МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНСТИТУТ ТЕХНОЛОГИЙ (ФИЛИАЛ) ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» В Г. ВОЛГОДОНСКЕ РОСТОВСКОЙ ОБЛАСТИ

(Институт технологий (филиал) ДГТУ в г. Волгодонске)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для проведения лабораторного практикума

по дисциплине

«Технологии программирования»

для обучающихся по направлению подготовки

09.03.02 Информационные системы и технологии

программа бакалавриата «Информационные системы»

Волгодонск 2021

Содержание

Знакомство с интегрированной средой разработки Microsoft Visual Studio 2010

Integrated Development Environment (интегрированная среда разработки), или, сокращенно, IDE – это программный продукт, объединяющий текстовый редактор, компилятор, отладчик и справочную систему. Далее приводятся минимально необходимые сведения для начала работы с интегрированной средой. Более подробную информацию можно извлечь из справочной системы Microsoft Visual Studio 2010.

Каждая программа, создаваемая в среде Microsoft Visual Studio 2010, даже такая простая, как «Hello, World!», всегда оформляется как отдельный проект (project).

Проект – это набор взаимосвязанных исходных, заголовочных файлов и, возможно, файлов ресурсов, компиляция и компоновка которых позволяет создать исполняемую программу. Но разработчики Visual Studio пошли еще дальше, стремясь удовлетворить потребности не только программистов-одиночек, но и больших коллективов разработчиков программных продуктов. Так появилось понятие «решение» (Solution).

Решение может содержать любое количество различных проектов, сгруппированных вместе для согласованной разработки: от отдельного приложения до библиотеки функций или целого программного пакета. Для решения учебных задач каждая программа будет воплощаться в виде одного проекта, поэтому решение всегда будет содержать ровно один проект.

Создание проекта

Для создания нового проекта типа «консольное приложение» выполните следующие действия:

- Выберите в строке меню следующую команду File▶New▶Project…
- Открывается диалоговое окно мастера создания проектов (см. Рисунке 1).

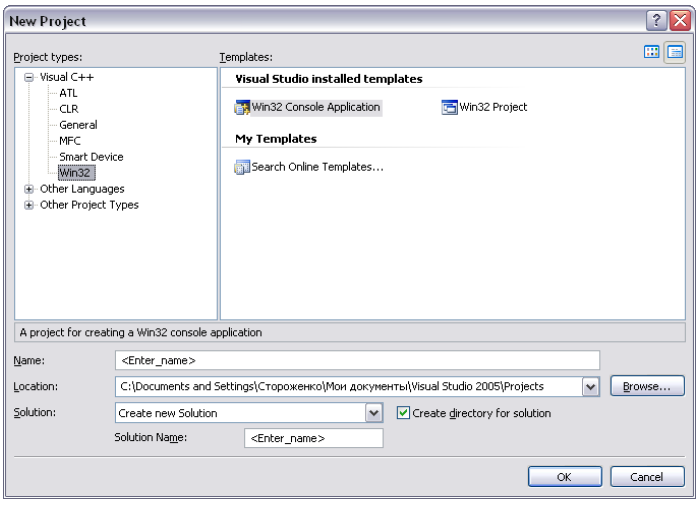

Рисунок 1 – Окно New Project.

- Выберите в дереве типов проектов (Project types) тип Win32;
- Выберите в поле доступных шаблонов (Templates) тип шаблона Win32 Console Application;

- введите имя проекта в текстовом поле Name, например, First;
- введите имя каталога размещения файлов проекта в текстовом поле Location (по указанному пути создается каталог, название которого совпадает с названием проекта);
- щелкните левой кнопкой мыши по кнопке OK.
- После щелчка запускается мастер приложений Application Wizard, который открывает диалоговое окно Win32 Console Application – название вашего проекта (см. Рисунок 2).

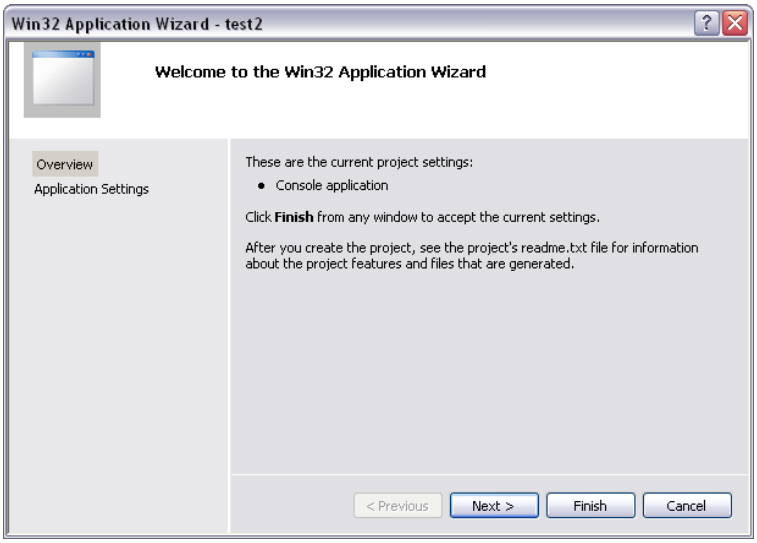

Рисунок - 2 Окно Win32 Application Wizard

Для завершения создания проекта щелкните левой кнопкой мыши по кнопке Finish.

Допустим, что в окне New Projects поле Name вы ввели имя проекта test1. После выполнения вышеуказанных шагов окно Visual Studio 2010 примет следующий вид (см. Рисунке 3).

