**Занятие 1. Типы данных в С++**

**Ввод и вывод данных.**

**1.1. Типы данных в С++**

Программы на С++ (как, впрочем, и на любом другом языке) работают с данными, расположенными в ячейках памяти. Каждая такая ячейка памяти имеет конкретный системный адрес. По этому адресу можно обратиться с целью получения хранящегося там значения данных с целью записи туда другого значения. Но для удобства программирования вместо системных адресов используют имена ячеек. Такая именованная ячейка памяти называется переменной. Переменные полезны тем, что позволяют присваивать короткий и легко запоминающийся идентификатор для данных, которые планируется использовать.

Переменные могут быть одного из следующих типов.

Таблица 1

Основные типы данных С++

|  |  |  |  |
| --- | --- | --- | --- |
| **Название типа** | **Обозначение** | Байт | Диапазон принимаемых значений |
| *1* | *2* | *3* | *4* |
| Целочисленный  Логический | Bool | 1 | 0 … 255  False (ложь) или True (истина) |
| Целочисленный (символьный) тип данных | Char  signed char  unsigned char | 1 | -128 … 127  0 … 255 |
| Целочисленные типы данных | | | |
| Короткое целое | short int | 2 | -32 768 … 32 767 |
| Короткое целое без знака | unsigned short int | 2 | 0 … 65 535 |
| Целое | int | 4 | -2 147 483 648 … 2 147 483 647 |
| Целое без знака | unsigned int | 4 | 0 … 4 294 967 295 |
| Длинное целое | long int | 4 | -2 147 483 648 … 2 147 483 647 |
| Длинное целое без знака | unsigned long int | 4 | 0 … 4 294 967 295 |
| Типы данных с плавающей точкой | | | |
| Вещественный одинарной точности | float | 4 | -2 147 483 648.0 … 2 147 483 647.0 |
|  | long float | 8 | -9 223 372 036 854 775 808 .0 … 9 223 372 036 854 775 807.0 |
| Вещественный двойной точности | double | 8 | -9 223 372 036 854 775 808 .0 …  9 223 372 036 854 775 807.0 |

В строгих языках программирования перед использованием переменных необходимо резервировать пространство в памяти ЭВМ. Оно будет использоваться при выполнении программы. Резервирование памяти под переменную называется ***объявлением переменной.*** В C++ объявление осуществляется по следующим правилам.

Имя переменной выбирает составитель программы; имя переменной должно начинаться с буквы латинского алфавита и может содержать буквы латинского алфавита, цифры и символы подчеркивания. Заглавные и строчные буквы считаются разными.

Примеры имен переменных: **х, y, summa, s1, srednee\_ar** и т.д.

Имена переменных не должны совпадать с ключевыми словами языка С++.

Чтобы использовать в программе переменную, необходимо:

1. *объявить переменную* в начале программы, явно указав тип данных для переменной.

Пример объявления переменных

**int a;** // объявление переменной a целого типа.

**float b;** // объявление переменной b типа данных с плавающей запятой.

1. *Инициализация переменной* – это присвоение начального значения при объявлении. Инициализатор должен быть константой, значение которой присваивается переменной или выражением с использованием значений переменных, которые были ранее определены.

Пример:

double c = 14.2; // инициализация переменной типа double.

char d = 's'; // инициализация переменной типа char.

bool k = true; // инициализация логической переменной k.

z=x+m;

*Примечание*.

Если переменная не будет объявлена, но будет использоваться далее в программе, то программа не запустится, компилятор выдаст ошибку.

Если переменная не будет проинициализирована, то компилятор не выдаст ошибки, но расчеты будут выполнены неверно.

Для числовых переменных определены простейшие арифметические операции, которые приведены в табл. 2. Для их использования **не нужно** подключать библиотеку.

Таблица 2

Основные арифметические операции С++

|  |  |  |
| --- | --- | --- |
| **Операция в С++** | **Описание** | **Пример в программе** |
| **=** | присвоение | х=0.4; |
| **+** | сложение | х=8+3;  y=x+7.9; |
| **-** | вычитание | x=8-3;  y=x-7.9; |
| **\*** | умножение | x=4\*2.5;  y=x\*4; |
| **/** | деление. Результатом деления целых чисел является целое число. | x=7.5/3.2;  y=x/2.6;  z=1/3; //z=0 (округление до целого, т.к. делятся целые числа)  z=1.0/3.0; //z=0.333(3) |
| % | Деление по модулю (нельзя применять к типам с плавающей точкой (float или double) | z=10 % 3 // z =1. |
| -- | Декремент  (уменьшает значение переменной на 1) | x-- |
| ++ | Инкремент  (увеличивает значение переменной на 1) | x++ |

Операторы в выражении вычисляются слева направо в соответствии с приоритетами арифметических операций. Изменить порядок вычисления можно с помощью скобок.

