Тема 6. Операторы цикла. Цикл с параметром. Циклы с предусловием и с постусловием. for (do..while, while)

Учебные вопросы:

1. Введение. Типы циклов.
2. Цикл с параметром for
3. Циклы с предусловием while.
4. Циклы с постусловием do..while
5. Заключение.

1. Введение. Типы циклов..

Цикл — это конструкция в программировании, позволяющая многократно выполнять определенный блок кода до тех пор, пока выполняется определенное условие.

Зачем нужны циклы:

Циклы помогают автоматизировать повторяющиеся задачи, экономя время и упрощая код.

Примеры задач, которые можно решить с помощью циклов:

- Перебор всех элементов массива.
- Повторение одной и той же операции над множеством данных.
- Создание многократно повторяющихся выводов или действий.

Основные компоненты цикла

Инициализация:

- Подготовка начальных условий для работы цикла (например, установка начального значения счетчика).

•**Условие продолжения**:

- Логическое выражение, которое проверяется перед каждой итерацией. Если условие истинно, выполняется следующая итерация; если ложно — цикл завершается.

•**Тело цикла**:

- Код, который выполняется на каждой итерации.

•**Изменение переменной**:

- Обновление значений переменных, которые участвуют в условии, чтобы приблизить момент завершения цикла (например, увеличение счетчика).

Типы циклов в C#

- **Цикл с параметром (for):** Используется, когда известно точное количество итераций.
- **Цикл с предусловием (while):** Выполняется до тех пор, пока условие истинно; может не выполниться ни разу, если условие с самого начала ложно.
- **Цикл с постусловием (do..while):** Гарантированно выполнит тело цикла хотя бы один раз, даже если условие ложно с самого начала.

2. Цикл с параметром for

Цикл for используется для выполнения набора инструкций определенное количество раз. Он особенно удобен, когда заранее известно количество итераций.

```
for (инициализация; условие; итерация)
{
    // тело цикла
}
```

Инициализация: установка начального значения счетчика.

Условие: проверка условия выполнения цикла. Если условие истинно, выполняется очередная итерация, иначе цикл завершается.

Итерация: изменение значения счетчика после каждой итерации

Пример: Вывод чисел от 1 до 5.

```
for (int i = 1; i <= 5; i++)
{
    Console.WriteLine(i);
}
```

Инициализация: **int i = 1** — создается переменная **i**, которой присваивается значение 1.

Условие: **i <= 5** — цикл продолжается, пока значение **i** меньше или равно 5.

Итерация: **i++** — после каждой итерации значение **i** увеличивается на 1.

Другой пример:

```
for (int i = -10; i < 10; i+=2)
{
    Console.WriteLine(i);
}
```

Что будет выведено в консоль?

