**Лабораторная работа 8**

1. **Переменные в C++**

Ясно, что всякая программа, так или иначе, обрабатывает данные. Эти данные, как правило, размещаются в переменных. Каждая переменная имеет **имя, тип, размер и значение**. Основные типы с указанием размеров занимаемой памяти и областью значений приведены в таблице 1.

*Таблица 1. Основные типы переменных в C++*

|  |  |  |  |
| --- | --- | --- | --- |
| **тип** | **Тип (по-русски)** | **Размер памяти** | **Интервал допустимых значений** |
| bool | логический | 1 байт | **true** или **false** |
| char | символьный | 1 байт | От –128 до 127 (код  символа) |
| int | целый | 4 байта | От –2 147 483 648  до 2 147 483 647 |
| long | целый | 4 байта | От –2 147 483 648  до 2 147 483 647 |
| short | Короткий целый | 2 байта | От –32 768 до 32 767 |
| unsigned char | Беззнаковый  символьный | 1 байт | От 0 до 255 |
| unsigned int | Беззнаковый  целый | 4 байта | От 0 до 4 294 967 295 |
| unsigned short | Беззнаковый  короткий целый | 2 байта | От 0 до 65535 |
| float | Вещественный  плавающий | 4 байта | От  3.4е-38 до  3.4е+38 |
| double | Вещественный двойной точности | 8 байтов | От  1.7е-308 до   1.7е+308 |

Объявление типов переменных делается соответствующим служебным словом с последующим перечислением имен переменных:

**int i,j,k,l,m** – перечисленные переменные будут целого типа и т.п. Причем, объявление типов можно совместить с присваиванием.

Присваивание значения переменным делается с помощью *команды присваивания* (в С++ используется *знак равенства*). Значения для символьных переменных заключаются в одинарные кавычки:

**int i=33; float a=.156; char f=’5’, f2=’$’, ff=’1’**

Основные арифметические и логические операции, допустимые в С++, приведены в таблице 2.

*Таблица 2. Основные арифметические и логические операции в C++*

|  |  |  |  |
| --- | --- | --- | --- |
| **Арифметические**  **операции** | **символы** | **Логические операции** | **символы** |
| сложение | + | равно | = = |
| вычитание | — | Не равно | != |
| умножение | \* | больше | > |
| деление | / | Меньше | < |
| Вычисление  остатка | % | Больше или равно | >= |
| присваивание | = | Меньше или равно | <= |
| отрицание | ! | Логическое умножение | && |
|  |  | Логическое сложение | | | |

Присваивание значений переменных делается командой присваивания: *j*=5; *i=j*. Кстати, в C++ можно такие операции совмещать: *i=j*=5;

В С++ имеются помимо привычных и сокращенные формы записи арифметических операций. Примеры:

***Операция присваивания***

a+=3 эквивалентно a=a+3 a– =3 эквивалентно a=a–3 a\*=3 эквивалентно a=a\*3 a/=3 эквивалентно a=a/3 a%=3 эквивалентно a=a%3 ***Операции инкремента***

i++ постфиксная форма i=i+1;

++i префиксная форма i=i+1;

***Операции декремента***

i– – постфиксная форма i=i–1

– – i префиксная форма i=i–1

Операции инкремента и декремента применяются к целым числам. Различие между постфиксной и префиксной формами иллюстрируется приведенным ниже примером.

Например, пусть имеем переменные целого типа *t* и *c*. Если

*с=*5, тогда при записи *t*=++*c* + 6 получим, что *t* равно 12.

Если же применить постфиксную форму (при том же *c*=5): *t*=*c*++ + 6, то получим, что *t*=11, потому что начальное значение c используется для вычисления выражения до того, как c увеличится на единицу операцией инкремента. Этот оператор эквивалентен следующим двум: *t*=*c*+6; ++*c*;

Эти же правила применимы и к операции декремента – – . На- пример, если *с=*5, то *t* = – –*c* +6 даст значение 10 для *t*, в то время как *t* = 6 + *с*– – даст значение 11 для переменной *t*.

1. **Математические функции в C++**

Поскольку язык C++ разрабатывался, прежде всего, для ре- шения научно-технических задач, то здесь имеется целый набор математических функций. Основные функции приводятся в таблице 1, более широкий набор дан в Приложении 1.

*Таблица 1. Основные математические функции в C++*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Матема- тическая запись** | **Запись на С++** | **Тип функции** | **тип аргумента** | **Примечание** |
| sin *x* | sin(x) | double | double | аргумент в радианах |
| cos *x* | cos(x) | double | double | аргумент в радианах |
| *tgx* | tan(x) | double | double | аргумент в радианах |
| *arctgx* | atan(x) | double | double | угол, тангенс кото-  рого есть X |
| ln *x* | log(x) | double | double | натуральный ло-  гарифм числа X |
| *ex* | exp(x) | double | double | Функция, обратная натуральному ло-  гарифму числа X |
| *x* | sqrt(x) | double | double | Квадратный корень  числа X>=0 |
| | *x* | | fabs(x) | double | double | модуль (абс. вели-  чина) числа Х |
| *x y* | pow(x,y) | double | double | число X в степени Y |

Для примера запишем несколько выражений с использованием математических функций по правилам языка C++.

