Программирование C#

3. Управляющие операторы

Карбаев Д.С., 2016

Управляющие операторы

- операторы выбора: if и switch;
- итерационные операторы, в том числе операторы цикла: for, while, do-while и foreach;
- операторы перехода: break, continue, goto, return и throw.

Оператор if

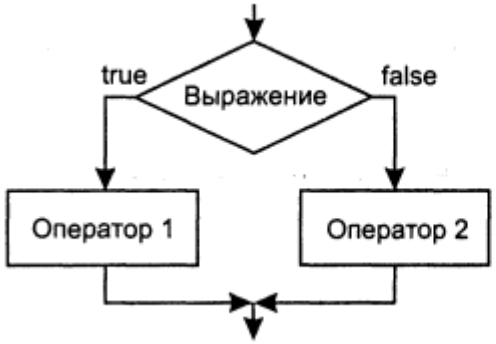
Форма записи этого оператора:

```
if(условие) оператор;
else оператор;
```

где условие — это некоторое условное выражение, а оператор — адресат операторов if и else.

Общая форма оператора if, в котором используются блоки операторов:

```
if (условие)
{
 последовательность операторов
}
else
{
 последовательность операторов
}
```



Оператор if

Пример: является ли число положительным или отрицательным.

```
static void Main(string[] args)  {
 int i;
 for (i = -5; i <= 5; i++) {
  Console.Write("Проверка " + i + ": ");
  if (i < 0)
    Console.WriteLine("отрицательное число");
  else
    Console.WriteLine("положительное число");
 }
}
```

Результат выполнения:

```
Проверка -5: отрицательное число
Проверка -4: отрицательное число
Проверка -3: отрицательное число
Проверка -2: отрицательное число
Проверка -1: отрицательное число
Проверка 0: положительное число
Проверка 1: положительное число
Проверка 2: положительное число
Проверка 3: положительное число
Проверка 4: положительное число
Проверка 5: положительное число
```

Вложенный оператор if

Пример:

```
if(i == 10) {
  if (j < 20) a = b;
  if(k > 100) c = d;
  else a = c; // этот оператор else связан с оператором if(k > 100)
}
else a = d; // этот оператор else связан с оператором if(i == 10)
```

Вложенный оператор if

Пример: является ли число положительным или отрицательным.

```
static void Main(string[] args) {
 int i;
 for (i = -5; i <= 5; i++) {
  Console.Write("Проверка " + i + " : ");
  if (i < 0)
    Console.WriteLine("отрицательное число");
  else if (i == 0)
    Console.WriteLine("число без знака");
  else Console.WriteLine("положительное число");
 }
}
```

Результат выполнения:

```
Проверка -5: отрицательное число
Проверка -4: отрицательное число
Проверка -3: отрицательное число
Проверка -2: отрицательное число
Проверка -1: отрицательное число
Проверка 0: число без знака
Проверка 1: положительное число
Проверка 2: положительное число
Проверка 3: положительное число
Проверка 4: положительное число
Проверка 5: положительное число
```

Конструкция if -else-if

Общая форма:

```
if (условие)
 оператор;
else if (условие)
 оператор;
else if (условие)
 оператор;
else
 оператор;
```

Конструкция if -else-if

Пример: Определить наименьший делитель заданного целого значения, состоящий из одной цифры.

Результат выполнения:

Наименьший делитель числа 2 равен 2

…

Наименьший делитель числа 10 равен 2

11 не делится на 2, 3, 5 или 7.

```
static void Main(string[] args) {
   int num;
   for (num = 2; num < 12; num++) {
    if ((num % 2) == 0)
     Console.WriteLine("Наименьший делитель числа " + num + " равен 2.");
    else if ((num % 3) == 0)
     Console.WriteLine("Наименьший делитель числа " + num + " равен 3.");
    else if ((num % 5) == 0)
     Console.WriteLine("Наименьший делитель числа " + num + " равен 5.");
    else if ((num % 7) == 0)
     Console.WriteLine("Наименьший делитель числа " + num + " равен 7.");
    else
     Console.WriteLine(num +
      " не делится на 2, 3, 5 или 7.");
   }
}
```

