36. Базовые конструкции языков программирования: условные и безусловные переходы, операторы выбора, циклы, процедуры и функции.

http://ips.ifmo.ru/courses/cpp/topic2/index.html

http://ips.ifmo.ru/courses/cpp/topic5/l1/index.html

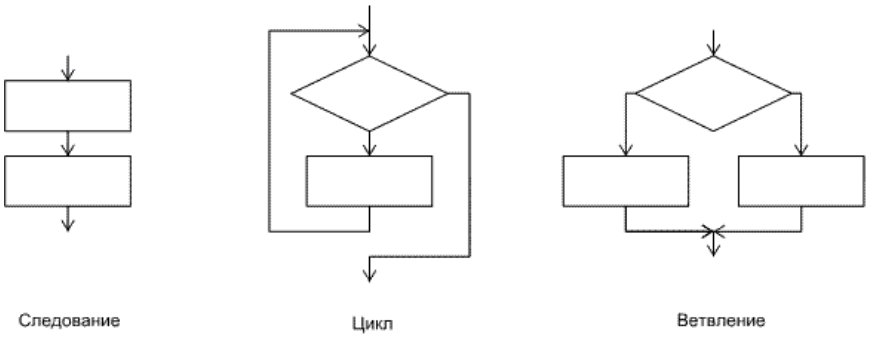
В теории программирования доказано, что программу для решения задачи любой сложности можно составить только из трех структур, называемых следованием, ветвлением и циклом. Их называют базовыми конструкциями структурного программирования.

Следованием называется конструкция, представляющая собой последовательное выполнение двух или более операторов (простых или составных).

Ветвление задает выполнение либо одного, либо другого оператора в зависимости от выполнения какого-либо условия.

Цикл задает многократное выполнение оператора.

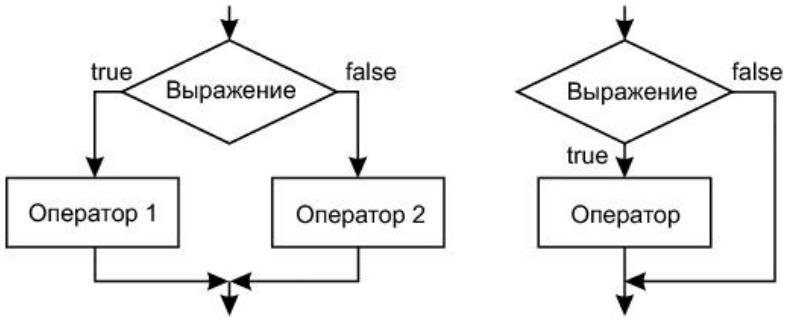
Особенностью базовых конструкций является то, что любая из них имеет только один вход и один выход, поэтому конструкции могут вкладываться друг в друга произвольным образом.



Условный оператор if используется для разветвления процесса вычислений на два направления. Сначала вычисляется логическое выражение. Если оно имеет значение true, выполняется первый оператор, иначе — второй. После этого управление передается на оператор, следующий за условным. Ветвь else может отсутствовать.

Формат оператора:

```
if ( логическое_выражение ) оператор_1; [ else оператор_2; ]
```



Оператор безусловного перехода goto используется в одной из трех форм:

```
goto метка;
goto case константное_выражение;
```

```
goto default;
```
В теле той же функции должна присутствовать ровно одна конструкция вида:

```
метка: оператор;
```

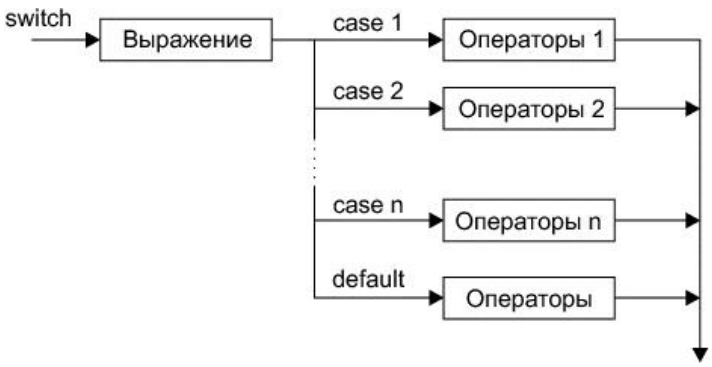
Оператор `goto метка` передает управление на помеченный оператор. Метка — это обычный идентификатор, областью видимости которого является функция, в теле которой он задан. Метка должна находиться в той же области видимости, что и оператор перехода.

Вторая и третья формы оператора goto используются в теле оператора выбора switch. Оператор `goto case константное_выражение` передает управление на соответствующую константному выражению ветвь, а оператор `goto default` — на ветвь default.

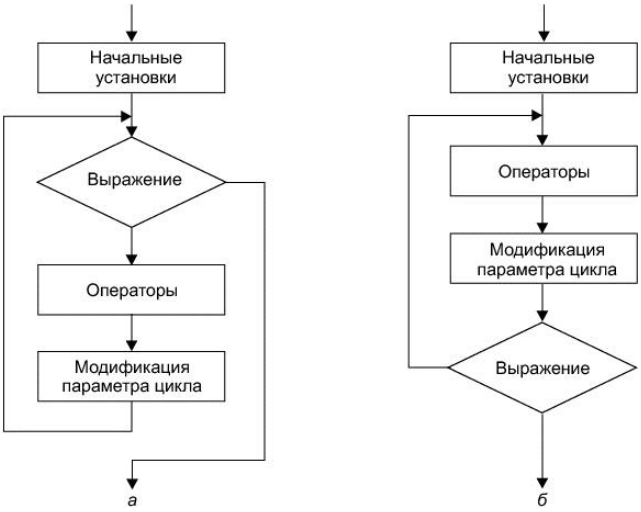
Оператор выбора switch (переключатель) предназначен для разветвления процесса вычислений на несколько направлений.

