PROGRAMMING

USING
C/C++

Модуль 3.
Инструкции

Что такое инструкция?

- **Инструкция** (англ. statement) – это минимальная единица программы, представляющая собой один шаг программы.
  - Одна инструкция может занимать несколько строк.
  - В одной строке может быть несколько инструкций.

- Любая инструкция может быть помечена **меткой** – специальным ***идентификатором***, за которым следует двоеточие. Например:

```
start:   cout  << “Starting program…” << endl;
```

Далее в программе можно воспользоваться конструкцией безусловного перехода **goto** для мгновенного перехода к метке:

```
goto start;
```

Использование **goto** в программе ***крайне(!!!)*** нежелательно. Это нарушает структуру программы и является жестким нарушением принципов структурного программирования.

Инструкции и выражения

- Любое **выражение**, которое заканчивается *точкой с запятой* (;) является ***инструкцией***:
  - Присваивания: `x = 10;` `x += 15;`
  - Инкремент и декремент: `x++;` `--x;`
  - Вызов функции: `pow(2, 3);`

Однако не все инструкции-выражения имеют смысл. Напрмер:

```
x + y;
```

данная инструкция ничего не изменяет и не выполняет действий, которые как-то отразились бы в программе.

Другими словами эта инструкция не имеет **побочного эффекта**.

Побочный эффект – это действие, которое неким образом *меняет состояние программы*.

Например, изменяет значение переменной, выводит что-то на экран и т.д.

Составная инструкция

- Несколько инструкций, которые заключены в фигурные скобки, образуют **составную инструкцию**:

Например:

```
{
        x++;
        cout << “x = ” << x << endl;
}
```

- Составная инструкция **синтаксически эквивалентна** одной инструкции.
- Составная инструкция может употребляться **везде**, *где допускается обычная инструкция*.
- Точка с запятой **не ставится** после закрывающей фигурной скобки.
- Составная инструкция является блоком и **может содержать объявления локальных переменных**.
- Эти переменные будут уничтожены после выполнения составной инструкции, т.е. после выхода из блока.

Пустая инструкция

- В языке С точка с запятой служит не только **признаком инструкции**, как например тут:

```
x++;
```

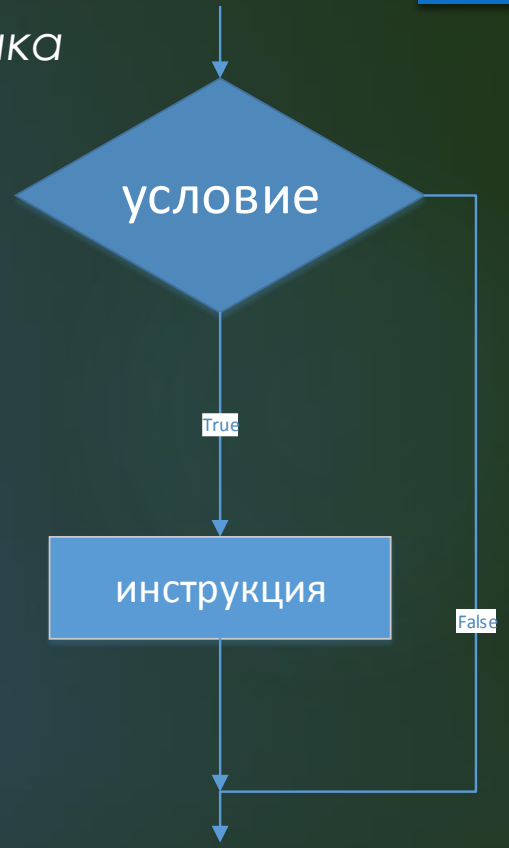
А является по сути **пустой инструкцией** и может употребляться отдельно в коде, как **самостоятельная инструкция**:

```
;
```

Условная инструкция

- **Условная инструкция if (англ. If statement)** – это *конструкция языка программирования*, которая позволяет выполнять **инструкцию** *в зависимости от истинности некоторого выражения*.
- Общий синтаксис:

```
if (условие)
    инструкция
```



- В данном случае **условие** – это ***выражение***, которое либо *дает в результате значение логического типа, либо любое другое значение, которое* ***возможно преобразовать к логическому типу***. (0 – false, всё остальное – true)
- Если условие будет истинно, то выполнится инструкция. Если ложно – инструкция будет пропущена.

Условная инструкция

- Когда необходимо обеспечить реакцию на ложность условия применяется конструкция if-else.
- Общий синтаксис:

```
if (условие)
      инструкция1
else
      инструкция2
```

- В данном случае, если условие будет истинным – будет выполнена только **инструкция1**, а затем выполнение кода продолжится с инструкций, которые написаны **после** if-else.
- А если условие будет ложным будет выполнена только **инструкция2**.
- Если необходимо выполнить несколько инструкций в теле if или else применяется составная инструкция:

```
if (условие)
{
      инструкция1
      инструкция3

}
```

Условная инструкция

- Условная инструкция является **единой инструкцией**. То есть, например:

```
if (n < 0)
    cout << “Отрицательное” << endl;
else
    cout << “Положительное или ноль” << endl;
```

воспринимается компилятором как **одна цельная инструкция**.

Следовательно возможна вложенность:

```
if (n <= 0)
    if (n < 0)
        cout << “Отрицательное” << endl;
    else
        cout << “Ноль” << endl;
else
    cout << “Положительное” << endl;
```

Условная инструкция

- Однако, иногда может возникнуть неопределенность с вложенностью. Например:

```
if (n>0)
     if (a>b)
          z=a;
else
     z=b;
```

Хоть код и отформатирован так, компилятор решает эту проблему таким правилом: **else относится к самому внутреннему if**.

```
if (n>0)
     if (a>b)
          z=a;
     else
          z=b;
```

If-else if-else

- Когда необходимо выбрать *один вариант* из **более чем двух**, применяется *конструкция if-else if*.
- Общий синтаксис:

```
if (условие1)
      инструкция1
else if (условие2)
      инструкция2
...
else if (условиеn)
      инструкцияn
else
      инструкция
```

  - Условия будут проверяться **по очереди до тех пор**, пока не будет найденно **истинное**, либо пока выполнение не дойдет до else.
  - Если истинное условие найдено – проверки **прекращаются**, выполняется указанная инструкция, а выполнение кода продолжается с кода, который написан **далее за if-else if-else.**