Объектно-ориентированное программирование

Лекция №2. Управляющие конструкции, циклы и функции

# Темы:

1. Блоки кода. Области видимости. Управление ходом программы. Условный оператор. Оператор выбора

2. Циклы. Цикл с постусловием. Цикл с предусловием. Цикл for. Ключевые слова break, pause continue.

3. Функции. Методы. Параметры методов. Передача параметров по значению, по ссылке. Выходной параметр. Возвращаемое значение.

# Выражения, блоки и пустые операторы

Любое выражение, завершающееся точкой с запятой, рассматривается как оператор, выполнение которого заключается в вычислении выражения. Частным случаем выражения является пустой оператор ; (он используется, когда по синтаксису оператор требуется, а по смыслу — нет). Примеры:

i++; // выполняется операция инкремента

а \*= b + с; // выполняется умножение с присваиванием

fun( i, k ); // выполняется вызов функции

while( true ); // цикл из пустого оператора (бесконечный)

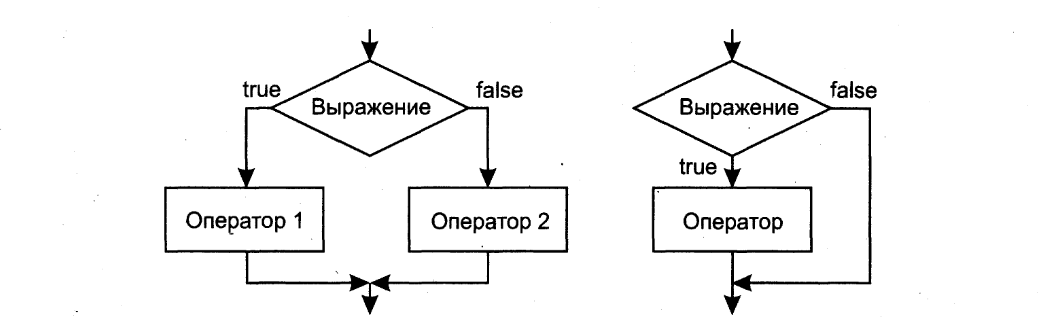
Блок, или составной оператор, — это последовательность описаний и операторов, заключенная в фигурные скобки. Блок воспринимается компилятором как один оператор и может использоваться всюду, где синтаксис требует одного оператора, а алгоритм — нескольких. Блок может содержать один оператор или быть пустым.

# Операторы ветвления

Операторы ветвления if и switch применяются для того чтобы в зависимости от конкретных значений исходных данных обеспечить выполнение разных последовательностей операторов. Оператор if обеспечивает передачу управления на одну из двух ветвей вычислений, а оператор switch — на одну из произвольного числа ветвей.

# Условный оператор if

Условный оператор if используется для разветвления процесса вычислений на два направления. Структурная схема оператора приведена на рис. 1.

  
Иллюстрация 1: Структурная схема условного оператора

Формат оператора:

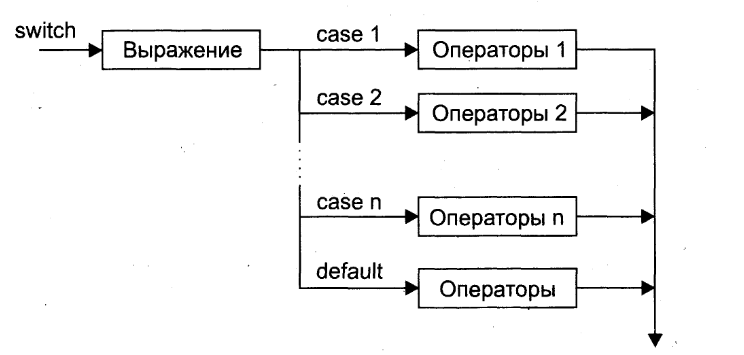
if ( логическое\_выражение ) оператор1; [ else оператор2; ]

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.