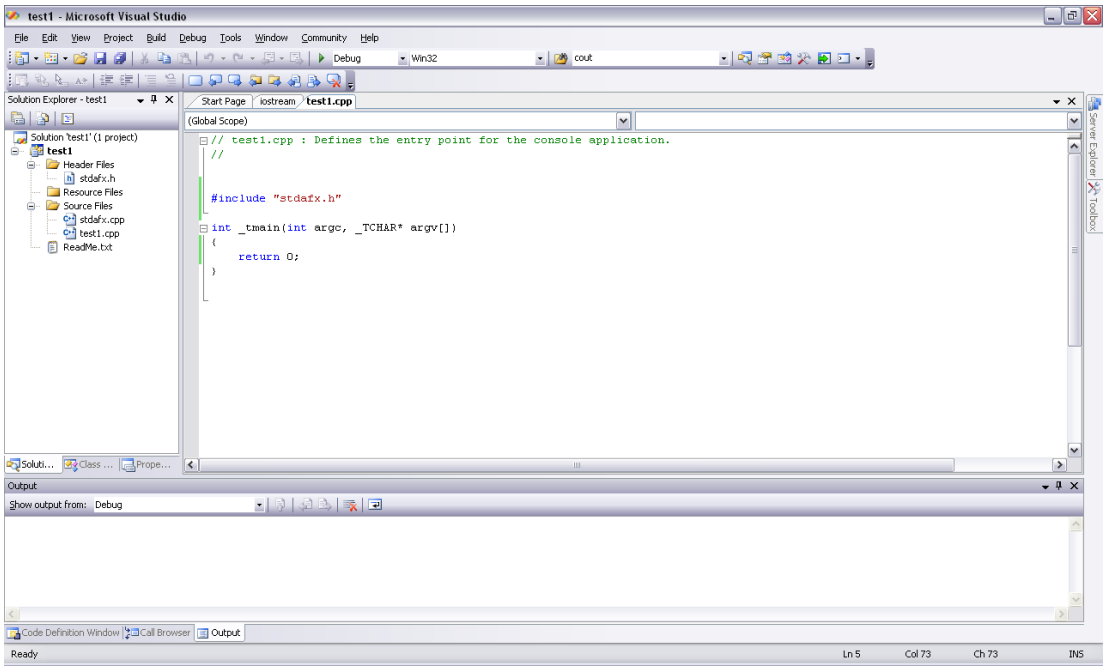

Рисунок 3 – Окно Microsoft Visual Studio 2010 после создания проекта test1

Компиляция, компоновка и выполнение проекта

Операции компиляции, компоновки и выполнения проекта могут быть выполнены либо через меню **Build** и **Debug** главного окна, либо с помощью кнопок панели инструментов. Опишем кратко команды меню **Build**:

- Build Solution – построение решения. Компилируется все файлы, в которых произошли изменения с момента последней компиляции. После компиляции происходит сборка (link) всех объектных модулей, включая библиотечные, в результирующий выполняемый файл. Сообщение об ошибках компоновки выводится в окне OutPut. Если обе фазы компоновки завершились без ошибок, то созданный выполняемый файл с расширением .exe может быть запущен на выполнение.
- Rebuild Solution – перестройка решения. Компилируются все файлы проекта независимо от того, были ли в них произведены изменения.
- Clean Solution – удаление из каталога, в котором располагаются файлы решения вспомогательных файлов Visual Studio.
- Compile – данный пункт меню по принципу выполняемых действий практически аналогичен Build Solution. Но после выполнения компиляции не запускается сборка решения.

В меню **Debug** нам понадобиться одна команда Start Debugging. При выполнении данной команды меню появляется окно, изображенное на Рисунке 4. Данное окно предлагает заново построить проект. После постройки проекта начнется его выполнение.

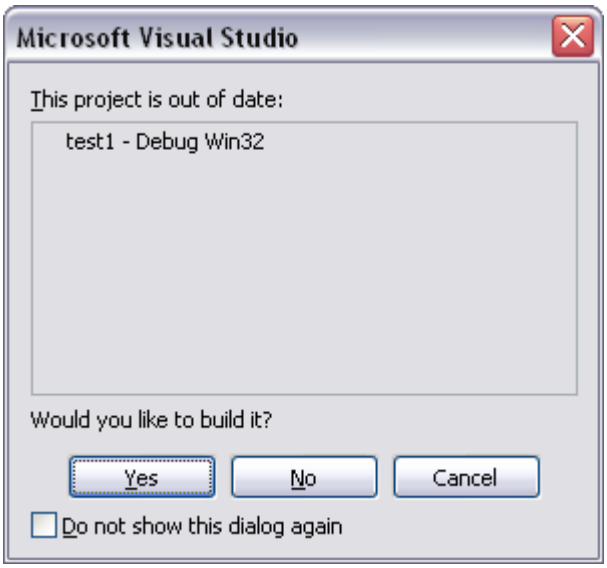

Рисунок 4 - Диалоговое окно появляющееся после выполнения команды Start Debugging

Операции Start Debugging удобнее выполнять через соответствующие кнопки панели инструментов Standard, часть панели Standard в увеличенном виде показана на Рисунке 5.

Рисунок 5 – Панель инструментов Standard.

Функции ввода-вывода

Работа в среде Visual Studio 2010 (в режиме консольных приложений) сопряжена с определенными неудобствами, вызванными различными стандартами кодировки символов кириллицы в операционных системах MS DOS и Windows. Это происходит вследствие того, что весь ввод-вывод в консольных приложениях проходит в кодировке стандарта ANSII, а текст в исходных файлах, набираемый в текстовом редакторе Visual Studio 2010 имеет кодировку в стандарте ANSI. Поэтому для нормального вывода строки, содержащей буквы русского алфавита, эту строку необходимо сначала «пропустить» через функцию CharToOemA(), а уже потом отправлять на консольный вывод.

С учетом выше сказанного можно написать вывод русских букв следующем образом:

```
#include "stdafx.h"
#include <iostream>
#include <windows.h>
	using namespace std;

char * RUS(const char * text);

int _tmain(int argc, _TCHAR* argv[])
{
	char c;
	cout<<RUS("Привет");
	cin>>c;
	return 0;
}
char bufRus[250];
char * RUS(const char * text)
{
	CharToOemA(text,bufRus);
	return bufRus;
}
```

Лабораторная работа №1

Оператор выбора if …else

Цель работы: освоить работу с оператором ветвления программы if … else.

Оборудование: персональный компьютер, Microsoft Visual Studio 2010.

Краткая теория

К операторам выбора (другое название операторы управления потоком выполнения программы) относят: условный оператор if … else и переключатель switch. Каждый из них служит для выбора пути выполнения программы.

Синтаксис условного оператора:

if (*выражение*) *оператор_1*; else *оператор_2*;

Выражение должно быть скалярным и может иметь арифметический тип или тип указателя. Если оно не равно нулю (или не есть пустой указатель), то условие считается истинным и выполняется *оператор_1*. В противном случае выполняется *оператор_2*. В качестве операторов нельзя использовать описания и определения. Однако здесь могут быть составные операторы и блоки:

```
if ( x > 0) { x = –x; f(x*2);}
  else { int i = 2; x *= i; f(x);}
```

При использовании блоков (т.е. составных операторов с определениями и описаниями) нельзя забывать о локализации определяемых в блоке объектов. Например, так как переменная i локализована в блоке и не существует вне него, то следующий листинг будет ошибочным:

```
if (j > 0) {int i; i = 2 * j;} else i = –j;
```

Допустима сокращенная форма условного оператора, в которой отсутствует else и *оператор_2*. В этом случае при ложности (равенстве нулю) проверяемого условия никакие действия не выполняются:

```
if (a < 0) a = –a;
```

В свою очередь, *оператор_1* и *оператор_2* могут быть условными, что позволяет организовать цепочку проверок условий любой глубины вложенности. В этих цепочках каждый из

условных операторов (после проверяемого условия и после else) может быть как полным условным, так и иметь сокращенную форму записи.

