



Ст. преп.
каф. ПОВТ
Масленников
Алексей
Александров
ич



Лекция № 2

Введение в программирование





Оператор	Значение
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления



```
#include <stdio.h>
int main()
{
    int a = 9,b = 4, c;

    c = a+b;
    printf("a+b = %d \n",c);

    c = a-b;
    printf("a-b = %d \n",c);

    c = a*b;
    printf("a*b = %d \n",c);

    c=a/b;
    printf("a/b = %d \n",c);

    c=a%b;
    printf("Остаток от деления на b = %d \n",c);
    return 0;
}
```



$$a+b = 13$$

$$a-b = 5$$

$$a*b = 36$$

$$a/b = 2$$

Остаток от деления на $b=1$



При этом существует особенность:

Пусть $a = 5.0$, $b = 2.0$, $c = 5$ и $d = 2$.

```
a/b = 2.5
```

```
a/d = 2.5
```

```
c/b = 2.5
```

```
c/d = 2    // Но!
```



Операторы инкремента и декремента



```
#include <stdio.h>
int main()
{
    int a = 10, b = 100;
    float c = 10.5, d = 100.5;

    printf("++a = %d \n", ++a);

    printf("--b = %d \n", --b);

    printf("++c = %f \n", ++c);

    printf("--d = %f \n", --d);

    return 0;
}
```



```
++a = 11
```

```
--b = 99
```

```
++c = 11.500000
```

```
--d = 99.500000
```



Оператор	Пример	Тоже самое, как
=	<code>a = b</code>	<code>a = b</code>
<code>+=</code>	<code>a += b</code>	<code>a = a+b</code>
<code>-=</code>	<code>a -= b</code>	<code>a = a-b</code>
<code>*=</code>	<code>a *= b</code>	<code>a = a*b</code>
<code>/=</code>	<code>a /= b</code>	<code>a = a/b</code>
<code>%=</code>	<code>a %= b</code>	<code>a = a%b</code>



```
#include <stdio.h>
int main()
{
    int a = 5, c;
    c = a;
    printf("c = %d \n", c);
    c += a; // c = c+a
    printf("c = %d \n", c);
    c -= a; // c = c-a
    printf("c = %d \n", c);
    c *= a; // c = c*a
    printf("c = %d \n", c);
    c /= a; // c = c/a
    printf("c = %d \n", c);
    c %= a; // c = c%a
    printf("c = %d \n", c);

    return 0;
}
```



```
c = 5
```

```
c = 10
```

```
c = 5
```

```
c = 25
```

```
c = 5
```

```
c = 0
```



Оператор	Значение	Пример
<code>==</code>	Равно	<code>5 == 3</code> returns 0
<code>></code>	Больше	<code>5 > 3</code> returns 1
<code><</code>	Меньше	<code>5 < 3</code> returns 0
<code>!=</code>	Не равно	<code>5 != 3</code> returns 1
<code>>=</code>	Больше, либо равно	<code>5 >= 3</code> returns 1
<code><=</code>	Меньше, либо равно	<code>5 <= 3</code> return 0



```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10;
    printf("%d == %d = %d \n", a, b, a == b); // true
    printf("%d == %d = %d \n", a, c, a == c); // false
    printf("%d > %d = %d \n", a, b, a > b); //false
    printf("%d > %d = %d \n", a, c, a > c); //false
    printf("%d < %d = %d \n", a, b, a < b); //false
    printf("%d < %d = %d \n", a, c, a < c); //true
    printf("%d != %d = %d \n", a, b, a != b); //false
    printf("%d != %d = %d \n", a, c, a != c); //true
    printf("%d >= %d = %d \n", a, b, a >= b); //true
    printf("%d >= %d = %d \n", a, c, a >= c); //false
    printf("%d <= %d = %d \n", a, b, a <= b); //true
    printf("%d <= %d = %d \n", a, c, a <= c); //true

    return 0;
}
```



5 == 5 = 1

5 == 10 = 0

5 > 5 = 0

5 > 10 = 0

5 < 5 = 0

5 < 10 = 1

5 != 5 = 0

5 != 10 = 1

5 >= 5 = 1

5 >= 10 = 0

5 <= 5 = 1

5 <= 10 = 1



Оператор	Назначение	Пример
&&	Логическое «И». Верно, когда все операнды верны	If c = 5 and d = 2 then, expression ((c == 5) && (d > 5)) equals to 0.
	Логическое «ИЛИ». Верно, когда хотя бы один из операндов верен	If c = 5 and d = 2 then, expression ((c == 5) (d > 5)) equals to 1.
!	Логическое отрицание, верно когда операнд имеет значение «ложь»	If c = 5 then, expression ! (c == 5) equals to 0.



