3. Простейшие операторы языка С и C++. Арифметические операции и арифметические выражения

Ввод и вывод данных в языке C++ можно осуществлять двояко: «в стиле C» и «потоково». Мы будем использовать оба способа (хотя в одной программе их смешивать не рекомендуется).

**Ввод и вывод в стиле C** требует включения заголовочного файла с именем stdio.h, т. е. присутствия в программе директивы препроцессора

#include <stdio.h>

Ввод данных осуществляется с помощью функции scanf. Её форма записи:

scanf(”*строка\_форматов”*, *список\_адресов\_величин*);

Здесь ”*строка\_форматов”* — это последовательность спецификаторов форматов для в вводимых величин, выписанная в порядке следования идущих далее адресов читаемых величин. Воспринимается компилятором как символьная строка, поэтому как и любая символьная строка должна заключаться в апострофы. **Спецификатор формата** — это условное обозначение, начинающееся с символа %, сообщающее о типе читаемых данных. Полный список спецификаторов можно найти в [2], здесь мы приведём только некоторые из них:

%c — одиночный символ (тип char);

%d — тип int;

%f — тип float;

%lf — тип double;

%e — вещественный формат с плавающей точкой (экспоненциальный формат);

%s — символьная строка.

Вывод данных осуществляется с помощью функции printf. Её форма записи:

printf(”*строка\_с\_форматами”*, *список\_вывода*);

Здесь ”*строка\_с\_форматами”* представляет собой окончательный вид того, что должно получиться на выходе, при этом на место выводимых величин (из идущего далее списка) ставятся их спецификаторы форматов (в порядке следования). Кроме спецификаторов формата могут использоваться **модификаторы**. Они ставятся между знаком % и соответствующей буквой и представляют собой либо одно число, либо два числа, написанные через точку, первое из которых задаёт общее количество позиций, отводимых под число, а второе (при его наличии) — сколько из этих позиций отводится под дробную часть числа. Если указанного общего числа позиций недостаточно, происходит его автоматическое расширение до необходимого минимума, если их, наоборот, больше, чем нужно, остаются пробелы слева (при наличии знака минус перед модификатором — справа).

В частности, команда

scanf(”%d%lf%lf”,&n,&x,&y);

означает, что будут прочитаны три переменные: целая n, и две вещественные с двойной точностью x и y.

Команда

printf(”n=%d, z=%5.2f\n”,n,z);

означает, что будут напечатаны два числа: n (целое) и z (вещественное, предварённые, соответственно, текстовыми пояснениями ”n=” и ”z=” и разделённые запятой c пробелом. Число z выйдет в привычном математическом виде (целая и дробная части, разделённые точкой), при этом для числа будут выделены 5 позиций, из них 2 для дробной части. После числа z будет сделан переход на новую строку, что обеспечивается наличием в конце строки специального управляющего символа ’\n’.

**Потоковый ввод-вывод** может использоваться в программе, написанной на зыке C++. Для этого требуется включение заголовочного файла с именем iostream. Обращаем внимание на отсутствие расширения .h — подобные файлы реализованы в более современном формате и требуют, кроме наличия соответствующей директивы препроцессора, подключения к пространству имён std:

#include <iostream>

using namespace std;

**Потоком** в программировании называется последовательность байт, связанная с каким-либо компьютерным устройством. В файле iostream определены потоки cin и cout, связанные, соответственно, с клавиатурой и монитором. Для потока cin определена (перегружена) операция >> (взятие из потока или чтение с клавиатуры), для потока cout — операция << (помещение в поток или вывод на монитор). Приведённые выше примеры команд scanf и printf при потоковом вводе-выводе могут быть записаны так:

cin>>n>>x>>y;

и

cout<<”n=”<<n<<”, z=”<<z<<’\n’;

В этом случае формат вывода компьютер определяет сам. В частности, целые числа выходят так, как они есть, вещественные (которые по абсолютной величине не слишком большие и не слишком маленькие) — в привычном математическим виде, при этом в дробной части ставится до 6 знаков, большие и маленькие вещественные числа — в экспоненциальном формате. Тем не менее, задавать формат при потоковом вводе-выводе возможно, например, с помощью манипуляторов [1], но в данном пособии мы этого вопроса касаться не будем. Одним из манипуляторов является endl, который означает действие по переходу на новую строку (так же, как и символ ’\n’). Соответственно, приведённая выше команда cout может быть записана ещё так:

cout<<”n=”<<n<<”, z=”<<z<<endl;

Над величинами какого-либо типа определён свой набор операций. В частности, над величинами числовых типов определены **арифметические бинарные операции** сложения, вычитания, умножения и деления (записываются, соответственно: +, –, \*, /), а также операция **унарный минус** (взятие противоположного числа). Над целыми числами определена также бинарная операция %, вычисляющая остаток от целочисленного деления левого операнда на правый. Необходимо заметить, что операция деления с помощью «дробной косой черты» (/), применённая к двум целым операндам, производится с округлением результата (путём отбрасывания его дробной части). Если в какой-либо операции участвуют операнды разных типов (целого и вещественного), то происходит неявное преобразование целого типа в вещественный.

