.1

6.3. Выполнение индивидуального задания

В программе предусмотреть сохранение вводимых данных в файле и возможность чтения из ранее сохраненного файла. Результаты выводить в окно просмотра и в текстовой файл.

1. В магазине формируется список лиц, записавшихся на покупку товара повышенного спроса. Каждая запись этого списка содержит: порядковый номер, Ф.И.О., домашний адрес покупателя и дату постановки на учет. Удалить из списка все повторные записи, проверяя Ф.И.О. и домашний адрес.

2. Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Вывести в алфавитном порядке список товаров, хранящихся больше месяца, стоимость которых превышает 10 000 р.

3. Для получения места в общежитии формируется список студентов, который включает Ф.И.О. студента, группу, средний балл, доход на члена семьи. Общежитие в первую очередь предоставляется тем, у кого доход на члена семьи

меньше двух минимальных зарплат, затем остальным в порядке уменьшения среднего балла. Вывести список очередности предоставления мест в общежитии.

4. В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны его номер, тип автобуса, пункт назначения, время отправления и прибытия. Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

5. На междугородной АТС информация о разговорах содержит дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента. Вывести для каждого города общее время разговоров с ним и сумму.

6. Информация о сотрудниках фирмы включает: Ф.И.О., табельный номер, количество проработанных часов за месяц, почасовой тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. Вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12% от суммы заработка.

7. Информация об участниках спортивных соревнований содержит: наименование страны, название команды, Ф.И.О. игрока, игровой номер, возраст, рост, вес. Вывести информацию о самой молодой, рослой и легкой команде.

8. Для книг, хранящихся в библиотеке, задаются: регистрационный номер книги, автор, название, год издания, издательство, количество страниц. Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.

9. Различные цеха завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают: наименование, количество, номер цеха. Для заданного цеха необходимо вывести в порядке убывания количество выпущенных изделий по каждому наименованию.

10. Информация о сотрудниках предприятия содержит: Ф.И.О., номер отдела, должность, дату начала работы. Вывести списки сотрудников по отделам в порядке убывания стажа.

11. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: Ф.И.О., адрес, оценки. Определить количество абитуриентов, проживающих в г.Минске и сдавших экзамены со средним баллом не ниже 4.5, вывести их фамилии в алфавитном порядке.

12. В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны: номер рейса, тип самолета, пункт назначения, время вылета. Вывести все номера рейсов, типы самолетов и время вылета в заданный пункт назначения в порядке возрастания времени вылета.

13. У администратора железнодорожных касс хранится информация о свободных местах в поездах дальнего следования на ближайшую неделю в следующем виде: дата выезда, пункт назначения, время отправления, число свободных мест. Оргкомитет международной конференции обращается к администратору с просьбой зарезервировать m мест до города N на k-й день недели с временем отправления поезда не позднее t часов вечера. Вывести время отправления или сообщение о невозможности выполнить заказ в полном объеме.

14. Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: Ф.И.О. абитуриента, оценки. Определить средний балл по университету и вывести список абитуриентов, средний балл которых выше среднего балла по университету. Первыми в списке должны идти студенты, сдавшие все экзамены на 5.

15. В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий(телевизор, радиоприемник и т. п.),марку изделия, дату приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести информацию о состоянии заказов на текущие сутки по группам изделий.

16. Разработать программу формирования ведомости об успеваемости студентов. Каждая запись этой ведомости должна содержать: номер группы, Ф.И.О. студента, оценки за последнюю сессию. Вывести списки студентов по группам. В каждой группе Ф.И.О. студентов должны быть расположены в порядке убывания среднего балла.

17. В исполкоме формируется список нуждающихся в улучшении жилищных условий. Каждая запись этого списка содержит: порядковый номер, Ф.И.О., величину жилплощади на одного члена семьи и дату постановки на учет. Исходя из заданного количества квартир, выделяемых ежегодно, вывести весь список с указанием ожидаемого года получения квартиры.

18. Имеется список женихов и список невест. Каждая запись списка содержит пол, имя, возраст, рост, вес, а также требования к партнеру: наименьший и наибольший возраст, наименьший и наибольший вес, наименьший и наибольший рост. Объединить эти списки в список пар с учетом требований к партнерам без повторений.

19. В библиотеке имеется список книг. Каждая запись этого списка содержит: фамилии авторов, название книги, год издания. Вывести информацию о книгах, в названии которых встречается некоторое ключевое слово (ввести с клавиатуры).

20. В магазине имеется список поступивших в продажу автомобилей. Каждая запись этого списка содержит: марку автомобиля, стоимость, расход топлива на 100 км, надежность (число лет безотказной работы), комфортность (отличная, хорошая, удовлетворительная). Вывести перечень автомобилей, удовлетворяющих требованиям покупателя, которые вводятся с клавиатуры в виде некоторого интервала допустимых значений.

21. Каждая запись списка вакантных рабочих мест содержит: наименование организации, должность, квалификацию (разряд или образование), стаж работы по специальности, заработную плату, наличие социального страхования (да/нет), продолжительность ежегодного оплачиваемого отпуска. Вывести список рабочих мест в соответствии с требованиями клиента.

22. В технической службе аэропорта имеется справочник, содержащий записи следующей структуры: тип самолета, год выпуска, расход горючего на 1000 км. Для определения потребности в горючем техническая служба запрашивает расписание полетов. Каждая запись расписания содержит

следующую информацию: номер рейса, пункт назначения, дальность полета. Вывести суммарное количество горючего, необходимое для обеспечения полетов на следующие сутки.

23. Поля шахматной доски характеризуются записью

```
Tun
 Pole=структура {
        Ver:(a,b,c,d,e,f,g,h);  {вертикальные координаты}
        Hor:1..8;     }          {горизонтальные координаты}
```

Вывести шахматную доску, пометив крестиками все поля, которые «бьет» ферзь, стоящий на поле с координатами Ver_i и Hor_i, и ноликами - остальные поля.

24. Поля шахматной доски характеризуются записью (см. задание 23)

```
Pole      Figur:;
```

Вывести сообщение, может ли конь за один ход перейти с поля $Figura_i$ на поле $Figura_j$.

25.

```
Tun
Karta= структура {
        m: (piki,trefi,bubni,chervi);                    {масть}
        d:(shest,sem,vosem,devjat,desjat,valet,dama,korol,tuz);  {достоинство}
      } k1,k2;
```

Вывести сообщение, «бьет» ли карта *k1* карту *k2*, с учетом того что масть m_i является козырной.

26. Для участия в конкурсе на замещение вакантной должности сотрудника фирмы желающие подают следующую информацию: Ф.И.О., год рождения, образование (среднее, специальное, высшее), знание иностранных языков (английский, немецкий, французский; владею свободно, читаю и перевожу со словарем), владение компьютером (MSDOS, Windows), стаж работы, наличие рекомендаций. Вывести список претендентов в соответствии с требованиями руководства фирмы.

27. При постановке на учет в ГАИ автолюбители указывают следующие данные: марка автомобиля, год выпуска, номер двигателя, номер кузова, цвет, номерной знак, Ф.И.О и адрес владельца. Вывести список автомобилей, проходящих техосмотр в текущем году, сгруппированных по маркам автомобилей. Учесть, что если текущий год четный, техосмотр проходят автомобили с четными номерами двигателей, иначе – с нечетными номерами.

28. Для участия в конкурсе исполнителей необходимо заполнить следующую анкету: Ф.И.О., год рождения, название страны, класс музыкального инструмента (гитара, фортепиано, скрипка, виолончель). Вывести список самых молодых лауреатов конкурса по классам инструментов в порядке занятых мест.

29. Список группы студентов содержит следующую информацию: Ф.И.О., рост и вес. Вывести Ф.И.О. студентов, рост и вес которых чаще всего встречаются в списке.

30. Список группы студентов содержит следующую информацию: Ф.И.О., рост и вес. Вывести Ф.И.О. студентов, рост и вес которых являются в списке уникальными.

ТЕМА 7. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ФАЙЛОВ

Цель лабораторной работы: изучить правила работы с компонентами TOpenDialog и TSaveDialog. Написать программу с использованием файлов и данных типа “структура”.

7.1. Работа с файлами

В языках C и C++ файл рассматривается как поток (stream), представляющий собой последовательность считываемых или записываемых байтов. При этом последовательность записи определяется самой программой.

C++ Builder позволяет работать с файлами тремя различными способами:

1) работа в стиле C;
2) работа в стиле C++;
3) использование библиотечных компонентов.

7.1.1. Работа с файлами в стиле C

Каждый файл в программе на C++ должен быть связан с некоторым указателем. Этот указатель имеет тип FILE (определен в stdio.h) и используется во всех операциях с файлами.

Синтаксис операции следующий:

```
# include < stdio.h >
        …
FILE *fin, *fout;
```

Для работы с файлом его необходимо открыть функцией **fopen**, первый параметр которой содержит имя файла и путь к нему, а второй – режим открытия файла. Функция возвращает указатель на файл. Часто используемые режимы открытия файла:

r	Открывает файл для чтения
r+	Открывает существующий файл для чтения и записи
w	Создает файл для записи. Если файл уже существует, его содержимое уничтожается
w+	Создает файл для чтения и записи. Если файл уже существует, его содержимое уничтожается
a	Открывает файл для записи данных в конец файла. Если файл отсутствовал, он создается
a+	Открывает файл для чтения или записи данных в конец файла. Если файл отсутствовал, он создается

После режима может добавляться символ “t” – текстовый файл или “b” – бинарный файл. Если символ не указан, то по умолчанию считается что файл текстовый. Например:

```
fin=fopen(“a:\\dat.txt”,”r”);
fout=fopen(“a:\\out.txt”,”w”);
```

При записи обмен происходит не непосредственно с файлом, а с некоторым буфером. Информация из буфера переписывается в файл только при его переполнении или при закрытии файла.

После окончания работы с файлом он обязательно должен быть закрыт функцией **fclose(FILE *).**

Если файл не удалось открыть, то возвращается нулевой указатель (NULL). Например:

```
# include < stdio.h >
        …
    FILE *lw;
        if ((lw=fopen("a.text","r")) == NULL)
        {
        Memo1->Lines->Add("Файл не открыт");
            return;
        }
        …
  fclose(lw);
```

7.1.1.1. Работа с текстовыми файлами

Для записи в текстовый файл наиболее часто используется функция **fprintf**:

int *fprintf(FILE *stream, const char *format[]);

где параметр format определяет строку форматирования аргументов, заданных своими адресами. Обычно эта строка состоит из последовательности символов “%”, после которых следует символ типа данных:

I или i	Десятичное, восьмеричное или шестнадцатеричное целое
D или d	Десятичное целое
U или u	Десятичное целое без знака
E или e	Действительное с плавающей точкой
s	Строка символов
c	Символ

Из отрытого текстового файла можно читать информацию как по строкам, так и посимвольно. Чтение строки осуществляется функцией **fgets** :

char *fgets(char *st, int n, FILE *stream);

где st – указатель на буфер, в который считается строка; n – число читаемых символов; stream – указатель на файл. Строка читается до тех пор, пока не будет прочитано n-1 символов, или до конца строки \n. В конце прочитанной строки ставится нулевой символ.