Оператор switch

Общая форма оператора:

```
switch(выражение) {
        case константа1:
         последовательность операторов
        break;
        case константа2:
         последовательность операторов
        break;
        case константа3:
         последовательность операторов
        break;
        default:
         последовательность операторов
        break;
}
```

Заданное выражение в операторе **switch** должно быть целочисленного типа (**char**, **byte**, **short** или **int**), перечислимого или же строкового. Выражения других типов, например с плавающей точкой, в операторе **switch** не допускаются.

Оператор switch

```
static void Main(string[] args)  {
 int i;
 for (i = 0; i < 10; i++)
  switch (i) {
   case 0:
    Console.WriteLine("i равно нулю");
    break;
   case 1:
    Console.WriteLine("i равно единице");
    break;
   case 2:
    Console.WriteLine("i равно двум");
    break;
   case 3:
    Console.WriteLine("i равно трем");
    break;
   case 4:
    Console.WriteLine("i равно четырем");
    break;
   default:
    Console.WriteLine("i равно или больше пяти");
    break;
  }
```

Результат выполнения:

```
i равно нулю.
i равно единице.
i равно двум.
i равно трем.
i равно четырем.
i равно или больше пяти.
i равно или больше пяти.
i равно или больше пяти.
i равно или больше пяти.
i равно или больше пяти.
```

Оператор switch

Использовать элементы типа char для управления оператором switch:

```
static void Main(string[] args)  {
 char ch;
 for (ch = 'A'; ch <= 'D'; ch++)
  switch (ch) {
   case 'A':
    Console.WriteLine("ch содержит A");
    break;
   case 'B':
    Console.WriteLine("ch содержит B");
    break;
   case 'C':
    Console.WriteLine("ch содержит C");
    break;
   case 'D':
    Console.WriteLine("ch содержит D");
    break;
  }
}
```

Результат выполнения:

```
ch содержит A
ch содержит B
ch содержит C
ch содержит D
```

Оператор switch

Пример "проваливания" пустых ветвей case :

```
static void Main(string[] args) {
 int i;
 for (i = 1; i < 5; i++)
  switch (i) {
   case 1:
   case 2:
   case 3: Console.WriteLine("i равно 1, 2 или 3");
    break;
   case 4: Console.WriteLine("i равно 4");
    break;
  }
}
```

Результат выполнения:

```
i равно 1, 2 или 3
i равно 1, 2 или 3
i равно 1, 2 или 3
i равно 4
```

Вложенные операторы switch

```
static void Main(string[] args)  {
 char ch1, ch2;
 // ...
 switch (ch1) {
  case 'A': Console.WriteLine("Эта ветвь A – часть " +
  "внешнего оператора switch.");
   switch (ch2){
     case 'A':
       Console.WriteLine("Эта ветвь A – часть " +
       "внутреннего оператора switch");
     break;
     case 'B': // ...
   } // конец внутреннего оператора switch
   break;
  case 'B': // ...
  }
}
```

Оператор цикла for

- Общая форма оператора for для повторного выполнения единственного оператора.

```
for (инициализация; условие; итерация) оператор;
```

- форма для повторного выполнения кодового блока

```
for(инициализация; условие; итерация)
{
   последовательность операторов;
}
```

- Пример: выполнение цикла for в отрицательном направлении

```
int x;
for(x = 100; x > -100; x -= 5)
Console.WriteLine(x);
```

- Пример невыполнения цикла:

```
for(count=10; count < 5; count++)
x += count; // этот оператор не будет выполняться
```

