

ОСНОВЫ ЯЗЫКА C++

Арифметические операции в C++

Алфавит языка C++

Алфавит — это фиксированный для данного языка набор основных символов, т.е. "букв алфавита", из которых должен состоять любой текст на этом языке — никакие другие символы в тексте не допускаются.

- Прописные и строчные латинские буквы (различаются в именах), знак подчеркивания
- Цифры (0... 9)
- Специальные знаки “ { }, | [] () + - * / % \ ; ‘ : ? < = > ! & ~ ^ . #
- Разделители (пробел, табуляция, перевод строки)

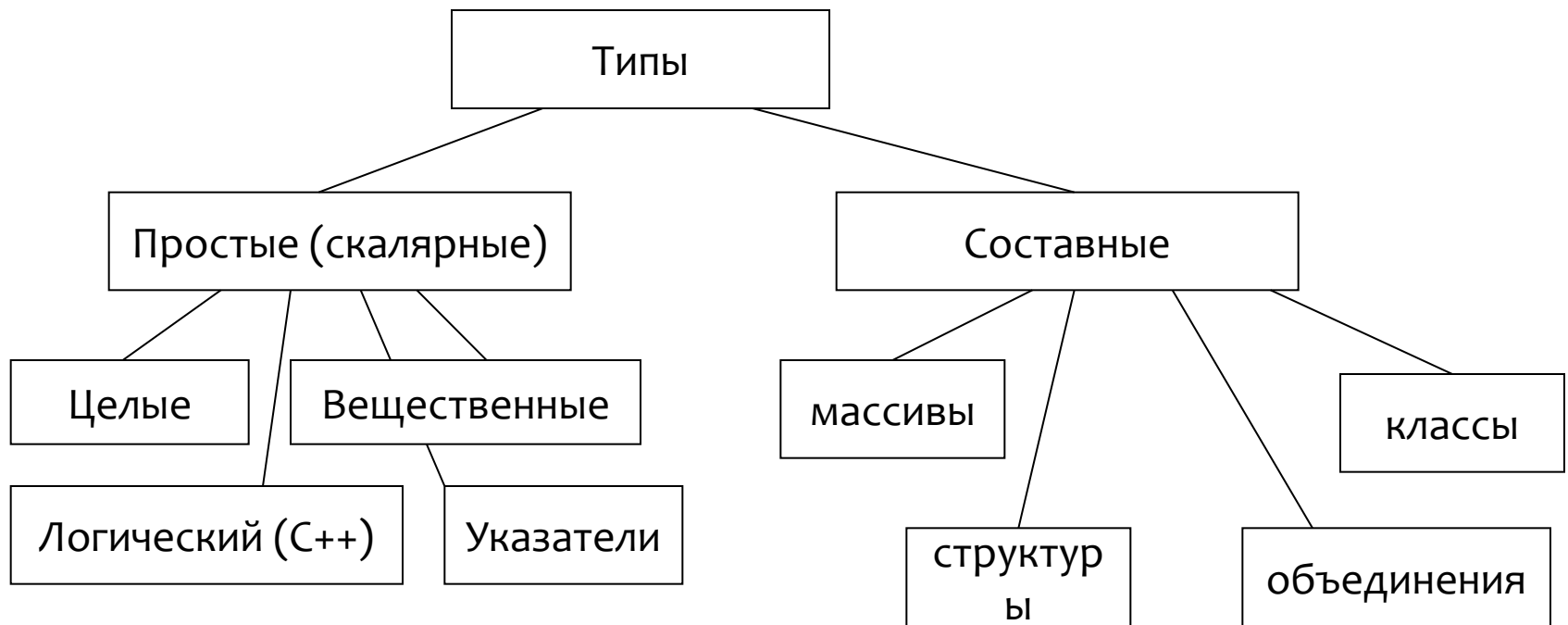
Лексемы C++

- Имена (не рекомендуется начинать с `_`)
- Ключевые слова
- Знаки операций (одно и двухсимвольные)
- Константы
- Разделители

Комментарии (ставятся в программе)

- однострочные `//` комментарий
- многострочные `/*` длинные `*/`

Типы данных C++



Базовые типы

	C/C++
Целые	char
	int (short int)
	unsigned char
	unsigned int (short)
	long int
	unsigned long int
Вещест.	float
	double
	long double