Для проверки достижения конца файла используется функция **feof(F)**.

Чтение форматированных данных можно осуществлять с помощью функции **fscanf**:

int *fscanf(FILE *stream, const char *format[]);

Строка форматирования строится аналогично fprintf.

Следует обратить внимание на то, что при чтении данных всегда указываются адреса переменных (&), а не сами переменные.

Пример записи и чтения данных из файла:

```
# include < stdio.h >
        …
Memo1->Clear();

 FILE *lw;

 // Запись данных в файл
if ((lw=fopen("a.text","wt")) == NULL)
        {
                Memo1->Lines->Add("Файл не удалось создать");
                return;
        }
 int num=10;
 char st[12] = "ИНФОРМАЦИЯ",sr[30];
          fprintf(lw,"%s \n В группе %i человек",st, num);
fclose(lw);

 // Чтение данных из файла
if ((lw=fopen("a.text","rt")) == NULL)
        {
                Memo1->Lines->Add("Файл не удалось открыть ");
                return;
        }
  while (!feof(lw)) {
     fgets(sr,30,lw);
    if (sr[strlen(sr)-1] =='\n') sr[strlen(sr)-1]=0;
       Memo1->Lines->Add(sr) ;
          }
fclose(lw);
```

7.1.1.2. Работа с двоичными файлами

Двоичный файл представляет собой последовательность символов без разделителей. Порядок следования символов такой же, как они хранятся в оперативной памяти. Двоичные файлы имеют меньший объем, читаются и записываются быстрее. Единственный недостаток – трудность отладки. Ввод и вывод данных чаще всего производится функциями **fwrit**e и **fread**:

```
fwrite (dd,size,n,f);
fread (dd,size,n,f);
```

где dd – указатель на вводимые данные; size – размер передаваемых данных в байтах; n – число передаваемых данных; f – указатель на файл.

Пример работы с двоичным файлом:

```
# include < stdio.h >
        …
Memo1->Clear();

 FILE *lw;

 // Ввод данных в файл
if ((lw=fopen("a.text","wb")) == NULL)
 {
 Memo1->Lines->Add("Файл не удалось создать ");
 return;
 }
 int i,num=10;
 char s1[80] = "ИНФОРМАЦИЯ", s2[80], s3[80] ;

 fwrite (s1,sizeof(char),strlen(s1)+1,lw);
 fwrite ("в группе",sizeof(char),9,lw);
 fwrite (&num,sizeof(int),1,lw);
 fwrite ("человек",sizeof(char),8,lw);
fclose(lw);

 // Чтение данных из файла
if ((lw=fopen("a.text","rb")) == NULL)
 {
 Memo1->Lines->Add("Файл не удалось открыть ");
 return;
 }
for (i=0; i<=80; i++) {
                fread (s1+i, sizeof(char),1,lw);
                 if (s1[i]=='\0') break;
                        }
 for (i=0; i<=80; i++) {
                fread (s2+i, sizeof(char),1,lw);
                  if (s2[i]=='\0') break;
                        }
 fread (&num,sizeof(int),1,lw);
 for (i=0; i<=80; i++) {
                fread (s3+i, sizeof(char),1,lw);
                  if (s3[i]=='\0') break;
                        }       Memo1->Lines->Add(s1);
       Memo1->Lines->Add(s2);
       Memo1->Lines->Add(IntToStr(num));
       Memo1->Lines->Add(s3);
fclose(lw);
```

В приведенном примере чтение проводилось последовательно. Узнать текущую позицию указателя можно с помощью функции **ftell**. Для произвольного чтения данных можно перемещать указатель в произвольную позицию с помощью функции **fseek**.

fseek(F, set, nn)

где F- указатель на файл; set – число байт, на которое производится сдвиг от точки отсчета; nn – точка отсчета (0 – начало файла; 1 – текущая позиция; 2 – конец файла).

Наиболее полезна эта функция в файлах, состоящих из записей одного размера.

7.1.2. Работа с использованием дескрипторов

В начале работы любой программы автоматически открываются три потока со своими дескрипторами (входной – клавиатура, выходной – экран и поток сообщений об ошибках). Программно можно открывать новые файлы с дескрипторами. Правила работы практически идентичны работе с двоичными файлами и различаются только синтаксисом.

7.1.3. Работа с файлами в стиле C++

В C++ определены три класса файлового ввода/вывода: **ifstream** – входные данные для чтения; **ofstream** – выходные файлы для записи; **fstream** – файлы для чтения и записи.

Очень удобно применять следующие операции: поместить в поток (<<) и извлечь из потока (>>).

Пример программы:

```
#include <fstream.h>
        …
Memo1->Clear();

 // Ввод данных в файл

 ofstream lw ("a.text");
if (!lw)
 {
 Memo1->Lines->Add("Файл не удалось создать ");
 return;
 }
 int num=10;
 double k=5.67;
 char s[20] = "ИНФОРМАЦИЯ";

 lw << num << ' ' <<k << ' ' << s << endl;
 lw.close();
```

```
// Чтение данных из файла
 ifstream lx ("a.text");
if (!lx)
 {
 Memo1->Lines->Add("Файл не удалось открыть ");
 return;
 }
 lx >> num >> k >> s;
      Memo1->Lines->Add(IntToStr(num));
      Memo1->Lines->Add(FloatToStr(k));
       Memo1->Lines->Add(s);
 lx.close();
```

Помимо этих операций поместить в поток можно еще с помощью функций **put** и **write**. Возможности ввода/вывода можно существенно расширить, используя манипуляторы потока.

7.1.4. Работа с файлами с помощью компонентов

Работа с файлами осуществляется в помощью методов **LoadFromFile** и **SaveToFile**. Через компоненты можно работать не только с текстовыми файлами, но и с файлами мультимедиа и изображениями.

7.2. Компоненты TOpenDialog и TSaveDialog

Компоненты TOpenDialog и TSaveDialog находятся на странице DIALOGS. Все компоненты этой страницы являются невизуальными, т.е. не видны в момент работы программы. Поэтому их можно разместить в любом удобном месте формы. Оба рассматриваемых компонента имеют идентичные свойства и различаются только внешним видом. После вызова компонента появляется диалоговое окно, с помощью которого выбирается имя программы и путь к ней. В случае успешного завершения диалога имя выбранного файла и маршрут поиска содержатся в свойстве FileName. Для фильтрации файлов, отображаемых в окне просмотра, используется свойство Filter, а для задания расширения файла, в случае, если оно не задано пользователем, – свойство DefaultExt. Если необходимо изменить заголовок диалогового окна, используется свойство Title.

7.3. Порядок выполнения задания

Задание: написать программу, вводящую в файл или читающую из файла ведомость абитуриентов, сдавших вступительные экзамены. Каждая запись должна содержать фамилию, а также оценки по физике, математике и сочинению. Вывести список абитуриентов, отсортированный в порядке уменьшения их среднего балла, и записать эту информацию в текстовой файл.

7.3.1. Настройка компонентов TOpenDialog и TSaveDialog

Для установки компонентов TOpenDialog и TSaveDialog на форму необходимо на странице Dialogs меню компонентов щелкнуть мышью по

пиктограммаме  или  и поставить её в любое свободное место формы. Установка фильтра производится следующим образом. Выбрав соответствующий компонент, дважды щелкнуть по правой части свойства Filter инспектора объектов. Появится окно Filter Editor, в левой части которого записывается текст, характеризующий соответствующий фильтр, а в правой части – маска. Для OpenDialog1 установим значения маски, как показано на рис. 7.1. Формат *.dat

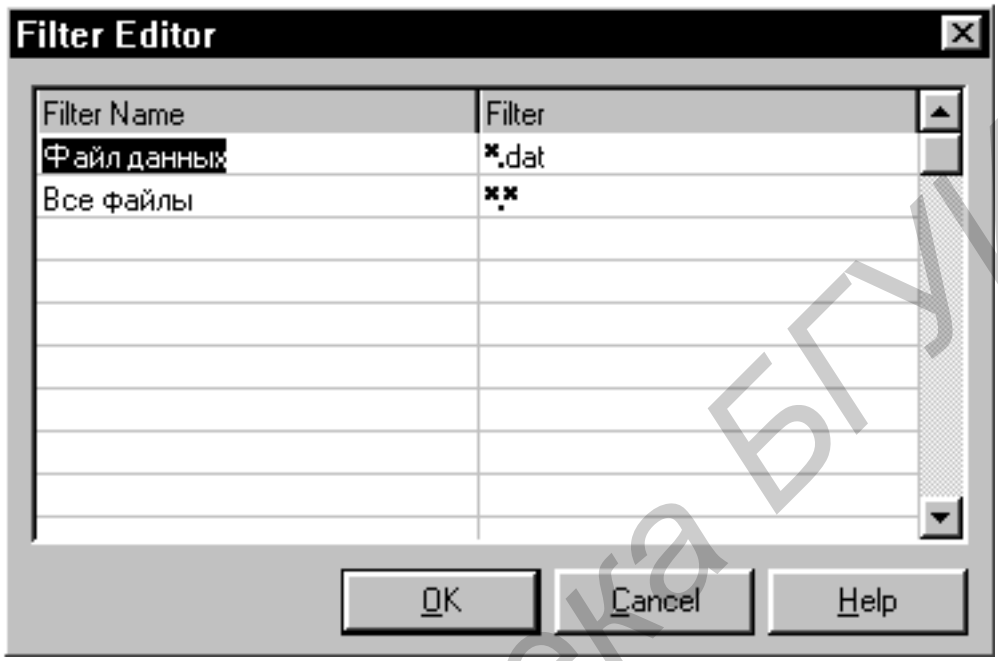


Рис. 7.1

означает, что будут видны все файлы с расширением dat, а формат *.* - что будут видны все файлы (с любым именем и с любым расширением).

Для того чтобы файл автоматически записывался с расширением .dat, в свойстве DefaultExt запишем требуемое расширение - .dat.

Аналогичным образом настроим SaveDialog1 для текстового файла (расширение .txt).

7.3.2. Работа с программой

После запуска программы на выполнение появится диалоговое окно программы. Кнопка “Ввести запись” видна не будет. Необходимо создать новый файл записей, нажав на кнопку “Создать”, или открыть ранее созданный, нажав на кнопку “Открыть”. После этого станет видна кнопка “Ввести запись” и можно будет вводить записи. При нажатии на кнопку “Сортировка” будет проведена сортировка ведомости по убыванию среднего балла и диалоговое окно примет вид, как на рис. 7.2. Затем при нажатии на кнопку “Сохранить” будет создан текстовой файл, содержащий отсортированную ведомость. Файл записей закрывается одновременно с программой при нажатии на кнопку “Close” или  .