p	q	$p \& q$	$p q$	$p \wedge q$
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1



```
#include <stdio.h>
int main()
{
    int a = 5, b = 5, c = 10, result;
    result = (a = b) && (c > b);
    printf("(a = b) && (c > b) = %d \n", result);
    result = (a = b) && (c < b);
    printf("(a = b) && (c < b) = %d \n", result);
    result = (a = b) || (c < b);
    printf("(a = b) || (c < b) = %d \n", result);
    result = (a != b) || (c < b);
    printf("(a != b) || (c < b) = %d \n", result);
    result = !(a != b);
    printf("!(a == b) = %d \n", result);

    result = !(a == b);
    printf("!(a == b) = %d \n", result);

    return 0;
}
```




```
(a = b) && (c > b) = 1
```

```
(a = b) && (c < b) = 0
```

```
(a = b) || (c < b) = 1
```

```
(a != b) || (c < b) = 0
```

```
!(a != b) = 1
```

```
!(a == b) = 0
```



Оператор	Назначение
&	Побитовое «И»
	Побитовое «ИЛИ»
^	Побитовое «XOR» (исключающее ИЛИ)
~	Дополнение
<<	Побитовый сдвиг влево
>>	Побитовый сдвиг вправо



12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Побитовое «ИЛИ» 12 и 25

00001100

| 00011001

00011101 = 29



```
#include <stdio.h>
int main()
{
    int a = 12, b = 25;
    printf("Вывод= %d", a | b);
    return 0;
}
```

Вывод= 29



12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

XOR 12 и 25

00001100

| 00011001

00010101 = 21



Пример «Побитовое исключающее ИЛИ»



```
#include <stdio.h>
int main()
{
    int a = 12, b = 25;
    printf("Вывод= %d", a^b);
    return 0;
}
```

Вывод= 21



35=00100011 (In Binary)

Для числа: 35

~ 00100011

11011100 = 220



```
#include <stdio.h>
int main()
{
    printf("=%d\n",~35);
    printf("=%d\n",~-12);
    return 0;
}
```

```
=-36
t=11
```




Для любого целого в С дополнение будет:

$$-(n+1)$$

Decimal	Binary	2's complement
0	00000000	$-(11111111+1) = -00000000 = -0(\text{decimal})$
1	00000001	$-(11111110+1) = -11111111 = -256(\text{decimal})$
12	00001100	$-(11110011+1) = -11110100 = -244(\text{decimal})$
220	11011100	$-(00100011+1) = -00100100 = -36(\text{decimal})$

Переполнение игнорируется

Дополнительный код отрицательного числа можно получить инвертированием модуля двоичного числа (первое дополнение) и прибавлением к инверсии единицы (второе дополнение), либо вычитанием числа из нуля.

Дополнительный код (дополнение до 2) двоичного числа получается добавлением 1 к младшему значащему разряду его дополнения до 1



212 = 11010100 (In binary)

212>>2 = 00110101 (In binary) [Right shift by two bits]

212>>7 = 00000001 (In binary)

212>>8 = 00000000

212>>0 = 11010100 (No Shift)

212 = 11010100 (In binary)

212<<1 = 110101000 (In binary) [Left shift by one bit]

212<<0 = 11010100 (Shift by 0)

212<<4 = 110101000000 (In binary) = 3392 (In decimal)



```
#include <stdio.h>
int main()
{
    int num=212, i;
    for (i=0; i<=2; ++i)
        printf("Правый сдвиг %d: %d\n", i, num>>i);

    printf("\n");

    for (i=0; i<=2; ++i)
        printf("Левый сдвиг %d: %d\n", i, num<<i);

    return 0;
}
```

```
=-36
t=11
```



Правый сдвиг 0: 212

Правый сдвиг 1: 106

Правый сдвиг 2: 53

Левый сдвиг 0: 212

Левый сдвиг 1: 424

Левый сдвиг: 848



Оператор позволяет получить размер того или иного объекта

```
#include <stdio.h>
int main()
{
    int a, e[10];
    float b;
    double c;
    char d;
    printf("Size of int=%lu bytes\n",sizeof(a));
    printf("Size of float=%lu bytes\n",sizeof(b));
    printf("Size of double=%lu bytes\n",sizeof(c));
    printf("Size of char=%lu byte\n",sizeof(d));
    printf("Size of integer type array having 10 elements = %lu bytes\n",
sizeof(e));
    return 0;
}
```



Оператор позволяет получить размер того или иного объекта

Size of int = 4 bytes

Size of float = 4 bytes

Size of double = 8 bytes

Size of char = 1 byte

Size of integer type array having 10 elements = 40 bytes



Запись if в форме:

```
conditionalExpression ? expression1 : expression2
```

```
conditionalExpression ? expression1 : expression2
```



Запись if в форме:

```
#include <stdio.h>
int main(){
    char February;
    int days;
    printf(«Если год високосный , то -1. Иначе –любое число: ");
    scanf("%c",&February);

    days = (February == '1') ? 29 : 28;

    printf(«Число дней в Феврале = %d",days);
    return 0;
}
```




Запись if в форме:

Если год високосный , то -1. Иначе –любое число: 1
Число дней в Феврале = 29



Спасибо за внимание !