Специальные ТИПЫ

bool – логический
(true/false) – в C++

void – пустой.
Используется для
обозначения функций
без значений и
нетипизированных
указателей

Вспомним структуру программы на языке C++

```
#include <iostream> //подключение модуля
using namespace std; // подключение множества
имен std
int main() // начало программы
{
setlocale(LC_ALL, ""); //подключение русского языка
int a, b; //описание переменных
cin >> a >> b; //ввод данных
cout << "сумма" << a+b; //вывод ответа
return 0; //возврат
}
```

Арифметические операции в C++

Большинство арифметических операций входят в состав стандартного модуля `<iostream>`.

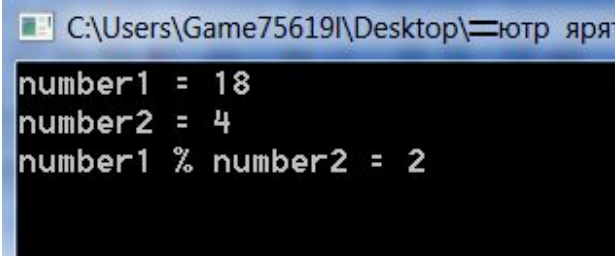
1. Деление по модулю - %

Применяется только к целочисленным переменным, нельзя делить по модулю на 0.

Возвращает остаток после целочисленного деления.

Например, $7 / 4 = 1$ с остатком 3, таким образом, $7 \% 4 = 3$. Еще пример: $25 / 7 = 3$ с остатком 4, таким образом, $25 \% 7 = 4$. Остаток составляет не дробь, а целое число. $36 \% 5 = 1$, в числе 36 только 35 делится на 5 без остатка, тогда $36 - 35 = 1$, 1 – остаток и результат.

```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL, "rus");
    int number1 = 18;
    int number2 = 4;
    cout << "number1 = " << number1 << endl;
    cout << "number2 = " << number2 << endl;
    cout << "number1 % number2 = " << number1 % number2 << endl;
    cout << endl;
    return 0;
}
```



```
C:\Users\Game756191\Desktop\...
number1 = 18
number2 = 4
number1 % number2 = 2
```

Если один из операндов оператора деления с остатком является отрицательным, то результаты могут быть как положительными, так и отрицательными! Например, ответом $-5 \% 2$ может быть как 1, так и -1. В спецификации C++ 11 решили сделать так, чтобы результат $a \% b$ был того же знака, что и значение a .

2. Арифметические операторы присваивания (такие же как в Python)

Оператор	Символ	Пример	Операция
Присваивание	=	$x = y$	Присваиваем значение y переменной x
Сложение с присваиванием	+=	$x += y$	Добавляем y к x
Вычитание с присваиванием	-=	$x -= y$	Отнимаем y от x
Умножение с присваиванием	*=	$x *= y$	Умножаем x на y
Деление с присваиванием	/=	$x /= y$	Делим x на y
Деление с остатком с присваиванием	%=	$x %= y$	Оставляем остаток от деления x / y в переменной x

До этого момента, когда вам нужно было добавить число 5 к переменной, вы делали так: $x = x + 5$

Это работает, но это *не совсем красиво* и требуется два оператора для выполнения.

Так как стейтменты, типа $x = x + 5$ являются очень распространенными, то C++ предоставил 5 арифметических операторов присваивания, для удобства. Вместо $x = x + 5$, вы можете написать $x += 5$. Вместо $x = x * y$, вы можете написать $x *= y$.

3. Деление целых чисел и чисел типа с плавающей точкой

Оператор деления имеет **два режима**. Если оба операнда – целые числа, то оператор выполняет целочисленное деление. Т.е. любая дробь (больше, меньше) отбрасывается и возвращается целое значение, округления нет. Например, $7 / 4 = 1$. Если один или оба операндов типа с плавающей точкой, то тогда будет выполняться деление типа с плавающей точкой. Здесь уже дробь присутствует. Например: $7.0 / 3 = 2.333$, $7 / 3.0 = 2.333$ и $7.0 / 3.0 = 2.333$.

