УРУМКАЛИЕВА Ч.Ч.
Кыргызский Национальный Университет им.Ж.Баласагына
УДК: 681.142.2+42/49

ОСОБЕННОСТИ СИНТАКСИСА УСЛОВНОГО ОПЕРАТОРА В ЯЗЫКЕ ПРОГРАМИРОВАНИЯ C# 180

В данной статье рассматривается отличительные особенности программирования условного оператора на языке си шарп. Условный оператор if используется для разветвления процесса обработки данных на два направления. К операторам ветвления относятся условный оператор if и оператор выбора switch. Он может иметь одну из форм: сокращенную или полную.

Бул статьяда си шарп программалоо тилиндеги тармактуу операторунун колдонуу эрежесинин озгочолугу жазылат.Тармактуу операторду катарына оператор if жана оператор switch кирет, жана толук же кыскартылган формада жазылат.

This article examines the distinctive features of the programming of the conditional operator in C Sharp. If statement is used to branch the process data in two directions. To the operators are branching and conditional operator if the operator selection switch. It can have one of the forms: condensed or complete.

Существует множество сред языков программирования. И у каждой из них есть свои преимущества и недостатки. Переняв многое от своих предшественников — языков C++, Java,— C# исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем. Для организации условного ветвления язык C# унаследовал от C и C++ конструкцию **if. . .else.** Ее синтаксис интуитивно понятен для любого, кто программировал на процедурных языках. Как в C++ и других языках программирования, в языке C# для выбора одной из нескольких возможностей используются две конструкции **- if и switch .** Первую из них обычно называют альтернативным выбором, вторую - разбором случаев.

Начнем с синтаксиса оператора if:

if (условие)
оператор(ы)
else оператор(ы)

Если по каждому из условий нужно выполнить более одного оператора, эти операторы должны быть объединены в блок с помощью фигурных скобок ({...}). (Это также касается других конструкций C# в которых операторы могут быть объединены в блок — таких как циклы for и while.)

Условный оператор if используется для разветвления процесса обработки данных на два направления. Он может иметь одну из форм: сокращенную или полную.

Форма сокращенного оператора if:

if (B) S;

где B - логическое или арифметическое выражение, истинность которого проверяется; S - оператор: простой или составной.

При выполнении сокращенной формы оператора if сначала вычисляется выражение B, затем проводится анализ его результата: если B истинно, то выполняется оператор S ; если B ложно, то оператор S пропускается. Таким образом, с помощью сокращенной формы оператора if можно либо выполнить оператор S, либо пропустить его.

Форма полного оператора if:

if (B) S1; else S2;

где B - логическое или арифметическое выражение, истинность которого проверяется; S1, S2 - оператор: простой или составной.

При выполнении полной формы оператора if сначала вычисляется выражение B, затем анализируется его результат: если B истинно, то выполняется оператор S1, а оператор S2 пропускается; если B ложно, то выполняется оператор S2, а S1 -

пропускается. Таким образом, с помощью полной формы оператора if можно выбрать одно из двух альтернативных действий процесса обработки данных.

Рассмотрим несколько примеров записи условного оператора if:

```
if (a > 0)  x=y;// Сокращенная форма с простым оператором
if (++i) // Сокращенная форма с составным оператором
 {
x=y; y=2*z;
}
if (a > 0 || b<0) x=y; else  x=z;// Полная форма с простым оператором
if (i+j-1) // Полная форма с составными операторами
{
x= 0; y= 1;
}
Else
{
x=1; y:=0;
}
```

Этот синтаксис подобен C++ и Java, но, опять-таки, отличается от Visual Basic. Разработчики Visual Basic должны отметить, что C# не имеет оператора, соответствующего End If. Вместо этого существует правило, что каждая часть if содержит только одну конструкцию. Если требуется указать более одного оператора, то они заключаются в фигурные скобки, что заставляет трактовать всю группу операторов как единый блок. Оператор if можно использовать без финального else. Можно также комбинировать else if, чтобы проверять множество условий:

if (выражение_1)
оператор_1
else if (выражение_2)
оператор_2
…
else if (выражение_K)
оператор_K
else оператор_N

Следует отметить следующую особенность синтаксиса: выражения **if** должны заключаться в круглые скобки и быть булевого типа. Точнее, выражения должны давать значения **true** или **false.** По правилам синтаксиса языка C++, then-ветвь оператора следует сразу за круглой скобкой без ключевого слова then, типичного для большинства языков программирования. Каждый из операторов может быть блоком - в частности, if-оператором. Поэтому возможна и такая конструкция:

if (выражение1) if (выражение2) if (выражение3) ...

