Карагандинский технический университет имени Абылкаса Сагинова
Кафедра «Кибербезопасность и искусственный интеллект»

СЛАЙД-ЛЕКЦИЯ

Тема: «Инструкции управления»

Дисциплина: Программирование на C#

Специальность: ВТиПО

Автор: Ст.преп. Сницарь Лилия Ринатовна

ПЛАН ЛЕКЦИИ

1. Инструкция if.
2. Инструкция switch.
3. Цикл for.
4. Цикл while.
5. Цикл do-while.
6. Цикл foreach.
7. Инструкция break.
8. Инструкция continue.
9. Инструкция goto.

1. Инструкция if

Для организации условного ветвления язык C# унаследовал от C и C++ конструкцию **if...else**.

Ее синтаксис должен быть интуитивно понятен для любого, кто программировал на процедурных языках.

Полный формат ее записи такой:

```
if(условие) инструкция;
else инструкция;
```

Здесь под элементом ***инструкция*** понимается одна инструкция языка C#. Часть `else` необязательна.

1. Инструкция if

Вместо элемента *инструкция* может быть использован блок инструкций. В этом случае формат записи *if-инструкции* принимает такой вид:

```
if (условие)
{
    последовательность инструкций;
else
    последовательность инструкций;
}
```

Условное выражение, управляющее выполнением *if-инструкции*, должно иметь тип bool.

1. Инструкция if

Рассмотрим простую программу, в которой используется if-else-инструкция для определения того, является число положительным или отрицательным.

```
for (int i = -5; i < 6; i++)
{
    Console.Write("Тестирование " + i + ": ") ;
    if (i < 0)
        Console.WriteLine("Число отрицательно");
    else
        Console.WriteLine("Число положительно");
}
```

```
Тестирование -5: Число отрицательно
Тестирование -4: Число отрицательно
Тестирование -3: Число отрицательно
Тестирование -2: Число отрицательно
Тестирование -1: Число отрицательно
Тестирование 0: Число положительно
Тестирование 1: Число положительно
Тестирование 2: Число положительно
Тестирование 3: Число положительно
Тестирование 4: Число положительно
Тестирование 5: Число положительно
```

1. Инструкция if

Вложенные if-инструкции образуются в том случае, если в качестве элемента *инструкция* используется другая if-инструкция.

Важно помнить, что else-инструкция всегда относится к ближайшей if-инструкции, которая находится внутри того же программного блока, но еще не связана ни с какой другой else-инструкцией.

1. Инструкция if

Пример:

```
if (i == 10)
{
    if (j < 20) a = b;
    if(k > 100) c = d;
    else a = c; // Эта else-инструкция
                 //относится к if(k > 100).
}
else a = d; // Эта else-инструкция
           //относится к if(i == 10).
```

1. Инструкция if

```
for (int i = -5; i < 6; i++)
{
    Console.Write("Тестирование " + i + ": ") ;
    if (i < 0)
        Console.WriteLine("Число отрицательно");
    else if (i==0)
        Console.WriteLine("Число без знака");
    else
        Console.WriteLine("Число положительно");
}
```

```
Тестирование -5: Число отрицательно
Тестирование -4: Число отрицательно
Тестирование -3: Число отрицательно
Тестирование -2: Число отрицательно
Тестирование -1: Число отрицательно
Тестирование 0: Число без знака
Тестирование 1: Число положительно
Тестирование 2: Число положительно
Тестирование 3: Число положительно
Тестирование 4: Число положительно
Тестирование 5: Число положительно
```

1. Инструкция if

Очень распространенной в программировании конструкцией, в основе которой лежит вложенная if-инструкция, является «лестница» if-else-if.

Ее можно представить в следующем виде:

```
if (условие)
    инструкция;
else if (условие)
    инструкция;
else if (условие)
    инструкция;
…
else
    инструкция;
```

2. Инструкция switch

Второй инструкцией выбора является **switch**.

Инструкция switch обеспечивает многонаправленное ветвление. Она позволяет делать выбор одной из множества альтернатив.

Хотя многонаправленное тестирование можно реализовать с помощью последовательности вложенных if-инструкций, для многих ситуаций инструкция switch оказывается более эффективным решением.

Она работает следующим образом. Значение выражения последовательно сравнивается с константами из заданного списка.

При обнаружении совпадения для одного из условий сравнения выполняется последовательность инструкций, связанная с этим условием.

2. Инструкция switch

Общий формат записи инструкции switch такой:

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

2. Инструкция switch

Элемент *выражение* инструкции switch должен иметь целочисленный тип (например, char, byte, short или int) или тип string.

Выражения, имеющие тип ***с плавающей точкой***, ***не разрешены***.

Очень часто в качестве управляющего switch-выражения используется просто переменная; case-константы должны быть литералами, тип которых совместим с типом заданного выражения.