Оператор цикла for

Выяснить, является ли число простым.

```
static void Main(string[] args) {
 int num, i, factor;
 bool isprime;
 for (num = 2; num < 20; num++){
  isprime = true;
  factor = 0;
  // Делится ли значение переменной num нацело?
  for (i = 2; i <= num / 2; i++){
   if ((num % i) == 0){
    // Значение переменной num делится нацело.
    // Следовательно, это непростое число,
    isprime = false;
    factor = i;
   }
  }
  if (isprime)
   Console.WriteLine(num + " – простое число.");
  else
   Console.WriteLine("Наибольший делитель числа " +
                     num + " равен " + factor);
 }
}
```

Результат выполнения:

```
2 — простое число
3 — простое число
Наибольший делитель числа 4 равен 2
5 — простое число
Наибольший делитель числа 6 равен 3
7 — простое число
Наибольший делитель числа 8 равен 4
Наибольший делитель числа 9 равен 3
Наибольший делитель числа 10 равен 5
11 — простое число
Наибольший делитель числа 12 равен 6
13 — простое число
Наибольший делитель числа 14 равен 7
Наибольший делитель числа 15 равен 5
Наибольший делитель числа 16 равен 8
17 — простое число
Наибольший делитель числа 18 равен 9
19 - простое число
```

For: переменные управления циклом

- использовать две или более переменных для управления циклом.

```
int i, j;
for(i=0, j=10; i < j; i++, j--)
  Console.WriteLine("i и j : " + i + " " + j);
```

результат

```
i и j : 0 10
i и j : 1 9
i и j : 2 8
i и j : 3 7
i и j : 4 6
```

For: переменные управления циклом

Использовать запятые в операторе цикла for для выявления наименьшего и наибольшего делителя числа:

Результат выполнения:

```
Наибольший делитель: 50
Наименьший делитель: 2
```

```
static void Main(string[] args){
 int i, j;
 int smallest, largest;
 int num;
 num = 100;
 smallest = largest = 1;
 for (i = 2, j = num / 2; (i <= num / 2) & (j >= 2); i++, j--) {
  if ((smallest == 1) & ((num % i) == 0))
   smallest = i;
  if ((largest == 1) & ((num % j) == 0))
   largest = j;
 }
 Console.WriteLine("Наибольший делитель: " + largest);
 Console.WriteLine("Наименьший делитель: " + smallest);
}
```

For: условное выражение

Условием выполнения цикла может служить любое выражение типа bool:

```
int i, j;
bool done = false;
for(i=0, j=100; !done; i++, j--) {
  if(i*i >= j) done = true;
  Console.WriteLine("i, j: "+i+" "+j);
}
```

Результат выполнения:

```
0 100
1 99
2 98
3 97
4 96
5 95
6 94
7 93
8 92
9 91
10 90
```

For: отсутствующие части цикла

- Отдельные части цикла for могут оставаться пустыми.

```
int i;
for(i = 0; i < 10; ) {
  Console.WriteLine("Проход №" + i);
  i++; // инкрементировать переменную управления циклом
}
```

результат

```
Проход №0
Проход №1
…
Проход №9
```

- Исключить еще одну часть из определения цикла for.

```
int i;
i = 0; // исключить инициализацию из определения цикла
for(; i < 10; ) {
  Console.WriteLine("Проход №" + i);}
```

For: бесконечный цикл

```
for(;;) // цикл, намеренно сделанный бесконечным
{
  //...
}
```

- Тело цикла может быть пустым.

```
int i;
int sum = 0;
// получить сумму чисел от 1 до 5
for(i = 1; i <= 5; sum += i++ ); // цикл без тела
Console.WriteLine("Сумма равна " + sum); //уже после цикла
```

Результат:

```
Сумма равна 15
```

Примечание:

```
sum += i++; //равнозначно следующим операциям
sum = sum + i;
i++ ;
```