При этом могут возникать ошибки неоднозначного сопоставления if и else. Синтаксис языка предполагает, что при вложениях условных операторов каждое else соответствует ближайшему к нему предшествующему if. Пример неверного толкования этого правила:

```
if (x == 1)
   if (y == 1) cout << “x равен 1 и y равен 1”;
      else cout << “x не равен 1”;
```

При x=1 и y=1, совершено справедливо печатается фраза “x равен 1 и y равен 1”. Однако фраза “x не равен 1” может быть напечатана только при x=1, и при y≠1, т.к. else относиться к ближайшему if. Внешний условный оператор, где выражением является x = 1, является сокращенным и в качестве оператора_1 включает полный условный оператор, в котором проверяется условие y = 1. Таким образом, проверка этого условия выполняется только при x=1. Простейшее правильное решение этой микрозадачи можно получить, применив фигурные скобки, т.е. построив составной оператор. Нужно фигурными скобками ограничить область действия внутреннего условного оператора, сделав его неполным. Тем самым внешний оператор превращается в полный условный оператор:

```
if (x == 1)
   {if (y == 1) cout << “x равен 1 и y равен 1”;}
   else cout << “x не равен 1”;
```

Теперь else относиться к первому if и выбор выполняется верно.

В качестве второго примера вложения условных операторов рассмотрим функцию, возвращающую максимальное из значений трех аргументов:

```
int max3( int x, int y, int z)
{  if (x < y)
        if (y < z) return z;
        else return y;
   else
        if (x < z) return z;
        else return x;
}
```

В тексте соответствие if и else показано с помощью отступов.

Ниже приведены логические операции и операции отношения (сравнения), которые можно применять для сравнения операндов в *выражении*.

<	меньше, чем	x<y
>	больше, чем	x>y
<=	меньше или равно	x<=y
>=	больше или равно	x>=y
==	равно	x==y
!=	не равно	x!=y
&&	И	x<y && x>0
||	ИЛИ	x<y || x>0
( )	группировка выражений	(x<y)||(x>0)

Порядок выполнения

1. С помощью данных методических указаний освоить работу с оператором ветвления программы if … else.
2. Составить алгоритм решения задачи и написать программу согласно варианту задания.
3. Оформить отчет.

Задания

1. Написать программу, которая по введенному значению аргумента вычисляет значение функции. Значение *a, b, c, d* ввести с клавиатуры.

Варианты

Вариант 1

$$F=\begin{cases} ax^2+b, x<0, b\neq 0 \\ \dfrac{x-a}{x-c}, x>0, b=0 \\ \dfrac{x}{c} \end{cases}$$

Вариант 2

$$F=\begin{cases} \dfrac{1}{ax}-b, x+5<0, c=0 \\ \dfrac{x-a}{x}, x+5>0, c\neq 0 \\ \dfrac{10x}{c-4} \end{cases}$$

Вариант 3

$$F=\begin{cases} ax^2+bx+c, a<0, c\neq 0 \\ \dfrac{-a}{x-c}, a>0, c=0 \\ a(x+c) \end{cases}$$

Вариант 4

$$F=\begin{cases} -ax-c, c<0, x\neq 0 \\ \dfrac{x-a}{-c}, c>0, x=0 \\ \dfrac{bx}{c-a} \end{cases}$$

Вариант 5

$$F=\begin{cases} a-\dfrac{x}{10+b}, x<0, b\neq 0 \\ \dfrac{x-a}{x-c}, x>0, b=0 \\ 3x+\dfrac{2}{c} \end{cases}$$

Вариант 6

$$F=\begin{cases} ax^2+b^2x, c<0, b\neq 0 \\ \dfrac{x+a}{x+c}, c>0, b=0 \\ \dfrac{x}{c} \end{cases}$$

Вариант 7

$$F=\begin{cases} -ax^2-b, x<5, c\neq 0 \\ \dfrac{x-a}{x}, x>5, c=0 \\ \dfrac{-x}{c} \end{cases}$$

Вариант 8

$$F=\begin{cases} -ax^2, c<0, a\neq 0 \\ \dfrac{a-x}{cx}, c>0, a=0 \\ \dfrac{x}{c} \end{cases}$$

Вариант 9

$$F=\begin{cases} ax^2+b^2x, a<0, x\neq 0 \\ x-\dfrac{a}{x-c}, a>0, x=0 \\ 1+\dfrac{x}{c} \end{cases}$$

Вариант 10

$$F=\begin{cases} ax^2-bx+c, x<3, b\neq 0 \\ \dfrac{x-a}{x-c}, x>3, b=0 \\ \dfrac{x}{c} \end{cases}$$

Вариант 11

$$F=\begin{cases} ax^2+\dfrac{b}{c}, x<1, c\neq 0 \\ \dfrac{x-a}{(x-c)^2}, x>1.5, c=0 \\ \dfrac{x^2}{c^2} \end{cases}$$

Вариант 12

$$F=\begin{cases} ax^3+b^2+c, x<0.6, b+c\neq 0 \\ \dfrac{x-a}{x-c}, x>0.6, b+c=0 \\ \dfrac{x}{c}+\dfrac{x}{a} \end{cases}$$

Вариант 13

$$F=\begin{cases} ax^2+b, x-1<0, b-x\neq 0 \\ \dfrac{x-a}{x}, x-1>0, b+x=0 \\ \dfrac{x}{c} \end{cases}$$

Вариант 14

$$F=\begin{cases} -ax^3-b, x+c<0, a\neq 0 \\ \dfrac{x-a}{x-c}, x+c>0, a=0 \\ \dfrac{x}{c}+\dfrac{c}{x} \end{cases}$$

Вариант 15

$$F=\begin{cases}-ax^2+b, x<0, b\neq 0\\ \dfrac{x}{x-c}+5.5, x>0, b=0\\ \dfrac{x}{-c}\end{cases}$$

Вариант 16

$$F=\begin{cases}a(x+c)^2-b, x=0, b\neq 0\\ \dfrac{x-a}{-c}, x\neq 0, b=0\\ a+\dfrac{x}{c}\end{cases}$$

Вариант 17

$$F=\begin{cases}ax^2-cx+b, x+10<0, b\neq 0\\ \dfrac{x-a}{x-c}, x+10>0, b=0\\ \dfrac{-x}{a-c}\end{cases}$$

