Практическая работа №3

МДК 0104: Системное программирование на .NET

Тема: Циклы в C#

Цель работы Сформировать практические навыки применения циклических конструкций (`for`, `while`, `do-while`, `foreach`) для организации повторяющихся действий и обработки коллекций данных в программах на C#.

Задачи

1. Закрепить синтаксис и семантику циклических конструкций.
2. Научиться выбирать оптимальный тип цикла для решения конкретной задачи.
3. Развить навыки работы с массивами и коллекциями в циклах.
4. Освоить использование операторов управления циклом (`break`, `continue`).
5. Приобрести опыт работы с системой контроля версий Git и платформой Gogs.

Краткая справочная информация

Основные циклические конструкции в C#

1. **Цикл `for`**: Используется, когда известно количество итераций.

```
for (инициализация; условие; итератор)
{
    // Тело цикла
}
// Пример: Вывод чисел от 0 до 9
for (int i = 0; i < 10; i++)
{
    Console.WriteLine(i);
}
```

2. **Цикл `while`**: Выполняется, пока условие истинно. Проверка условия происходит *перед* каждой итерацией.

```
while (условие)
{
    // Тело цикла
}
// Пример: Чтение ввода, пока не будет введено "exit"
string input = "";
while (input != "exit")
{
    input = Console.ReadLine();
}
```

3. **Цикл `do-while`**: Выполняется, пока условие истинно. Проверка условия происходит *после* каждой итерации. Тело цикла выполнится как минимум один раз.

```
do
{
    // Тело цикла
} while (условие);
// Пример: Запрос пароля до тех пор, пока он не будет верным
string password;
do
{
    Console.Write("Введите пароль: ");
    password = Console.ReadLine();
} while (password != "secret");
```

4. **Цикл `foreach`**: Используется для итерации по элементам коллекции (массивы, списки и т.д.). Не требует управления индексом.

```
foreach (тип элемент in коллекция)
{
    // Тело цикла
}
// Пример: Вывод всех элементов массива
int[] numbers = { 1, 2, 3, 4, 5 };
foreach (int number in numbers)
{
    Console.WriteLine(number);
}
```

5. **Операторы управления циклом**:
    - `break` - немедленно завершает выполнение цикла.
    - `continue` - немедленно переходит к следующей итерации цикла, пропуская оставшийся код в теле текущей итерации.

Ход работы

1. Получить вариант задания у преподавателя.
2. На локальной машине создать новое консольное приложение .NET с именем `LoopPractice`.
3. Реализовать логику согласно своему варианту. Код должен быть хорошо читаемым, содержать комментарии и использовать соответствующие типы циклов.
4. Протестировать программу на различных входных данных, включая граничные случаи.
5. Сделать не менее 3-х скриншотов работы программы с разными входными данными.
6. Создать новый приватный репозиторий на предоставленном Git-сервере **Gogs** с названием `loop-practice`.
7. Настроить файл `.gitignore` для проекта .NET (можно сгенерировать на gitignore.io).

8. Добавить в репозиторий исходный код проекта (файл `Program.cs`), папку `images` со скриншотами и оформить файл `README.md`.
9. Предоставить ссылку на репозиторий преподавателю для проверки.

Требования к репозиторию на Gogs

Структура репозитория должна выглядеть следующим образом:

```
loop-practice/
├── LoopPractice/
│   ├── Program.cs
│   └── LoopPractice.csproj
├── images/
│   ├── test_1.png
│   ├── test_2.png
│   └── test_3.png
├── .gitignore
└── README.md
```

Содержание `README.md`:

- Название работы и номер варианта.
- Цель и задачи.
- Полное текстовое описание задания из варианта.
- Инструкция по сборке и запуску (`dotnet run`).
- Краткое описание логики решения.
- Описание примеров, приведенных на скриншотах (какие данные вводились и какой результат получен).

Варианты заданий

Вариант 1: Таблица умножения Напишите программу, которая выводит на экран таблицу умножения (от 1 до 10) для числа, введенного пользователем. Используйте цикл `for`.

