



# ЕН.Ф.02 – Информатика и программирование

## Лекция 3. Операторы C++

Конова Елена Александровна  
E\_Konova@mail.ru

# Определение и классификация

**Оператор** – предложение, описывающее одно действие по обработке данных (один шаг алгоритма).

**Назначение** операторов:

- 1) преобразование данных;
- 2) управление ходом выполнения программы (программное управление).

**Классификация** операторов

1. Операторы преобразования данных.
2. Операторы управления.
3. Вызовы функций (операторы-функции и операторы-выражения).

# Операторы преобразования данных

1. Выражение-оператор.
2. Пустой оператор.
3. Составной оператор (блок).

# Выражение–оператор

Оператора присваивания в С нет.

Выражение-оператор, это выражение, в составе которого есть операция присваивания (и ее клоны) или операции изменения данных (++ — и другие).

Примеры:

```
x = y + 4;
```

```
y += 4;    // Выражение
```

```
x ++;    // Выражение-оператор.
```

Операция присваивания правоассоциативна, поэтому возможна цепочка присваиваний:

```
x = y = z = 1;
```

```
3 ← 2 ← 1
```

# Выражение присваивания

Назначение: преобразование данных в соответствии с выражением правой части.

**Синтаксис:**

Имя\_переменной = выражение;

Имя\_переменной = (тип) выражение;

Семантика:

- 1) вычисляется выражение правой части (определен тип);
- 2) присваивается переменной левой части.

При этом происходит неявное или явное преобразование и приведение типов.

# Пустой оператор

Имеет вид

```
    ; ;  
for (int i=0; i<5;i++) ; // 5 раз выполняется  
    S += i; // пустой оператор
```

# Составной оператор (блок)

**Составной** оператор

```
{  
  // Только исполнимые операторы  
}
```

**Блок**

```
{  
  // Объявления объектов  
  // и выполнимые операторы  
}
```

Назначение – объединение группы операторов в один.

Объявленные объекты существуют только внутри блока.

Пример блока – тело любой функции, в т.ч. **main**.

# Операторы управления

Операторы программы выполняются в том порядке, в котором записаны.

Каждый оператор выполняет один шаг алгоритма, тем самым управляя работой компьютера (принцип программного управления).

Поэтому порядок выполнения выражений называют **потокком управления**.



# Операторы управления

Операторы управления позволяют изменить направление потока управления.

Условный оператор **if**

Оператор цикла **while**

Оператор цикла **do {...} while**

Оператор цикла типа прогрессия **for**

Оператор прерывания (цикла) **break**

Оператор продолжения (цикла) **continue**

Оператор переключатель **switch**

Оператор выхода из функции **return**

# Операторы вызова функции

Назначение – передача управления (и данных) функции, получение результата.

Пример.

```
printf("x=%d", x); // Вызов функции  
scanf("%d%f", &x, &y); //  
pow(x, a); //.
```

Синтаксис:

Имя\_функции (фактические\_параметры)

// a + b или sin(x)

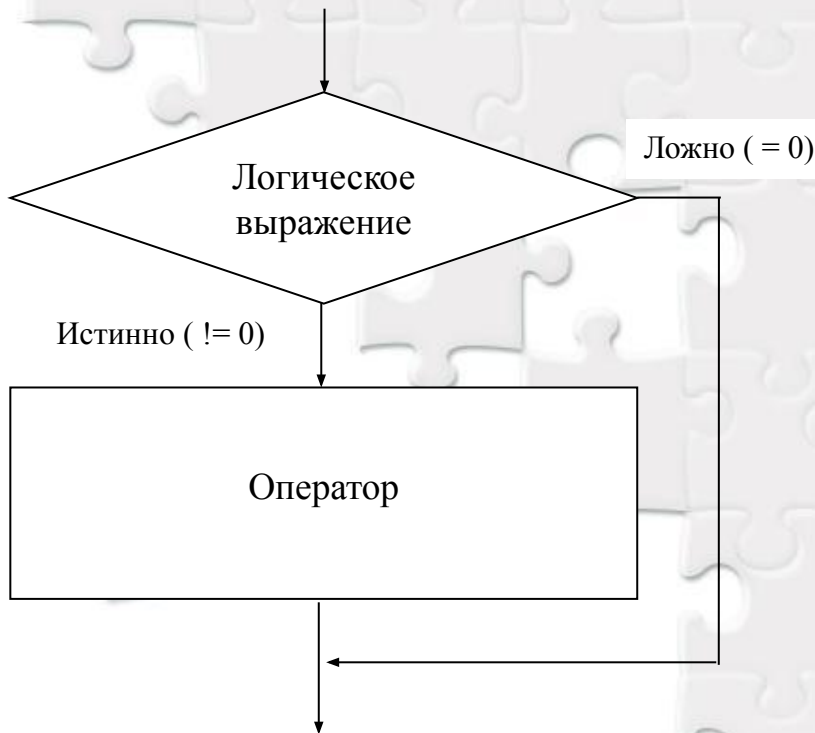
Операнды – имя функции и параметры.