** 2  **

*ax*2  *bx*  *c*

* *b*  *b*2  4*ac*

2*a*

запишется так: **(**alfa\*alfa + beta) / (a\*x\*x + b\*x + c)

запишется так**:** (–b + sqrt(b\*b – 4\*a\*c))/(2\*a)

Запишем еще следующие выражения:

*a*) 8 *x* 8  8*x*

*б* ) *xyz*  3,3 | *x*  4 *y* | *в*)

**  sin2 ** 4

107 

*tg*4!

cos2 **  | *ctg* |

*а)* sqrt(sqrt(sqrt (pow(x,8)+8\*x)))

*б*) (x\*y\*z – 3.3\*fabs(x+pow(y, 1.0/4)))/(1e7+sqrt(tan(2\*3\*4)))

*в*)(b+sin(pow(pi,4))\*sin(pow(pi,4)))/(cos(g)\*cos(g)+fabs(1/tan(g)))

## ЛИНЕЙНАЯ ПРОГРАММА

### Оформление линейной программы

Программа на С++ состоит из одной или более функций. Причем, начинается выполнение программы с функции **main()**, которая, по сути, есть главный элемент программы. Каждая функция, вообще говоря, должна возвращать какой-либо результат. Поэтому и функция **main()** должна заканчивать командой возврата **return 0;,** показывающей, что программа завершена. Причем у функции (как и у переменных) указывается тип. Функция **main()** имеет тип **int**. Собственно алгоритм (исполняемая часть программы) заключается в фигурные скобки {} после выражения **int main()**. Каждая фраза алгоритма заканчивается точкой с запятой **;**.

Обычно перед функцией **main** дают так называемые директивы препроцессору. Такие директивы начинаются со знака # (произносится «хаш»). Например, директива

###### # include <iostream>

подключает встроенные в C++ функции, которые обеспечивают ввод/вывод.

Для *вывода на экран* служит команда

**cout <<** (читается «си-аут»). В ней для вывода текстовых сообщений помещают их в кавычки. Также там можно разместить управляющие последовательности для разделения строк вывода:

“\**n**” – дает команду начать вывод с новой строки

‘ ‘ (пробел в одинарных кавычках) – разделяет пробелом выводимые знаки или строки. Иначе – весь вывод сливается в одну строку.

“**\t**” - табуляция;

Для *ввода в программу* (в процессе ее выполнения) служит команда:

**cin**>> (читается «си-ин»).

В ней указывается переменная, которой и будет присвоено значение, введенное с экрана.

В программе можно размещать комментарии, которым предшествует **//** (двойная дробная черта).

Важной в начале исходного текста программы является директива объявления пространства имен:

###### using namespace std;

Эффект от ее применения состоит в том, что вы можете свободно применять, в частности, вышеупомянутые команды ввода-вывода **cout** и **cin**. И процессор при этом будет четко понимать, что это команды, а не переменные.

Изученных сведений достаточно для составления простой программы на C++. По аналогии со структурным языком назовем такую программу, исполняющуюся прямолинейно от начала и до конца – *линейной программой*. Рассмотрим простой пример.

***Пример 3-1.*** Вычислить периметр и площадь прямо– угольника, длина которого равна *a*, а ширина равна *b*. ***пример: a=3 м, b=4 м Ответ: периметр=14 м, площадь=12 м2***

Приведем сначала блок-схему алгоритма для решения данной за- дачи.

//вычисление периметра и площади прямоугольника #include <iostream>

using namespace std; int main()

{ float a,b,p,s;

cout<<"Vvedi storony pryamoug" <<"\n"; cin>>a>>b;

p=2\*(a+b); s=a\*b; cout<<"Perimetr="<<p<<" Ploshad'="<<s; cout<<"\n";

return 0;

}

Результат работы данного примера приведен на ***рис. 1.7*** и ***1.8***.

***Замечание.*** Одна из проблем при программировании в Visual C++ 2005 (при создании консольного приложения) в том, что в консольном

«черном» окне не отображаются сообщения, написанные по-русски. Это связано с тем, что в Windows (где мы пишем текст программы) и в MS DOS (где отображается ее результат) используется разная кодировка русских букв. Чтобы справится с этой проблемой, нужно использовать

специальную функцию AnsiToOem, которая преобразует текст из одной кодировки в другую, причем, надо подключить дополнительный заголо- вочный файл windows.h .