Ветви **else** и **if,** позволяющие организовать выбор из многих возможностей, могут отсутствовать. Может быть опущена и заключительная else-ветвь. В этом случае краткая форма оператора **if** задает альтернативный выбор - делать или не делать - выполнять или не выполнять then-оператор.

Выражения **if** проверяются в порядке их написания. Как только получено значение **true,** проверка прекращается и выполняется оператор (это может быть блок), который следует за выражением, получившим значение **true.** С завершением этого оператора завершается и оператор **if.** Ветвь **else,** если она есть, относится к ближайшему открытому **if.**

```
static void Main(string[] args)
{
Console.WriteLine("Введите строку");
string input;
```

```
input = Console.ReadLine ();
if (input == "")
Console.WriteLine ("Вы ввели пустую строку");
else if (input.Length < 3)
Console.WriteLine("Строка содержит менее 3 символов");
else if (input.Length < 10)
Console.WriteLine("Строка содержит от 3-х, но не более 10 символов");
Console.WriteLine("Была введена строка " + input);
}
```

Количество else if, добавляемых к единственному if, не ограничено. Как видите, этот пример кода объявляет строковую переменную input, приглашает пользователя ввести текст в командной строке, помещает его в input, затем проверяет длину строковой переменной. Этот код также показывает, насколько легко осуществляются манипуляции со строками в C#. Например, чтобы определить длину input, используется input. Length.

Один момент, который следует отметить касательно if: не обязательно использовать фигурные скобки, если в условной ветви присутствует только один оператор: if

Однако для согласованности многие программисты предпочитают использовать фигурные скобки с оператором if в любом случае.

Представленный пример с if также иллюстрирует некоторые операции **C#**, предназначенные для сравнения значений. В частности, отметим, что в C#, подобно C++ и Java, для проверки на равенство используется операция == Не используйте для этой цели = Одиночный знак равенства служит для присваивания значений.

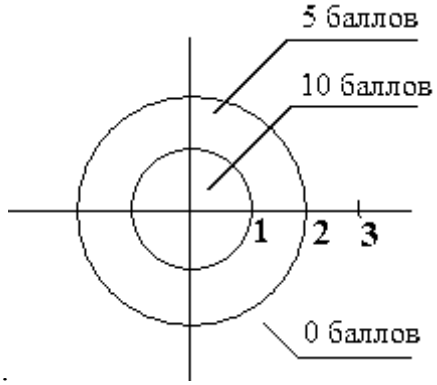
Рассмотрим пример использования условного оператора.

```
static void Main()
{
 Console.Write("x= ");
 float x = float.Parse(Console.ReadLine());
    Console.Write("y=");
 float y = float.Parse(Console.ReadLine());
 if (x < y ) Console.WriteLine("min= "+x);
 else Console.WriteLine("min= "+y);
}
```

Результат работы программы:

x	y	min
0	0	0
1	-1	-1
-2	2	-2

Рассмотрим пример программы, использующей вложенные условные операторы.



.

Дана мишень подсчитать количество очков после выстрела по данной мишени.

```
static void Main()
{
```

```
  int Ball=0;
     Console.Write("x= ");
  float x = float.Parse(Console.ReadLine());
  Console.Write("y= ");
  float y = float.Parse(Console.ReadLine());
  if (x * x + y * y <=1) Ball = 10;            //окружность с радиусом 1
  else  if (x * x + y * y <= 4) Ball = 5;      //окружность с радиусом 2
  Console.WriteLine("Ball= "+ Ball);
}
```

Результат работы программы:

x	y	Ball
0	0	10
1	-1	5
-2	2	0

Частным, но важным случаем выбора из нескольких вариантов является ситуация, при которой выбор варианта определяется значениями некоторого выражения. Соответствующий оператор C#, унаследованный от C++, но с небольшими изменениями в синтаксисе, называется оператором switch.

Оператор выбора switch предназначен для разветвления процесса вычислений по нескольким направлениям. Формат оператора:

```
switch ( <выражение> )
{
 case <константное_выражение_1>:
  [<оператор 1>];  <оператор перехода>;
 case <константное_выражение_2>:
  [<оператор 2>]; <оператор перехода>;
 ...
 case <константное_выражение_n>:
  [<оператор n>]; <оператор перехода>;
  [default: <оператор>; ]
}
```

Выражение, записанное в квадратных скобках, является необязательным элементом в *операторе switch*. Если оно отсутствует, то может отсутствовать и оператор перехода.

