Арифметические операции

Почти всякое выражение кроме чисел и переменных содержит еще арифметические операции, такие как сложение, вычитание, умножение и деление. Для умножения правила несколько отличаются от математических: в качестве значка для умножения используется звездочка, и пропускать его нельзя. Деление имеет одну особенность: если оно используется для целых чисел, т. е. оба его операнда имеют целый тип, результат будет целым (дробная часть отбрасывается), так что выражение 1/3 имеет достаточно странный результат 0. Если нам нужно получить вещественный результат, нужно явно преобразовать один из операндов к вещественному типу, например так: 1.0/3 . Если в некоторой операции участвуют операнды разных типов, то один из них преобразуется к типу другого (целый и вещественный — к вещественному, занимающие разный объем памяти — к занимающему больший объем, знаковый и беззнаковый — к беззнаковому). Можно, однако, и явно преобразовать результат выражения к другому типу, если это необходимо, при помощи операции преобразования типов. Синтаксис ее таков: (новый_тип)выражение. Смысл этой операции в том, чтобы превратить результат выражения в значение нового_типа, по возможности изменив само значение минимальным образом. Например, результат выражения (double)5 будет 5.0. Эту операцию можно использовать при делении значения одной переменной целого типа на значение другой, тоже целого типа, чтобы результат получился вещественным:

```
int a,b;
cin>>a>>b;
cout<<(double)a/b<<endl;
```

А вот обратное преобразование типов (вещественное значение к целому типу) в большинстве случаев уже меняет значение (происходит округление в сторону 0, т. е. то, что идет после точки, отбрасывается; для отрицательных чисел этот способ округления отличается от взятия целой части на 1).

Здесь же уместно сказать о том, что операция ^ в C++ также имеется, но означает она вовсе не возведение в степень, а что она означает, будет сказано несколько позже.

Кроме того, для целых чисел имеется дополнительная операция % (остаток от деления), которая дает целый результат. Также, разумеется, можно использовать скобки, но только круглые (смысл квадратных скобок будет ясен несколько позже).

Примеры.

1. Вычислить выражение

$$\frac{3+5(6+3)-8\cdot 3+1}{21+50/(3+4(1+2))}$$

и вывести его результат на экран.

Решение:

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
    cout<<(3+5*(6+3)-8*3+1)/(21+50.0/(3+4*(1+2)))<<endl;
    return EXIT_SUCCESS;
}
```

2. Вычислить выражение

$$\frac{(0,25-0,12)0,81+0,13^2-2,7/3,1}{3/7+2,97(8,05-8,1\cdot 6,07)}$$

и вывести его результат на экран.

Решение:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout<<((0.25-0.12)*0.81+0.13*0.13-2.7/3.1)/
          (3.0/7+2.97*(8.05-8.1*6.07))<<endl;
    return EXIT_SUCCESS;
}
```

3. Ввести с клавиатуры 3 вещественных числа a, b и c и вывести значение выражения

$$a + b - c\frac{3ab + a^2}{bc} - (c + \frac{ab}{c})^2$$

на экран.

Решение:

```
#include <iostream>
#include <cstdlib>
#include <cmath>

using namespace std;

int main()
{
    double a = 0, b = 0, c = 0;
    cout<<"a = "; cin>>a;
    cout<<"b = "; cin>>b;
    cout<<"c = "; cin>>c;
    cout<<"Результат = "
        <<(a+b-c*(3*a*b+a*a)/(b*c)-(c+a*b/c)*(c+a*b/c))
        <<endl;
    return EXIT_SUCCESS;
}
```

Задачи.

1. Вычислить выражение

$$\frac{3,1}{5,23^3 + 1} + \frac{4.1^3}{2,72^2 + 1}$$

и вывести его результат на экран.

2. Ввести с клавиатуры 3 вещественных числа a, b и c и вывести значение выражения

$$\frac{(ab + 7c)^3 - (a^2 + b^2 + c^2)}{b - ac - a(b + c))}$$

на экран.

