СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ. ВЕТВЛЕНИЕ

ЛЕКЦИЯ 3.1

ПРЕПОДАВАТЕЛЬ: МЕЛЬНИКОВА ЕКАТЕРИНА СЕРГЕЕВНА

ТРЕБОВАНИЯ К ИЗУЧЕНИЮ МАТЕРИАЛОВ ЛЕКЦИИ

Студентам необходимо внимательно изучить материалы лекции. Определения, классификации, списки, важную информацию, выделенную **КРАСНЫМ ЦВЕТОМ** необходимо законспектировать в тетрадь (проверка конспектов будет проводиться на каждом очном практическом/лабораторном занятии). На последнем слайде презентации будут располагаться контрольные вопросы. На них необходимо подготовить ответы УСТНО к концу текущей лекции.

ПРИНЦИПЫ СТРУКТУРНОГО ПРОГРАММИРОВАНИЯ

Любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:

1) Следование
2) Ветвление
3) Цикл

В программе базовые конструкции могут быть вложены друг в друга произвольным образом, но никаких других средств управления последовательностью выполнения операций не предусматривается.

Повторяющиеся фрагменты программы (либо не повторяющиеся, но представляющие собой логически целостные вычислительные блоки) могут оформляться в виде подпрограмм - процедур или функций. В этом случае в тексте основной программы, вместо помещённого в подпрограмму фрагмента, вставляется инструкция вызова подпрограммы. При выполнении такой инструкции выполняется вызванная подпрограмма, после чего исполнение программы продолжается с инструкции, следующей за командой вызова подпрограммы.

Разработка программы ведётся пошагово, методом «сверху вниз».

ТИП ДАННЫХ BOOL

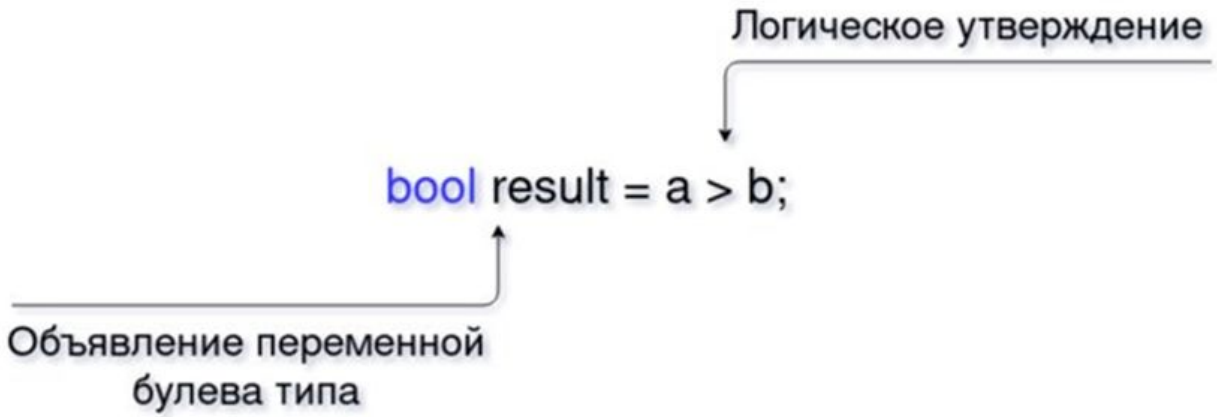
Тип данных bool, в котором хранится значение либо истина, либо ложь часто используется при ветвлении и следовании.

Чтобы поместить в переменную булева типа какое-то значение, можно самостоятельно написать его (true или false) или ввести какое-либо логическое утверждение:

- число a больше числа b;
- имя пользователя — Игорь;
- сегодня четверг и так далее.

Дальше компьютер проверяет правдивость такого утверждения и возвращает результат.

.



Для написания таких утверждений необходимы логические операторы:

- > — больше;
- < — меньше;
- == — равно;
- != — не равно;
- >= — больше или равно;
- <= — меньше или равно.

Вот несколько примеров:

```
bool result = 5 > 6; //false
result = 5 < 6; //true
result = 5 == 6; //false
result = 5 != 6; //true
```

При попытке вывести значение переменной result программа выведет 1 или 0.