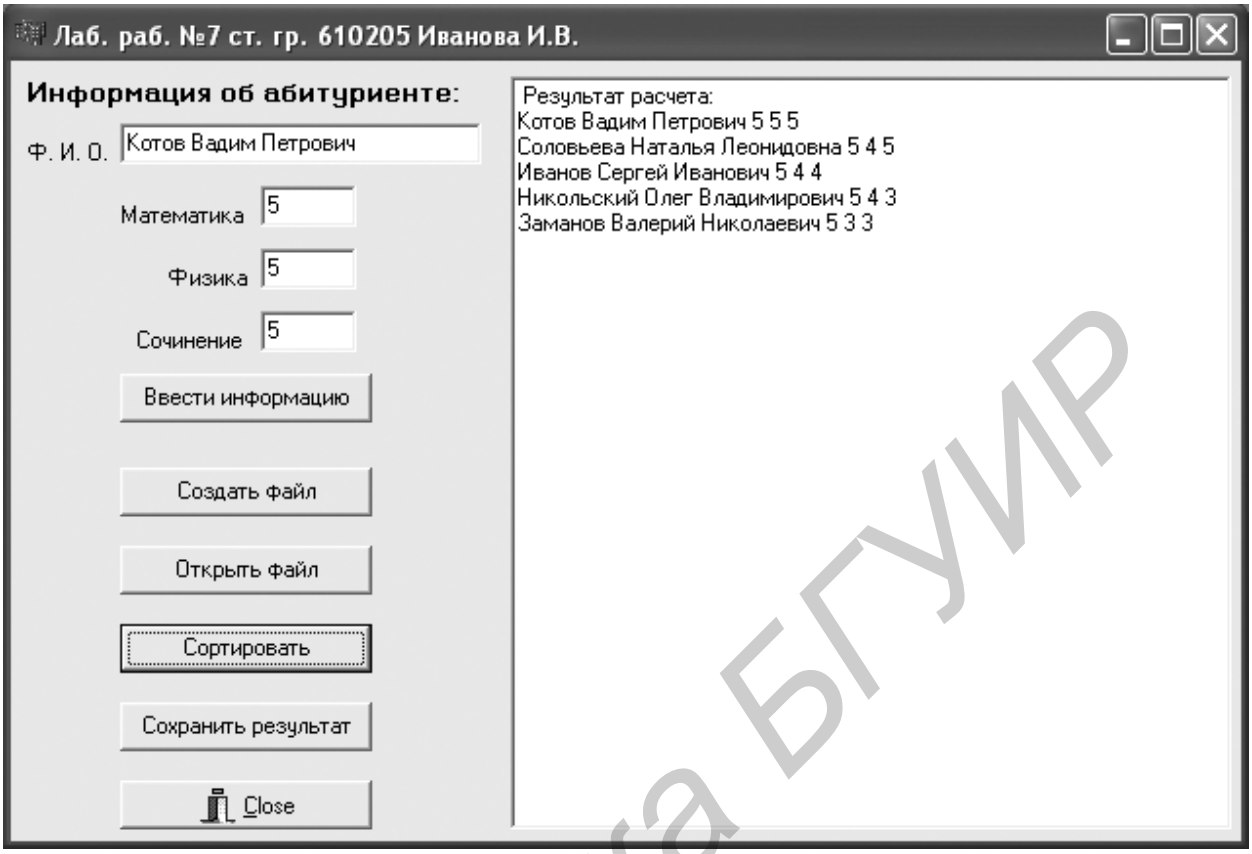


Рис. 7.2

Текст программы:

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Lab7F.h"
#include <stdio.h>;
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
typedef struct {
            char FIO[30];
              Byte otc[3];
            Single sball;
        } TStudent;

TStudent Stud[100];
int nzap=0;
```

```
FILE *Fs;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
Edit1->Clear();
Edit2->Clear();
Edit3->Clear();
Edit4->Clear();
RichEdit1->Clear();
Button1->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
OpenDialog1->Title="New File";
if (OpenDialog1->Execute())
{
 char *FileNameS= OpenDialog1->FileName.c_str();
   if ((Fs=fopen(FileNameS,"wb"))==NULL)  {
         ShowMessage("File no created");
         return;                    }
  Button1->Show();
}
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 strcpy(Stud[nzap].FIO,Edit1->Text.c_str());
Stud[nzap].otc[0]=StrToInt(Edit2->Text);
Stud[nzap].otc[1]=StrToInt(Edit3->Text);
Stud[nzap].otc[2]=StrToInt(Edit4->Text);
Stud[nzap].sball=(Stud[nzap].otc[0]+Stud[nzap].otc[1]+Stud[nzap].otc[2])/3.0;

RichEdit1->Lines->Add(AnsiString(Stud[nzap].FIO)+" "
      +IntToStr(Stud[nzap].otc[0])+" "
      +IntToStr(Stud[nzap].otc[1])+" "
      +IntToStr(Stud[nzap].otc[2])+" ");
```

```
fwrite(&Stud[nzap],sizeof(TStudent),1,Fs);
nzap++;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
OpenDialog1->Title="Open File";
if (OpenDialog1->Execute())
{
 char *FileNameS= OpenDialog1->FileName.c_str();
   if ((Fs=fopen(FileNameS,"rb"))==NULL)  {
        ShowMessage("File is not opened");
        return;                    }
  Button1->Show();

   RichEdit1->Clear();
   nzap=0;
 do
  {
  fread(&Stud[nzap],sizeof(TStudent),1,Fs);
   if (feof(Fs)) break;
RichEdit1->Lines->Add(AnsiString(Stud[nzap].FIO)+" "
     +IntToStr(Stud[nzap].otc[0])+" "
     +IntToStr(Stud[nzap].otc[1])+" "
     +IntToStr(Stud[nzap].otc[2])+" ");
           nzap++;
  }
  while(True);
  Button1->Show();
 }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button4Click(TObject *Sender)
{
 TStudent tmp;
for(int i=0;i<nzap-1;i++)
 for(int j=i;j<nzap;j++)
 if (Stud[i].sball<Stud[j].sball)
 {
 tmp=Stud[i];
 Stud[i]=Stud[j];
 Stud[j]=tmp;
  }
```

```
    RichEdit1->Clear();
    RichEdit1->Lines->Add(" Результат:");
 for(int i=0;i<nzap;i++)
      RichEdit1->Lines->Add(AnsiString(Stud[i].FIO)+" "
      +IntToStr(Stud[i].otc[0])+" "
      +IntToStr(Stud[i].otc[1])+" "
      +IntToStr(Stud[i].otc[2])+" ");
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button5Click(TObject *Sender)
{
SaveDialog1->Title="Save File";
if (SaveDialog1->Execute())
{
 AnsiString FileNameR = SaveDialog1->FileName;
 RichEdit1->Lines->SaveToFile(FileNameR);
 }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
 fclose(Fs);
}
```

7.4. Выполнение индивидуального задания

По указанию преподавателя выбрать вариант задания из темы 6. Предусмотреть запись исходных данных в файл и возможность чтения из него. Результат вывести на экран и в файл.

ТЕМА 8. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ УКАЗАТЕЛЕЙ. ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ

8.1. Объявление указателей

Указатель – это переменная, которая содержит адрес памяти, по которому можно найти значение другой переменной. Объявление указателя:

<тип> *<идентификатор>

При объявлении указателя необходимо указывать тип переменной, на которую он будет указывать. Каждая переменная, объявляемая как указатель, должна иметь перед своим именем символ “*”. Например:

```
int  *pA, B;     // *pA -  указатель на элемент целого типа;
                 // B – переменная целого типа
double *pC;   // *C -  указатель на переменную действительного типа
void *pD;      // *D – указатель на любой тип данных (перед применением
               // необходимо инициировать конкретным типом данных
```

Указатель обязательно должен быть инициирован либо при объявлении, либо во время выполнения программы (до его первого использования). В качестве значения указатель может получить либо конкретный адрес, либо 0 или NULL, последнее указывает на то, что указатель ни на что не указывает.

Для указателей определены 2 операции:

***** - операция косвенной адресации. Применяется для получения данных, расположенных по указанному адресу. Работает следующим образом: в области памяти, выделенной под указатель, содержится информация об адресе переменной объявленного типа. По этому адресу извлекается информация, которая затем используется для вычислений.

& - операция адресации. Применяется для получения адреса какой-либо переменной.

Пример:

```
int A=9, *pB;
 pB=&A;
*pB=44;
      Memo1->Lines->Add("A= "+IntToStr(A));
 A=28;
       Memo1->Lines->Add("*pB= "+IntToStr(*pB));
```

Результат:

```
A= 44
*pB= 28
```

8.2. Указатели на массив

Массивы и указатели в C++ тесно связаны и могут использоваться одинаковым образом. Имя массива является указателем на его первый элемент. Указатель также содержит адрес первого элемента массива. Указатели могут бить использованы при любых операциях с массивами. Например имеется объявление:

int A[5] = [1,2,3,4,5], *pmas;

Тогда для того чтобы pmas указывал на массив, надо записать:

pmas=A; или pmas=&A[0];

Обращение, например, к третьему элементу массива можно записать:

A[2] или *(pmas+2), или *(A+2)

Указатели можно индексировать так же, как и массивы. Все операции, которые можно проводить с массивами, можно применить к массивам указателей. Массивы указателей, как правило, применяются для работы с массивами данных.

Пример. Отсортировать массив по возрастанию, не перемещая элементы в памяти компьютера.

```
int mas[]={1,8,6,3,9};
 int *pmasmin[5], *px;

    for (int i=0; i<5; i++)
     pmasmin[i] = &mas[i];

  for (int i=0; i<4; i++)
   for (int j=i+1; j<5; j++)
    if (*pmasmin[i]> *pmasmin[j])
    {
     px=pmasmin[i];
     pmasmin[i]=pmasmin[j];
     pmasmin[j]=px;
    }
   for (int i=0; i<5; i++)    Memo1->Lines->Add(IntToStr(*pmasmin[i]));
```

Результат: 1 3 6 8 9.

Для двухмерных массивов используются указатели на указатели. Например

```
 int mas[5][5];
 int **pmas;
```

Обращение может проводиться: *(*(mas+i)+j).

С помощью массивов указателей также удобно работать с массивами строк. Например:

```
char *pmas[]={"One","Save","Men","Write","Builder"};
 char   *px;
```

```
for (int i=0; i<4; i++)
 for (int j=i+1; j<5; j++)
   if (*pmas[i]> *pmas[j])
  {
   px=pmas[i];
   pmas[i]=pmas[j];
   pmas[j]=px;
  }
 for (int i=0; i<5; i++)
        Memo1->Lines->Add(pmas[i]);
```

Результат: Builder Men One Save Write.

Как видно из примера, каждый элемент массива является указателем на первый символ строки. При хранении в памяти каждая строка заканчивается нулевым символом. Число символов в строке может быть различным.

8.3. Особенности применения указателей

При работе с указателями можно использовать следующие операции: сложение, вычитание, операции сравнения и отношения. Однако имеются особенности их применения.

Операции сложения и вычитания. Эти операции отличаются тем, что например, прибавление единицы означает прибавление к указателю числа байт, содержащихся в переменной, на которую он указывал. Операции сложения и вычитания имеют смысл только для массивов, где данные располагаются в памяти строго друг за другом. Применение этих операций для разнотипных указателей может привести к непредсказуемым результатам.

Операции сравнения. Так же как и операции сложения и вычитания имеют смысл только для однотипных массивов.

Операции отношения (== и !=). Имеют смысл для любых указателей. Если два указателя равны между собой, то они указывают на одну и ту же переменную.

Указатели можно присваивать друг другу только в случае, если они имеют одинаковый тип (исключение – указатель на void).

8.4. Динамическое размещение данных

Динамическое размещение данных используется в случаях, когда число необходимых для работы объектов заранее неизвестно и задается непосредственно во время выполнения программы, либо когда некоторые объекты используются только в определенном месте программы и далее становятся ненужными. При этом достигается значительная экономия вычислительных ресурсов.

Для динамически размещаемых данных выделяется специальная область памяти – **heap**. Имеются следующие функции для работы с динамической памятью (находятся в файлах **stdlib.h alloc.h**):

void *malloc(size) – выделяет блок памяти размером **size** байт. Если выделение прошло удачно, то функция возвращает указатель на выделенную область, иначе – возвращается **NULL**.

void *calloc(n, size) – выделяет **n** блоков памяти, каждый размером **size** байт. Если выделение прошло удачно, то функция возвращает указатель на выделенную область, иначе – возвращается **NULL**.

void *realloc(*bl, size) – изменяет размер ранее выделенного блока памяти с адресом ***bl** на новый размер, равный **size** байт. Если изменение прошло удачно, то возвращает указатель на выделенную область, иначе – возвращается **NULL**. Если ***bl** равен **NULL**, то функция работает так же, как и функция **malloc**. Если **size** равен нулю, то выделенный по адресу ***bl** блок памяти освобождается, и функция возвращает **NULL**.

void free(*bl) – освобождает ранее выделенный блока памяти с адресом ***bl.**

Пример динамического выделения освобождения памяти для двухмерного массива:

```
int  **pA;
    pA=(int **) calloc (N,sizeof(int*));
 for (i=0; i<N; i++)
         pA[i]=(int *) calloc (M,sizeof(int));
 …

 for (i=0; i<N; i++) free(pA[i]);
    free(pA);
```

Другим, более предпочтительным подходом к динамическому распределению памяти является использование операций **new** и **delete**:

void *объект new void – возвращает указатель на динамически размещенный объект;

delete объект – освобождение памяти.

Например:

```
double *M = new double;
double K = *new double;
double *A = new double[100];
 . . .
delete M;
delete K;
delete [] A;
```

8.5. Порядок выполнения задания

Задание: написать программу, которая определяет, является ли заданная матрица N-го порядка магическим квадратом. Магический квадрат – это матрица, в которой сумма элементов во всех столбцах и во всех строках одинакова. Элементы матрицы необходимо разместить в памяти динамически.
Панель диалога приведена на рис. 8.1.

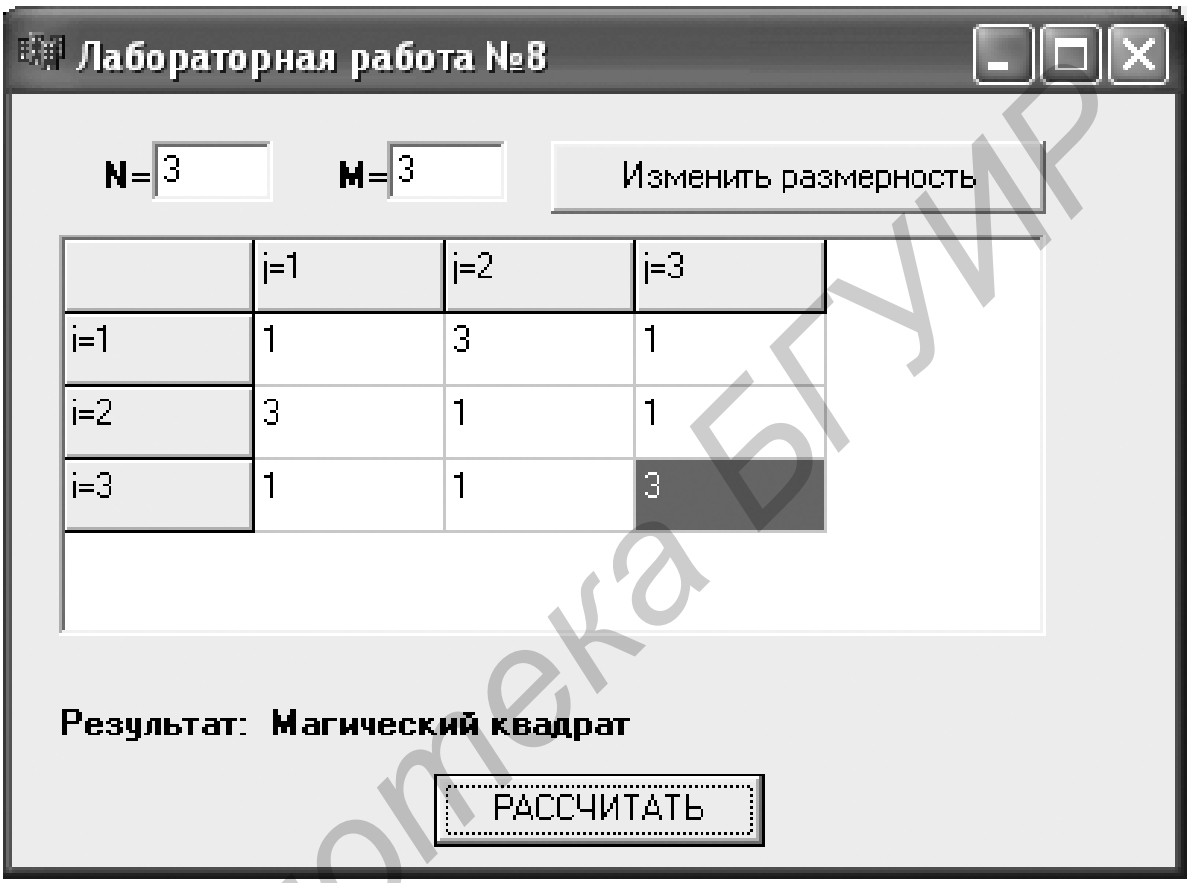


Рис. 8.1

Текст программы:

```
#include <vcl.h>
#pragma hdrstop

#include "lr7.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

int  **pA;
int N,M;
```

```
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 N=3; M=3;
Edit1->Text="3";
Edit2->Text="3";
 StringGrid1->RowCount=N+1;
 StringGrid1->ColCount=M+1;
  for (int i=1; <=N; i++){
  StringGrid1->Cells[i][0]="j="+IntToStr(i);
  StringGrid1->Cells[0][i]="i="+IntToStr(i);
                              }
 Label4->Caption="";
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
 N=StrToInt(Edit1->Text);
 M=StrToInt(Edit2->Text);
 StringGrid1->RowCount=N+1;
 StringGrid1->ColCount=M+1;
   for (int i=1;i<=N;i++) StringGrid1->Cells[0][i]="i="+IntToStr(i);
   for (int i=1;i<=M;i++) StringGrid1->Cells[i][0]="j="+IntToStr(i);
 }
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 int i,j,s1=0, sn;
       pA = new int*[N];
      for (i=0; i<N; i++)
                        pA[i] = new int[M];
 for (i=0;i<N;i++)
   for (j=0;j<M;j++)
                        pA[i][j]=StrToInt(StringGrid1->Cells[j+1][i+1]);
     bool met=True;
  for (i=0;i<N;i++)
                    s1+=pA[i][1];
   for (i=0;i<N && met;i++) {
```

```
    sn=0;
   for (j=0; j<M; j++) sn+=pA[i][j];
    if (s1!=sn) met=False;
                                    }
   for (j=0; j<M && met; j++)   {
     sn=0;
   for (i=0; i<N; i++)
                    sn+=pA[i][j];
    if (s1!=sn) met=False;
                                     }
if (met) Label4->Caption="Магический квадрат";
    else Label4->Caption="Немагический квадрат";
  for (i=0;i<N;i++)
                delete[] pA[i];
        delete [] pA;
}
```

8.6. Индивидуальные задания

По указанию преподавателя выберите вариант задания из темы 4. Для размещения массива использовать динамическое выделение памяти.

ТЕМА 9. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ПОДПРОГРАММ И МОДУЛЕЙ

Цель лабораторной работы: изучить возможности C++ Builder для написания подпрограмм и создания модулей. Составить и отладить программу, использующую внешний модуль с подпрограммой.

9.1. Использование подпрограмм

Подпрограмма – это именованная, определенным образом оформленная группа операторов, которая может быть вызвана любое количество раз из любой точки основной программы.

Подпрограммы используются в том случае, когда одна и та же последовательность операторов в тексте программы повторяется несколько раз. Эта последовательность заменяется вызовом подпрограммы, содержащей необходимые операторы. Подпрограммы применяются также для создания специализированных библиотечных модулей, содержащих набор подпрограмм определенного назначения, для использования их другими программистами.

В языке C++ в качестве подпрограмм используют функции. Функция должна быть объявлена (декларирована) до ее первого использования. Предварительное описание функции называется прототипом. Прототип обычно размещается в начале программы на C++ (расширение .cpp) либо в заголовочном файле (расширение .h) и сообщает компилятору о том, что далее в программе будет приведен полный текст функции.

Прототип имеет следующую структуру:

<тип_результата> <имя_функции> (<список параметров>);

Полное определение функции имеет следующий вид:

<тип_результата><имя_функции>(<список параметров>)
{
<тело функции>
return <результат>
}

Результат возвращается функцией в точку ее вызова при помощи оператора возврата **return** и соответствует типу, указанному в определении функции. Если тип функции не указан (**void**), то по умолчанию предполагается тип **int**. Список параметров состоит из перечня типов и имени параметров, разделенных запятыми. В том случае, если функция не имеет параметров, после имени располагаются только круглые скобки .

Функция передает только один параметр. Если необходимо возвратить несколько параметров, то следует в списке параметров использовать ссылки либо указатели. Если функция не возвращает параметров, то следует использовать пустой тип **void**.

Функции могут быть использованы в качестве формальных параметров подпрограмм. Для этого определяется тип-указатель на функцию:

typedef <тип_результата> (*<имя_указателя на функцию>)(<список типов>);

9.2. Использование модулей

В случае разработки сложных проектов можно часть функций поместить в другие программы и подключать их при необходимости.

Модуль – программная единица, включающая в себя файл программы с расширением (.cpp), где находится определение функций, и заголовочный файл с расширением (.h), где находятся прототипы функций, описание классов, объявления глобальных переменных.

Заголовочный файл подключается к файлу программы с помощью директивы препроцессора **#include**. По этой директиве препроцессор до компиляции программы добавляет содержимое заголовочного файла в файл программы.

9.3. Порядок выполнения задания

Задание: написать программу вывода на экран таблицы функции, которую оформить в виде процедуры. В качестве функции использовать по выбору Tg(x), ch(x) и $\sin^2(x)$.

9.3.1. Создание модуля

Для создания модуля в меню File выбрать New – Unit. В результате будет создан файл с заголовком Unit Unit1. Имя модуля можно сменить на другое, отвечающее внутреннему содержанию модуля, например MyUnit.

9.3.2. Подключение модуля

Для того чтобы подключить модуль к проекту, необходимо в меню Project выбрать опцию Add to Project… и выбрать файл, содержащий модуль. После этого подключить заголовочный файл модуля с помощью директивы препроцессора **#include**. Теперь в проекте можно использовать функции, содержащиеся в модуле.

Панель диалога будет иметь вид рис. 9.1.

Тексты модуля и вызывающей программы приведены ниже.

Текст заголовочного файла модуля:

```
//---------------------------------------------------------------------

#ifndef MyUnitH
#define MyUnitH
#include <StdCtrls.hpp>
//---------------------------------------------------------------------
typedef double (*TFun)(double);
void Tab(TMemo *mem,TFun f,double xn,double xk,double h);
#endif
```

Текст файла модуля:

```
#pragma hdrstop
#include "MyUnit.h"
#pragma package(smart_init)

void Tab(TMemo *mem,TFun f,double xn,double xk,double h)
{
double x,y;
x=xn;
while(x<xk)
{
y=(*f)(x);
mem->Lines->Add("x="+FloatToStrF(x,ffFixed,8,3)+
            " y="+FloatToStrF(y,ffFixed,8,3));
x+=h;
}
}
```

Текст вызывающей программы:

```
//---------------------------------------------------------------------------


#include <vcl.h>
#include <Math.h>
#include "MyUnit.h"

#pragma hdrstop
#include "lr9.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------

double cTg( double x); // Функция для вычисления котангенса
double Ch(double x); // Функция для вычисления гиперболического синуса
double Sin2(double x); // Функция для вычисления квадрата синуса
//---------------------------------------------------------------------------
```

```
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm1::FormCreate(TObject *Sender)
{
Edit1->Text="0,1";
Edit2->Text="3";
Edit3->Text="0,3";
Memo1->Clear();
RadioGroup1->ItemIndex=0;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
Memo1->Clear();
double xn=StrToFloat(Edit1->Text);
double xk=StrToFloat(Edit2->Text);
double h=StrToFloat(Edit3->Text);
switch(RadioGroup1->ItemIndex)
{
case 0:Tab(Memo1,cTg,xn,xk,h); break;
case 1:Tab(Memo1,Ch,xn,xk,h); break;
case 2:Tab(Memo1,Sin2,xn,xk,h); break;
}
}
double cTg( double x)
{
return cos(x)/sin(x);
}
```

```
double Ch(double x)
 {
return (exp(x)-exp(-x))/2;
}
double Sin2(double x)
{
return pow(sin(x),2);
}
```

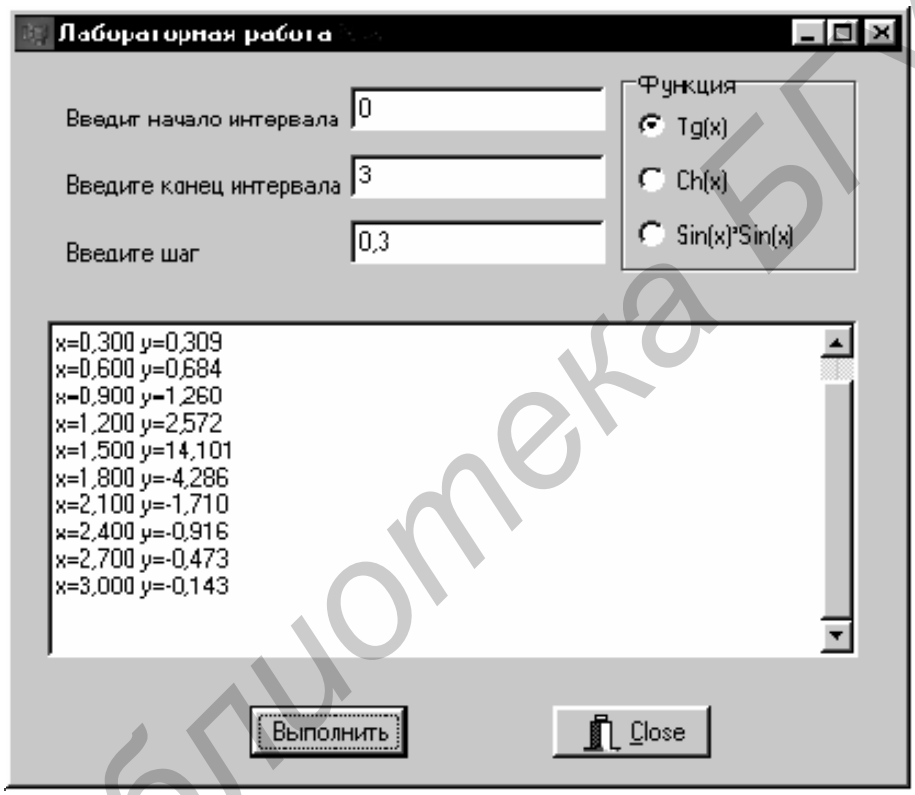


Рис. 9.1

9.4. Выполнение индивидуального задания

По указанию преподавателя выберите вариант задачи из заданий, приведенных в теме 3. Предусмотрите возможность выбора функции, для которой будет рассчитываться таблица. Функции поместите в отдельный модуль. Вызывать выбранную функцию должна процедура, использующая в качестве входного параметра имя соответствующей функции.

ТЕМА 10. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МЕХАНИЗМА ОБРАБОТКИ ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЙ

Цель лабораторной работы: изучить средства обработки исключительных ситуаций. Написать программу обработки данных, представленных в заданной системе счисления с использованием механизма обработки исключительных ситуаций.

10.1. Обработка исключительных ситуаций

Под исключительной ситуацией понимается некое ошибочное состояние, возникающее при выполнении программы, требующее выполнения определённых действий для продолжения работы программы или корректного ее завершения. Стандартный обработчик, вызываемый по умолчанию, информирует пользователя о возникновении ошибки и предпринимает определённые действия в связи с возникшей ошибкой. Для реализации необходимой создателю программы реакции на возникновение ошибки можно использовать такую синтаксическую структуру, которая перехватывает исключительную ситуацию и дает возможность разработчику обеспечить выполнение требуемых действий при ее возникновении.

До появления C++ Builder в Borland C++ выделяли два вида управления исключительными ситуациями:

управление исключительными ситуациями, характерное для языка C++;

структурное, или структурированное (structured exception) управление, характерное для языка C.

В C++ Builder добавились механизмы обработки исключительных ситуаций, обусловленных применением библиотеки визуальных компонентов (VCL — Visual Component Library).

Для обработки исключительных ситуаций, связанных с применением VCL и управлением исключительными ситуациями C++, следует использовать в общем случае конструкцию вида:

```
try
{<операторы>}
 catch (<описание исключительной ситуации E1>)
  {<операторы обработчика исключительной ситуации E1>}
catch (<описание исключительной ситуации E2>)
  {<операторы обработчика исключительной ситуации E2>}
                                        …
catch (<описание исключительной ситуации En>)
  {<операторы обработчика исключительной ситуации En>}
```

catch (…)
{<операторы для обработки не предусмотренной выше исключительной ситуации>**}**

Для каждой i-й исключительной ситуации, связанной с применением VCL, описание исключительной ситуации Ei **в общем случае** должно иметь вид:

const <имя класса исключительной ситуации Ei> **&Ei**

Наличие ключевого слова ***const*** и переменной ***Ei*** необязательно.

Некоторые наиболее распространенные классы исключительных ситуаций, связанных с применением VCL, представлены в табл. 10.1.

Таблица 10.1

Имя класса исключительной ситуации	**Причина возникновения исключительной ситуации**
EAbort	Намеренное прерывание выполнения программы
EAccessViolation	Ошибочная попытка обращения к недоступной области памяти
EConvertError	Ошибка преобразования данных одного типа в данные другого типа
EDivByZero	Попытка целочисленного деления на ноль
EIntOverflow	При выполнении операций с целыми числами получился слишком большой результат
EInOutError	Ошибки при файловом вводе/выводе
EZeroDivide	Деление на ноль числа с плавающей точкой

При обработке исключительных ситуаций C++ описание i-й исключительной ситуации следует в общем случае представлять в виде:

<имя типа данных, соответствующее исключительной ситуации Ei> ***Ei***

Указание переменной ***Ei*** необязательно. В качестве имени типа данных, соответствующего исключительной ситуации ***Ei***, может выступать ***int***, ***const char****, ***void****, ссылка на класс, определённый пользователем, и.т.д.

Реализация механизмов обработки исключительных ситуаций, характерных для C (которые, разумеется, широко применяются в текстах программ на C++), осуществляется с помощью одной из следующих конструкций:

try
{операторы**}**
__except(<фильтр>**)**
{обработка исключительной ситуации **}**

try
{операторы**}**
__finally
{операторы завершения работы блока try**}**

С помощью конструкции **try … __except** реализуется так называемое ***кадрированное управление***, а с помощью **try…__finally** — ***завершающее управление*** исключительными ситуациями. Операторы, следующие за ключевым словом **try**, заключённые в фигурные скобки, получили название ***тела защищённого кода***.

Во всех указанных выше конструкциях ключевое слово **try** предназначено для указания на начало блока, в который помещают операторы, выполнение которых может привести к возникновению исключительной ситуации.

Когда возникает необходимость в самостоятельной обработке исключительной ситуации, то применяются конструкции **try … catch** и **try … __except**. Если исключительная ситуация не возникает, то выполняются только операторы, расположенные внутри фигурных скобок, стоящих после слова **try**. В этом случае операторы, расположенные после **catch** и **__except** не выполняются. При возникновении исключительной ситуации выполнение сразу передаётся в один из соответствующих блоков **catch** или соответственно в блок **__except** (при определённом значении фильтра), которые содержат операторы, определяющие реакцию приложения на ту или иную исключительную ситуацию. В случае конструкции **try…catch**, если программа не находит обработчика, соответствующего возникшей исключительной ситуации, то выполняются операторы, находящиеся в блоке **catch**, в котором в качестве описания исключительной ситуации указано многоточие **(…)**.

Определённые отличия имеются в работе конструкции **try…__finally**.

Если исключительная ситуация не возникает, то сначала выполняются операторы ***тела защищённого кода***, а затем – операторы внутри блока **__finally** (так называемый ***завершающий блок***). При возникновении исключительной ситуации выполнение сразу передается в блок **__finally,** который обычно используется для корректного завершения программы, т.е. закрытия файлов, очистки динамически выделенной памяти и т.д.

В конструкции **try…__except** фильтр, или точнее ***фильтрующее выражение*** может принимать одно из трёх возможных значений: 1)*EXCEPTION_EXECUTE_HANDLER* (такое значение фильтра указывает, что управление должно быть передано соответствующему обработчику исключительной ситуации); 2) *EXCEPTION_CONTINUE_SEARCH* (в этом случае не будет происходить выполнение того обработчика, который непосредственно связан с данным **__except**, а будет осуществлён поиск иного обработчика); 3) *EXCEPTION_CONTINUE_EXECUTION* (это значение фильтра позволяет вновь вернуть управление выполнением программы в место, где была заявлена исключительная ситуация). В круглых скобках после **__except** может находиться либо одна из данных констант, либо функция, возвращающая в качестве результата такую константу.

Следует отметить, что интегрированная среда разработки C++ Builder сама отслеживает и обрабатывает возникающие исключительные ситуации. Для отладки программы, содержащей собственную обработку исключительных

ситуаций, надо отключить опции **Stop on Delphi Exceptions** и **Stop on C++ Exceptions,** находящиеся в **Tools – Debbuger Options …** , закладка **Language Exceptions**.

Возникновение исключительной ситуации может быть инициировано преднамеренно. Для этого можно использовать ключевое слово **throw** или функцию **RaiseException()**. Ключевое слово **throw** позволяет “выбросить” (если применять терминологию языка C++) исключение. За словом **throw** может следовать операнд (если это требуется)**.** Он определяет тип генерируемой исключительной ситуации, а следовательно, и то, какой обработчик будет обрабатывать исключительную ситуацию. Допустимо использовать как встроенные типы языка C++, так и определённые пользователем. Ключевое слово **throw** используется и для генерации исключительных ситуаций, связанных с применением VCL. Для того чтобы “заявить” (по терминологии языка C++) исключительную ситуацию, можно использовать функцию

RaiseException (< Код исключительной ситуации >**,**
<тип исключительной ситуации>**,**
<количество элементов в массиве аргументов>**,**
<адрес массива аргументов>**)**

Первые три параметра имеют тип **DWORD**, четвёртый — **const DWORD***/

В качестве типа исключительной ситуации указывается либо константа *EXCEPTION_CONTINUABLE*, либо *EXCEPTION_NONCONTINUABLE*, чем определяется, возобновима или нет исключительная ситуация.

10.2. Системы счисления

Под позиционной системой счисления понимают способ записи чисел с помощью цифр, при котором значение цифры определяется ее порядком в записи числа. Число R в p-ичной системе счисления можно представить в развернутом виде

$$R = a_n a_{n-1} \dots a_1 a_0, a_{-1} \dots a_{-k} = a_n p^n + a_{n-1} p^{n-1} + \dots + a_1 p^1 + a_0 p^0 + a_{n-1} p^{n-1} + \dots + a_{-k} p^{-k},$$

где a_i – цифры, p – основание системы счисления. Количество цифр равно p. Для записи цифр в общем случае может быть использован любой набор p символов. Обычно при $p \le 10$ используются символы $0-9$, для $p \ge 10$ добавляются буквы латинского алфавита A, B, C, D, E, F, которые в десятичной системе представляют числами 10, 11, 12, 13, 14, 15. Например,

$$R = (2A,B)_{16} = 2 \cdot 16^1 + A \cdot 16^0 + B \cdot 16^{-1} = 2 \cdot 16 + 10 \cdot 1 + 11/16 = (42,6875)_{10}$$

В компьютерной технике обычно используются системы с основанием, равным степени двойки: двоичная, восьмеричная и шестнадцатеричная. Имеются процессоры, реализующие троичную систему счисления. Для удобства пользователей ввод – вывод и операции над числами в компьютере производят в десятичной системе счисления.

При переводе числа из десятичной в другую систему счисления целая и дробная части числа переводятся различным образом.

При переводе целой части она делится на основание новой системы счисления, остаток представляет очередную цифру , а частное снова делится на основание. Процесс повторяется до тех пор, пока частное не станет равным нулю.

Заметим, что цифры получаются в порядке, обратном порядку их следования в записи числа.

При переводе дробной части она умножается на основание системы счисления. Целая часть полученного числа представляет очередную цифру , а дробная часть опять умножается на основание системы. Расчеты ведут до получения требуемого количества цифр.

10.3. Порядок выполнения индивидуального задания

Задание. Написать обучающую программу , позволяющую освоить операции сложения, вычитания, умножения, целочисленного деления и нахождения остатка от целочисленного деления для любых двух целых чисел, записанных в семнадцатеричной системе счисления. Для обозначения чисел использовать цифры от 0 до 9 и буквы от a до g (прописные или строчные).

Панель диалога будет иметь вид (рис. 10.1).

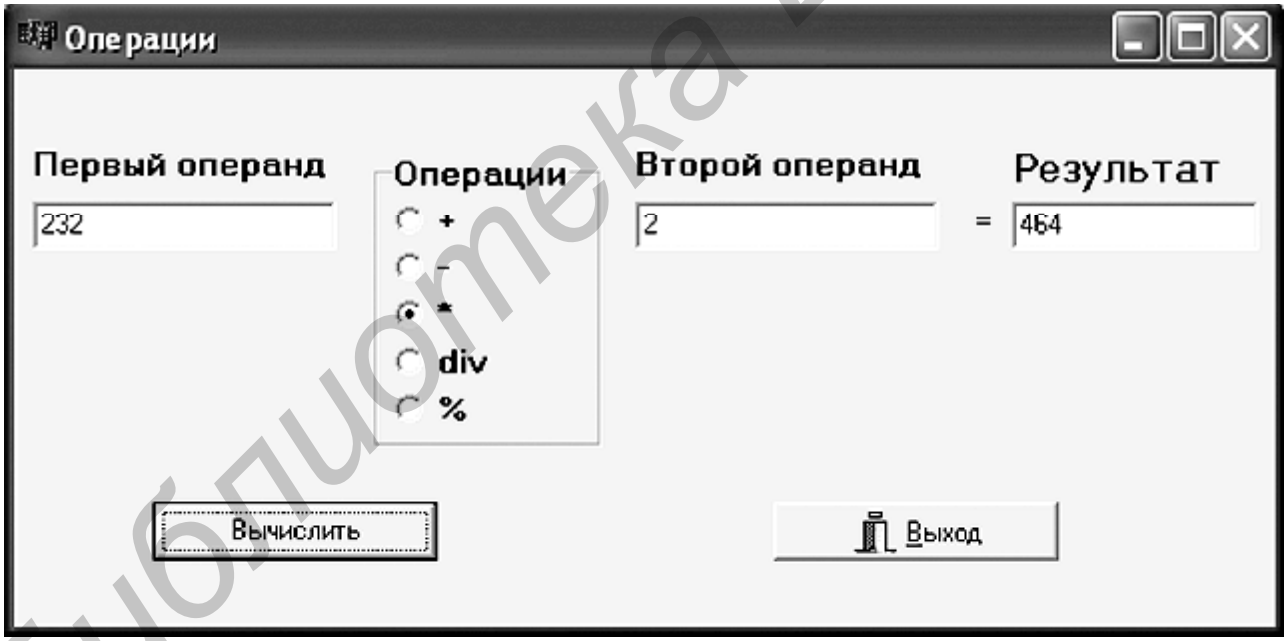


Рис. 10.1

Текст программы:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
```

```
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
long value;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Edit1Change(TObject *Sender)
{
char *endptr;
AnsiString C="ABCDEFGabcdefg0123456789";
if (Edit1->Text!="")
{try
          {Edit1->Color=clYellow;
          if (Edit2->Text!="") Edit3->Color=clFuchsia;
          if((Edit1->Text[1]=='-')&&(Edit1->Text.Length()==1)) return;
          if (C.AnsiPos(Edit1->Text[Edit1->Text.Length()])==0)
           throw EConvertError("");
           value=strtol(Edit1->Text.c_str(),&endptr,17);
           if ((Edit1->Text.Length()>11)&&(value==MaxInt))
              throw EIntOverflow("");
          } catch (EConvertError &)
      {if (Edit1->Text!='-')
           ShowMessage("Возможны лишь цифры от 0 до 9 и буквы
            от A до G,или "+  AnsiString(" от a до g!"));
       Edit1->Text=Edit1->Text.SetLength(Edit1->Text.Length()-1);
       Edit1->SelStart=Edit1->Text.Length();
       }
         catch (EIntOverflow &)
```

```
        {ShowMessage("Слишком большое число!");
        Edit1->Text=Edit1->Text.SetLength(Edit1->Text.Length()-1);
        Edit1->SelStart=Edit1->Text.Length();
        }
        catch (...)
  {ShowMessage("Возникла неизвестная исключительная ситуация!");}
  }
}

//---------------------------------------------------------------------------
void __fastcall TForm1::Edit2Change(TObject *Sender)
{char *endptr;
 int j=3;
AnsiString C,CComp="ABCDEFGabcdefg0123456789";
  if (Edit2->Text!="")
{try
        {Edit2->Color=clYellow;
        if (Edit1->Text!="") Edit3->Color=clFuchsia;
        if((Edit2->Text[1]=='-')&&(Edit2->Text.Length()==1)) return;
        C=Edit2->Text[Edit2->Text.Length()];
         try
         {j=j/CComp.AnsiPos(C);}
         __except(EXCEPTION_EXECUTE_HANDLER)
         {Application->MessageBox("Вы ввели недопустимый символ!",
         "Ошибка",MB_OK);
         throw EAssertionFailed("");}
        value=strtol(Edit2->Text.c_str(),&endptr,17);
         if((Edit2->Text.Length()>11)&&(value==MaxInt))
         throw EIntOverflow("");
         }
        catch (EAssertionFailed &)
        {if (Edit2->Text!='-')
```

```
        ShowMessage("Возможны лишь цифры от 0 до 9 и буквы от A до
G,или "+
        AnsiString(" от a до g!"));
        Edit2->Text=Edit2->Text.Delete(Edit2->Text.Length(),1);
        Edit2->SelStart=Edit2->Text.Length();
        }
          catch (EIntOverflow &)
        {ShowMessage("Слишком большое число!");
        Edit2->Text=Edit2->Text.Delete(Edit2->Text.Length(),1);
        Edit2->SelStart=Edit2->Text.Length();}
catch (...)
{ShowMessage("Возникла неизвестная исключительная ситуация!");}
}
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
  long x,y;
  char s[11],s1[11],*endptr;
  try {
       try
    {x=strtol(Edit1->Text.c_str(),&endptr,17);
     y=strtol(Edit2->Text.c_str(),&endptr,17);
     switch (RadioGroup1->ItemIndex)
     {case 0:x=x+y;break;
      case 1:x=x-y;break;
      case 2:x=x*y;break;
      case 3:x=x/y;break;
      case 4:x=x%y;break;
      }
      if (x>=0) itoa(x,s,17);
      else{itoa(-x,s,17);strcpy(s,strcat(strcpy(s1,"-"),s));}
```

```
      if (stricmp(s,Edit3->Text.c_str())!=0) Abort();
      else MessageDlg("Вы полностью правы!",mtInformation,
                              TMsgDlgButtons()<<mbOK,0);}
      catch (EIntOverflow &)
      {MessageDlg("Переполнение при выполнении операции!",mtError,
                        TMsgDlgButtons()<<mbOK,0);}
      catch (EDivByZero &)
      {MessageDlg("Делить на ноль нельзя!",mtError,
                        TMsgDlgButtons()<<mbOK,0);}
      catch(EConvertError &)
      {MessageDlg("Ошибка исходных данных!",
                        mtError, TMsgDlgButtons()<<mbOK,0);}
      catch(EAbort &){MessageDlg("Вы неправильно вычислили!",mtWarning,
                        TMsgDlgButtons()<<mbOK,0);
      Edit3->Text=AnsiString(s);}
      catch(...)
      {MessageDlg("Неизвестная исключительная ситуация!",
                         mtError,TMsgDlgButtons()<<mbOK,0);
}
    }
    __finally
    {Edit1->Color=clWindow;
     Edit2->Color=clWindow;
     Edit3->Color=clWindow;}
   }
//---------------------------------------------------------------------------

void __fastcall TForm1::RadioGroup1Click(TObject *Sender)
{
if (Edit1->Text!="") Edit1->Color=clYellow;
          else Edit1->Color=clWindow;
if (Edit2->Text!="") Edit2->Color=clYellow;
          else Edit2->Color=clWindow;
```

```
if ((Edit1->Text!="")&&(Edit2->Text!="")) Edit3->Color=clFuchsia;
        else Edit3->Color=clWindow;
}
//---------------------------------------------------------------------------
```

Индивидуальные задания

Создать приложение Windows, использующее механизм обработки исключительных ситуаций, позволяющее выполнять перевод вещественного числа из одной системы счисления в другую. Для обозначения использовать цифры 0 … 9 и буквы латинского алфавита (прописные и строчные).