3. Написать программу, вводящую коэффициенты квадратного уравнения и печатающую его дискриминант.

4. Написать программу, вводящую длину радиуса окружности и печатающую ее длину и площадь ограничиваемого ею круга.

5. Направления кодируются следующим образом: 0 — север, 1 — запад, 2 — юг и 3 — восток. Ввести направление и вывести результат его поворота на 90° а) против часовой стрелки; б) по часовой стрелке.

6. Ввести целое число и вывести его цифру в разряде тысяч.

7*. Грузовик может увезти 32 коробки. Ввести число коробок, которые необходимо увезти и вывести минимальное достаточное для этого число грузовиков.

8. Вычислить 20!. Объяснить результат программы.

Одной из особенностей C и C++ является то обстоятельство, что тип переменных для хранения одиночных символов (он называется char) является целочисленным (соответствующая переменная на самом деле хранит код символа, т. е. его номер в специальной таблице символов), и потому символьные переменные вполне могут участвовать в арифметических выражениях, а также имеется возможность превращать целые числа в те символы, которые имеют соответствующий номер.

Пример. Ввести целое число от 1 до 26 и вывести большую латинскую букву, имеющую данный номер в алфавите.

Решение:

```
#include <iostream>
#include <cstdlib>

using namespace std;

int main()
{
```

```
    int n;
    cout<<"Input the number: "; cin>>n;
    cout<<"Numbered letter = "<<(char)('A'-1+n)<<endl;
    return EXIT_SUCCESS;
}
```

Задача.

Ввести оценку по 99-балльной шкале (от 0 до 99) и вывести разряд, соответствующий этой оценке (оценкам от 90 до 99 соответствует разряд 'A', от 80 до 89 — 'B', от 70 до 79 — 'C' и т. д.).

Также в C++ имеются и менее стандартные арифметические операции. К ним относятся операции инкремента и декремента, префиксные и постфиксные. Эти операции имеют один аргумент и могут применяться только к переменным, т. е. так называемым левым выражениям — выражениям, которые могут встречаться в левой части операции присваивания. Это могут быть как обычные переменные из одного из предыдущих разделов, так и ряд других выражений — какие именно, будет сказано несколько позже.

Операции инкремента и декремента, в отличие от обычных операций, рассмотренных ранее, имеют так называемый побочный эффект — они изменяют значение той переменной, к которой применяются.

В следующей таблице показано, как нужно использовать эти операции:

Операция	Выражение	Результат	Значение переменной n после вычисления операции
Префиксный инкремент	++n	$n+1$	$n+1$
Постфиксный инкремент	n++	n	$n+1$
Префиксный декремент	−−n	$n-1$	$n-1$
Постфиксный декремент	n−−	n	$n-1$

Здесь n — переменная, а n — ее значение до начала вычисления соответствующего выражения.

На использование этих операций в выражениях имеется одно ограничение: нельзя в одном и том же выражении, к одной и той же переменной, применять какие-либо из этих операций более одного раза.

Пример.

Следующая программа нарушает вышеприведенное правило и потому ее результат достаточно неожиданный.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int n = 1;
    cout<<n++<<' '<<n++<<' '<<n++<<endl;
    return EXIT_SUCCESS;
}
```

Что нужно знать в результате (проверяется на зачете):

- Какие арифметические операции имеются в C++?
- Особенности операции умножения.
- Особенности операции деления.
- Что происходит, если в одной арифметической операции участвуют операнды разных типов?
- Синтаксис и смысл операции преобразования типа.
- Использование этой операции для вещественного деления целых переменных.
- Синтаксис и смысл операций инкремента и декремента, префиксного и постфиксного.
- Что такое побочный эффект выражения?
- Правило употребления этих операций в выражениях и последствия нарушений этого правила.

Что нужно уметь в результате (задача на контрольной работе):

- Составлять арифметические выражения для решения несложных вычислительных задач.
- Записывать составленные арифметические выражения по правилам C++.