Лабораторная работа № 20-21

«Программная реализация циклического алгоритма. Операторы передачи управления: goto, break, continue, return.»

Цель работы:
получение навыков составления и отладки программ, с использованием операторов передачи управления: goto, break, continue, return на языке C#.

Теория

Оператор goto

Имеющийся в C# оператор goto представляет собой оператор безусловного перехода. Когда в программе встречается оператор goto, ее выполнение переходит непосредственно к тому месту, на которое указывает этот оператор. Он уже давно "вышел из употребления" в программировании, поскольку способствует созданию "макаронного"

кода. Хотя в некоторых случаях он оказывается удобным и дает определенные преимущества, если используется благоразумно. Главный недостаток оператора goto с точки зрения программирования заключается в том, что он вносит в программу беспорядок и делает ее практически неудобочитаемой. Но иногда применение оператора goto может, скорее, прояснить, чем запутать ход выполнения программы.

Для выполнения оператора goto требуется метка — действительный в C# идентификатор с двоеточием. Метка должна находиться в том же методе, где и оператор goto, а также в пределах той же самой области действия.

Пример использования оператора goto:

```
// Обычный цикл for выводящий числа от 1 до 5

Console.WriteLine("Обычный цикл for:");
for (int i = 1; i <= 5; i++)
    Console.Write("\t{0}",i);

// Реализуем то же самое с помощью оператора goto
Console.WriteLine("\n\nA теперь используем goto:");

int j = 1;
link1:
Console.Write("\t{0}",j);
j++;
if (j <= 5) goto link1;
```

```
Console.ReadLine();
```

Репутация оператора goto такова, что в большинстве случаев его применение категорически осуждается. Вообще говоря, он, конечно, не вписывается в рамки хорошей практики объектно-ориентированного программирования.

Оператор break

С помощью оператора break можно специально организовать немедленный выход из цикла в обход любого кода, оставшегося в теле цикла, а также минуя проверку условия цикла. Когда в теле цикла встречается оператор break, цикл завершается, а выполнение программы возобновляется с оператора, следующего после этого цикла. Оператор break можно применять в любом цикле, предусмотренном в C#.

// В данном цикле выведутся числа от 1 до 5 вместо 100

```
for (int i = 1; i < 100; i++)
    if (i <= 5)
        Console.WriteLine(i);
    else break;
Console.ReadLine();
```

Обратите внимание если оператор break применяется в целом ряде вложенных циклов, то он прерывает выполнение только самого внутреннего цикла.

В отношении оператора break необходимо также иметь в виду следующее. Во-первых, в теле цикла может присутствовать несколько операторов break, но применять их следует очень аккуратно, поскольку чрезмерное количество операторов break обычно приводит к нарушению нормальной структуры кода. И во-вторых, оператор break, выполняющий выход из оператора switch, оказывает воздействие только на этот оператор, но не на объемлющие его циклы.

Оператор continue

С помощью оператора continue можно организовать преждевременное завершение шага итерации цикла в обход обычной структуры управления циклом. Оператор continue осуществляет принудительный переход к следующему шагу цикла, пропуская любой код, оставшийся невыполненным. Таким образом, оператор continue служит своего рода дополнением оператора break.

В циклах while и do-while оператор continue вызывает передачу управления непосредственно условному выражению, после чего продолжается процесс выполнения цикла. А в цикле for сначала вычисляется итерационное выражение, затем условное выражение, после чего цикл продолжается:

```
// Выводим числа кратные 5
```

```
for (byte i = 1; i <= 100; i++)
{
    if (i % 5 != 0) continue;
    Console.Write("\t{0}", i);
}
Console.ReadLine();}}
```

Результат:

Оператор continue редко находит удачное применение, в частности, потому, что в C# предоставляется богатый набор операторов цикла, удовлетворяющих большую часть прикладных потребностей. Но в тех особых случаях, когда требуется преждевременное прерывание шага итерации цикла, оператор continue предоставляет структурированный способ осуществления такого прерывания.