Для каждого из арифметических операторов есть форма, в которой одновременно с заданной операцией выполняется присваивание:

|  |  |
| --- | --- |
| оператор | эквивалентный оператор присваивания |
| X+=Y | X=X+Y |
| X-=Y | X=X-Y |
| X\*=Y | X=X\*Y |
| X/=Y | X=X/Y |
| X%=Y | X=X%Y |

В языке С++ допускаются обе формы записи операции присвоения.

Для сравнения переменных и чисел используются встроенные знаки сравнения, которые приведены в табл. 3. Для их использования **не нужно** подключать библиотеку.

Таблица 3

Операторы сравнения и логические операторы С++

|  |  |  |
| --- | --- | --- |
| **Сравнение в С++** | **Описание** | **Пример в программе** |
| > | больше чем | x>0 |
| < | меньше чем | y<z |
| >= | больше или равно | y>=x |
| <= | меньше или равно | z<=8.56 |
| == | проверка на равенство | x==0.7 |
| != | не равно | x!=y |
| && | логическое И | x>0 && x<1 // двойное неравенство 0<x<1 |
| || | логическое ИЛИ | s<8 || s>10 |

В противоположность большинству языков, С поддерживает все существующие битовые операторы. Поскольку С создавался, чтобы заменить ассемблер, то была необходимость поддержки всех (или по крайней мере большинства) операции, которые может выполнить ассемблер. Битовые операции — это тестирование, установка или сдвиг битов в байте или слове, которые соответствуют стандартным типам языка С char и int. Битовые операторы не могут использоваться с float, double, long double, void и другими сложными типами. Таблица содержит имеющиеся операторы. Примеры приведены для значений *a=5* (00000101) и *b=6* (00000110).

Таблица 4

Битовые операторы

| **Оператор** | **Описание** | **Пример в программе** | **Результат вычисления** |
| --- | --- | --- | --- |
| & | И | c=a&b | 00000100==4 |
| | | ИЛИ | d=a|b | 00000111==7 |
| ^ | Исключающее ИЛИ | e=a^b | 00000011==3 |
| ~ | Дополнение(Отрицание) | f=~a | 11111010==250 |
| >> | Сдвиг вправо | c=a>>1 | 00000010==3 |
| << | Сдвиг влево | c=a<<2 | 00010100==20 |

**Программа на С++** состоит из объявлений **(**переменных, констант, типов, классов, функций)и описаний функций.

Главной и обязательной функцией для консольных проектов является функция **main( )** для Win32 проектов wWinmain( ). Функция main( ) выполняется сразу после запуска готового приложения. Программа состоит из модулей, деление программы на модули осуществляют в соответствии с выполняемыми функциями задачами. Именно в модуле происходит программирование задачи. Каждый модуль состоит из заголовочного файла с расширением **.h** и файла реализации с расширением **.cpp**. В заголовочном файле содержатся объявления классов, функций, переменных и т.д. Увидеть эти файлы можно в окне обозревателя решений. После компиляции программы для каждого **.cpp** файла создается объектный файл **.obj**, затем компоновщик объединяет все объектные файлы в один исполняемый загрузочный модуль с расширением **exe**.

Заголовочные файлы включаются в текст программы с помощью директивы препроцессора **#include**. Директивы препроцессора начинаются со знака "диез" (#), который должен быть самым первым символом строки. Программа, которая обрабатывает эти директивы, называется препроцессором (в современных компиляторах препроцессор обычно является частью самого компилятора).

Директива **#include** включает в программу содержимое указанного файла. Имя файла может быть указано двумя способами:

#include <some\_file.h>

#include "my\_file.h"

Если имя файла заключено в угловые скобки (<>), считается, что нам нужен некий стандартный заголовочный файл, и компилятор ищет этот файл в предопределенных местах. (Способ определения этих мест сильно различается для разных платформ и реализаций.) Двойные кавычки означают, что заголовочный файл - пользовательский, и его поиск начинается с того каталога, где находится исходный текст программы.

**2. Операции ввода/вывода**

Для ввода/вывода значений переменных в программу С++ располагает следующими возможностями.