Вариант 18

$$F=\begin{cases}ax^3+bx^2, x<0, b\neq 0\\ \dfrac{x-a}{x-c}, x>0, b=0\\ \dfrac{x+5}{c(x-10)}\end{cases}$$

Вариант 19

$$F=\begin{cases}a(x+7)^2-b, x<5, b\neq 0\\ \dfrac{x-cd}{ax}, x>5, b=0\\ \dfrac{x}{c}\end{cases}$$

Вариант 20

$$F=\begin{cases}-\dfrac{2x-c}{cx-a}, x<0, b\neq 0\\ \dfrac{x-a}{x-c}, x>0, b=0\\ -\dfrac{x}{c}+\dfrac{-c}{2x}\end{cases}$$

Контрольные вопросы

1. Объясните, для чего нужен оператор выбора if … else.
2. Приведите синтаксис оператора выбора if … else, полную и сокращенную форму.
3. Сколько операторов можно писать после … if(выражение)?
4. Сколько операторов можно писать после … else?
5. Укажите, с помощью, какой конструкции после … if(выражение) и … else можно писать несколько операторов.
6. Перечислите операторы условия, которые можно использовать в конструкции if … else.
7. Перечислите логические операторы, которые можно использовать в конструкции if … else.
8. Напишите на C# программу вычисляющую, следующую задачу: Если a > b и c < 5, то a=b+c, иначе a=c.
9. Напишите на C# программу вычисляющую, следующую задачу: Если a <10 или c< 1, то a=b, иначе a=c.
10. Напишите на C# программу вычисляющую, следующую задачу: Если a > b и c < 7, то a=b+c.

Лабораторная работа №2

Оператор выбора switch

Цель работы: освоить работу с оператором ветвления программы switch.

Оборудование: персональный компьютер, Microsoft Visual Studio 2010.

Краткая теория

К операторам выбора (другое название операторы управления потоком выполнения программы) относят: условный оператор if … else и переключатель switch. Каждый из них служит для выбора пути выполнения программы.

Переключатель является наиболее удобным средством для организации мультиветвлений. Синтаксис переключателя выглядит следующим образом:

```
switch (переключающее_выражение)
    {
        case константное_выражение_1: операторы_1;
        case константное_выражение_2: операторы_2;
        . . .
        case константное_выражение_n: операторы_n;
        default: операторы;
    }
```

Управляющая конструкция switch передает управление к тому из помеченных, с помощью зарезервированного слова «case», операторов, для которых значение константного выражения совпадает со значением переключающего выражения. Переключающее выражение должно быть целочисленным или его значение приводится к целому. Значение константных выражений, помещаемых за служебными словами case, приводится к типу переключающего выражения. В одном переключателе все константные выражения должны иметь различные значения, но быть одного типа. Любой из операторов, помещенных в фигурных скобках после конструкции switch(. . .), может быть помечен одной или несколькими метками вида case *константное_выражение*:.

Если значение переключающего выражения не совпадает ни с одним из константных выражений, то выполняется переход к оператору, отмеченному меткой default:. В каждом переключателе должно быть не более одной метки default, но эта метка может и отсутствовать. В случае отсутствия метки default при несовпадении переключающего выражения ни с одним из константных выражений, помещенных вслед за case, в переключателе не выполняется ни один из операторов.

Сами по себе метки case *константное_выражение_j*: и default: не изменяют последовательности выполнения операторов. Если не предусмотрен переход из переключателя, то в нем последовательно выполняются все операторы начиная с той метки, на которую передано управление. Пример программы с переключателем:

```
#include <iostream.h>
void main()
{
        int ic;
        cout << “Введите любую десятичную цифру:”;
        cin >> ic;
        cout << endl;
        switch (ic)
                {
                        case 0: case 1: cout << “один, “;
                        case 2: case 3: cout << “три, “;
                        case 4: case 5: cout << “пять, “;
                        case 6: case 7: cout << “семь, “;
                        case 8: case 9: cout << “девять. “;
                                        break;
                                default:  cout << “Ошибка! Это не цифра!”;
                }
}
```

Результаты двух выполнений программы:

```
Введите любую десятичную цифру: 4 <Enter>
пять, семь, девять

Введите любую десятичную цифру: z <Enter>
Ошибка! Это не цифра!
```

Кроме сказанного о возможностях переключателя, приведенная программа иллюстрирует действие оператора break. С его помощью выполняется выход из переключателя. Если поместить операторы break после вывода каждой цифры, то программа будет печатать название только одной нечетной цифры.

В переключателе могут находиться описания и определения объектов, т.е. составной оператор, входящий в переключатель, может быть блоком. В этом случае нужно избегать ошибок «перескакивания» через определения:

```
switch(n)                                   //Переключатель с ошибками
{
        char d='D';                         //Никогда не обрабатывается
        case 1: float f = 3.14;             //Обрабатывается только для n == 1
        case 2: if  (int(d) != int(f)) …    // d и (или) f не определены
```

Порядок выполнения

1. С помощью методических указаний к работе освоить работу с оператором ветвления switch(…).
2. Составить алгоритм решения задачи и написать программу согласно варианту задания.
3. Оформить отчет.

Задания

1. Написать программу, которая по введенному значению аргумента вычисляет значение функции. Значение a, b, c, d ввести с клавиатуры.

Варианты

Вариант 1

$$F=\begin{cases} ax^2+b, d=1 \\ \dfrac{x-a}{x-c}, d=2 \\ \dfrac{x}{c} \end{cases}$$

Вариант 2

$$F=\begin{cases} \dfrac{1}{ax}-b, d=5 \\ \dfrac{x-a}{x}, d=1 \\ \dfrac{10x}{c-4} \end{cases}$$

Вариант 3

$$F=\begin{cases} ax^2+bx+c, d=1 \\ \dfrac{-a}{x-c}, a>0, d=13 \\ a(x+c) \end{cases}$$

Вариант 4

$$F=\begin{cases} -ax-c, d=3 \\ \dfrac{x-a}{-c}, d=8 \\ \dfrac{bx}{c-a} \end{cases}$$

Вариант 5

$$F=\begin{cases} a-\dfrac{x}{10+b}, d=4 \\ \dfrac{x-a}{x-c}, d=6 \\ 3x+\dfrac{2}{c} \end{cases}$$

Вариант 6

$$F=\begin{cases} ax^2+b^2x, d=2 \\ \dfrac{x+a}{x+c}, d=3 \\ \dfrac{x}{c} \end{cases}$$

Вариант 7

$$F=\begin{cases} -ax^2-b, d=7 \\ \dfrac{x-a}{x}, d=8 \\ \dfrac{-x}{c} \end{cases}$$