Оператор return

Оператор return организует возврат из метода. Его можно также использовать для возврата значения. Имеются две формы оператора return: одна — для методов типа void, т.е. тех методов, которые не возвращают значения, а другая — для методов, возвращающих конкретные значения.

Для немедленного завершения метода типа void достаточно воспользоваться следующей формой оператора return:

```
return;
```

Когда выполняется этот оператор, управление возвращается вызывающей части программы, а оставшийся в методе код пропускается.

Для возврата значения из метода в вызывающую часть программы служит следующая форма оператора return:

```
return значение;
```

Давайте рассмотрим применение оператора return на конкретном примере:

```
static void Main(string[] args)
{
    int result = Sum(230);
    Console.WriteLine("Сумма четных чисел от 1 до 230 равна: " + result);
    Console.ReadLine();
}
// Метод, возвращающий сумму всех четных чисел
// от 1 до s
static int Sum(int s)
{
    int mySum = 0;
    for (int i = 1; i <= s; i++)
```

```
        if (i % 2 == 0)
            mySum += i;
    return mySum;
}
```

Ход работы:

4. Задана точка с координатами (x, y). Определить, на какой оси или в каком квадранте она находится.

```
Random rnd = new Random();
int x = rnd.Next(-10, 10);
int y = rnd.Next(-10, 10);
string line = "======Ответ=======\nТочка лежит на";
Console.WriteLine($"====Координаты====\n({x};{y})");
if (x == 0) Console.WriteLine($"{line} координате Y");
if (y == 0) Console.WriteLine($"{line} координате X");
if (x > 0 && y > 0) Console.WriteLine($"{line} I квадранте");
if (x < 0 && y > 0) Console.WriteLine($"{line} II квадранте");
if (x < 0 && y < 0) Console.WriteLine($"{line} III квадранте");
if (x > 0 && y < 0) Console.WriteLine($"{line} IV квадранте");
Console.ReadKey();
```

выполнение:

```
====Координаты====
(4;0)
======Ответ=======
Точка лежит на координате X
```

Контрольные вопросы:

1. Для чего и каким образом используются в операторах цикла операторы передачи управления break, continue, return, goto?

- С помощью оператора break можно специально организовать немедленный выход из цикла в обход любого кода, оставшегося в теле цикла, а также минуя проверку условия цикла.

```
for (int i = 1; i < 100; i++)
    if (i <= 5)
        Console.WriteLine(i);
    else break;
Console.ReadLine();
```

- С помощью оператора continue можно организовать преждевременное завершение шага итерации цикла в обход обычной структуры управления циклом.

```
// Выводим числа кратные 5

for (byte i = 1; i <= 100; i++)
{
    if (i % 5 != 0) continue;
    Console.Write("\t{0}", i);
}
Console.ReadLine();}}
```

- Когда в программе встречается оператор goto, ее выполнение переходит непосредственно к тому месту, на которое указывает этот оператор.

```


// Обычный цикл for выводящий числа от 1 до 5

Console.WriteLine("Обычный цикл for:");
for (int i = 1; i <= 5; i++)
    Console.Write("\t{0}",i);

// Реализуем то же самое с помощью оператора goto
Console.WriteLine("\n\nA теперь используем goto:");

int j = 1;
link1:
Console.Write("\t{0}",j);
j++;
if (j <= 5) goto link1;
Console.ReadLine();


```

- Оператор return организует возврат из метода. Его можно также использовать для возврата значения.

```
static void Main(string[] args)
{
    int result = Sum(230);
    Console.WriteLine("Сумма четных чисел от 1 до 230 равна: " + result);
    Console.ReadLine();
}
// Метод, возращающий сумму всех четных чисел
// от 1 до s
```

```
static int Sum(int s)
{
    int mySum = 0;
    for (int i = 1; i <= s; i++)
        if (i % 2 == 0)
            mySum += i;
    return mySum;
}
```

2. Что такое метка?
    - метка — действительный в C# идентификатор с двоеточием.
3. Какая форма оператора служит для возврата значения из метода в вызывающую часть программы?
    - `return`