Если в какой-либо ветви требуется выполнить несколько операторов, их необходимо заключить в блок, иначе компилятор не сможет понять, где заканчивается ветвление. Блок может содержать любые операторы, в том числе описания и другие условные операторы (но не может состоять из одних описаний). Необходимо учитывать, что переменная, описанная в блоке, вне блока не существует.

# Оператор выбора switch

Оператор switch (переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Структурная схема оператора приведена на рис. 2.

  
Иллюстрация 2: Структурная схема оператора switch

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 ]

}

Выполнение оператора начинается с вычисления выражения. Тип выражения чаще всего целочисленный (включая char) или строковый1. Затем управление передается первому оператору из списка, помеченному константным выражением, значение которого совпало с вычисленным. Все константные выражения должны быть неявно приводимы к типу выражения в скобках. Если совпадения не произошло выполняются операторы, расположенные после слова default (а при его отсутствии управление передается следующему за switch оператору).

Каждая ветвь переключателя должна заканчиваться явным оператором перехода, а именно оператором break, goto или return:

* оператор break выполняет выход из самого внутреннего из объемлющих его операторов switch, for, while и do
* оператор goto выполняет переход на указанную после него метку, обычно это метка case одной из нижележащих ветвей оператора switch
* оператор return выполняет выход из функции, в теле которой он записан

Оператор goto обычно используют для последовательного выполнения нескольких ветвей переключателя, однако поскольку это нарушает читабельность программы, такого решения cледует избегать.

Оператор switch предпочтительнее оператора if в тех случаях, когда в программе требуется разветвить вычисления на количество направлений большее двух и выражение, по значению которого производится переход на ту или иную ветвь, не является вещественным. Часто это справедливо даже для двух ветвей, поскольку повышает наглядность программы.

# Операторы цикла

Операторы цикла используются для вычислений, повторяющихся многократно. В C# имеется четыре вида циклов: цикл с предусловием while, цикл с постусловием repeat, цикл с параметром for и цикл перебора foreach. Каждый из них состоит из определенной последовательности операторов.

Блок, ради выполнения которого и организуется цикл, называется телом цикла. Остальные операторы служат управления процессом повторения: это начальные установки, проверка условия продолжения цикла и модификация параметра цикла (рис. 4.4). 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, схема которого показана на

рис. 3, а), либо после тела цикла (цикл с постусловием, рис. 3, б). Разница между ними состоит в том, что тело цикла с постусловием всегда выполняется хотя бы один раз, после чего проверяется, надо ли его выполнять еще раз. Проверка необходимости выполнения цикла с предусловием делается до тела цикла, поэтому возможно, что он не выполнится ни разу.

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. Количество повторений такого цикла можно определить заранее. Параметр есть не у всякого цикла. В так называемом итеративном цикле условие продолжения содержит переменные, значения которых изменяются в цикле по рекуррентным формулам1. 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. Передавать управление извне внутрь цикла запрещается (при этом возникает ошибка компиляции).