*2.1. Потоковый ввод-вывод в С++.*

В С++ имеются специфические средства ввода-вывода. Это библиотека классов, подключаемая к программе с помощью файла **iostream.h**. В этой библиотеке определены в качестве объектов стандартные символьные потоки со следующими именами:

**cin** – стандартный поток ввода с клавиатуры;

**cout** – стандартный поток вывода на экран.

Ввод данных интерпретируется как извлечение из потока **cin** и присваивание значений соответствующим переменным. В С++ определена операция извлечения из стандартного потока, знак которой **>>**. Например, ввод значений в переменную **х** реализуется оператором: **cin>>x**;

Вывод данных интерпретируется как помещение в стандартный поток **cout** выводимых значений. Выводиться могут тексты, заключенные в двойные кавычки, и значения выражений. Знак операции помещения в поток **<<.**

Примеры использования потокового ввода/вывода:

cout<<"fraza"; //выведет на экран слово fraza. Может вывести любой текст.

cout<<x; //выведет на экран число, хранящееся в переменной х.

cin>>x; //присваивает переменной х значение, введенное с клавиатуры.

cin>>x>>y; //присваивает переменной х первое введенное с клавиатуры значение, переменной у – второе.

// endl – оператор перевода каретки на экране на следующую строку.

cout<<a+b;

cout<<"\nРезультат="<<Y;

cout<<"x="<<x<<" y="<<y<<" z = "<<z<<endl;

В выходном потоке можно использовать управляющие символы, перед каждым элементом вывода нужно ставить знак операции <<. Элемент вывода **endl** является манипулятором, определяющим перевод курсора на новую строку (действует аналогично управляющему символу **\n**).

В процессе потокового ввода-вывода происходит преобразование из формы внешнего символьного представления во внутренний формат и обратно. Формат определяются автоматически.

Запись

#include <iostream> // директива препроцессора

#include <cmath > // директива препроцессора для функции sqrt

using namespace std; // директива использования пространства имен std

int main()

{

float a, b, c, p, s;

cout<<"\na="; cin>>a;

cout<<"\nb="; cin>>b;

cout<<"\nc="; cin>>c;

p=(a+b+c)/2;

s=sqrt(p\*(p-a)\*(p-b)\*(p-c) ) ;

cout<<"\nПлощадь треугольника="<<s;

system(«pause»);

}

Примеры

#include "stdafx.h" // директива препроцессора

#include <iostream> // директива препроцессора

using namespace std; // директива использования пространства имен std

int main()

{

setlocale(LC\_ALL, "rus"); // функция для отображения символов русского алфавита в консоль

cout << "Come up and C++ me sometime."<<endl; // вывод в консоль

cout << "You won't regret it"<<endl;

cout << "ПРИВЕТ!!!"<<endl;

system(«pause»); // консоль ожидает ввода символа для дальнейшего выполнения (закрытия)

return 0;

}

#include "stdafx.h" // директива препроцессора

#include <iostream> // директива препроцессора

using namespace std; // директива использования пространства имен std

int main()

{

setlocale(LC\_ALL, "rus"); // функция для отображения символов русского алфавита в консоль

int carrot;

carrot = 25;

cout << "У меня " << carrot << " морковок :))"<< endl; // вывод в консоль

carrot = carrot - 1;

cout << "ХРУМ -хрум. А сейчас у меня " << carrot << " морковки" << endl;

system(«pause»); // консоль ожидает ввода символа для дальнейшего выполнения (закрытия)

return 0;

}

*2.2. Использование манипуляторов ввода-вывода*

Управляющие символы влияют на расположение на экране выводимых знаков. Признаком управляющего символа является значок \. Ниже приводится список основных управляющих символов:

\n — перевод строки;

\t — горизонтальная табуляция;

\r — возврат курсора к началу новой строки;

\а — сигнал-звонок;

Манипуляторы позволяют встраивать инструкции форматирования в выражение ввода-вывода. Наиболее часто используемые стандартные манипуляторы описаны в таблице 3.