Вариант 8

$$F=\begin{cases} -ax^2, d=10 \\ \dfrac{a-x}{cx}, d=11 \\ \dfrac{x}{c} \end{cases}$$

Вариант 9

$$F=\begin{cases} ax^2+b^2x, d=5 \\ x-\dfrac{a}{x-c}, d=13 \\ 1+\dfrac{x}{c} \end{cases}$$

Вариант 10

$$F=\begin{cases} ax^2-bx+c, d=7 \\ \dfrac{x-a}{x-c}, x>3, d=3 \\ \dfrac{x}{c} \end{cases}$$

Вариант 11

$$F=\begin{cases} ax^2+\dfrac{b}{c}, d=2 \\ \dfrac{x-a}{(x-c)^2}, d=7 \\ \dfrac{x^2}{c^2} \end{cases}$$

Вариант 12

$$F=\begin{cases} ax^3+b^2+c, d=5 \\ \dfrac{x-a}{x-c}, d=8 \\ \dfrac{x}{c}+\dfrac{x}{a} \end{cases}$$

Вариант 13

$$F=\begin{cases} ax^2+b, d=1 \\ \dfrac{x-a}{x}, d=4 \\ \dfrac{x}{c} \end{cases}$$

Вариант 14

$$F=\begin{cases} -ax^3-b, d=3 \\ \dfrac{x-a}{x-c}, d=10 \\ \dfrac{x}{c}+\dfrac{c}{x} \end{cases}$$

Вариант 15

$$F=\begin{cases} -ax^2+b, d=7 \\ \dfrac{x}{x-c}+5.5, d=9 \\ \dfrac{x}{-c} \end{cases}$$

Вариант 16

$$F=\begin{cases} a(x+c)^2-b, d=10 \\ \dfrac{x-a}{-c}, x\neq 0, d=0 \\ a+\dfrac{x}{c} \end{cases}$$

Вариант 17

$$F=\begin{cases} ax^2-cx+b, d=7 \\ \dfrac{x-a}{x-c}, d=1 \\ \dfrac{-x}{a-c} \end{cases}$$

Вариант 18

$$F=\begin{cases} ax^3+bx^2, d=8 \\ \dfrac{x-a}{x-c}, d=3 \\ \dfrac{x+5}{c(x-10)} \end{cases}$$

Вариант 19

$$F=\begin{cases} a(x+7)^2-b, d=1 \\ \dfrac{x-cd}{ax}, x>5, d=0 \\ \dfrac{x}{c} \end{cases}$$

Вариант 20

$$F=\begin{cases} -\dfrac{2x-c}{cx-a}, d=10 \\ \dfrac{x-a}{x-c}, x>0, d=5 \\ -\dfrac{x}{c}+\dfrac{-c}{2x} \end{cases}$$

Контрольные вопросы

1. Объясните, для чего нужна конструкция switch … case.
2. Приведите синтаксис конструкции switch … case.
3. Объясните логику работы конструкции switch … case.
4. Расскажите, в каких случаях и для чего необходимо использовать команду break в конструкции switch … case.
5. Какую операцию сравнения выполняет структура switch … case?
6. В каком месте структуры switch … case указывается переменная, по которой происходит сравнение?
7. Объясните для чего необходимо использование и приведите пример использования описателя default.
8. Напишите и объясните, как выполняется на языке C++ следующая задача: пользователь вводит число от 0 до 9 и программа выводит название только нечетных цифр от 0 до 9 включительно.
9. Напишите и объясните, как выполняется на языке C++ следующая задача: пользователь вводит число от 0 до 9 и программа выводит название только четных цифр до 9 включительно.

Лабораторная работа №3

Операторы цикла

Цель работы: освоить работу с оператором цикла for и while.

Оборудование: персональный компьютер, Microsoft Visual Studio 2010.

Краткая теория

Операторы цикла – это операторы, которые циклически повторяют заданную последовательность операторов фиксированное число раз или до тех пор, пока не будет удовлетворено условие проверки.

Определены три разных оператора цикла:

- цикл с предусловием:

  while (*выражение_условие*)

  тело_цикла

- цикл с постусловием:

  do

  тело_цикла

  while (*выражение_условие*)

- итерационный цикл:

  for (*инициализация_цикла*;

  выражение_условие;

  список_выражений)

  тело_цикла

Цикл while проверяет выражение и в том случае, если оно истинно, выполняет одиночный или составной оператор:

```
while (i < 100) str[i++] =” ”;
```

Сначала проверяется *выражение_условие*. Если i < 100, то выполняется тело цикла (str[i++] =””). Если i≥100, то цикл передает управление следующему за ним оператору. Важной характеристикой цикла while является то, что оператор никогда не выполнится, если условие изначально было ложно.

В предыдущем примере i меняется на каждом шаге цикла. Часто встречается ошибка, когда условие окончания цикла так никогда и не удовлетворяется. Это может происходить потому, что управляющая переменная не изменяется в теле цикла, в результате цикл становится бесконечным. В

таких случаях заметить ошибку довольно легко. При написании цикла убедитесь, что граничное условие достижимо.

Вместе с операторами цикла часто используется пустой оператор. В следующем примере происходит копирование одной строки символов в другую:

```
while (*strl ++ =  *str2++);
```

Оператор do – это оператор цикла похожий на while. Единственное отличие цикла while от цикла do … while состоит в том, что проверка условия происходит не в начале, а в конце цикла. Таким образом, тело цикла обязательно выполняется, по крайней мере, один раз. Например:

```
int i, total =0;
do {
        cout « "Введите число (0 – выход из программы). \n";
        cin » i;
        total += i;
} while (i) ;
cout « "Итого: " « total « endl;
```

Пользователю хотя бы раз надо предложить ввести число. Цикл do в данном случае подходит больше всего. На следующем листинге дан пример двух простых циклов: первый – цикл while, второй – do.

```
#include <iostream.h>
void main()
{
        int count = 0;
        while (count++ < 10)
                cout « "счетчик цикла while: " « count « endl;
        count = 0;
        do
                cout « " счетчик цикла do: " « count « endl;
        while (count++ < 10);
}
```

Результат действия программы:

```
счетчик цикла while: 12345678910 счетчик цикла do: 012345678910
```

Цикл do прокручивается 11 раз, а цикл while – 10. В обоих случаях конечное значение счетчика одинаково – 11.