УСЛОВНЫЕ ОПЕРАТОРЫ IF-ELSE

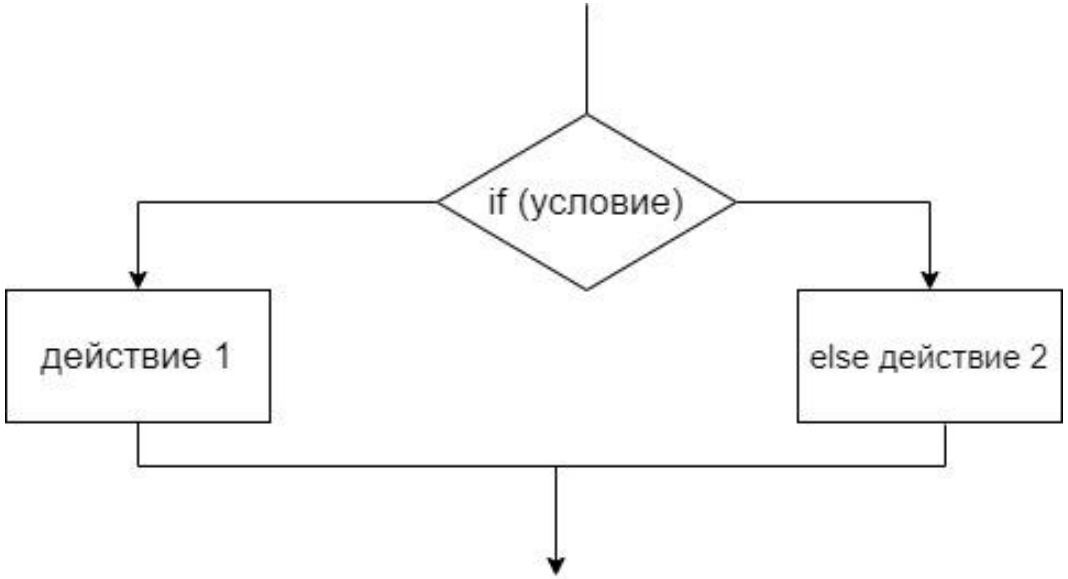
Оператор if служит для того, чтобы выполнить какую-либо операцию в том случае, когда условие является верным. Условная конструкция всегда записывается в круглых скобках после оператора if. Общая форма оператора следующая:

if (выражение) оператор;

else оператор;

Пример:

```
if (x<0) y=-x;
else y=x;
```



Зарисовать блок-схему

Оператор if является вложенным, если он вложен, т.е. находится внутри другого оператора if или else. Во вложенном условном операторе фраза else всегда ассоциирована с ближайшим if в том же блоке, если этот if не ассоциирован с другой фразой else.

Пример:

```
if(i){
    if(j) statement 1;
    if(k) statement 2; /* этот if */
    else statement 3; /* ассоциирован с этим else */
}
else statement 4; /* ассоциирован с if(i) */
```

В программах часто используется конструкция, которую называют *лестницей if-else-if*. Общая форма:

```
if (выражение) оператор;
else if (выражение) оператор;
. . .
else опepftop;
```

Условные выражения операторов if вычисляются сверху вниз. После выполнения некоторого условия, т.е. когда встретится выражение, принимающее значение ИСТИНА, выполняется ассоциированный с этим выражением оператор, а оставшаяся часть лестницы пропускается. Если все условия ложны, то выполняется оператор в последней фразе else. Эта структура операторов соответствует алгоритмической структуре «Выбор».