Выражение, стоящее за ключевым словом switch, должно иметь арифметический, символьный, строковый тип или тип указатель. Все константные выражения должны иметь разные значения, но их тип должен совпадать с типом выражения, стоящим после switch или приводиться к нему. Ключевое слово case и расположенное после него константное выражение называют также меткой case.

Выполнение оператора начинается с вычисления выражения, расположенного за ключевым словом switch. Полученный результат сравнивается с меткой case. Если результат выражения соответствует метке case, то выполняется оператор, стоящий после этой метки, за которым обязательно должен следовать оператор перехода: break, goto и т.д. При использовании оператора break происходит выход из switch и управление передается оператору, следующему за switch. Если же используется оператор goto, то управление передается оператору, помеченному меткой, стоящей после goto.

Когда разбор случаев предполагает проверку попадания в некоторый диапазон значений, приходится прибегать к оператору **if** для формирования специальной переменной. Разбор случаев - это часто встречающаяся ситуация в самых разных задачах Этот прием демонстрируется в следующем примере.

```
/// Определяет период в зависимости от возраста - age и использование ветвящегося оператора if
public void SetPeriod() {
```

```
        if ((age > 0) && (age < 7))
          period = 1;
        else if ((age >= 7) && (age < 17))
          period = 2;
        else if ((age >= 17) && (age < 22))
          period = 3;
        else if ((age >= 22) && (age < 27))
          period = 4;
        else if ((age >= 27) && (age < 37))
          period = 5;
        else period = 6;
}
```

Этот пример демонстрирует применение ветвящегося оператора **if.** С содержательной точки зрения он интересен тем, что в поля класса пришлось ввести специальную переменную **period,** позволяющую в дальнейшем использовать разбор случаев в зависимости от периода жизни:

```
/// Определяет статус в зависимости от периода - period и  использование разбора случаев - оператора Switch

public void SetStatus() {
        switch (period) {
        case 1:
          status = "child"; break;
        case 2:
          status = "schoolboy"; break;
        case 3:
          status = "student"; break;
        case 4:
          status = "junior researcher"; break;
        case 5:
          status = "senior researcher"; break;
        case 6:
          status = "professor"; break;
        default:
          status = "не определен"; break;
        }
        Console.WriteLine("Имя = {0}, Возраст = {1}, Статус = {2}", name, age, status);
}
```

//По заданному виду арифметической операции (сложение, вычитание, умножение и деление) и двум операндам, вывести на экран результат применения данной операции к операндам.

```
        static void Main()
        {
          Console.Write("OPER= ");
          char oper=char.Parse(Console.ReadLine());
          bool ok=true;
          Console.Write("A= ");
          int a=int.Parse(Console.ReadLine());
          Console.Write("B= ");
          int b=int.Parse(Console.ReadLine());
          float res=0;
          switch (oper)
```

```
    {
     case '+': res = a + b; break;
     case '-': res = a - b; break;
     case '*': res = a * b; break;
     case ':': if (b != 0)
     {
       res = (float)a / b; break;
     }
     else goto default;
     default: ok = false; break;
    }
    if (ok) Console.WriteLine("{0} {1} {2} = {3}", a, oper, b, res);
    else Console.WriteLine("error");
  }
```

Результат выполнения программы:

oper	x	y	rez
+	4	5	9
:	4	0	error

Ветвь default в операторе switch не является обязательной. Когда она не нужна, ее можно просто опустить. Переход последовательности операторов, связанной с одной ветвью case, в следующую ветвь case считается ошибкой, поскольку в C#должна непременно соблюдаться правило недопущения провалов в передаче управления ходом выполнения программы. Именно поэтому последовательность операторов в каждой ветви case оператора switch оканчивается оператором break(можно также с помощью оператора goto) но для данной цели чаще применяется оператор break. Последовательность операторов ветви default должна быть лишена провалов, поэтому она завершается оператором break. Правило недопущение провалов относится к тем особенностям языка C#, которыми он отличается от C, C++и Java.

Литература:

1. Фленов М.Е. «Библия C#» «БХВ-Петербург»2012г.
2. Г. Шилдт. C#4.0: полное руководство.:-М.:ООО «И.Д. Вильямс»,2011г.
3. П. Франка C++ учебный курс-Спб: Питер Ком, 1999г.
4. Н.Культин Основы программирования в Microsoft Visual C#2010-СПб.:БХВ-Петербург,2011.
5. Симонович С.В.Информатика. Базовый курс/Симонович С.В. и др. — СПб.: издательство "Питер", 2000. — 640 с.