# Схема изложения сведений об операторах

1. Назначение.
2. Синтаксис.
3. Механизм исполнения (семантика).
4. Примеры.
5. Особенности.

# Условный оператор – первая форма

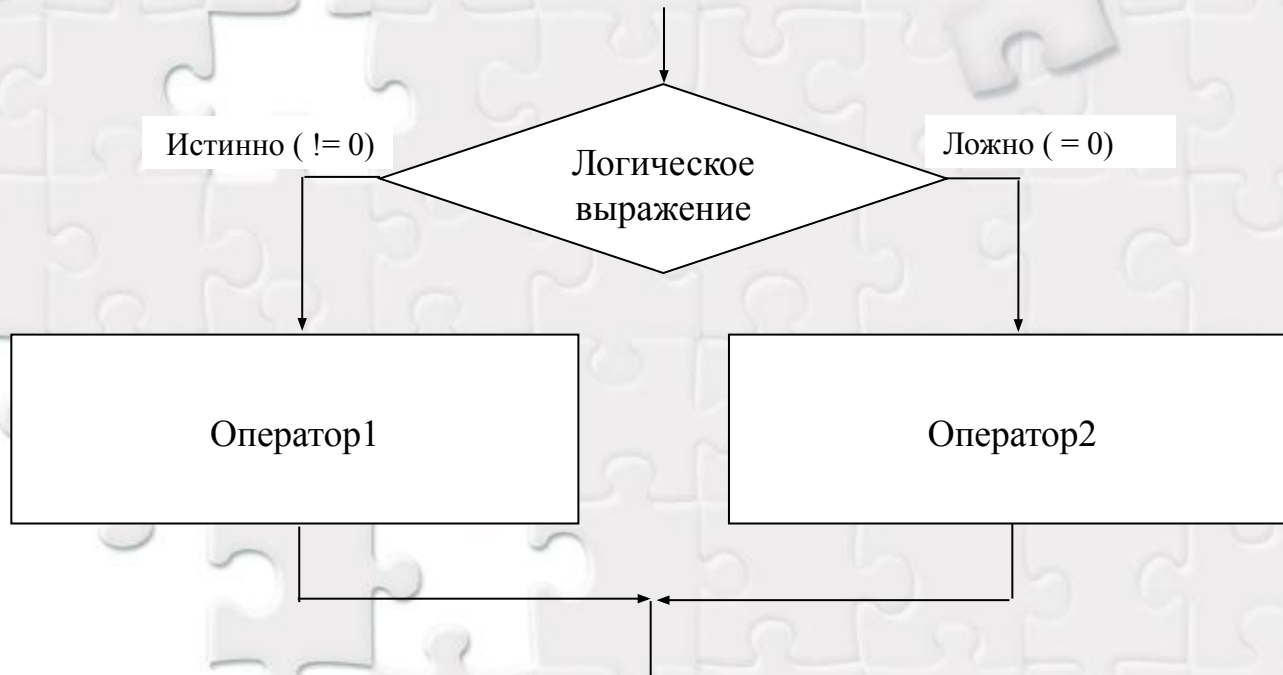
**Назначение** – выбор одного из двух возможных путей исполнения алгоритма (программы) в зависимости от условий, сложившихся при ее выполнении.