При этом никакие две case-константы в одной switch-инструкции не могут иметь идентичных значений.

2. Инструкция switch

```
for (int 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;
    }
```

```
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.
```

2. Инструкция switch

В C# считается ошибкой, если последовательность инструкций, относящаяся к одной case-ветви, переходит в последовательность инструкций, связанную со следующей.

Здесь должно действовать правило запрета на передачу управления вниз, на «провал», как говорят программисты. Поэтому case-последовательности чаще всего оканчиваются инструкцией break.

Инструкция **break**, завершающая последовательность case-инструкций, приводит к выходу из всей конструкции switch и передаче управления к следующей инструкции, находящейся вне конструкции switch.

Последовательность инструкций default-ветви также не должна «проваливаться» и обычно завершается инструкцией break.

2. Инструкция switch

```
for(int 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;
      }
```

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

3. Цикл for

Циклы являются управляющими конструкциями, позволяя в зависимости от определенных условий выполнять некоторое действие множество раз.

В C# имеются следующие виды циклов:

- `for`
- `foreach`
- `while`
- `do...while`

3. Цикл for

Цикл for в C# предоставляет механизм итерации, в котором определенное условие проверяется перед выполнением каждой итерации.

Синтаксис:

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

Если цикл for предназначен для повторного выполнения программного блока, то его общий формат выглядит так:

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

3. Цикл for

Элемент *инициализация* обычно представляет собой инструкцию присваивания, которая устанавливает *управляющую переменную цикла* равной начальному значению.

Эта переменная действует в качестве счетчика, который управляет работой цикла.

Элемент *условие* представляет собой выражение типа bool, в котором тестируется значение управляющей переменной цикла. Результат этого тестирования определяет, выполнится цикл for еще раз или нет.

Элемент *итерация* – это выражение, которое определяет, как изменяется значение управляющей переменной цикла после каждой итерации.

3. Цикл for

Обратите внимание на то, что все эти элементы цикла for должны отделяться точкой с запятой.

Цикл for будет выполняться до тех пор, пока вычисление элемента *условие* дает *истинный результат*.

Как только условие станет ложным, выполнение программы продолжится с инструкции, следующей за циклом for.

Управляющая переменная цикла for может изменяться как с положительным, так и с отрицательным приращением, причем величина этого приращения также может быть любой.

3. Цикл for

Пример:

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

3. Цикл for

Важно понимать, что условное выражение всегда тестируется в начале выполнения цикла for.

Это значит, что если первая же проверка условия даст значение ЛОЖЬ, код тела цикла не выполнится ни разу.

Пример:

```
for(count=10; count < 5; count++)
  x += count; // Эта инструкция не будет выполнена вовсе
```

Этот цикл никогда не выполнится, поскольку уже при входе в него значение его управляющей переменной count больше пяти.

Это делает условное выражение (count < 5) ложным с самого начала. Поэтому даже одна итерация этого цикла не будет выполнена.

3. Цикл for

Цикл for особенно полезен в тех случаях, когда известно количество его повторений.

Например, следующая программа использует два цикла for для отыскания простых чисел в диапазоне от 2 до 20.

Если число не простое, программа отобразит его самый большой множитель.

3. Цикл for

```
int num, i, value;
bool isprime;
// Внешний цикл проверяет числа от 2 до 19.
for (num = 2; num < 20; num++)
{
    isprime = true; // Предполагаем, что число простое.
    value = 0; // Начальное значение множителя.
    // Узнаем, делится ли num на i без остатка.
    for (i = 2; i <= num / 2; i++)
    {
        if ((num % i) == 0)
        {
            // Если num делится на i без остатка,
            // значит num — число не простое,
            isprime = false;
            value = i; //Сохраняем текущий множитель.
        }
    }
    // Проверяем, является ли num простым числом.
    if (isprime)
        Console.WriteLine(num + " -- простое число.");
    else
        Console.WriteLine("Максимальный множитель числа " + num + " равен " + value);
}
```

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

3. Цикл for

Для управления циклом for можно использовать две или больше переменных.

В этом случае инструкции инициализации и итерации для каждой из этих переменных отделяются запятыми.

Пример:

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

```
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
```

3. Цикл for

Приведем пример практического использования двух управляющих переменных в цикле for.

```
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 множитель: " + largest);
Console.WriteLine("Наименьший множитель: " + smallest);
```

```
Наибольший множитель: 50
Наименьший множитель: 2
```

4. Цикл while

Подобно for, **while** также является циклом с предварительной проверкой.

Синтаксис его аналогичен, но циклы while включают только одно выражение:

```
while (условие) инструкция;
```

Здесь под элементом *инструкция* понимается либо одиночная инструкция, либо блок инструкций.