Таблица 4

Основные манипуляторы

|  |  |  |
| --- | --- | --- |
| **Манипулятор** | **Назначение** | **Функция** |
| endl | Выводит символ новой строки и "сбрасывает" поток, т.е. переписывает содержимое буфера, связанного с потоком, на соответствующее устройство | Вывод |
| flush | "Сбрасывает" поток | Вывод |
| hex | Устанавливает флаг hex | Ввод-вывод |
| oct | Устанавливает флаг oct | Ввод-вывод |
| right | Устанавливает флаг right | Вывод |
| scientific | Устанавливает флаг scientific | Вывод |
| setbase  (int base) | Устанавливает основание системы счисления равной значению base | Вывод |
| setfill(int ch) | Устанавливает символ-заполнитель равным значению параметра ch | Вывод |
| setprecision  (int p) | Устанавливает количество цифр точности (после десятичной точки) | Вывод |
| setw (int w) | Устанавливает ширину поля, равной значению параметра w | Вывод |

При использовании манипуляторов, которые принимают аргументы, необходимо включить в программу заголовок **<iomanip>.**

Манипулятор используется как часть выражения ввода-вывода. Вот пример программы, в которой показано, как с помощью манипуляторов можно управлять форматированием выводимых данных.

#include "stdafx.h"

#include <iostream>

#include <iomanip>

using namespace std;

main()

{

cout << setprecision (2) << 1000.243<< endl;

cout << setw(20) << "Всем привет! ";

}

Результаты выполнения этой программы таковы.

*1е+003*

*Всем привет!*

// урок на использование esc-символов

#include <iostream> // директива препроцессора

using namespace std; // директива использования пространства имен std

/\*

\b удаление последнего выведенного символа

\n перейти на начало новой строки

\t перейти к следующей позиции табуляции

\\ вывести обратную косую черту \

\" вывести двойную кавычку "

\' вывести одинарную кавычку '

\*/

int main()

{

setlocale(LC\_ALL, "rus"); // функция для отображения символов русского алфавита в консоль

cout << "Come up and C++ me sometime.\n"; // вывод в консоль и переход на новую строку

cout << "You won't regret it \n\n"; // вывод в консоль и переход на новую строку дважды

cout << "\"ПРИВЕТ!!!\"\t\t";

cout << "Hallo!!!\t\t";

system(“pause”);

return 0;

}

*2.3. Форматированный ввод/вывод с клавиатуры*

Оператор вызова функции scanf () имеет следующую структуру:

**scanf (форматная\_\_строка, список\_аргументов);**

Данная функция осуществляет чтение символов, вводимых с клавиатуры, и преобразование их во внутреннее представление в соответствии с типом величин.

Символьную последовательность, вводимую с клавиатуры и воспринимаемую функцией scanf(), принято называть входным потоком. Функция scanf () разделяет этот поток на отдельные вводимые величины, интерпретирует их в соответствии с указанным типом и форматом и присваивает переменным, содержащимся в списке аргументов. Разделителем в потоке ввода между различными значениями может быть любое количество пробелов, а также другие пробельные символы: знак табуляции, конец строки. Только после нажатия на клавишу Enter вводимые значения присвоятся соответствующим переменным. До этого входной поток помещается в буфер клавиатуры и может редактироваться.

Список аргументов — это перечень вводимых переменных, причем перед именем каждой переменной ставится значок &. Это знак операции «взятие адреса переменной». Форматная строка заключается в кавычки (как и для printf) и состоит из списка спецификаций. Каждая спецификация начинается со знака %.

Оператор вызова функции printf () имеет следующую структуру:

**printf (форматная\_строка, список\_аргументов);**

Форматная строка ограничена двойными кавычками и может включать в себя произвольный текст, управляющие символы и спецификаторы формата. Список аргументов может отсутствовать или же состоять из выражений, значения которых выводятся на экран (в частном случае из констант и переменных).

Спецификатор формата определяет форму внешнего представления выводимой величины. Вот некоторые спецификаторы формата:

%с — символ;

%d — целое десятичное число (тип int);

%u — целое десятичное число без знака (тип unsigned);

%f — вещественные числа в форме с фиксированной точкой.

К спецификатору формата могут быть добавлены числовые параметры: ширина поля и точность. Ширина – это число позиций, отводимых на экране под величину, а точность – число позиций под дробную часть (после точки). Параметры записываются между значком % и символом формата и отделяются друг от друга точкой. Если в пределы указанной ширины поля выводимое значение не помещается, то этот параметр игнорируется и величина будет выводиться полностью.