Попытки деления на 0 (или на 0.0) станут причиной сбоя в вашей программе, это правило не следует забывать!

4. Операция "запятая"

Дополнительная операция (,) не работает непосредственно с данными, а приводит к вычислению выражения слева направо. Эта операция позволяет Вам использовать в одной строке несколько выражений, разделенных запятой.

Пример

```
i = 10;  
j=(i=12,i+8);
```

Результат выполнения: $j=20$. Сначала i получает значение 10, затем использование операции "запятая" приведет к тому, что i получит значение 12, а затем значение $i+8$, т.е. $12+8=20$. Результат будет присвоен j .

5. Префиксный и постфиксные операции

В языке C++ предусмотрены две уникальные операции, которые увеличивают или уменьшают значение переменной на 1.

Оператор	Пример	Описание	Эквивалентное выражение
++	i ++;	Постфиксная	i =i+1; или i+=1;
++	++ i;	Префиксная	i =i+1; или i+=1;
--	i --;	Постфиксная	i =i-1; или i-=1;
--	-- i;	Префиксная	i =i-1; или i-=1;

Префиксный и постфиксные операции различаются приоритетом. Префиксные операции имеют самый большой приоритет и выполняются до любой другой операции. Постфиксные операции имеют самый маленький приоритет и выполняются после всех остальных операций.

Пример.

```
float a, b=2, c=1, d=1;
```

```
a = b + c++;
```

```
cout << " a=" << a <<"c= " << c;
```

```
/* Даст результат a=3 c=2.
```

Используется постфиксный инкремент. Сначала произойдет сложение b и c, результат запишется в a, затем c будет увеличена на 1 */

```
a = ++d + b;
```

```
cout << " a=" << a <<"d= " << d;
```

```
/* Даст результат a=4 d=2.
```

Используется префиксный инкремент. Сначала d будет увеличена на 1 (и станет равной 2), затем произойдет сложение d и b, результат запишется в a */

Остальные операции находятся в модуле `#include <cmath>`

6. Возведения в степень — функция `pow()`.

`pow(a, x)` – это эквивалент a^x

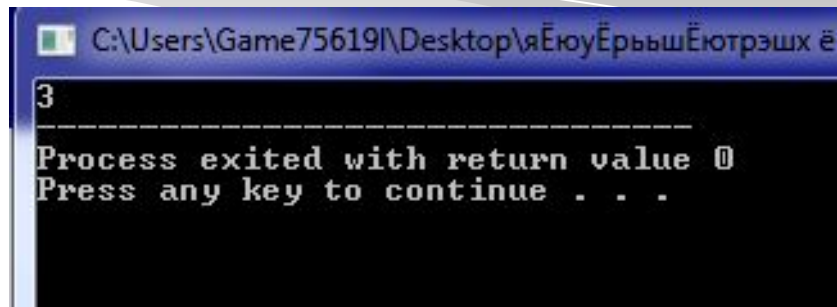
Стоит отметить, что параметры `pow()` — типа `double`, поэтому можно использовать не только целые числа, но и дробные.

7. Модуль числа – `fabs(число)`.

8. Квадратный корень - `sqrt(x)`.

9. Преобразование вещественного числа в целое:

```
float x=3.5;
int y;
y=(int)x;
cout<<y;
```



```
C:\Users\Game75619\Desktop\яЁюЁрььШЁютрэшх ё+
3
-----
Process exited with return value 0
Press any key to continue . . .
```

10. Использование `static_cast` <> в делении

Мы можем использовать `static_cast` <> для конвертации целого числа в число типа `floating point`. Таким образом, вместо целочисленного деления, будет деление типа с плавающей точкой. Например:

Результат:

int / int = 1

double / int = 1.75

int / double = 1.75

double / double = 1.75

```
1 #include <iostream>
2
3 int main()
4 {
5     int x = 7;
6     int y = 4;
7
8     std::cout << "int / int = " << x / y << "\n";
9     std::cout << "double / int = " << static_cast<double>(x) / y << "\n";
10    std::cout << "int / double = " << x / static_cast<double>(y) << "\n";
11    std::cout << "double / double = " << static_cast<double>(x) / static_cast<double>(y) << "\n";
12
13    return 0;
14 }
```

Задание

Задание 1: Составить программу, демонстрирующую работу арифметических операторов присваивания.