**Синтаксис:**

```
if (Логическое_выражение)  
    Оператор;  
//
```

# Условный оператор – вторая форма



```
if (Логическое_выражение)  
    Оператор1;  
else //Альтернативная часть  
    Оператор2;
```

# Семантика

Логическое выражение – любое типа `int`.

Оператор – совокупность действий, один или несколько операторов. Если несколько операторов входят в ветвь `if`, то это блок:

```
{  
    Оператор;  
}
```

## Семантика

1. Вычисляется логическое выражение.
2. Если оно  $\neq 0$  (истина), то выполняется Оператор1.
3. Если оно  $== 0$  (ложь), то выполняется Оператор2 (или ничего в первой форме).

# Замечания по синтаксису

1. Логическое выражение записывается в скобках, это сколь угодно сложное выражение, вычисляющее целочисленное значение. Для записи используются все знаки логических операций.

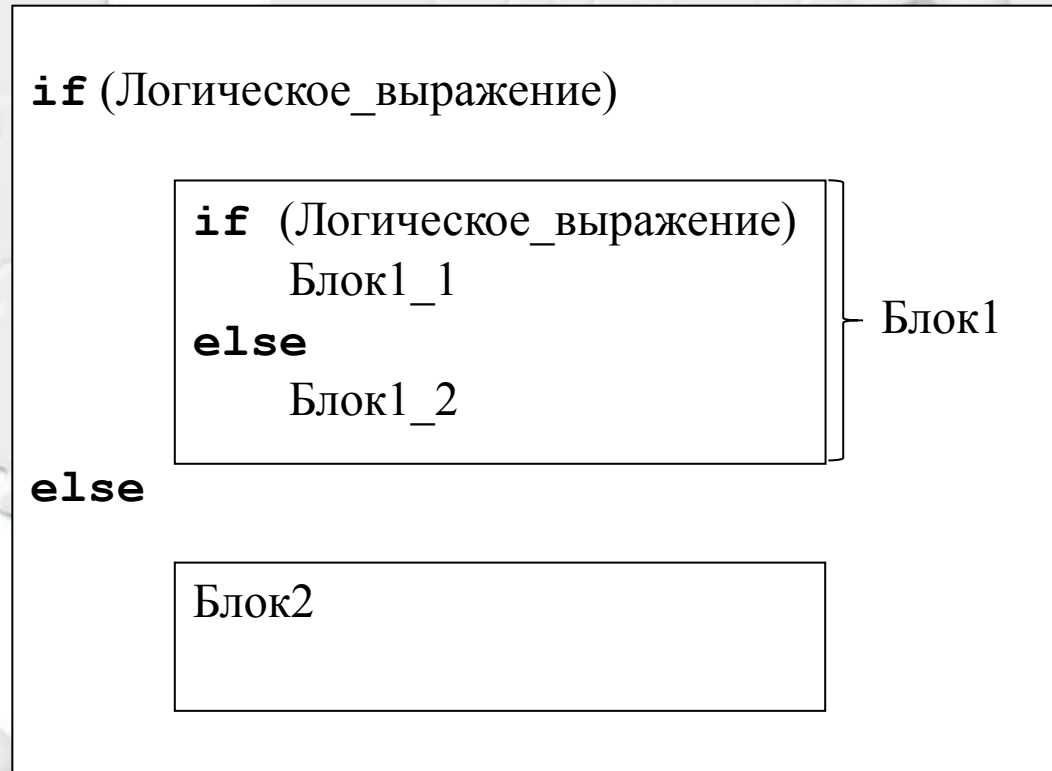
2. В общем случае, «Оператор», это блок:

```
if (выражение)  
    {  
    }  
else  
    {  
    }
```

В блоке могут быть объявлены переменные, они известны только в этом блоке.

3. Структура условного оператора подчеркивается отступами.

# Вложенный условный оператор



Правила организации: любой внутренний уровень полностью принадлежит одной из ветвей внешнего уровня.