**ЗАДАНИЕ 1**

|  |  |  |  |
| --- | --- | --- | --- |
| № | F(x) | Проверочные значения | |
| X1 | X2 |
| 1 |  | 2 | 4 |
| 2 |  | 1 | 3 |
| 3 |  | 1 | 3 |
| 4 |  | 2 | 5 |
| 5 |  | 1 | 4 |
| 6 |  | 1 | 5 |
| 7 |  | 1 | 4 |
| 8 |  | 1 | 2 |
| 9 |  | 3 | 5 |
| 10 |  | 4 | 6 |
| 11 |  | 1 | 3 |
| 12 |  | 1 | 5 |
| 13 |  | 1 | 3 |
| 14 |  | 2 | 5 |
| 15 |  | 1 | 5 |

**Занятие 2.**

**Линейные алгоритмы**

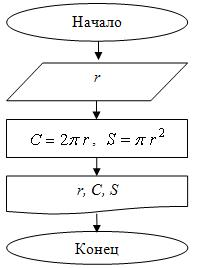
## 2.1. Примеры

1.Условие задачи

*Найти длину окружности и площадь круга, если известен радиус.*

Решение. Введем обозначения: r – радиус, который будет считываться с клавиатуры (т.к. он не задан конкретным числом); C – длина окружности, вычисляемая по формуле C=2\pi r; S – площадь круга, вычисляемая по формуле S=\pi r^2.

Приведем блок-схему на рис. 1.



**Рис. 1.** Блок-схема для примера

Код программы (Visual Studio):

#include <iostream>

#include <math.h> // для некоторых версий #include <cmath>

using namespace std;

int main()

{

setlocale(LC\_ALL,»rus»);

int r;

float l, s;

const float PI = 3.14;

cout << «введите радиус окружности:»;

cin >> r;

l = 2.0 \* PI \* r ;

s=PI\*pow(r, 2.0);

cout << «Длина окружности=« << l << endl;

cout << «Площадь окружности=« << s << endl;

system(«pause»);

return 0;

}

2.Условие задачи

*По условиям договора зарплату за месяц выдают дважды: сначала часть ее в виде аванса, а по истечении месяца – остальную часть. При этом при окончательном расчете из заработной платы удерживают аванс и величину подоходного налога (13%).*

*Составить программу, которая по введенному с клавиатуры значению зарплаты и процентной доле аванса выводит на экран значения обоих частей зарплаты.*

2. Алгоритм

Начало

Ввести значение оклада

Ввести долю аванса

Рассчитать аванс

Рассчитать налог

Рассчитать окончательную заработную плату

Вывести значение аванса

Вывести значение заработной платы

Конец

3. Программа

#include «stdafx.h»

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, «rus»);

double oklad, d, avance, nalog, plata;

cout << «Введите величину оклада:»;

cin >> oklad;

cout << «Введите % долю аванса:»;

cin >> d;

avance = oklad \* d / 100;

nalog = oklad \* 0.13;

plata = oklad - avance - nalog;

cout << «Аванс=« << avance << « руб.» <<endl;

cout << «Расчет=« <<plata << « руб.» << endl;

system(«pause»);

return 0;

}

**ЗАДАНИЕ 2**

1. В классе имеется N1 мальчиков и N2 девочек. Вычислить сколько процентов от численности класса составляют мальчики и сколько девочки.
2. Известны количество жителей в государстве и площадь его территории. Определить плотность населения в этом государстве.
3. Известны объем и масса тела. Определить плотность материала этого тела.
4. Вам под коттедж отвели участок площадью S м2. Но оказалось, что S1 м2 этого участка занята болотом. Вычислить сколько % участка занимает болото.
5. С владельцев автомобилей взымается так называемый транспортный налог. Рассчитывается он следующим образом:

***Налог = Мощность двигателя \* 5.***

Составить программу расчета транспортного налога на ваш автомобиль.

1. Вступительные экзамены в колледж сдавало N0 человек. Из них получили двойки N1 человек, а на одни пятерки сдало N2 человек, которые и прошли в колледж без конкурса. Остальные абитуриенты вынуждены были участвовать в конкурсе на поступление и доказывать свое право учиться в колледже с помощью дополнительных показателей. Определить, сколько процентов абитуриентов от общего их числа вынуждены были участвовать в дополнительном конкурсе.
2. На курсе числится N0 студентов. Из них N1 студентов участвуют только в спортивных мероприятиях, N2 студентов участвуют только в художественной самодеятельности, а N3 участвуют и в художественной самодеятельности и в спортивных мероприятиях. Определить сколько % студентов не участвуют ни в никаких мероприятиях.
3. Плата за обучение в вузе в первый год обучения равнялась S1 руб. На следующий год она стала равна S2 руб. На третий год – S3 руб. Определить величину роста оплаты (в %) после первого и второго года обучения.
4. Некая студентка решила поучаствовать в кастинге конкурса «Мисс Очарование». Для этого она решила сесть на специальную диету. Начальный ее вес составлял M0 кг. Через неделю ее вес стал равен M1 кг, а еще через неделю – M2 кг. Определить, по сколько процентов веса каждую неделю теряла студентка, сидя на диете.
5. Некая студентка на день св. Валентина получила N1 валентинок, а сама послала N2 валентинок. Определить сколько процентов от общего числа валентинок, прошедших через ее руки, составили полученные валентинки.
6. Студентке на карманные расходы было выдано S0 руб. Из них S1 руб. было потрачено на косметику, а остальные – на походы в дискотеки. Определить, сколько процентов от общей суммы составила каждая статья расходов.
7. У студента в семестре по плану должно было быть N0 занятий. Из них по уважительной причине было пропущено N1 занятий, а по неуважительной – N2 занятий. Определить, сколько % от общего числа занятий было посещено студентом.