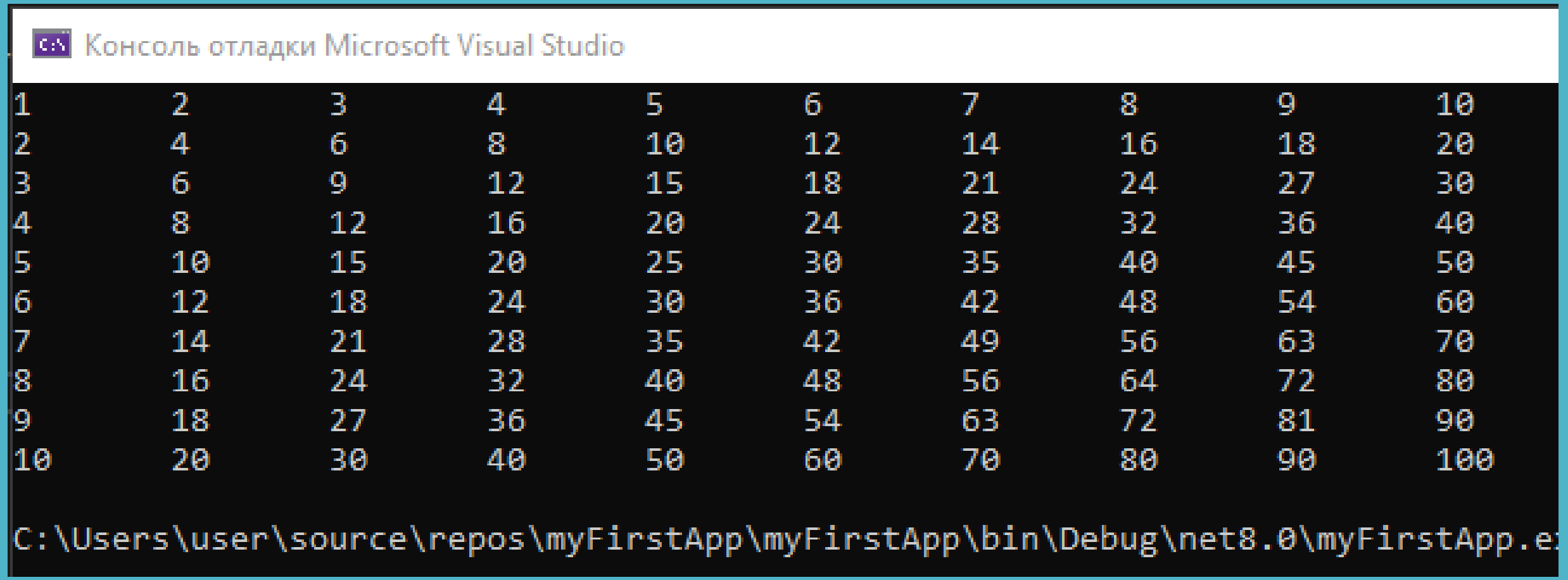
Область применения цикла for

- Когда известно точное количество итераций.
- Для работы с массивами или списками.
- Для создания таблиц, выполнения вычислений с фиксированным количеством шагов.

```
int sum = 0;
for (int i = 1; i <= 10; i++)
{
    sum += i;
}
Console.WriteLine("Sum = " + sum);
```

Вложенные циклы — это циклы, которые находятся внутри других циклов. Они используются, когда необходимо выполнить несколько операций для каждого значения внешнего цикла. Например вывод таблицы умножения:

```
for (int i = 1; i <= 10; i++)
{
    for (int j = 1; j <= 10; j++)
    {
        Console.Write((i * j) + "\t");
    }
    Console.WriteLine();
}
```


Консоль отладки Microsoft Visual Studio
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
C:\Users\user\source\repos\myFirstApp\myFirstApp\bin\Debug\net8.0\myFirstApp.e

Частые ошибки при использовании цикла for

Бесконечный цикл: Если условие в цикле for никогда не становится ложным, цикл будет выполняться бесконечно.

```
for (int i = 0; i >= 0; i++)
{
    Console.WriteLine(i);
}
```

Некоторые **ключевые слова**, которые используются для управления выполнением циклов и других управляющих конструкций.

- **break** используется для **немедленного завершения** выполнения цикла или switch-case блока.

Когда **break** встречается внутри цикла, выполнение цикла прерывается, и программа продолжает выполнение с первой строки кода после цикла.

```
for (int i = 1; i <= 10; i++)
{
    if (i == 5)
    {
        break; // Прерывает цикл, когда i равно 5
    }
    Console.WriteLine(i);
}
```

- **continue** используется для перехода к **следующей итерации** цикла, пропуская оставшийся код в текущей итерации.

Когда **continue** встречается внутри цикла, оставшаяся часть кода в теле цикла пропускается, и выполнение продолжается с проверки условия для следующей итерации.

```
for (int i = 1; i <= 10; i++)
{
    if (i % 2 == 0)
    {
        continue; // Пропускаем четные числа
    }
    Console.WriteLine(i);
}
```

goto — это оператор, который позволяет немедленно передать управление **на указанную метку** в коде.

Однако, его использование **не рекомендуется**, так как оно может сделать код сложным для понимания и сопровождения.

```
for (int i = 1; i <= 10; i++)
{
    if (i == 5)
    {
        goto End; // Переход на метку End
    }
    Console.WriteLine(i);
}

End:
Console.WriteLine("Цикл завершен.");
```

3. Цикл с предусловием (while)

Цикл **while** — это цикл, который выполняет блок кода до тех пор, **пока условие истинно**. Проверка условия происходит перед каждой итерацией, что позволяет циклу **не выполниться ни разу**, если условие с самого начала ложно.

```
while (условие)
{
    // тело цикла
}
```

Условие: Логическое выражение, которое проверяется перед каждой итерацией.

Тело цикла: Код, который выполняется, если условие истинно.

```
int i = 1;
while (i <= 5)
{
    Console.WriteLine(i);
    i++;
}
```

Инициализация: int i = 1 — создается переменная i, которой присваивается значение 1.

Условие: i <= 5 — цикл продолжается, пока значение i меньше или равно 5.

Тело цикла: Console.WriteLine(i); i++; — выводим текущее значение i и увеличиваем его на 1.