Как именно это делается, проиллюстрируем на том же ***при-***

***мере 3.1***. (нужные дополнения выделены **жирным** шрифтом)

//вычисление периметра и площади прямоугольника #include <iostream>

###### #include <windows.h>

using namespace std; int main()

{ char str[256]; float a,b,p,s;

###### AnsiToOem("Введи стороны прямоугольника ", str);

cout<< str <<"\n"; cin>>a>>b; p=2\*(a+b); s=a\*b;

###### AnsiToOem("Периметр=", str);

cout<< str<<p;

###### AnsiToOem(" Площадь=", str);

cout<<str<<s; cout<<"\n"; return 0;

}

Как видим, сначала каждое русское сообщение преобразуется функцией AnsiToOem и помещается в строку str (она описана как char str[256], т.е. может содержать до 256 символов). За- тем уже эта строка str и выводится. Обратите внимание, что подключен заголовочный файл windows.h**.**

Итак, **порядок разработки программы на Visual C++ 2005 (консольное приложение)**:

1. Создать проект: дать команду ***File/New/Project*** (или нажать комбинацию клавиш **Ctrl-Shift-N**)
2. В возникшем окне на левой панели выбрать **Win32** и на пра- вой выберем **Win32 Console Application (**для создания кон- сольного приложения), в строке **Name** введем имя проекта (при этом должна быть установлена «птичка» **Create directory for solution** создать каталог для проекта) и нажать ОК
3. в очередном окне появляется список подпапок данного про- екта. Для создания исходного файла следует щелкнуть правой клавишей по папке **Source files** и выбрать ***Add/New Item...*** (см. ***рис. 1.4***)
4. в возникшем окне (***рис. 1.5***) выбрать **Code**, указать **C++ File (.cpp)**, ввести имя для исходного файла (в строке **Name**) и на- жать **Add**.
5. в окне редактора вводим текст программы и сохраняем его командой ***File/Save all*** (или клавишами **Ctrl-Shift-S**)
6. для запуска программы на выполнение, нужно сначала про- вести компиляцию и компоновку — дать команду ***Build/Build Solution*** (или клавиша **F7**).
7. если все в порядке — в окне **Output** должно быть «0 failed» (см. ***рис. 1.7***), то собственно запускаем на исполнение — кла- вишами **Ctrl-F5** (***Debug/Start Without Debugging***).
8. в результате откроется «черное окно», где вводим исходные данные и получаем результат — см***. рис. 1.8.***

Для демонстрации решим одну задачу.

1. Составить программу для нахождения значения функции: c=sina+cos2b в режиме Win32 Console Application.

**Пример.** Программа может иметь вид:

 #include <iostream >//подключаем файл для организации ввода/вывода

#include <math.h>//подключаем файл для использования алгебраических функций

#include <conio.h>//подключаем файл для вызова функции очистки экрана

using namespace::std; // перевод на русский язык

int main()//заголовок главной программы

{ double a,b,c; //описываем три переменных вещественного типа

cout<<"Введите значения a и b:"; //выдаем текстовую подсказку на экран

cin>>a>>b; //запрашиваем ввод с клавиатуры двух переменных: a и b

c=sin(a)+pow(cos(b),2); //считаем значение функции

cout<<"Функция равна:"<<c<<"\n";//выводим результат на экран

cout<<"Для продолжения нажмите любую клавишу. . .";

getch();//делаем паузу для просмотра результата

return 0; //завершаем работу главной программы

}

1. **Составить программу для нахождения значения функции в режиме CRL WINDOUS FORM.**

Панель диалога программы организовать в виде, представленном на рис:

Для вывода результатов работы программы в программе используется текстовое окно, которое представлено обычным элементом управления. После установки свойства Multiline в True появляется возможность растягивать элемент управления не только по горизонтали, но и по вертикали.

## 

Затем ввести кнопку очистить, для удаления данных из textBox.

**Пример.** Составить программу для нахождения значения функции: z=sinx+cos2y в режиме CRL WINDOUS FORM.

В самом начале программы (после #pragma once)

#pragma once

#include <math.h>

double x, y;// тип переменных для математических функций

Для кнопки «Вычислить»:

// Получение исходных данных из TextBox

x=double::Parse(textBox1->Text);// считыванеие значения х из textBox1

y=double::Parse(textBox2->Text); // считыванеие значения у из textBox2

z=double::Parse(textBox3->Text); // считыванеие значения z из textBox3

// Вычисление выражения

double z= sin(x)+pow(cos(y),2);; // вычисление функции

// Ввод исходных данных в окно результатов

textBox3->Text=textBox3->Text+"Янущик Ольга Владимировна, ШБИП "+"\r\n"+"x="+System::Convert::ToString(x)+"\r\n"+"y="+System::Convert::ToString(y)+

// Вывод результата

textBox3->Text=textBox3->Text+"\r\n"+"z="+System::Convert::ToString(z); //вывод текста в textBox3

// \r\n вывод на новую строку

//Очистка всех введенных данных и полученного результата

Для кнопки «Очистить»

textBox1->Clear();// очистка textBox1

textBox2->Clear();

textBox3->Clear();

## Индивидуальные задания

1. Составить программу для расчета функции по формулe в режиме Win32 Console Application.