**Занятие 3. Линейные алгоритмы.**

**Математические возможности С++**

**3.1. Общие сведения**

В С++, как и в любом современном языке программирования, имеется большой набор стандартных математических операций и встроенных функций. Их список приводится в таблицах.

Для числовых переменных определены простейшие арифметические операции, которые приведены в табл. 3.1. Для их использования не нужно подключать библиотеку.

Таблица 3.1

Основные арифметические операции С++

|  |  |  |
| --- | --- | --- |
| **Операция в С++** | **Описание** | **Пример в программе** |
| = | присвоение | х=0.4; |
| + | сложение | х=8+3;  y=x+7.9; |
| - | вычитание | x=8-3;  y=x-7.9; |
| \* | умножение | x=4\*2.5;  y=x\*4; |
| / | Деление. Результатом деления целых чисел является целое число. | x=7.5/3.2;  y=x/2.6;  z=1/3; //z=0 (округление до целого, т.к. делятся целые числа)  z=1.0/3.0; //z=0.333(3) |
| % | Деление по модулю (нельзя применять к типам с плавающей точкой (float или double) | z=10 % 3 // z =1. |
| -- | Декремент | x-- |
| ++ | Инкремент | x++ |

Математические функции, выполняющие некоторые часто используемые математические задачи, в С++ определены в заголовочном файле <cmath>. Например, нахождение корня, возведение в степень, sin(), cos() и многие другие. В таблице 3.2 собраны основные математические функции. Все они имеют вещественные аргументы и возвращаемый результат.

Таблица 3.2

Основные математические функции С++

|  |  |  |
| --- | --- | --- |
|  | | |
| **Математическая функция** | **Программная запись** | **Описание** |
| |x| | fabs(x) | Модуль числа. |
| \sin x | sin(x) | Синус числа, аргумент в радианах. |
| \cos x | cos(x) | Косинус числа, аргумент в радианах. |
| \tg x | tan(x) | Тангенс числа, аргумент в радианах. |
| e^x | exp(x) | Экспонента числа. |
| \ln x | log(x) | Натуральный логарифм числа. |
| \lg x | log10(x) | Десятичный логарифм числа. |
| x^y | pow(x, y) | х в степени y. |
| 10^x | pow10(x) | Степень десяти. |
| \sqrt{x} | sqrt(x) | Квадратный корень из числа. |
| \arcsin x | asin(x) | Арксинус числа, в радианах. |
| \arccos x | acos(x) | Арккосинус числа, в радианах. |
| \arctg x | atan(x) | Арктангенс числа, в радианах. |
| \pi | M\_PI | Число \pi=3.141593 |

**3.2. Вычисления сложных выражений**

Вычисления в сложных выражениях производятся слева направо с учетом приоритета операций. Сначала выполняются операции высшего приоритета, затем более низкого. Если приоритет соседних операций одинаков, то сначала выполняется левая. Операции в порядке уменьшения приоритета располагаются следующим образом:

**Функции > ^** **> \* = / = \ = Mod > + = -**

Используя круглые скобки можно изменить порядок вычислений

# Примеры выражений

Таблица 3.3

|  |  |
| --- | --- |
| Выражение | Запись в C++ |
| *х*+3 | *х*+3 |
|  | (*x*+*y*)/4 |
|  | x+y/4 |
|  | log(2\*pow(x,2.0)) |
|  | exp(-fabs(*x*)) |
|  | pow((*x+*5),(1.0/3)) |
|  | log10(3*\*x+*5) |
|  | pow(sin(*x*),2)+ cos(pow(*x*,2)) |
|  | pow(log(fabs(pow(*x*,(5.0/3))))/Log(7),2) |

При вычислении сложных выражений удобнее разбить это выражение на ряд простых, вычислить их отдельно и затем вычислить исходное.