Формат оператора:

```
switch ( выражение ){
    case константное_выражение_1: [ список_операторов_1 ]
    case константное_выражение_2: [ список_операторов_2 ]
    ...
    case константное_выражение_n: [ список_операторов_n ]
    [ default: операторы ]
}
```



Операторы цикла используются для вычислений, повторяющихся многократно. Блок, ради выполнения которого и организуется цикл, называется телом цикла. Остальные операторы служат для управления процессом повторения вычислений: это начальные установки, проверка условия продолжения цикла и модификация параметра цикла. Один проход цикла называется итерацией.



Начальные установки служат для того, чтобы до входа в цикл задать значения переменных, которые в нем используются. Проверка условия продолжения цикла выполняется на каждой итерации либо до тела цикла (тогда говорят о цикле с предусловием), либо после тела цикла (цикл с постусловием). Параметром цикла называется переменная, которая используется при проверке условия продолжения цикла и принудительно изменяется на каждой итерации, причем, как правило, на одну и ту же величину. Если параметр цикла целочисленный, он называется счетчиком цикла. Цикл завершается, если условие его продолжения не выполняется. Возможно принудительное завершение как текущей итерации, так и цикла в целом. Для этого служат операторы break, continue, return и goto.

Цикл с предусловием while (выражение вычисляется перед каждой итерацией цикла. Если при первой проверке выражение равно false, цикл не выполнится ни разу)

Формат оператора:

```
while ( выражение ) оператор
```

Цикл с постусловием do (сначала выполняется простой или составной оператор, образующий тело цикла, а затем вычисляется выражение (оно должно иметь тип bool). Если выражение истинно, тело цикла выполняется еще раз, и проверка повторяется. Цикл завершается, когда выражение станет равным false или в теле цикла будет выполнен какой-либо оператор передачи управления, этот вид цикла применяется в тех случаях, когда тело цикла необходимо обязательно выполнить хотя бы один раз).

```
do оператор while выражение;
```

Цикл с параметром for

```
for ( инициализация; выражение; модификации ) оператор;
```

Инициализация служит для объявления величин, используемых в цикле, и присвоения им начальных значений. В этой части можно записать несколько операторов, разделенных запятой.

Областью действия переменных, объявленных в части инициализации цикла, является цикл. Инициализация выполняется один раз в начале исполнения цикла.

Выражение типа bool определяет условие выполнения цикла: если его результат равен true, цикл выполняется.

Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметров цикла. В части модификаций можно записать несколько операторов через запятую.

Простой или составной оператор представляет собой тело цикла. Любая из частей оператора for может быть опущена (но точки с запятой надо оставить на своих местах).

Оператор foreach

Оператор foreach применяется для перебора элементов в специальным образом организованной группе данных. Массив является именно такой группой. Удобство этого вида цикла заключается в том, что нам не требуется определять количество элементов в группе и выполнять их перебор по индексу: мы просто указываем на необходимость перебрать все элементы группы. Синтаксис оператора:

```
foreach ( тип имя in выражение ) тело_цикла
```

Имя задает локальную по отношению к циклу переменную, которая будет по очереди принимать все значения из массива выражение (в качестве выражения чаще всего применяется имя массива или другой группы данных). В простом или составном операторе, представляющем собой тело цикла, выполняются действия с переменной цикла. Тип переменной должен соответствовать типу элемента массива.

Функция — это именованная последовательность описаний и операторов, выполняющая какое-либо законченное действие. Функция может принимать параметры и возвращать значение. Функции, которые не возвращают значений, иногда называют **процедурами**. В некоторых языках программирования объявления функций и процедур имеют различный синтаксис, в частности, могут использоваться различные ключевые слова. В объектно-ориентированном программировании функции, объявления которых являются неотъемлемой частью определения класса, называются **методами**.

С помощью функций задача может быть разделена на более простые и обозримые, после чего программу можно рассматривать в более укрупненном виде — на уровне взаимодействия функций. Использование функций является первым шагом к повышению степени абстракции программы и ведет к упрощению ее структуры.

Разделение программы на функции позволяет также избежать избыточности кода, поскольку функцию записывают один раз, а вызывать ее на выполнение можно из разных точек программы многократно.

Процесс отладки программы, содержащей функции, можно лучше структурировать. Часто используемые функции можно помещать в библиотеки. Таким образом создаются более простые в отладке и сопровождении программы.

Функция начинает выполняться в момент вызова. Любая функция должна быть объявлена и определена. Как и для других величин, объявлений может быть несколько, а определение только одно.

Объявление функции должно находиться в тексте раньше ее вызова для того, чтобы компилятор мог осуществить проверку правильности вызова.

Объявление функции (прототип, заголовок, сигнатура) задает ее имя, тип возвращаемого значения и список передаваемых параметров.

Определение функции содержит, кроме объявления, тело функции, представляющее собой последовательность операторов и описаний в фигурных скобках:

```
[ атрибуты ] [ спецификаторы ] тип имя_метода ( [ параметры ] )
    тело_метода
```