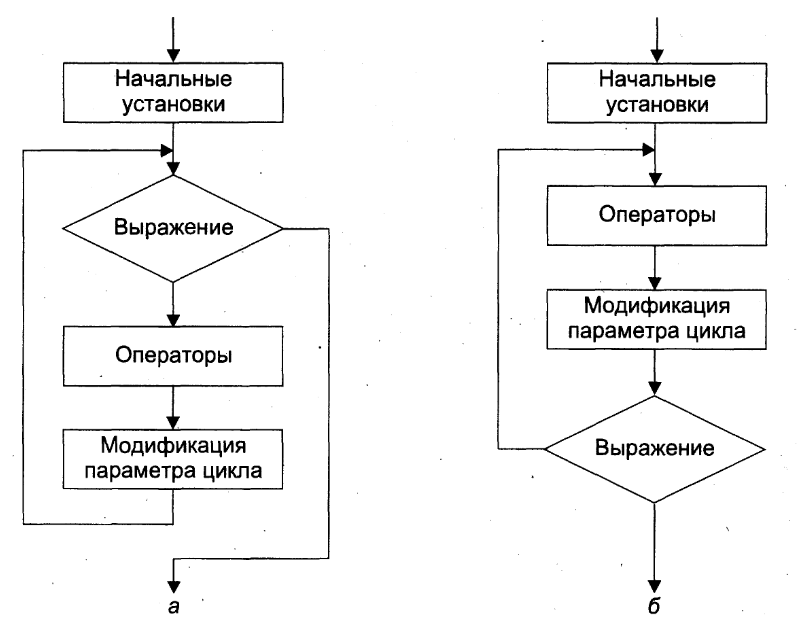


Иллюстрация 3: Структурные схемы операторов цикла

## Цикл с предусловием while

Формат оператора прост:

while ( выражение ) оператор

Выражение должно быть логического типа. Например, это может быть операция отношения или просто логическая переменная. Если результат вычисления выражения равен true, выполняется простой или составной оператор (блок). Эти действия повторяются до того момента, пока результатом выражения не станет значение false. После окончания цикла управление передается на следующий за ним оператор. Выражение вычисляется перед каждой итерацией цикла. Если при первой проверке выражение равно false, цикл не выполнится ни разу. Если в теле цикла необходимо выполнить более одного оператора, необходимо заключить их в блок с помощью фигурных скобок.

## Цикл с постусловием do

Цикл с постусловием реализует структурную схему, приведенную на рис. 3, б, и имеет вид

do оператор while выражение;

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-либооператор передачи управления. Этот вид цикла применяется в тех случаях, когда тело цикла необходимо обязательно выполнить хотя бы один раз, например, если в цикле вводятся данные и выполняется их проверка. Если же такой необходимости нет, предпочтительнее пользоваться циклом с предусловием.

## Цикл с параметром for

Цикл с параметром имеет следующий формат:

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, например:

for ( int i =О, j = 20; ...

int k, m;

for ( k = 1, m = 0; ...

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. Цикл с параметром реализован как цикл с предусловием.

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

for ( int i = 0, j = 20; i < 5 && j > 10; i++, j-- ) ...

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!).

# Методы

Метод — это функциональный элемент класса, который реализует вычисления или другие действия, выполняемые классом или экземпляром. Методы определяют поведение класса. Метод представляет собой законченный фрагмент кода, к которому можно обратиться по имени. Он описывается один раз, а вызываться может столько раз, сколько необходимо. Один и тот же метод может обрабатывать различные данные, переданные ему в качестве аргументов.

Синтаксис метода:

[ атрибуты ] [ спецификаторы ] тип имя\_метода ( [ параметры ] )

телометода

Рассмотрим основные элементы описания метода. Первая строка представляет собой заголовок метода. Тело метода, задающее действия, выполняемые методом, чаще всего представляет собой блок — последовательность операторов в фигурных скобках. Тип определяет, значение какого типа вычисляется с помощью метода. Часто употребляется термин «метод возвращает значение», поскольку после выполнения метода происходит возврат в то место вызывающей функции, откуда был вызван метод, и передача туда значения выражения, записанного в операторе return. Если метод не возвращает никакого значения, в его заголовке задается тип void, а оператор return отсутствует.

Параметры используются для обмена информацией с методом. Параметр представляет собой локальную переменную, которая при вызове метода принимает значение соответствующего аргумента. Область действия параметра — весь метод.

Имя метода вкупе с количеством, типами и спецификаторами его параметров представляет собой сигнатуру метода — то, по чему один метод отличают от других. В классе не должно быть методов с одинаковыми сигнатурами.

## Параметры методов

Рассмотрим более подробно, каким образом метод обменивается информацией с вызвавшим его кодом. При вызове метода выполняются следующие действия:

1. Вычисляются выражения, стоящие на месте аргументов.
2. Выделяется память под параметры метода в соответствии с их типом.
3. Каждому из параметров сопоставляется соответствующий аргумент (аргументы как бы накладываются на параметры и замещают их).
4. Выполняется тело метода.
5. Если метод возвращает значение, оно передается в точку вызова; если метод имеет тип void, управление передается на оператор, следующий после вызова. При этом проверяется соответствие типов аргументов и параметров и при необходимости выполняется их преобразование. При несоответствии типов выдается диагностическое сообщение.