Отметим, что для использования математических функций нужно подключить к программе заголовочный файл <math.h>.

2.

1. 

При x=14.26, y=-1.22, z=3.5× t=0.564849.

2. 

При x=-4.5, y=0.75×,z=0.845 u=-55.6848.

3. 

При x=3.74, y=-0.825, z=0.16, *v*=1.0553.

4. 

При x=0.4 ,y=-0.875, z=-0.475 *w*=1.9873.

5. 

При x=-15.246, y=4.642, z=20.001 =-182.036.

6. 

При x=16.55, y=-2.75, z=0.15 =-38.902.

7. 

При x=0.1722, y=6.33, z=3.25 =-172.025.

8. 

При x=-2.235, y=2.23, z=15.221 =39.374.

9. 

При x=1.825, y=18.225, z=-3.298 =1.2131.

10. 

При x=3.981, y=-1.625, z=0.512 *a*=1.26185.

11. 

При x=6.251, y=0.827, z=25.001 *b*=0.7121.

12.

При x=3.251, y=0.325, z=0.466 *c*=4.025.

13. .

При x=17.421, y=10.365, z=0.828 *f*=0.33056.

14. .

При x=12.3, y=15.4, z=0.252 *g*=82.8257.

15. .

При x=2.444, y=0.869, z=-0.13 -0.49871.

**Приложение 1. Список библиотечных функций** Приведенный ниже список содержит основные функции с указанием их типа и типов аргументов (без указания имени аргумента). При использовании функции следует добавлять **.h** к имени

заголовочного файла.

Математические функции

|  |  |  |
| --- | --- | --- |
| **Прототип** | **Описание** | **Заголовочный файл** |
| int abs(int) | Абсолютное значение | stdlib |
| long labs(long) | Абсолютное значение | stdlib |
| double fabs(double) | Абсолютное значение | math |
| double sqrt(double) | Квадратный корень | math  complex |
| double pow(double, dou-  ble) | Первый аргумент в степени вто-  рого | math  complex |
| double exp(double x) | Функция **ex (**где x – аргумент) | math  complex |
| double log(double) | Натуральный логарифм (**ln**) | math |
| double log10(double) | Логарифм по основанию 10 (**lg**) | math |
| double ceil(double) | Наименьшее целое большее или  равное аргументу | math |
| double floor(double) | Наибольшее целое меньшее или  равное аргументу | math |
| int rand % n | Случайное число от 0 до n-1 | stdlib |
| void srand(unsigned int) | Реинициализирует генератор  случайных чисел | stdlib |
| double acos(double) | арккосинус | math |
| double asin(double) | арксинус | math |
| double atan(double) | арктангенс | math |
| double cos(double) | косинус | math |
| double cosh(double) | косинус гиперболический | math |
| double sin(double) | синус | math |
| double sinh(double) | синус гиперболический | math |
| double tan(double) | тангенс | math |
| double tanh(double) | тангенс гиперболический | math |

Строковые функции (для работы с символьными массивами)

|  |  |  |
| --- | --- | --- |
| **Прототип** | **Описание** | **Заголовочный файл** |
| char\* strcat(char \*Dest, const char \*Source) | Добавляет строку Source в  строку Dest и возвращает ука- затель на Dest | string |
| char\* strchr(const char\* string, int ch) | Выполняет поиск символа с кодом ch слева направо в стро- ке string; возвращает указатель на первое вхождение символа. Если символ не обнаружен,  возвращает NULL (нулевой указатель) | string |
| char\* strstr(const char\* string, const char\* Search) | Выполняет поиск строки ch слева направо в строке string; возвращает указатель на пер- вое вхождение символа. Если символ не обнаружен, воз-  вращает NULL (нулевой указа- тель) | string |
| char\* strcpy(char \*Dest, const char \*Source) | Копирует строку Source в строку Dest и возвращает ука-  затель на Dest | string |
| char\* strncpy(char \*Dest, const char \*Source, size\_t count) | Копирует не более чем count символов из строки Source в  строку Dest и возвращает ука- затель на Dest | string |
| int strcmp(const char\* string1, const char\* string2) | Сравнивает строки string1, string2 лексикографически: возвращает –1, если string1< string2; 0, если равны; +1, если  string1> string2 | string |
| double atof (const char\* str) | Преобразует строку str в ве-  щественное число | stdlib |
| int atoi (const char\* str) | Преобразует строку str в целое  число | stdlib |