При выполнении цикла for происходит следующее:

1. Выполняется *начальное_выражение*. Это происходит только один раз – при входе в цикл.
2. Проверяется условие. Если его значение равно true, то переходим к третьему пункту. Если же оно ложно, то управление передается оператору, следующему за блоком цикла for. Если это значение изначально было равно false, то операторы в теле цикла не выполняются ни разу.
3. Выполняется последовательность операторов.
4. Вычисляется выражение управления циклом.
5. Возврат в пункт 2.

Инициализация_цикла, выражение_условие, список_выражений - одно из них или более могут быть опущены, но знак точка с запятой обязан присутствовать. То есть корректна следующая запись:

```
for (;;) {// Последовательность операторов }
```

Это эквивалентно такому циклу while:

```
while(1) { // Последовательность операторов }
```

Обычно *инициализация_цикла* используется для инициализации и, возможно, объявления переменной управления циклом. Здесь можно объявлять и инициализировать и другие переменные. Вот обычный пример использования цикла, инициализирующего массив:

```
int ia[10];
for (int i = 0, j = 100; i < 10; i++, j-=10) ia[i] = j;
```

Область видимости переменных, объявленных в *инициализации_цикла*, распространяется только на цикл for. *выражение_условие* чаще всего используется для проверки условия завершения цикла. Как правило, в нем происходит сравнение переменной управления циклом с

некоторым значением. Так же, как и в циклах while и do, необходимо следить за тем, чтобы условие рано или поздно выполнялось.

Инструкция *список_выражений* обычно используется для приращения счетчика цикла. В данном случае безразлично, использовать ли инкремент в префиксной или же постфиксной форме, поскольку на значение управляющего выражения это никак не влияет.

Порядок выполнения

1. С помощью методических указаний освоить работу с операторами циклов.
2. Составить алгоритм решения задачи и написать программу согласно варианту задания.
3. Оформить отчет.

Задания

1. Вычислить и вывести на экран в виде таблицы значение функции F на интервале от Хнач. до Хкон. с шагом dX. Значения a, b, c, d, Хнач., Хкон., dX ввести с клавиатуры.

Варианты

Вариант 1

$$F=\begin{cases} ax^2+b, x<0, b\neq 0 \\ \dfrac{x-a}{x-c}, x>0, b=0 \\ \dfrac{x}{c} \end{cases}$$

Вариант 2

$$F=\begin{cases} \dfrac{1}{ax}-b, x+5<0, c=0 \\ \dfrac{x-a}{x}, x+5>0, c\neq 0 \\ \dfrac{10x}{c-4} \end{cases}$$

Вариант 3

$$F=\begin{cases} ax^2+bx+c, a<0, c\neq 0 \\ \dfrac{-a}{x-c}, a>0, c=0 \\ a(x+c) \end{cases}$$

Вариант 4

$$F=\begin{cases} -ax-c, c<0, x\neq 0 \\ \dfrac{x-a}{-c}, c>0, x=0 \\ \dfrac{bx}{c-a} \end{cases}$$

Вариант 5

$$F=\begin{cases} a-\dfrac{x}{10+b}, x<0, b\neq 0 \\ \dfrac{x-a}{x-c}, x>0, b=0 \\ 3x+\dfrac{2}{c} \end{cases}$$

Вариант 6

$$F=\begin{cases} ax^2+b^2x, c<0, b\neq 0 \\ \dfrac{x+a}{x+c}, c>0, b=0 \\ \dfrac{x}{c} \end{cases}$$

Вариант 7

$$F=\begin{cases}-ax^2-b, x<5, c\neq 0\\ \dfrac{x-a}{x}, x>5, c=0\\ \dfrac{-x}{c}\end{cases}$$

Вариант 8

$$F=\begin{cases}-ax^2, c<0, a\neq 0\\ \dfrac{a-x}{cx}, c>0, a=0\\ \dfrac{x}{c}\end{cases}$$

Вариант 9

$$F=\begin{cases}ax^2+b^2x, a<0, x\neq 0\\ x-\dfrac{a}{x-c}, a>0, x=0\\ 1+\dfrac{x}{c}\end{cases}$$

Вариант 10

$$F=\begin{cases}ax^2-bx+c, x<3, b\neq 0\\ \dfrac{x-a}{x-c}, x>3, b=0\\ \dfrac{x}{c}\end{cases}$$

Вариант 11

$$F=\begin{cases}ax^2+\dfrac{b}{c}, x<1, c\neq 0\\ \dfrac{x-a}{(x-c)^2}, x>1.5, c=0\\ \dfrac{x^2}{c^2}\end{cases}$$

Вариант 12

$$F=\begin{cases}ax^3+b^2+c, x<0.6, b+c\neq 0\\ \dfrac{x-a}{x-c}, x>0.6, b+c=0\\ \dfrac{x}{c}+\dfrac{x}{a}\end{cases}$$

Вариант 13

$$F=\begin{cases}ax^2+b, x-1<0, b-x\neq 0\\ \dfrac{x-a}{x}, x-1>0, b+x=0\\ \dfrac{x}{c}\end{cases}$$

Вариант 14

$$F=\begin{cases}-ax^3-b, x+c<0, a\neq 0\\ \dfrac{x-a}{x-c}, x+c>0, a=0\\ \dfrac{x}{c}+\dfrac{c}{x}\end{cases}$$

Вариант 15

$$F=\begin{cases}-ax^2+b, x<0, b\neq 0\\ \dfrac{x}{x-c}+5.5, x>0, b=0\\ \dfrac{x}{-c}\end{cases}$$

Вариант 16

$$F=\begin{cases}a(x+c)^2-b, x=0, b\neq 0\\ \dfrac{x-a}{-c}, x\neq 0, b=0\\ a+\dfrac{x}{c}\end{cases}$$

Вариант 17

$$F=\begin{cases}ax^2-cx+b, x+10<0, b\neq 0\\ \dfrac{x-a}{x-c}, x+10>0, b=0\\ \dfrac{-x}{a-c}\end{cases}$$

Вариант 18

$$F=\begin{cases}ax^3+bx^2, x<0, b\neq 0\\ \dfrac{x-a}{x-c}, x>0, b=0\\ \dfrac{x+5}{c(x-10)}\end{cases}$$

Вариант 19

$$F=\begin{cases} a(x+7)^2-b, x<5, b\neq 0 \\ \dfrac{x-cd}{ax}, x>5, b=0 \\ \dfrac{x}{c} \end{cases}$$

Вариант 20

$$F=\begin{cases} -\dfrac{2x-c}{cx-a}, x<0, b\neq 0 \\ \dfrac{x-a}{x-c}, x>0, b=0 \\ -\dfrac{x}{c}+\dfrac{-c}{2x} \end{cases}$$

2. Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от Хнач до Хкон с шагом dx точностью ε. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