For: управляющие переменные в цикле

Объявить переменную управления циклом
в самом цикле for:

```
static void Main(string[] args)
{
 int sum = 0;
 int fact = 1;
 // вычислить факториал чисел от 1 до 5
 for (int i = 1; i <= 5; i++)
 {
  sum += i; // Переменная i действует в цикле.
  fact *= i;
 }
 //А здесь переменная i недоступна.
 Console.WriteLine("Сумма равна " + sum);
 Console.WriteLine("Факториал равен " + fact);

}
```

Оператор цикла while

- Общая форма оператора while.

```
while (условие) оператор;
```

- Пример: вычислить порядок величины целого числа.

```
static void Main(string[] args)  {
   int num = 435679;
   int mag=0;
   Console.WriteLine("Число: " + num);
   while (num > 0) {
     mag++;
     num = num / 10;
   }
   Console.WriteLine("Порядок величины: " + mag);
}
```

Результат

```
Число: 435679
Порядок величины: 6
```

Оператор цикла while

Вычислить целые степени числа 2:

```
static void Main(string[] args) {
   int e;
   int result;
   for (int i = 0; i < 10; i++) {
     result = 1;
     e = i;
     while (e > 0) {
       result *= 2;
       e--;
     }
     Console.WriteLine("2 в степени " + i
        + " равно " + result);
   }
}
```

Результат выполнения:

```
2 в степени 0 равно 1
2 в степени 1 равно 2
2 в степени 2 равно 4
2 в степени 3 равно 8
2 в степени 4 равно 16
2 в степени 5 равно 32
2 в степени 6 равно 64
2 в степени 7 равно 128
2 в степени 8 равно 256
2 в степени 9 равно 512
```

Оператор цикла do-while

- Общая форма оператора do-while.

```
do {
  операторы;
} while (условие);
```

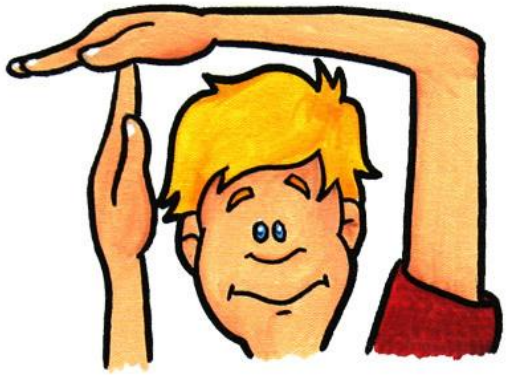
- Пример: отобразить цифры целого числа в обратном порядке.

```
static void Main(string[] args) {
    int num=198;
    int nextdigit;
    Console.WriteLine("Число: " + num);
    Console.Write("Число в обратном порядке: ");
    do {
      nextdigit = num % 10;
      Console.Write(nextdigit);
      num = num / 10;
    } while (num > 0);
    Console.WriteLine();
}
```

Результат

```
Число: 198
Число в обратном
порядке: 891
```

Оператор break



- Использовать оператор break для выхода из цикла.

```
for(int i = -10; i <= 10; i++) {
  if(i > 0) break; // завершить цикл,
            //как только значение
            // переменной i станет положительным
  Console.Write (i + " ");
}
Console.WriteLine ("Готово!");
```

результат

```
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 Готово!
```

- Применить оператор break для выхода из цикла do-while.

```
int i;
i = -10;
do {
  if(i > 0) break;
  Console.Write (i + " ");
  i++;
} while (i <= 10);
Console.WriteLine("Готово!");
```

Оператор break

- Выявить наименьший делитель числа.

```
int factor = 1;
int num = 1000;
for (int i=2; i <= num/i; i++) {
  if((num%i) == 0) {
   factor = i;
   break; // прервать цикл, как только будет
   // выявлен наименьший делитель числа
  }
}
Console.WriteLine("Наименьший делитель равен " + factor);
```

