Операции, операторы, операнды

Оператор

- **Операция** — действие над данными, приводящие к определенному результату.

Оператор

- **Оператор** — конструкция языка позволяющая производить различные действия над данными, приводящие к определенному результату.

Оператор

- **Операнд** — данные, над которыми совершаются действия приводящие к определённому результату.

Операторы

- **Унарные** — операторы, которым необходим, только один операнд (данные, над которыми производится действие).
- **Бинарные** — операторы, которым необходимо два операнда слева и справа от оператора.
- **Тернарные** — операторы, которым необходимо три операнда.

Приоритет операторов

- Все операторы имеют приоритет.

Знаки и приоритет операций (начало)

Приоритет операций	Знаки операций	Порядок выполнения операций с равным приоритетом
1	( ) [ ] -> :: .	слева направо
2	! ~ + − ++ −− & * (<*имя типа*>) sizeof new delete <*имя типа*>()	справа налево
3	.* ->*	слева направо
4	* / %	слева направо
5	+ −	слева направо
6	<< >>	слева направо
7	< <= >= >	слева направо
8	== !=	слева направо
9	&	слева направо

Арифметические операции

Название операции	Символ, для обозначения в языке С	Краткое описание. Пример.
Сложение	+	Складывает два значения вместе, результатом является сумма операндов: 5+18 результат 23
Вычитание	–	Вычитает значение, находящееся справа из значения, находящегося слева от оператора. Результат — разность операндов: 20–15 результат 5
Умножение	*	Перемножает два значения, результатом является произведение операндов: 5*10 результат 50
Деление	/	Делит значение, находящееся слева на значение, находящееся справа от оператора. Например: 20/4 результат 5
Деление по модулю	%	Результатом этой операции является остаток от целочисленного деления, например, если мы делим 11 на 3, то целых частей у нас получается 3, (так как 3*3=9), в остатке будет 2, это число и будет результатом деления по модулю: 11/3 3 целых 2 в остатке 11%3 = 2 (остаток)

Деление по модулю

1. Операцию деления по модулю, можно применять только к целочисленным данным. Попытки нарушить данное правило приведут к ошибке на этапе компиляции.

2. Если меньшее число делится на большее с помощью %, то результатом будет само меньшее число. 3%10 = 3

3. Делить по модулю на нуль нельзя, это приведет к некорректной работе программы на этапе выполнения

Инкремент и декремент

- **Инкремент** — обозначается конструкцией ++.
- Данный оператор увеличивает содержимое любой переменной на единицу и перезаписывает значение переменной.
- Например:

```
int a=8;                // на экране число
cout<<a<<endl;          8
a++;                    // на экране число
cout<<a<<endl;          9
```

Инкремент и декремент

- **Декремент** — обозначается конструкцией – –.
- Данный оператор уменьшает содержимое любой переменной на единицу и перезаписывает значение переменной.
- Например:

```
int a=8;      // на экране число
cout<<a;      8
a--;          // на экране число
cout<<a;      7
```

Инкремент и декремент

- Имеет значение то, с какой стороны стоят знаки инкремент и декремент (постфиксная и префиксная формы).
- int a=8;
- cout<<++a<<endl; *// на экране число 9*
- cout<<a <<endl; *// на экране число 9*
- cout<<a++<<endl; *// на экране число 9*
- cout<<a<<endl; *// на экране число 10*

Последовательность выполнения операторов

- Принцип выполнения команд в языке С неоднозначен.

1. Если кроме постфиксной формы инкремента или декремента, в строке есть еще какая-либо команда, то сначала выполняется эта команда, и только потом инкремент или декремент независимо от расположения команд в строке.

2. Если кроме префиксной формы инкремента или декремента, в строке есть еще какая-либо команда, то все команды в строке выполняются справа налево согласно приоритету операторов.

Инкремент и декремент

- Операции **инкрементирования** и **декрементирования** – это **унарные** операции, так как для их совершения требуется только один операнд.

Последовательность выполнения операторов

ОПЕРАТОР	НАПРАВЛЕНИЕ
() [] . ->	СЛЕВА НАПРАВО
* / % + −	
<< >> & ^ |	
<< = >> = == != =	
&& ||	
Унарный − унарный + ! ++ −−	СПРАВА НАЛЕВО
?:	
+ = − = / = * = % = & = ^ = | =	

Пример

```
#include <iostream>
using namespace std;
void main()
{
```

```
    const float PI = 3.141592;
    const short COEFFICIENT_FOR_CALCULATION = 2;
    float radius, circumference, area;
cout << «Программа расчёта круга\n\n";
cout << «Введите радиус окружности\n\n";
cin >> radius;
cout << "\n\n";
area = PI*radius*radius; // подсчет площади круга
circumference = PI*(radius * COEFFICIENT_FOR_CALCULATION);
cout << “Площадь круга: " << area << "\n\n";
cout << «Длина окружности: " << circumference << "\n\n";
cout << «Спасибо! Пока!\n\n";
```