Параметр цикла можно изменять на любое значение, например на 2, для получения только четных значений:

```
int i = -10;
while (i < 10)
{
    Console.WriteLine(i);
    i += 2;
}
```

Циклы могут быть вложенными. Пример вложенного цикла для вывода таблицы умножения:

```
int i = 1;
while (i <= 10)
{
    int j = 1;
    while (j <= 10)
    {
        Console.Write((i * j) + "\t");
        j++;
    }
    Console.WriteLine();
    i++;
}
```

Область применения цикла while:

- Когда неизвестно точное количество итераций заранее.
- Когда выполнение цикла зависит от состояния переменных, которое может измениться в процессе выполнения программы.

Ожидание ввода от пользователя

```
string input = "";
while (input != "exit")
{
    Console.WriteLine("Enter something (type 'exit' to quit):");
    input = Console.ReadLine();
}
```

Бесконечный цикл while

Бесконечный цикл — это цикл, который никогда не завершится самостоятельно. Такой цикл обычно создается случайно, когда условие никогда не становится ложным.

```
while (true)
{
    Console.WriteLine("Этот цикл будет выполняться вечно!.");
}
```

Обычно такие циклы прерываются с помощью ключевого слова **break:**

```
while (true)
{
    Console.WriteLine("Введите 'exit' для выхода:");
    string input = Console.ReadLine();
    if (input == "exit")
    {
        break; // Прерываем цикл
    }
}
```

Частые ошибки при использовании **while**

- Пропуск изменения состояния: Если забыть изменить переменные, участвующие в условии, цикл может стать бесконечным.

```
int i = 1;
while (i <= 5)
{
    Console.WriteLine(i);
    // i++; // Если не увеличить i, цикл будет бесконечным
}
```

- Неправильная инициализация переменной:

Если переменная инициализируется с неправильным значением, цикл может не выполниться или выполниться неверно.

```
int i = 10;
while (i <= 5) // Условие сразу ложно, цикл не выполнится
{
    Console.WriteLine(i);
    i++;
}
```

4. Циклы с постусловием do..while

Цикл **do..while** выполняет блок кода как минимум один раз и продолжает его выполнение, пока условие остается истинным.

```
do
{
    // Код, который выполняется хотя бы один раз и затем повторяется
} while (условие);
```

Пример. Вывод чисел от 0 до 5:

```
int count = 0;

do
{
    Console.WriteLine("Count is: " + count);
    count++;
} while (count <= 5);
```

5. Заключение.

Каждый из циклов for, while и do..while используется для повторения блока кода, но их использование и поведение различаются.

Выбор цикла:

- Используйте **for**, если количество итераций известно заранее.
- Используйте **while**, если условие выхода из цикла известно до его выполнения.
- Используйте **do..while**, если необходимо выполнить блок кода хотя бы один раз, прежде чем проверять условие.

Контрольные вопросы:

- Что такое цикл в программировании и для чего он используется?
- Опишите основные компоненты цикла. Какие из них обязательны для всех типов циклов?
- Какой цикл вы бы выбрали, если заранее известно количество итераций, и почему?
- Чем отличаются циклы for, while и do..while? Приведите примеры использования каждого.
- Какие ошибки могут возникнуть при использовании цикла for? Как их избежать?
- Объясните, как работает цикл while и в каких ситуациях его стоит применять.
- Что такое бесконечный цикл, и как его можно предотвратить?
- Каково назначение операторов break и continue в циклах?
- Опишите, как работает цикл do..while и в чем его ключевое отличие от цикла while.
- Когда стоит использовать вложенные циклы? Приведите пример задачи, решаемой с их помощью.

Домашнее задание:

1. Повторить материал лекции.
2. Учебное пособие, с 41..48
3. Решить задачи: Учебное пособие, с. 48.

Материалы лекций:

https://github.com/ShViktor72/Education

Обратная связь:

colledge20education23@gmail.com

Список литературы:

1. Жаксыбаева Н.Н. Основы объектно-ориентированного программирования: язык C#. Часть 1./ Учебное пособие предназначено для учащихся технического и профессионального образования, Алматы, 2010,

2. https://learn.microsoft.com/ru-ru/dotnet/csharp/

3. https://www.youtube.com/watch?v=9ynxfp-SfEM&list=PLNmV1bEI9uO-r58OI34Q6-_KKPddrrAU6&index=8&ab_channel=DevCodeпрограммированиеC%23%7CАнтонСтупеньков