Вариант 1

$$\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+...)|x|>1$$

Вариант 2

$$e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-...|x|<20$$

Вариант 3

$$e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-...x>-20$$

Вариант 4

$$\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+...-1<x\le 1$$

Вариант 5

$$\ln\frac{1+x}{1-x}=2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=2(x+\frac{x^3}{3}+\frac{x^5}{5}+...)|x|<1$$

Вариант 6

$$\ln(1-x)=-\sum_{n=0}^{\infty}\frac{x^n}{n}=-(x+\frac{x^2}{2}+\frac{x^4}{4}+...)-1\le x<1$$

Вариант 7

$$arcctg(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} x^{2n+1}}{2n+1} = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} + ... |x| \le 1$$

Вариант 8

$$arctg(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + ... x > 1$$

Вариант 9

$$arctg(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + ... x \le 1$$

Вариант 10

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{n!} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + ... 0 < x < 90$$

Контрольные вопросы

1. Объясните, в каких случаях необходимо использовать цикл for.
2. Объясните, в каких случаях необходимо использовать цикл while.
3. Расскажите об отличии цикла for от цикла while.
4. Объясните, для чего и в каких случаях необходимо использовать команду break в цикле for.
5. Объясните, для чего и в каких случаях необходимо использовать команду break в цикле while.
6. Приведите синтаксис цикла for.
7. Приведите синтаксис цикла while.
8. Приведите пример бесконечного цикла for.
9. Приведите пример бесконечного цикла while.
10. Можно ли заменить цикл for циклом while и, на примере, покажите, как это сделать?
11. Можно ли заменить цикл while циклом for и, на примере, покажите, как это сделать?
12. Цикл for выполняется пока условие верно или пока условие неверно?
13. Цикл while выполняется пока условие верно или пока условие неверно?
14. Перечислите операторы условия, которые можно использовать в цикле for.
15. Перечислите логические операторы, которые можно использовать в цикле for.
16. Перечислите операторы условия, которые можно использовать в цикле while.

Лабораторная работа № 4

Массивы

Цель работы: освоить работу с массивами.

Оборудование: персональный компьютер, Microsoft Visual Studio 2010.

Краткая теория

Массив – это один из структурированных типов языка C++. От других структурированных данных массив отличается тем, что все его элементы имеют один и тот же тип, и что элементы массива расположены в памяти подряд.

Создание одномерного массива:

тип *имя_массива* [*константное_выражение*]

Здесь тип – это тип элементов массива, *имя_массива* – идентификатор, *константное_выражение*, если оно присутствует, определяет размер массива, т.е. количество элементов в массиве. В некоторых случаях допустимо описание массива без указания количества его элементов, т.е. без конкретного выражения в квадратных скобках.

Например:

```
extern unsigned long UL[ ];
```

Такое определение массива означает описание внешнего массива, который определен в другой части программы, где ему выделена память и (возможно) присвоены начальные значения его элементам.

При определении массива может выполняться его инициализация, т.е. элементы массива получают конкретные значения. Инициализация выполняется по умолчанию (без вмешательства программиста), если массив статический или внешний. В этих случаях всем элементам массива компилятор автоматически присваивает нулевые значения:

```
void f(void)
{
        static float F[4]; // Внутренний статический массив
        long double A[10]; // Массив автоматической памяти
}
// Описание массива
void main()
```

```
{
        extern int D[];
        …
        f();
        …
}

int D[8];       // Внешний массив (определение)
```

Массивы D[8] и F[4] инициализированы нулевыми значениями. В основной программе main() массив D описан без указания количества его элементов. Массив A[10] не получает конкретных значений своих элементов при определении.

Явная инициализация элементов массива разрешена только при его определении и возможна двумя способами: либо с указанием размера массива в квадратных скобках, либо без явного указания (без конкретного выражения) в квадратных скобках:

```
char CH[] = { 'A1', 'B1', 'C', 'D'}; // Массив из 4 элементов
int IN[6] = { 10, 20, 30, 40 }; // Массив из 6 элементов
char STR[] = "ABCD";       // Массив из 5 элементов
```

Количество элементов массива CH компилятор определяет по числу начальных значений в списке инициализации, помещенном в фигурных скобках при определении массива. В массиве IN шесть элементов, но только первые четыре из них явно получают начальные значения. Элементы IN[4], IN[5], либо не определены, либо имеют нулевые значения, когда массив внешний или статический. В массиве STR элемент STR[4] равен ' \0' , а всего в этом массиве пять элементов.

При отсутствии константного выражения в квадратных скобках список начальных значений в определении массива обязателен. Если размер массива явно задан, то количество элементов в списке начальных значений не должно превышать размера массива. Ошибочные определения:

```
float A[]; // Ошибка в определении массива - нет размера
double B[4] = { 1, 2, 3, 4, 5, 6 }; // Ошибка инициализации
```

В тех случаях, когда массив не определяется, а описывается, список начальных значений задавать нельзя. В описании массива может отсутствовать и его размер:

```
extern float E[]; // Правильное описание внешнего массива
```

Предполагается, что в месте определения массива E для него выделена память и выполнена инициализация.

Описание массива (без указания размера и без списка начальных значений) может использоваться в списке формальных параметров определения функции и в спецификации параметров прототипа функции. Примеры:

```
float MULTY(float G[], float F[]) // Определение функции
// MULTY
{ .. .
тело_функции
…
}
void print_array(int I[]);  // Прототип функции print_array
```

Доступ к элементам массива возможен с помощью индексированных переменных. При использовании индексированных элементов необходимо помнить, что все массивы в C++ начинаются с 0. Поэтому, если берется значение элемента с индексом 1, то обращаемся ко второму элементу, пример:

```
int m[10]={1,2,3,4,5,6,7,8,9,0};
. . .
m[5]=1; //Присвоение 6-му элементу значение 1
. . .
```

Существует еще один прием, позволяющий контролировать диапазон изменения индекса массива при его "просмотре", например в цикле. С помощью операции sizeof (*имя_массива*) можно определить размер массива в байтах, т.е. размеры участка памяти, выделенного для массива. Так как все элементы массива имеют одинаковые размеры, то частное sizeof(*имя_массива*)/sizeof(*имя_массива*[0]) определяет количество элементов в массиве. Следующий фрагмент программы печатает значения всех элементов массива:

```
int m[]={10, 20, 30, 40} ;
for (int i=0;i <sizeof(m)/sizeof(m[0]);i++) cout «"m["«  i  «  "]  =  "«m[i]«"    ";
```

Результат на экране дисплея: i[0]= 10 m[l] = 20 m[2] = 30 m[3] = 40

Отметим, что для первого элемента массива индекс равен 0. Цикл завершается при достижении i значения 4.