1. Перевод числа из двоичной системы счисления в троичную.
2. Перевод числа из троичной системы счисления в четырнадцатеричную
3. Перевод числа из четверичной системы счисления в пятеричную.
4. Перевод числа из восьмеричной системы счисления в шестеричную.
5. Перевод числа из двоичной системы счисления в семеричную.
6. Перевод числа из тринадцатеричной системы счисления в восьмеричную.
7. Перевод числа из одиннадцатеричной системы счисления в девятеричную.
8. Перевод числа из семеричной системы счисления в одиннадцатеричную.
9. Перевод числа из восемнадцатеричной системы счисления в двенадцатеричную.
10. Перевод числа из двадцатеричной системы счисления в тринадцатеричную.
11. Перевод числа из девятеричной системы счисления в четверичную.
12. Перевод числа из восьмеричной системы счисления в пятнадцатеричную.
13. Перевод числа из троичной системы счисления в шестнадцатеричную.
14. Перевод числа из пятнадцатеричной системы счисления в восемнадцатеричную.
15. Перевод числа из двоичной системы счисления в девятнадцатеричную.

Создать диалоговое приложение, использующее механизм обработки исключительных ситуаций, позволяющее осуществлять контроль знаний по теме “Свойства логарифмов” (сложение логарифмов, вычитание и т.д.). Основание логарифма должно задаваться в режиме диалога. Программа должна позволять осуществлять вычисления в указанных системах счисления с заданной в режиме диалога точностью:

16. В десятичной и троичной системах счисления.
17. В десятичной и четверичной системах счисления.
18. В десятичной и девятеричной системах счисления.
19. В десятичной и пятнадцатеричной системах счисления.
20. В десятичной и двадцатеричной системах счисления.

Создать диалоговое приложение, использующее механизм обработки исключительных ситуаций, позволяющее осуществлять контроль знаний по теме “Свойства комплексных чисел” (сложение, умножение, формы представления и т.д.).

Программа должна позволять осуществлять вычисления в указанных системах счисления с заданной в режиме диалога точностью:

21. В десятичной и троичной системах счисления.
22. В десятичной и четверичной системах счисления.
23. В десятичной и девятеричной системах счисления.
24. В десятичной и пятнадцатеричной системах счисления.
25. В десятичной и двадцатеричной системах счисления.

Создать приложение, использующее механизм обработки исключительных ситуаций, позволяющее выполнять вычисление $\sqrt[n]{x}$ с заданной точностью (x, n и точность задаются в режиме диалога) в соответствующих системах счисления:

26. В десятичной и троичной системах счисления.
27. В десятичной и четверичной системах счисления.
28. В десятичной и девятеричной системах счисления.
29. В десятичной и пятнадцатеричной системах счисления.
30. В десятичной и двадцатеричной системах счисления.

Приложение 1

КРАТКИЕ СВЕДЕНИЯ ПО ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ C++ В СРЕДЕ BUILDER

1. Понятие алгоритма и способы его записи

Фундаментальным понятием при программировании является АЛГОРИТМ. Написанию программы предшествует разработка алгоритма решения задачи.

Алгоритм – это последовательность действий, в результате выполнения которых из исходных данных получается требуемый результат (решение).

Алгоритмы можно классифицировать по виду выполняемых действий. Например, правила ГАИ – это набор алгоритмов, описывающих действия водителей и пешеходов, поваренная книга – набор алгоритмов приготовления пищи.

Назовем ***вычислительными*** алгоритмы, последовательность действий которых «умеет» выполнять компьютер.

Программа представляет запись вычислительного алгоритма на языке, который умеет читать компьютер.

Мы уже знаем, что основными действиями, которые умеет выполнять компьютер, являются операции над содержимым ячеек памяти. Составление программы в командах машины крайне сложно и неудобно. И после того как был накоплен опыт применения ЭВМ для решения различных классов задач, было осознано, что компьютер – это универсальный инструмент, который может выполнять любую формализованную работу по переработке информации. Именно такой задачей является перевод программы с одного формального языка на другой. Возникли первые ***языки*** программирования, которые сейчас принято называть ***языками высокого*** уровня, удобные для записи тех или иных классов алгоритмов.

В языках программирования вместо номеров ячейкам принято давать имена (идентификаторы), а содержимое ячеек называть ***переменными*** или ***константами***, в зависимости от того, изменяется оно или нет в процессе работы.

Во всех языках фундаментальным понятием является ***оператор***, который представляет описание определенного набора действий ЭВМ. Программа, написанная на языке программирования, состоит из последовательности операторов.

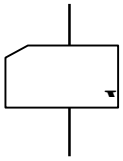
Одним из распространенных операторов является ***оператор присваивания*** ячейке памяти результата арифметических операций над содержимым других ячеек. Последовательность таких операций записывается, например, следующим образом:

```
b=0; c=1; f=5;
```

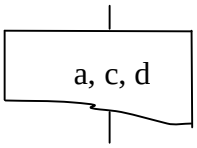
```
a=b+c-f; c=a; b=b+c;
```

Здесь = означает присвоить; a, b, c, f – имена ячеек (переменных); каждый оператор в C++ заканчивается точкой с запятой.

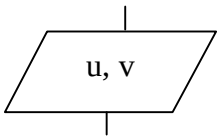
Во всех языках программирования имеется набор операторов, реализующих следующие основные действия:



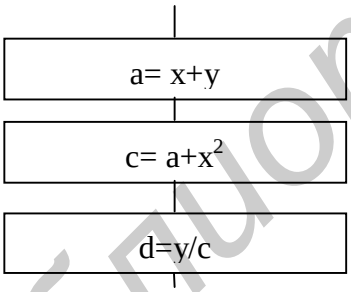
Ввод данных с внешнего носителя



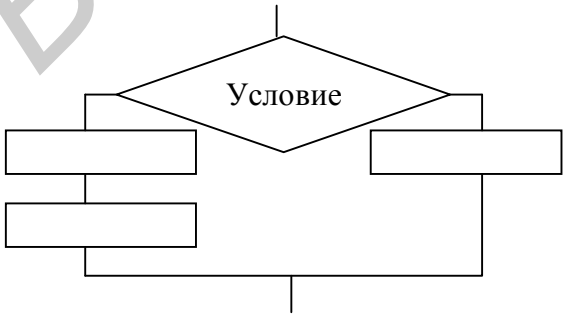
Вывод результатов на внешний носитель



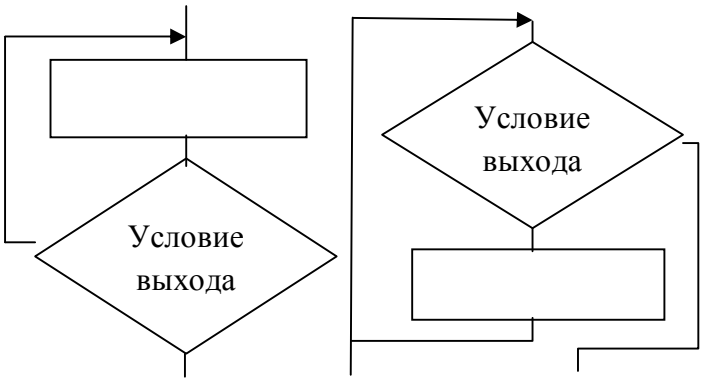
Обмен информацией между программами



Линейный вычислительный процесс



Разветвление алгоритма в зависимости от условия



Циклы, т.е. организация повторений некоторой последовательности операторов

2. Общая характеристика языка С++

Язык С был создан в 1972г. в лаборатории “Bell Laboratories” Деннисом Ритчи. Целью создания языка было написать операционную систему UNIX для компьютера PDP-11, который имел 24 килобайта памяти и один жесткий диск объемом 512 килобайт. Впоследствии созданный язык оказался настолько хорош, что был выбран для проектирования системных программ.

Язык программирования, известный как С++ , был разработан в начале 80-х годов прошлого столетия Бьярном Страуструпом. Реально С++ не является новым языком, так как включает все операторы и средства языка С, добавив только некоторые новые. Иначе говоря, С++ представляет собой надмножество языка С. Изучая С, вы по большей части изучаете и язык С++.

Приемущество С++ в том, что он позволяет с большей легкостью разрабатывать более сложные программы за счет более модульного подхода. Кроме того, С++ является языком объектно-ориентированного программирования.

3. Правила написания программ

При написании программы на языке С++ следует соблюдать следующие правила:

1. Операторы при записи алгоритма должны располагаться последовательно слева направо и сверху вниз. Порядок чтения может изменяться при помощи специальных операторов (см. ниже).

2. Все константы, типы, переменные и функции должны быть объявлены до их первого использования в любом месте программы.

3. При объявлении и использовании констант, типов, переменных и функций необходимо помнить, что в языке С++ прописное и строчное написание одной и той же буквы считается различными символами.

4. Каждый оператор заканчивается точкой с запятой (исключения см. ниже), поэтому в одной строке можно располагать несколько операторов или один оператор можно располагать на нескольких строках (не разбивая идентификаторы).

5. Для удобства разработки можно использовать комментарии, которые не обрабатываются компилятором и служат для улучшения читабельности программы. Комментарием считается все заключенное в скобки /* … */ или расположенное правее //.

6. Если необходимо объединить несколько операторов, то используется составной оператор. Составным оператором считается все, заключенное в фигурные скобки { … }.

4. Алфавит языка

Алфавит C++ включает латинские прописные и строчные буквы, цифры и специальные знаки. Специальные символы позволяют задавать операторы и знаки операций. Некоторые из них представлены в табл. П.1.

Таблица П.1

Символ	Операция
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Деление по модулю (остаток от деления)
++	Увеличение на единицу
--	Уменьшение на единицу
=	Присваивание
*=	Умножение с присваиванием
/=	Деление с присваиванием
%=	Деление по модулю (остаток от деления) с присваиванием
+=	Сложение с присваиванием
-=	Вычитание с присваиванием
&&	Логическое И
||	Логическое ИЛИ
!	Логическое отрицание
= =	Сравнение на равенство
>	Сравнение на больше
> =	Сравнение на больше или равно
<	Сравнение на меньше
< =	Сравнение на меньше или равно
! =	Сравнение на не равно
>>	Сдвиг вправо
<<	Сдвиг влево

5. Данные и их типы

Величины, с которыми должна работать программа, принято называть ***данными***. Все данные при работе программы размещаются в различные ячейки памяти, им присваиваются оригинальные имена (идентификаторы) и указываются

их типы. Идентификатор состоит из последовательности букв и цифр. Первой должна быть буква или символы _ и $. Пробелы, точки и другие специальные символы не могут входить в имя.