Результат работы:

```
C:\WINDOWS\system32\cmd.exe
number1 = 10
number2 = 4
Результат от += : number1 = 14
Результат от -= : number1 = 10
Результат от *= : number1 = 40
Результат от /= : number1 = 10
Результат от %= : number1 = 2
```

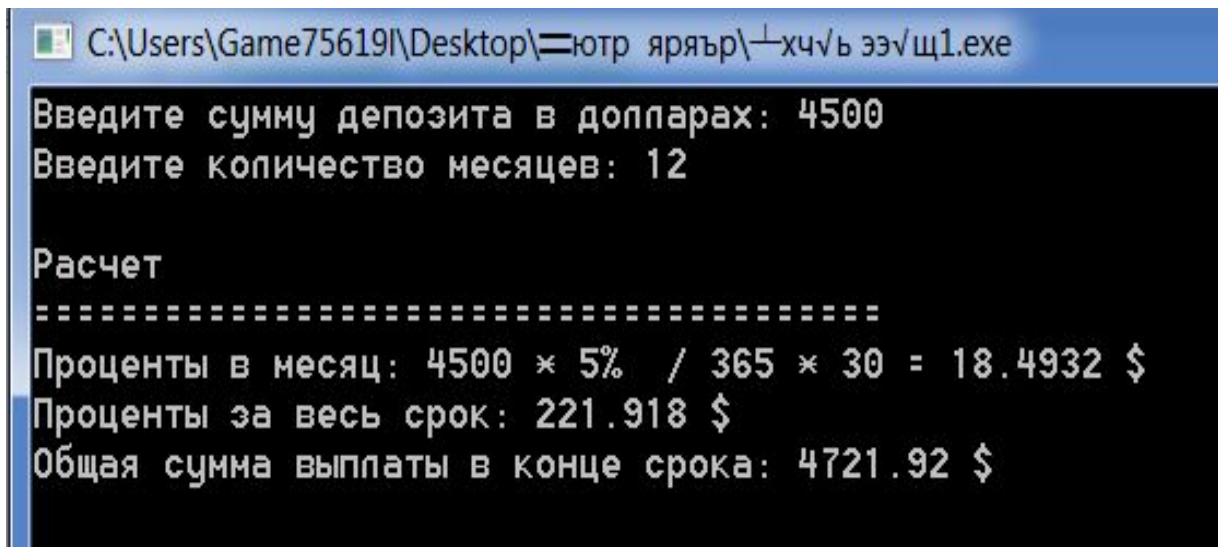
Почему в результате /= получилось 10?

Задание 2: Дано четырехзначное число ($x=5678$), вывести на экран в обратном порядке цифры из которых это число состоит. То есть мы должны увидеть на экране 8765. Подсказка: чтобы взять из числа отдельные цифры, надо применять деление по модулю на 10.

Результат:

```
C:\WINDOWS\system32\cmd.exe
Дано целое число: 5678
Число наизнанку: 8765
```

Задание 3. На сайте практически любого коммерческого банка, можно встретить так называемый Депозитный калькулятор, который позволяет людям, не желающим углубляться в формулы расчета процентных ставок, узнать какую прибыль они получат. Для этого им достаточно заполнить определённые поля, нажать на кнопку и увидеть результат. Это простая программа, которую уже сможет написать каждый из вас. Итак, задача: Пользователь вводит сумму депозита и количество месяцев хранения денег в банке. Необходимо провести расчет и показать на экран прибыль с депозита в месяц, за весь срок депозита, и общую сумму к выплате в конце срока. Валюта пусть будет — доллар США. Процентная ставка — 5% годовых. Формула расчета процентов в месяц — $\text{СуммаДепозита} * (\text{ПроцентнаяСтавка} / 100) / \text{ДнейВГоду} * \text{ДнейВМесяце}$.



```
C:\Users\Game756191\Desktop\...щ1.exe
Введите сумму депозита в долларах: 4500
Введите количество месяцев: 12

Расчет
=====
Проценты в месяц: 4500 * 5% / 365 * 30 = 18.4932 $
Проценты за весь срок: 221.918 $
Общая сумма выплаты в конце срока: 4721.92 $
```