Работой цикла управляет элемент *условие*, который представляет собой любое допустимое выражение типа bool.

Элемент инструкция выполняется до тех пор, пока условное выражение возвращает значение ИСТИНА. Как только это условие становится ложным, управление передается инструкции, которая следует за этим циклом.

4. Цикл while

Пример, в котором цикл while используется для вычисления порядка заданного целого числа.

```
int num;
int mag;
num = 435679;
mag = 0;
Console.WriteLine("Число: " + num);
while (num > 0)
{
    mag++;
    num = num / 10;
}
Console.WriteLine("Порядок: " + mag);
```

```
Число: 435679
Порядок: 6
```

5. Цикл do-while

В отличие от циклов for и while, в которых условие проверяется при входе, цикл **do-while** проверяет условие при выходе из цикла.

Это значит, что цикл do-while всегда выполняется хотя бы один раз.

Его общий формат имеет такой вид:

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

5. Цикл do-while

В следующей программе цикл do-while используется для отображения в обратном порядке цифр, составляющих заданное целое число.

```
int num;
int nextdigit;
num = 198;
Console.WriteLine("Число: " + num);
Console.Write("Число с обратным порядком цифр: ");
do
{
    nextdigit = num % 10;
    Console.Write(nextdigit);
    num = num / 10;
} while (num > 0);
Console.WriteLine();
```

```
Число: 198
Число с обратным порядком цифр: 891
```

6. Цикл foreach

Цикл **foreach** предназначен для опроса элементов *коллекции*.

Коллекция – это группа объектов. В C# определено несколько типов коллекций, среди которых можно выделить массив.

7. Инструкция break

С помощью инструкции **`break`** можно организовать немедленный выход из цикла, опустив выполнение кода, оставшегося в его теле, и проверку условного выражения.

При обнаружении внутри цикла инструкции break цикл завершается, а управление передается инструкции, следующей после цикла.

7. Инструкция break

Пример:

```
for (int i = -10; i <= 10; i++)
{
    if (i > 0) break; // Завершение цикла при i > 0.
     Console.Write(i + " ");
}
Console.WriteLine("Готово!");
```

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

8. Инструкция continue

Помимо средства «досрочного» выхода из цикла, существует средство «досрочного» выхода из текущей его итерации.

Этим средством является инструкция **`continue`**.

Она принудительно выполняет переход к следующей итерации, опуская выполнение оставшегося кода в текущей.

Инструкцию continue можно расценивать как дополнение к более «радикальной» инструкции break.

8. Инструкция continue

В этой программе используется инструкция continue для «ускоренного» поиска четных чисел в диапазоне от 0 до 100.

```
// Выводим четные числа между 0 и 100.
for(int i = 0; i <= 100; i++)
{
    if((i%2) != 0) continue; // Переход на следующую
                             //итерацию.
Console.WriteLine(i);
}
```

9. Инструкция goto

Инструкция **goto** – это C#-инструкция безусловного перехода. При ее выполнении управление программой передается инструкции, указанной с помощью метки.

Инструкция goto требует наличие в программе метки.

Метка – это действительный в C# идентификатор, за которым поставлено двоеточие.

Метка должна находиться в одном методе с инструкцией goto, которая ссылается на эту метку.

Например, с помощью goto можно организовать следующий цикл на 100 итераций:

9. Инструкция goto

```
for(int i=1; i < 5; i++) {
switch(i) {
case 1:
Console.WriteLine("В ветви case 1");
        goto case 3;
        case 2:
Console.WriteLine("В ветви case 2");
        goto case 1;
        case 3:
Console.WriteLine("В ветви case 3");
        goto default;
        default:
Console.WriteLine("В ветви default");
        break;
Console.WriteLine
// goto case 1; // Ошибка! Нельзя впрыгнуть
// в инструкцию switch.
```

```
В ветви case 1
В ветви case 3
В ветви default

В ветви case 2
В ветви case 1
В ветви case 3
В ветви default

В ветви case 3
В ветви default

В ветви default
```

Контрольные вопросы

1. Назовите три категории управляющих инструкций.
2. Полный формат ее записи if-else.
3. Что делает break?
4. Особенность цикла do-while?
5. Для чего используется continue?

Список литературы

1. Вагнер Б. C# Эффективное программирование М.: ЛОРИ, 2013. - 320 с.

2. Ишкова Э. А. Самоучитель C#. Начало программирования М.: Наука и техника, 2013. - 496 с.

3. Подбельский В. В. Язык C#. Базовый курс М.: Финансы и статистика, Инфра-М, 2011. - 384 с.

4. Троелсен Э. Язык программирования C# 2010 и платформа .NET4, М.: Москва: Огни, 2016. - 238 с.

ЛЕКЦИЯ ОКОНЧЕНА,
СПАСИБО ЗА ВНИМАНИЕ!