результат

```
Наименьший делитель равен 2
```

Оператор break

Применить оператор break во вложенных циклах:

```
static void Main(string[] args)  {
 for (int i = 0; i < 3; i++) {
  Console.WriteLine("Подсчет во внешнем цикле: " + i);
  Console.Write(" Подсчет во внутреннем цикле: ");
  int t = 0;
  while (t < 100) {
    if (t == 10) break; // прервать цикл,
                        //если t равно 10
    Console.Write(t + " ");
    t++;
   }
   Console.WriteLine();
  }
  Console.WriteLine("Циклы завершены.");
 }
```

Результат выполнения:

```
Подсчет во внешнем цикле: 0
Подсчет во внутреннем цикле: 0123456789
Подсчет во внешнем цикле: 1
Подсчет во внутреннем цикле: 0123456789
Подсчет во внешнем цикле: 2
Подсчет во внутреннем цикле: 0123456789
Циклы завершены
```

Оператор continue

- вывести четные числа от 0 до 100.

```
for(int i = 0; i <= 100; i++) {
 if((i%2) != 0) continue; // перейти к следующему шагу итерации
 Console.WriteLine (i);
}
```

Перечисления

- Перечисление представляет собой множество именованных целочисленных констант.
- Общая форма объявления перечисления:

```
enum имя {список_перечисления};
```

где имя — это имя типа перечисления,

список_перечисления — список идентификаторов, разделяемый запятыми.

- Указание базового типа перечисления:

```
enum Apple : byte { Jonathan, GoldenDel, RedDel, Winesap, Cortland,
Mcintosh };
```

- Инициализация перечисления :

```
enum Apple { Jonathan, GoldenDel, RedDel = 10, Winesap, Cortland,
Mcintosh };
```

Результат:

```
Jonathan 0    Winesap 11
GoldenDel 1   Cortland 12
RedDel 10     Mcintosh 13
```

Применение перечислений

```
enum Apple  {
 Jonathan, GoldenDel, RedDel, Winesap,
 Cortland, Mcintosh
};
static void Main() {
 string[] color = {
  "красный", "желтый", "красный", "красный",
  "красный", "красновато-зеленый"
 };
 Apple i; // объявить переменную перечислимого типа
 // Использовать переменную i для циклического
 // обращения к членам перечисления.
 for (i=Apple.Jonathan; i<=Apple.Mcintosh; i++)
   Console.WriteLine(i+" имеет значение "+(int)i);
 Console.WriteLine();
 // Использовать перечисление для индексирования массива.
 for (i = Apple.Jonathan; i <= Apple.Mcintosh; i++)
  Console.WriteLine("Цвет сорта "+i+" - "+color[(int)i]);
}
```

Результат выполнения:

```
Jonathan имеет значение 0
GoldenDel имеет значение 1
RedDel имеет значение 2
Winsap имеет значение 3
Cortland имеет значение 4
Mcintosh имеет значение 5
Цвет сорта Jonathan - красный
Цвет сорта GoldenDel - желтый
Цвет сорта RedDel - красный
Цвет сорта Winsap - красный
Цвет сорта Cortland - красный
Цвет сорта Mcintosh - красновато-зеленый
```

Применение перечислений

Сымитировать управление лентой конвейера:

```
// Перечислить команды конвейера.
enum Action { Start, Stop, Forward, Reverse };
static void Conveyor(Action com) {
 switch (com) {
  case Action.Start:
    Console.WriteLine("Запустить конвейер.");
    break;
  case Action.Stop:
    Console.WriteLine("Остановить конвейер.");
    break;
  case Action.Forward:
    Console.WriteLine("Переместить конвейер вперед.");
  break;
  case Action.Reverse:
    Console.WriteLine("Переместить конвейер назад.");
  break;
 }
}
static void Main() {
   Conveyor(Action.Start);
   Conveyor(Action.Forward);
   Conveyor(Action.Reverse);
   Conveyor(Action.Stop);
}
```

Результат выполнения:

```
Запустить конвейер.
Переместить конвейер вперед.
Переместить конвейер назад.
Остановить конвейер.
```