Например.

Составить программу для вычисления следующего выражения:



Выражение очень сложное. В частности, при его наборе очень легко ошибиться в расстановке скобок. Для снижения вероятности ошибок его можно разбить на три части и вычислить отдельно каждую из них. Тогда текст вычислительной части программы будет иметь вид:

a = 1 + exp(-pow(x, 2)) + sqrt(tan(abs(x)) - 1 / tan(1.0 / x));

b = log(1 + pow(x, 4)) / log(3) - pow(sin(x) + cos(x), 1.0 / 5);

c = atan(x / 2.0) - 3;

y = a / b \* c;

3. Программа

// пример на вычисление значения функции

#include «stdafx.h»

#include <iostream>

#include <math.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, «rus»);

int x, y, c, a, b;

cout << «введите значение x=«;

cin >> x;

a = 1 + exp(-pow(x, 2)) + sqrt(tan(abs(x)) - 1 / tan(1.0 / x));

b = log(1 + pow(x, 4)) / log(3) - pow(sin(x) + cos(x), 1.0 / 5);

c = atan(x / 2.0) - 3;

y = a / b \* c;

cout << «y=« <<y << endl;

system(«pause»);

return 0;

}

## Условные операторы

## 

**4.1 Условный оператор if** позволяет выбрать одну из двух ветвей вычислительного процесса

Общий вид оператора

if (выражение)

{

Операторы1

}

[Else

{

Операторы2

}]

Выражение в скобках после **if** является логическим выражением, значением которого является одно из двух логических значений true (истина) или false (ложь). В первом случае выполняются Операторы1, во втором – Операторы2.

Вторая ветвь (Операторы2) может отсутствовать. Об этом говорит наличие квадратных скобок в определении оператора. (Здесь и далее то, что заключено в квадратные скобки, не является обязательным.)

После выполнения какой-либо одной из ветвей условный оператор считается выполненным. Наличие фигурных скобок не является обязательным, если какая-либо ветвь содержит один оператор, в котором нет объявления переменных. Однако рекомендуется фигурные скобки использовать всегда, что делает структуру программы более наглядной.

Например,

int i = 1, x;

if (i <= 10)

{

x = 5;

}

else

x = 0;

cout<<x;

Если требуется выполнить более одного оператора после if или после else, то их все обязательно помещают внутри скобок {}. Эти операторы образуют *блок*. Например,

int i = 15, x;

if (i <= 10)

{

x = 5;

cout<<x;

}

else

{

x = 2;

cout<<x;

}

Каждый из двух блоков условного оператора (наличие обоих блоков необязательно) может содержать любые операторы, в том числе условные операторы. Если условный оператор входит в состав блока Операторы1, выполняемого после if, то он записывается и выполняется по общим правилам. Если проверка условия осуществляется после else, то используется оператор else if (условие), использование которого демонстрирует следующая схема:

if (условие1)

{

Операторы1

}

else if (условие2)

{

Операторы2

}

[else

{

Операторы3

}]

Если истинно условие1, то выполняются Операторы1 и условный оператор заканчивает работу. Если условие1 ложно, то проверяется условие2, и, если оно истинно, то выполняются Операторы2, в противном случае выполняются Операторы3, если присутствует последний else,или конструкция не выполняет никаких действий. Последний else всегда относится к последнему if, для которого еще не было соответствующего else. Например,

#include <iostream>

using namespace std;

int main()

{

setlocale(0, ««);

double num;

cout << «Введите произвольное число: «;

cin >> num;

if (num < 10) // Если введенное число меньше 10.

cout << «Это число меньше 10.» << endl;

else if (num == 10)

cout << «Это число равно 10.» << endl;

else // иначе

cout << «Это число больше 10.» << endl;

return 0;

}

*Замечание*.

В рассмотренном примере после каждого условия следует один оператор, поэтому фигурные скобки необязательны.

В качестве условия может быть любое выражение с логическим результатом. Логические выражения могут быть очень сложными и при их составлении можно использовать следующие логические операторы (табл. 1). Для их использования **не нужно** подключать библиотеку.