# Примеры

Примеры

# Запись логических выражений

Простые логические выражения используют отношения:

```
x + 3 > 0
```

```
x < 0.5*y
```

```
Angle==90
```

Операция отношения выполняется корректно, когда сравниваются значения одинаковых типов. Если операнды разного типа, то перед выполнением сравнения компилятор выполнит приведение типов. Результат операции может быть отличным от ожидаемого.

Например,

```
My_float >= My_Int //
```

В этом случае необходимо применять явное преобразование типа.

# Запись логических выражений

Сложные логические выражения используют знаки логических операций:

**&&** – логическое И;

**||** – логическое ИЛИ;

**!** – логическое отрицание НЕ.

`Age >= 12 && Age <=17 // Подростковый – от 12 до 17 лет.`

Логическое выражение истинно, когда оба условия выполняются одновременно.

`Age < 12 || Age > 17 // Не подросток.`

Логическое выражение истинно, когда хотя бы одно или оба условия выполнены.

`! (Age >=12 && Age <=17) // То же самое.`

# Приоритеты логических операций

1. ! // Правоассоциативна
2. / % \*
3. + -
4. < <= > >=
5. == != // Меньший приоритет
6. &&
7. ||
8. =

# Ошибки записи логических выражений

1. Знаки логических операций  $\&\&$  и  $\|\|$  похожи на знаки операций поразрядного сравнения  $\&$  и  $|$ .

$a \|\| b$  (0 или 1)

$a | b$  (001 | 010 = 3)

# Ошибки записи логических выражений

2. Знак операции = отличен от знака ==

```
Key = getch ();
```

```
...
```

```
if ( Key = 27) // 27 = код Esc,
```

```
{
```

```
    printf ("Завершение работы.\n");
```

```
    return;
```

```
}
```

```
else
```

```
    printf ("Продолжение работы.\n");
```

Независимо от первоначального значения `Key`, будет выполнено присваивание `Key = 27`, и значение выражения равно 1.

# Ошибки записи логических выражений

3. Пропуск знаков логических операций.

Абстрактная математическая запись:

$$-3 < x < 6$$

Запись на C++:

$$-3 < x \ \&\& \ x < 6$$

В противном случае

$$-3 < x < 6$$

Схема вычисления выражения:

$$1) \ (-3 < x) = 0 \text{ или } 1$$

$$2) \ (0 \text{ или } 1) < 6 = \text{True}$$

# Ошибки записи логических выражений

4. Операции сравнения для вещественных типов не выполняются точно в силу особенностей представления данных с плавающей точкой. Поэтому проверку на точное равенство `==` выполняют приближенно.

```
float x=1./3.;
float y;
y = x+x+x;
if (y==3*x)
    printf("Равны\n");
else
    printf("Не равны\n");
```

Проверка на приближенное равенство выполняется так:

```
(fabs (y-3*x)<0.001)
```



# Выводы

Примеры в `Primer_If`

# Стиль записи программ

Из элементов стиля можно выделить на начальном этапе:

- 1) **система именования** программных объектов;
- 2) наличие **комментариев**;
- 3) **структурирование текста** программы.

Система именования программных объектов требует, содержательного именования переменных и других объектов, например,

```
float x1, y1; // Координаты точки.
```

```
FILE *My_file_1; // Файл данных.
```

Имена `define` определенных констант – большими.

```
#define SIZE 100
```

```
#define
```

# Структура текста программы

1. Каждый оператор пишется в одной строке.
2. Пустые строки разделяют однородные группы операторов или логические фрагменты.
3. Пробелы в тексте улучшают читабельность кода.
4. Отступы подчеркивают структуру программы, особенно вложения для циклов, условий.

Операторы высшего уровня записываются с левого края.

Отступ на 3 позиции для зависимых операторов.

Фигурные скобки располагаются на уровне блока, который они охватывают, одна под другой.

# Пример

```
if (условие1)
{
    оператор1
}
else if (условие2)
{
    оператор2
}
else
{
    оператор3
}
```

```
if (условие1) оператор1 else if (условие2) {оператор2}
else {оператор3}
```