Многомерный массив в соответствии с синтаксисом языка есть массив массивов, т.е. массив, элементами которого служат массивы. Определение многомерного массива в общем случае должно содержать сведения о типе, размерности и количествах элементов каждой размерности:

type *имя_массива*[*K1*][*K2*]...[*KN*];

Здесь type – допустимый тип (основной или производный), *имя_массива* – идентификатор, *N* – размерность массива, *K1* – количество в массиве элементов размерности N-1 каждый и т.д. Например:

int ARRAY[4][3][6] ;

Трехмерный массив ARRAY состоит из четырех элементов, каждый из которых – двухмерный массив с размерами 3 на 6 элемента. В памяти массив ARRAY размещается в порядке возрастания самого правого индекса, т.е. самый младший адрес имеет элемент ARRAY[0] [0] [0], затем идет ARRAY [0] [0] [1] и т.д.

Как и в случае одномерных массивов, доступ к элементам многомерных массивов возможен с помощью индексированных переменных и с помощью указателей. Возможно объединение обоих способов в одном выражении.

Порядок выполнения

1. С помощью методических указаний освоить работу с массивами.
2. Составить алгоритм решения задачи и написать программу согласно варианту задания.
3. Оформить отчет.

Задания

1. Создать массив, инициализировав его элементы произвольными значениями. Выполнить задания, содержащиеся в варианте. Значение A, B, C ввести с клавиатуры.

Вариант 1

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Сумму отрицательных элементов массива;
2. Произведение элементов массива, расположенных между максимальным и минимальным элементами.

Вариант 2

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Сумму положительных элементов массива;
2. Произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Вариант 3

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. Сумму элементов массива, расположенных между первым и последним нулевыми элементами.
2. Сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

Вариант 4

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Сумму элементов массива с нечетными номерами;
2. Сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Вариант 5

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Максимальный элемент массива;
2. Сумму элементов массива, расположенных до последнего положительного элемента.

Вариант 6

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Минимальный элемент массива;
2. Сумму элементов массива, расположенных между первым и последним положительными элементами.

Вариант 7

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. Номер максимального элемента массива;
2. Произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Вариант 8

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Номер минимального элемента массива;
2. Сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Вариант 9

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Максимальный по модулю элемент массива;
2. Сумму элементов массива, расположенных между первым и вторым положительными элементами.

Вариант 10

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. Минимальный по модулю элемент массива
2. Сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Вариант 11

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Номер минимального по модулю элемента массива;
2. Сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Вариант 12

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Номер максимального по модулю элемента массива;
2. Сумму элементов массива, расположенных после первого положительного элемента.

Вариант 13

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество элементов массива, лежащих в диапазоне от A до B;
2. Сумму элементов массива, расположенных после максимального элемента.

Вариант 14

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество элементов массива, равных 0;
2. Сумму элементов массива, расположенных после минимального элемента.

Вариант 15

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество элементов массива, больших C;
2. Произведение элементов массива, расположенных после максимального по модулю элемента.

Вариант 16

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество отрицательных элементов массива;
2. Сумму модулей элементов массива, расположенных после минимального по модулю элемента.

Вариант 17

В одномерном массиве, состоящем из n целых элементов, вычислить:

1. Количество положительных элементов массива;
2. Сумму элементов массива, расположенных после последнего элемента, равного нулю.

Вариант 18

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Количество элементов массива, меньших C;
2. Сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

Вариант 19

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Произведение отрицательных элементов массива;
2. Сумму положительных элементов массива, расположенных до максимального элемента.

Вариант 20

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

1. Произведение положительных элементов массива;
2. Сумму элементов массива, расположенных до минимального элемента.

Контрольные вопросы

1. Сформулируйте понятие массив.
2. Укажите, как создать массив целых чисел размерностью 5.
3. Укажите, как создать двухмерный массив целых чисел размерностью 5 и 3.
4. Укажите, как можно получить доступ к элементам массива.
5. Как можно получить доступ к элементу массива при помощи переменной хранящей индекс элемента, и какой тип может иметь переменная, указывающая индекс массива.
6. Объясните, как можно при помощи цикла for обойти все элементы массива.
7. Объясните, как можно при помощи цикла while обойти все элементы массива.
8. Напишите и объясните, как выполняется на C++ следующая задача – вывод на экран только нечетных элементов массива.
9. Напишите и объясните, как выполняется на C++ следующая задача – вывод на экран только четных элементов массива.
10. Напишите и объясните, как выполняется на C++ следующая задача – вывод на экран 2,3,5,8,13,21 и т.д..

Рекомендованная литература.

1. Подбельский В.В Язык С++: Учеб. пособие. – 2-е. изд., перераб. и доп. – М.: Финансы и статистика, 1996.
2. Шмидский Я.К. Программирование на языке С/С# (Си). Самоучитель, 2004.
3. Б. Керниган, Д. Ритчи, А. Фьюер. Язык программирования Си. Задачи по языку Си. – М.: Финансы и статистика, 1985.
4. М. Уэйт, С. Прата, Д. Мартин. Язык Си. Руководство для начинающих. – М.: Мир, 1988.
5. М. Болски. Язык программирования Си. Справочник. – М.: Радио и связь, 1988.
6. Л. Хэнкок, М. Кригер. Введение в программирование на языке Си. – М.: Радио и связь, 1986.
7. М. Дансмур, Г. Дейвис. ОС UNIX и программирование на языке Си. – М.: Радио и связь, 1989.
8. Р. Берри, Б. Микинз. Язык Си. Введение для программистов. – М.: Финансы и статистика, 1988.
9. М. Беляков, А. Ливеровский, В. Семик, В. Шяудкулис. Инструментальная мобильная операционная система ИНМОС. – М.: Финансы и статистика, 1985.
10. К. Кристиан. Введение в операционную систему UNIX. – М.: Финансы и статистика, 1985.
11. Р. Готье. Руководство по операционной системе UNIX. – М.: Финансы и статистика, 1986.
12. М. Банахан, Э. Раттер. Введение в операционную систему UNIX. – М.: Радио и связь, 1986.
13. С. Баурн. Операционная система UNIX. – М.: Мир, 1986.
14. П. Браун. Введение в операционную систему UNIX. – М.: Мир, 1987.
15. M. Bach. The design of the UNIX operating system. – Prentice Hall, Englewood Cliffs, N.J., 1986.
16. S. Dewhurst, K. Stark. Programming in C#. – Prentice Hall, 1989.
17. M. Ellis, B. Stroustrup. The annotated C# Reference Manual. – Addison-Wesley, 1990.