Из операндов, знаков операций и скобок (круглых) строятся **выражения** (формулы), которые задают правила вычисления чего-либо. Очерёдность действий в выражении определяется следующим образом. То, что стоит в скобках, вычисляется в первую очередь. Скобки могут быть вложенными, в этом случае вычисление начинается с самых внутренних скобок. Фрагмент выражения, не содержащий скобок, вычисляется в порядке **приоритета** участвующих операций. Чем выше приоритет, тем раньше исполняется операция. Операции равного приоритета исполняются слева направо. В языке C/C++ имеются 14 градаций приоритета. Некоторые из них будут нами приведены далее, пока что заметим, что операции \*, / и % исполняются раньше операций + и – (т. е. имеют более высокий приоритет). В частности, дробь  записывается так: (a+b\*c)/(x–y). При вычислении сложных формул, содержащих различные математические функции, следует использовать стандартные функции, прописанные в заголовочном файле math.h (их список см. в [1,2]).

***Замечание.*** Для корректного подключения файла math.h именовать программный файл следует с расширением .cpp (т. е. отнести программу к языку не C, а C++).

Знак = на языке C/C++ означает **операцию присваивания** («положить равным»). Записывается она следующим образом:

*переменная* = *выражение*;

и действует так: вначале вычисляется значение выражения, стоящее в правой части, затем это значение записывается в переменную, стоящую в левой части.

Допускается также использование **сложных операций присваивания**: +=, –=, \*=, /= и т. п., которые к текущему значению переменной в левой части применяют соответствующую арифметическую операцию со вторым операндом — значением выражения в правой части. Так, например, оператор a+=3; означает обновление значения переменной a путём прибавления к нему числа 3. Операции увеличения и уменьшения на 1 (инкремент и декремент) могут ещё записываться, соответственно, как ++ и -- и имеют две формы записи: префиксную (например, ++a) и постфиксную (например, a++). Префиксная форма означает увеличение операнда на 1 до его использования, постфиксная форма означает увеличение операнда на 1 после его использования.

**Пример 1.** Программа вычисляет длину окружности, площадь круга и объём шара при заданном радиусе r, вводимом с клавиатуры (в стиле C). Для корректного вычислении объема шара используется явное преобразование целого типа в вещественный. Другой вариант — записать число 4 не в целом, а в вещественном виде: 4.0.

#include <stdio.h>  
int main() {

const float pi=3.14159;  
 float r,l,s,v;  
 printf("The program for the calculation of

length, area and volume\n");  
 printf("Input r");  
 scanf("%f",&r);  
 l=2\*pi\*r; //оператор присваивания  
 s=pi\*r\*r; //оператор присваивания  
 v=(float)4/3\*pi\*r\*r\*r; //оператор присв.  
 printf("l=%7.4f\n",l);  
 printf("s=%7.3f\n",s);  
 printf("v=%7.3f\n",v);  
 return 0;

}

*Примечание.* В связи с тем, что функция main имеет тип int, она должна возвращать результат типа int. Это объясняет присутствие формального оператора return 0; в конце текста программы. В современных средах программирования его наличие не обязательно.

**Пример 2.** По заданным *x*, *y* вычислить



В программе используется потоковый ввод-вывод и заголовочный файл math.h (см. [1, 2]), содержащий математические функции, из которых в данном примере используются функции fabs, log, tan, pow.

#include <iostream>

using namespace std;

#include <math.h>  
int main() {

double x,y,z;

cout<<”input x,y\n”; // запрос на ввод

cin>>x>>y;

z:=(fabs(1-x\*x\*x)-log(1+x\*x+y\*y)/log(2))/

(tan(x-2\*y+3)+pow(y\*y\*y\*y+1,1.0/3));

cout<<”z=”<<z<<endl;

return 0;

}

**Пример 3**, представляющий использование операций целочисленного деления. В трёхподъездном девятиэтажном доме (1–9 этажи, по 4 квартиры на этаже) требуется по номеру квартиры *n* (1–108) определить номер подъезда и номер этажа.

#include<stdio.h>  
#include<math.h>  
int main() {  
 int p,k,n,e;  
 printf("Введите n\n"); //запрос на ввод  
 scanf("%d",&n);  
 p=(n-1)/36+1;  
 printf("Номер подъезда =%d\n",p);  
 k=(n-1)%36+1;  
 e=(k-1)/4+1;  
 printf("Номер этажа =%d\n",e);  
 return 0;  
 }

**Задачи**

1. Составить программу для вычисления площади трапеции по заданным основаниям *a*, *b* и высоте *h*.
2. Найти периметр и площадь равнобедренной трапеции с основаниями  и  () и углом при большем основании (угол дан в радианах).
3. Вычислить расстояние между двумя точками на плоскости с заданными координатами (*x*1, *y*1) и (*x*2, *y*2).
4. Треугольник на плоскости задан своими координатами своих вершин. Найти периметр и площадь треугольника (по формуле Герона: , где *a*, *b*, *c* — стороны треугольника, а *p* — его полупериметр).
5. Мельхиор – это сплав никеля и меди, массы которых пропорциональны числам 2 и 9. Сколько потребуется никеля и меди для выплавки *С* *кг* мельхиора?
6. Даны *x*, *y*. Вычислить 
7. От начала суток прошло *t* секунд. *t* целое, в пределах 86400 (86400 – число секунд в сутках). Определить текущее время (часы, минуты, секунды).
8. Ввести целое число от 1 до 99. Найти его сумму цифр.
9. Сделать предыдущую задачу для числа от 1 до 999.
10. Дано число . Дописать в начало и в конец числа цифру *k* и вывести результат.