Вариант 2: Сумма и среднее арифметическое Запросите у пользователя 10 чисел. Используя цикл `for` или `while`, вычислите и выведите их сумму и среднее арифметическое.

Вариант 3: Факториал числа Напишите программу, которая вычисляет факториал числа, введенного пользователем. Используйте цикл `for`. Учтите, что факториал отрицательного числа не определен.

Вариант 4: Обратный отсчет Реализуйте программу обратного отсчета, которая запрашивает у пользователя число и затем выводит все числа от этого числа до 0, а в конце выводит "Старт!". Используйте цикл `while`.

Вариант 5: Поиск в массиве Создайте массив из 15 случайных целых чисел в диапазоне от 0 до 100. Попросите пользователя ввести число. Используя цикл `foreach`, проверьте, содержится ли это число в массиве, и выведите соответствующее сообщение.

Вариант 6: Проверка пароля Напишите программу, которая дает пользователю 3 попытки для ввода правильного пароля. Используйте цикл `for` для отслеживания количества попыток. При успешном вводе цикл должен прерваться оператором `break`.

Вариант 7: Рисование фигур Напишите программу, которая с помощью вложенных циклов `for` рисует в консоли прямоугольник из символов `*`. Высоту и ширину прямоугольника задает пользователь.

Вариант 8: Последовательность Фибоначчи Запросите у пользователя количество элементов последовательности Фибоначчи для вывода. Используя цикл `for`, сгенерируйте и выведите последовательность.

Вариант 9: Калькулятор с повторением Модифицируйте простой калькулятор (из предыдущих работ). После вычисления результата программа должна спрашивать пользователя, хочет ли он выполнить еще один расчет (`да`/`нет`). Используйте цикл `do-while` для организации повторения.

Вариант 10: Поиск простых чисел Напишите программу, которая находит все простые числа в диапазоне от 2 до числа N, введенного пользователем. Для проверки каждого числа на простоту используйте вложенный цикл `for`. Для оптимизации используйте `continue` при нахождении первого делителя.

Вариант 11: Анализ строки Попросите пользователя ввести строку. Используя цикл `foreach`, пройдитесь по всем символам строки и подсчитайте количество цифр и букв в ней.

Вариант 12: Угадай число (с подсказками) Программа загадывает случайное число от 1 до 100. Пользователь пытается его угадать. Цикл `while` должен продолжаться, пока число не угадано. После каждой попытки давайте подсказку "Больше" или "Меньше". В конце выведите количество сделанных попыток.

Вариант 13: Вывод четных/нечетных чисел Запросите у пользователя два числа: начало и конец диапазона. Предложите выбрать, что выводить: четные или нечетные числа. Используя цикл `for` и оператор `continue`, выведите только выбранные числа из заданного диапазона.

Вариант 14: Работа со списком товаров Создайте список (`List<string>`) для названий товаров. Используя цикл `while`, организуйте меню: 1 - Добавить товар, 2 - Вывести все товары, 3 - Очистить список, 4 - Выход. Для выхода используйте `break`.

Вариант 15: Сортировка пузырьком Реализуйте алгоритм сортировки пузырьком для массива из 10 целых чисел, введенных пользователем. Используйте вложенные циклы `for`. Выводите промежуточное состояние массива после каждого прохода внешнего цикла.

Контрольные вопросы

1. В чем ключевое различие между циклами `while` и `do-while`? Приведите пример задачи, для которой предпочтительнее использовать `do-while`.
2. Объясните, для каких задач предпочтительнее использовать цикл `for`, а для каких — `foreach`.
3. Как оператор `continue` влияет на ход выполнения цикла? В чем его отличие от `break`?
4. Почему при обходе коллекции с помощью цикла `foreach` нельзя изменять саму коллекцию (добавлять/удалять элементы)?

5. Что такое "бесконечный цикл" и как его можно создать? Какие операторы позволяют выйти из бесконечного цикла?