ТЕРНАРНЫЙ ОПЕРАТОР ? :

Его можно использовать вместо оператора if-else, записанного в форме:
if (условие) переменная = выражение; else переменная = выражение;
Оператор ? является тернарным, потому что он имеет три операнда.
Его общая форма следующая:
Выражение1 ? Выражение2 : Выражение3;

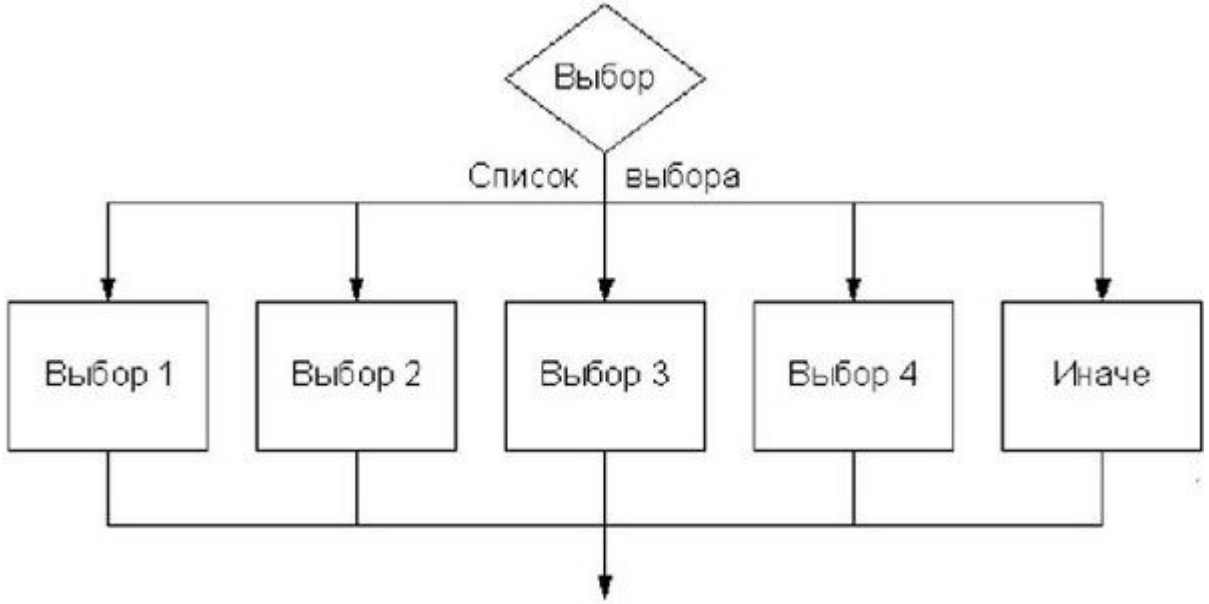
Результат операции ? определяется следующим образом. Сначала вычисляется Выражение1. Если оно имеет значение ИСТИНА, вычисляется Выражение2 и его значение становится результатом операции ?. Если Выражение1 имеет значение ЛОЖЬ, вычисляется Выражение3 и его значение становится результатом операции ?.

Пример:

```
x = 10;
y = x>9 ? 100 : 200;  //y=200
```

ОПЕРАТОР МНОЖЕСТВЕННОГО ВЫБОРА SWITCH

Оператор switch. Более удобная конструкция для реализации «Выбора». Этот оператор (переключатель) предназначен для выбора ветви вычислительного процесса исходя из значения управляющего выражения. (При этом значение управляющего выражения сравнивается со значениями в списке целых или символьных констант. Если будет найдено совпадение, то выполнится ассоциированный с совпавшей константой оператор.)



Зарисовать блок-схему

Общая форма оператора switch следующая:

```
switch (выражение)
{
case постоянная1: последовательность операторов; break;
case постоянная2: последовательность операторов; break;
case постоянная3: последовательность операторов; break;
default: последовательность операторов;
}
```

Оператор **break** — это один из операторов безусловного перехода. Он может применяться не только в операторе switch, но и в циклах. Когда в теле оператора switch встречается оператор break, программа выходит из оператора switch и выполняет оператор, следующий за фигурной скобкой } оператора switch.

Об операторе switch очень важно помнить следующее: оператор switch отличается от if тем, что в нем управляющее выражение проверяется только на равенство с постоянными. В одном и том же операторе switch никакие два оператора case не могут иметь равных постоянных.

Пример:

```
ch = getchar(); /* чтение клавиш */
switch(ch){
    case '1': printf(“Введена 1!”); break;
    case '2': printf(“Введена 2!”); break;
    case '3': printf(“Введена 3!”); break;
    default : printf("Не введена цифра от 1 до 3");
}
```