Количество аргументов должно соответствовать количеству параметров. На имена никаких ограничений не накладывается: имена аргументов могут как совпадать, так и не совпадать с именами параметров.

Существуют два способа передачи параметров: по значению и по ссылке.

При передаче по значению метод получает копии значений аргументов, и операторы метода работают с этими копиями. Доступа к исходным значениям аргументов у метода нет, а следовательно, нет и возможности их изменить.

При передаче по ссылке (по адресу) метод получает копии адресов аргументов, он осуществляет доступ к ячейкам памяти по этим адресам и может изменять исходные значения аргументов, модифицируя параметры.

В C# для обмена данными между вызывающей и вызываемой функциями предусмотрено четыре типа параметров:

* параметры-значения;
* параметры-ссылки — описываются с помощью ключевого слова ref;
* выходные параметры — описываются с помощью ключевого слова out;
* параметры-массивы — описываются с помощью ключевого слова params.

Ключевое слово предшествует описанию типа параметра. Если оно опущено, параметр считается параметром-значением. Параметр-массив может быть только один и должен располагаться последним в списке.

## Параметры-значения

Параметр-значение описывается в заголовке метода следующим образом:

тип имя

Пример заголовка метода, имеющего один параметр-значение целого типа:

void Р( int х )

Имя параметра может быть произвольным. Параметр х представляет собой локальную переменную, которая получает свое значение из вызывающей функции при вызове метода. В метод передается копия значения аргумента.

Механизм передачи следующий: из ячейки памяти, в которой хранится переменная, передаваемая в метод, берется ее значение и копируется в специальную область памяти — область параметров. Метод работает с этой копией, следовательно, доступа к ячейке, где хранится сама переменная, не имеет. По завершении работы метода область параметров освобождается. Таким образом, для параметров-значений используется, как вы догадались, передача по значению. Ясно, что этот способ годится только для величин, которые не должны измениться после выполнения метода, то есть для его исходных данных.

При вызове метода на месте параметра, передаваемого по значению, может находиться выражение, а также, конечно, его частные случаи — переменная или константа. Должно существовать неявное преобразование типа выражения к типу параметра.

## Параметры-ссылки

Во многих методах все величины, которые метод должен получить в качестве исходных данных, описываются в списке параметров, а величина, которую вычисляет метод как результат своей работы, возвращается в вызывающий код с помощью оператора return. Очевидно, что если метод должен возвращать более одной величины, такой способ не годится. Еще одна проблема возникает, если в методе требуется изменить значение каких-либо передаваемых в него величин. В этих случаях используются параметры-ссылки.

Признаком параметра-ссылки является ключевое слово ref перед описанием параметра:

ref тип имя

Пример заголовка метода, имеющего один параметр-ссылку целого типа:

void Р( ref int х )

При вызове метода в область параметров копируется не значение аргумента, а его адрес, и метод через него имеет доступ к ячейке, в которой хранится аргумент. Таким образом, параметры-ссылки передаются по адресу (чаще употребляется термин «передача по ссылке»). Метод работает непосредственно с переменной из вызывающей функции и, следовательно, может ее изменить, поэтому если в методе требуется изменить значения параметров, они должны передаваться только по ссылке.

## Выходные параметры

Довольно часто возникает необходимость в методах, которые формируют несколько величин, например, если в методе создаются объекты или инициализируются ресурсы. В этом случае становится неудобным ограничение параметров-ссылок: необходимость присваивания значения аргументу до вызова метода. Это ограничение снимает спецификатор out. Параметру, имеющему этот спецификатор, должно быть обязательно присвоено значение внутри метода, компилятор за этим следит. Зато в вызывающем коде можно ограничиться описанием переменной без инициализации.

# Использованная литература

1. Павловская Т.А. C# Программирование на языке высокого уровня, Питер 2014г. 432 с.