Данные могут быть ***константами*** и ***переменными.***

5.1. Константы

Константами называются неизменяемые величины в программе. Константы могут быть четырех типов: целые (десятичное, восьмеричное или шестнадцатеричное целое число), с плавающей точкой (десятичное число, представляемое в виде действительной величины с фиксированной или плавающей точкой), символьные (символы заключенные в одинарые скобки ‘ … ’ или, если это строка, – в двойные “ … “), перечисляемые (некая последовательность имен, которая автоматически нумеруется начиная с 0). Для улучшения читабельности программ константами им можно давать имена, например:

```
const M=5;          //  Константа целого типа
const float Pi2=3.14/2; //  Константа действительного типа
char const *Name = “Alex”; //  Указатель на строковую
                                    //    константу
```

5.2. Переменные

Переменными называются идентификаторы, значения которых могут меняться в процессе выполнения программы. Переменная может объявляться отдельным оператором до ее первого использования:

```
int  i, k=5;
double a, b;
```

или внутри операторов:

```
for (int i=0, I<3,  I++);
```

Тип данных определяет, какие значения они имеют, какая структура ячеек для их размещения используется, и какие операции над ними можно выполнять. В языке C++ имеется развитая система типов данных. Их можно разбить на две основные группы: скалярные (простые) и структурированные (составные). Отдельно стоит тип **void**, который определяет отсутствие значения.

К ***скалярному типу*** относятся данные, представляемые одним значением (числом, символом) и размещаемые в одной ячейке из нескольких байтов.

Структурированные типы определяются пользователем через скалярные и описанные ранее структурированные с помощью оператора typedef следующим образом

```
typedef double mas2[Nmax][Nmax];
typedef double mas1[Nmax];
```

Характеристики четырех основных скалярных базовых типов и их наиболее часто используемые имена представлены в табл. П.2. В скобках указано количество байт, рядом – диапазон изменения целых чисел и количество десятичных цифр действительных чисел.

Таблица П.2

Данные	Имя типа	Имя	Имя
Целые числа	***short int*** (2) -32,768 – 32,767	***int*** (4) -2,147,483,648 – 2,147,483,647	***unsigned int*** (4) 0 – 4,294,967,295
Действи-тельные числа	***float*** (4) $10^{-38} < \|X\| < 10^{38}$	**double** (8) $2.23*10^{-308} < \|X\| < 1.79*10^{308}$	***long double*** (10) $10^{-4932} < \|X\| < 1.18*10^{4932}$
Логические ***True,False***	***bool*** (4)		
Символы	***char*** (1)		

Для экономии памяти следует пользоваться более «короткими» типами, а для повышения точности более «длинными». Если ваш компьютер оснащен сопроцессором, то не рекомендуется использовать тип float

5.3. Операции над переменными основных скалярных типов

Во-первых, следует всегда помнить, что в операторе присваивания

a=<выражение>;

результат выражения (арифметического, логического) необязательно должен соответствовать типу переменной a, т.е. здесь можно смешивать типы, при этом может теряться информация. Исключение составляет возможность присваивать сложные структуры. Над переменными ***логического типа,*** определенными, например, как

```
bool  bl1,bl2,bl3,Cl,b1,d11,d12,d13;
int   x,y;
```

допустимы четыре логические операции **!,&&,** || и *операция присваивания*. Например:

```
cl=x>y;
bl=y<=10;
d11=b1 && c1;
d12=b1 || c1;
d13=!b1;
```

Здесь переменная dl1=True, если и bl и cl принимают значение true, и dl1=False, в противном случае; переменная dl2=True, если одна из b1 или c1 принимает значение True и d12=False в противном случае; переменная dl3=true, если bl=False, и dl3=False, если bl=True;

Над переменным ***символьного типа,*** определенным, например, как

```
char Sh, h;
```

ввиду их упорядоченности, допустимы такие логические операции:

```
Ch='a';        h='b';
b1=ch<h;       b1=ch>h;       b1=ch>=h;
b1=ch<=h;      b1=ch<>h;
```

Заметим, что символы >=, <=, пишутся слитно.

Над переменными ***целого типа,*** определенными, например, как

```
int     I, j, k, l, m, n;
```

наряду с операцией присваивания возможны следующие целочисленные арифметические операции и логические отношения (табл. П.3):

Таблица П.3.

Код операции	Операция	Пример	Логические отношения
+	Сложить	7=5+2	bl=m>n;
-	Отнять	2=5-3	bl=k<n;
*	Умножить	10=5*2	bl=i=j;
%	Остаток от деления	1=5%2	bl=j<=1;
			bl=i<>j;(не равно)

Над переменными ***действительного типа*** определенными, например, как

```
double      x, y, a, b;
```

возможны операции (+ - *) и логические отношения такие же, что и для целых. Для деления используется /. Результат операции 5/2.0, будет действительным, равным 2.5, в отличие от %.

В C++ имеется ряд функций для работы с действительными числами:

int **abs**(int n); - $|x|$;

double **fabs**(double x); - $|x|$;

double **cos**(double x);- cos(x);

double **cosh**(double x); - cosh(x);

double **sin**(double x); - sin(x);

double **sinh**(double x); -sinh(x);

double **tan**(double x); - tg(x);

double **floor**(double x); - округление к меньшему;

double **ceil**(double x); - округление к большему;

double **log10**(double x); - lg(x);

double **log**(double x); - ln(x);

double **exp**(double x); - e^x;

double **sqrt**(double x); - $\sqrt{x}$;

double **pow**(double x, double y); - x^y;

double **acos**(double x); - arccos(x);

double **asin**(double x); - arcsin(x);

double **atan**(double x); - arctg(x);

double **atan2**(double x, double y); - arctg(x); с учетом четверти.

6. Директивы препроцессора

Перед компиляцией программы происходит обработка программы препроцессором. На этом этапе выполняется подключение макросов, символических констант и других файлов. Также определяются режимы компиляции и выполняются директивы препроцессора. Каждая директива препроцессора располагается на отдельной строке, начинаются с символа “#” и в конце не ставится “;”.

6.1. Директива #include

Служит для включения указанного в ней файла в то место, где находится директива, т.е. директива убирается, а на ее место ставится файл. Имя включаемого файла может заключаться либо в угловые скобки (< >), либо в обычные (“ ”). В первом случае поиск указанного файла будет начинаться в стандартных каталогах C++ Builder, во втором случае – в текущем каталоге.

6.2. Директива #pragma

Данная директива служит для установки параметров компилятора. Обычно эти установки определяют другим способом, – используя диалог Project Options.

6.3. Директива #define

Служит для создания макросов и символических констант. Если в программе появляется строка, совпадающая с именем директивы, то она будет заменена текстом директивы. Для аннулирования макроса служит директива #undef.

Приложение 2

ПРОЦЕДУРЫ И ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

Для работы со строками применяются следующие процедуры и функции (в квадратных скобках указываются необязательные параметры)

Подпрограммы преобразования строк в другие типы	
Function StrToCurr(St: String): Currency;	Преобразует символы строки St в целое число типа Currency. Строка не должна содержать ведущих или ведомых пробелов
Function StrToFloat(St: String): Extended;	Преобразует символы строки St в вещественное число. Строка не должна содержать ведущих или ведомых пробелов
Function StrToInt(St: String): Integer;	Преобразует символы строки St в целое число. Строка не должна содержать ведущих или ведомых пробелов
Подпрограммы обратного преобразования	
Function FloatToStr(Value: Extended): String;	Преобразует вещественное значение Value в строку символов
Function FloatToStrF(Value: Extended; Format: TFloatFormat; Precision, Digits: Integer) : String;	Преобразует вещественное значение Value в строку символов с учетом параметров Precision и Digits (см. пояснения ниже)
Function FormatFloat(Format: String; Value: Extended): String;	Преобразует вещественное значение Value в строку
Function IntToStr(Value: Integer) : String;	Преобразует целое значение Value в строку символов

Правила использования параметров функции FloatToStrF показаны ниже:

Значение Format	Описание
fFfExponent	Precision задает общее количество десятичных цифр мантиссы. Digits - количество цифр в десятичном порядке XX. Число округляется с учетом первой отбрасываемой цифры: 3.1416E+00
ffFixed	Формат с фиксированным положением разделителя целой и дробной частей. Precision задает общее количество десятичных цифр в представлении числа. Digits - количество цифр в дробной части. Число округляется с учетом первой отбрасываемой цифры: 3,14
ffGeneral	Универсальный формат, использующий наиболее удобную для чтения форму представления вещественного числа. Соответствует формату ffFixed, если количество цифр в целой части меньше или равно Precision, а само число - больше или равно 0,00001, в противном случае соответствует формату ffExponent: 3,1416

ЛИТЕРАТУРА

1. Березин Б.И., Березин С.Б. Начальный курс С и С++. – М.: Диалог-МРТИ, 1999.

2. Керниган Б., Ритчи Д. Язык программирования СИ. – М.: Финансы и статистика, 1992.

3. Касаткин А.И., Вольвачев А.Н. Профессиональное программирование на языке СИ: от Турбо-С до Borland С++. Справ. пособие. – Мн.: Выш. шк., 1992.

4. Страуструп Б. Язык программирования С++. 2-е изд.: В 2 т. Киев: ДиаСофт, 1993.

5. Больски М.Н. Язык программирования СИ. Справочник. – М.: Радио и связь. 1988.

6. С++. Язык программирования. – М.: И.В.К.-СОФТ, 1991.

7. Архангельский А.Я. Программирование в С++ Builder 6. – М.: ЗАО “Издательство БИНОМ”, 2002.

Св. план 2004, поз. 40

Учебное издание

Синицын Анатолий Константинович,
Навроцкий Анатолий Александрович
Щербаков Александр Владимирович и др.

ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ В СРЕДЕ Builder C++

Лабораторный практикум по курсам “Программирование” и «Основы алгоритмизации и программирование» для студентов 1 – 2-го курсов всех специальностей БГУИР дневной и вечерней форм обучения
В 2-х частях
Часть 1

Редактор Т.Н. Крюкова
Корректор Е.Н. Батурчик

Подписано в печать 03.02.2004.	Формат 60x84 1/16.	Бумага офсетная.
Печать ризографическая.	Гарнитура “Times”	Усл. печ. л. 5,46.
Уч. изд. л. 5,0.	Тираж 350 экз.	Заказ 563.

Издатель и полиграфическое исполнение:
Учреждение образования
“Белорусский государственный университет информатики и радиоэлектроники”
Лицензия ЛП №156 от 30.12.2002.
Лицензия ЛП №509 от 03.08.2001.
220013, Минск, П. Бровки, 6