Таблица 1

Логические операции С++

|  |  |  |
| --- | --- | --- |
| **Сравнение в С++** | **Описание** | **Пример в программе** |
| > | больше чем | x>0 |
| < | меньше чем | y<z |
| >= | больше или равно | y>=x |
| <= | меньше или равно | z<=8.56 |
| == | проверка на равенство | x==0.7 |
| ! | не (отрицание) | X!=5 |
| && | логическое И | x>0 && x<1 // двойное неравенство 0<x<1 |
| || | логическое ИЛИ | s<8 || s>10 |

Их применение позволяет иногда резко упростить конструкцию **If** .Например:

**If (x > 0 && x < 10) …**

Следует иметь в виду, что и для логических операций имеется понятие приоритета. Для них приоритет понижается в следующей последовательности:

**! > && > ||**

Поэтому при составлении логических выражений, как и в случае математических операций, для придания нужного порядка вычислений необходимо использовать круглые скобки.

Например:

If (x==2 && y<0 || y>3) …

If (x==2 && (y<0 || y>3)) …

Первое условие выполнится, при x=2 и одновременно y<0 или вне зависимости от значения x при y>3.

Второе условие выполнится, если x=2 и одновременно либо y<0 либо y>3.

**ЗАДАНИЕ 3**

Необходимо решить на компьютере задачу вычисления значения функции y = f(x).

|  |  |
| --- | --- |
| 1. | 2. |
| 3. | 4. |
| 5. | 6. |
| 7. | 8. |
| 9. | 10. |
| 11. | 12. |
| 13. | 14. |
| 15. |  |

**Задание 4 (ЕГО ДЕЛАТЬ НЕ НАДО)**

4.3 Написать программу, которая определяет, попадает ли точка с введенными с клавиатуры координатами (x,y) в заштрихованную область.

В качестве примера рассмотрим задачу для фигуры, приведенной на рис. 1.

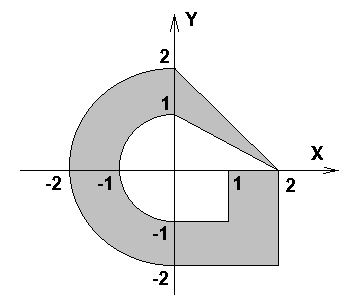


Рис. 1.

Определяем апроксимирующие функции. В левой части фигуры это полукольцо, образованное полуокружностями радиуса 1 и 2, и центром в начале координат, а в правой – отрезки прямых. В первой четверти уравнение верхней прямой имеет вид: y=2 – x, а уравнение нижней – y=1 – x / 2.

Остальные этапы алгоритма проще всего показать в виде следующей программы.

#include <iostream>

#include <cmath>

#include <cstdlib>

using namespace std;

int main()

{

setlocale(LC\_ALL,»rus»);

double x, y, l;

bool p1, p2, p3, p;

cout << «введите значение координаты х=«; cin >> x;

cout << «введите значение координаты y=«; cin >> y;

p1 =(x > 0) && (y > 0) && (y < 2-x) && (y>1-x / 2); // Условия попадания в первую

четверть

p2 =(x > 0) && (x<2) && (y < 0) && (y>-2) && (x > 1 || y < -1); // Условия попадания

в четвертую четверть

l = sqrt(x\*x+y\*y); // уравнение окружности

p3 = x < 0 && l < 2 && l > 1; // Условия попадания во вторую и третью четверти

p = p1 || p2 || p3; // общее условие попадания точки в заданную область

if (p)

cout << «Точка попала в фигуру» << endl;

else

cout << «Точка не попала в фигуру» << endl;

system(«pause»);

return 0;

}

Обратите внимание на скобки в выражении условия попадания точки в четвертой четверти. Они необходимы для нужного порядка вычислений.

|  |  |
| --- | --- |
| 1.  뿷膒bbᡭ뿷ⳟ腲 | 2.  뿷膒bbᡭ뿷᏷腲 |
| 3.  뿷膒bbᡭ뿷ⱗ腲 | 4.  뿷膒bbᡭ뿷ⱗ腲 |
| 5.  뿷膒bbᡭ뿷ⱗ腲 | 6.  뿷膒bbᡭ뿷ⳟ腲 |
| 7.  뿷膒bbᡭ뿷ⱗ腲 | 8.  뿷膒bbᡭ뿷᏷腲 |
| 9.  뿷膒bbᡭ뿷ⱗ腲 | 10.  뿷膒bbᡭ뿷ⱗ腲 |
| 11.  뿷膒bbᡭ뿷᏷腲 | 12.  뿷膒bbᡭ뿷ⱗ腲 |
| 13.  뿷膒bbᡭ뿷ⱗ腲 | 14.  뿷膒bbᡭ뿷ⳟ腲 |
| 15. |  |