Ильяхова Алиса, Б9122-02.03.01сцт

Лабораторная работа №7. Типы потоков. Управление потоками

Вариант 9

1. Цель и задачи работы

Цель:

Освоить принципы создания и управления потоками.

Задачи:

- Научиться создавать фоновые и приоритетные потоки;
- Научиться работать с пулом потоков;
- Научиться решать практические задачи с использованием различных типов потоков.

2. Реализация индивидуального задания

2.1. Условие варианта 9

Согласно таблице на стр. 57:

- **Задача**: Метод находит логическое значение, указывающее, существует ли заданное число в массиве целых случайных чисел.
- **Тип элемента коллекции**: Класс, описывающий массив случайных чисел (размер выбирается случайно, искомое число задаётся в конструкторе).

2.2. Класс DataItem

Реализован класс для хранения данных:

- int[] Array — массив случайных чисел;
- int Target — искомое число.

2.3. Класс ThreadWorker

Метод ProcessData:

- Принимает object item (требование ParameterizedThreadStart);
- Выполняет поиск числа в массиве;
- Имитирует долгую операцию через Thread.Sleep(2000);

- Выводит информацию о потоке: ID, фоновый/обычный, приоритет.

2.4. Управление типами потоков

В методе Main созданы три потока:

1. **Обычный поток** — IsBackground = false (по умолчанию), Priority = Normal.
2. **Фоновый поток** — IsBackground = true.
3. **Поток высокого приоритета** — Priority = ThreadPriority.Highest.

2.5. Синхронизация завершения

- Выполнено ожидание завершения **обычного** и **высокоприоритетного** потоков через Join().
- **Фоновый поток** завершится автоматически при завершении основного потока.

3. Ответы на контрольные вопросы

1. **Какие типы потоков вы знаете? Чем они отличаются?**
    a. **Обычные (foreground)**: программа не завершается, пока они работают.
    b. **Фоновые (background)**: завершаются автоматически, когда завершается основной поток.

Отличие — в поведении при завершении программы.

2. **Для чего применяются приоритеты потоков? Как задать приоритет потока?**

Приоритет определяет, насколько часто поток получает доступ к процессору.

Задаётся через свойство Thread.Priority (значения: Lowest, BelowNormal, Normal, AboveNormal, Highest).

3. **Что такое пул потоков? Какой метод запускает поток в пуле?**

Пул потоков — это набор заранее созданных потоков, которые можно использовать повторно.

Запуск задачи в пуле: ThreadPool.QueueUserWorkItem(callback, state).

4. Экранные формы и листинг программы

4.1. Консольный вывод программы

```
=== Лабораторная работа №7. Вариант 9 ===
Типы потоков. Управление потоками

[Поток 11]
[Поток 9]
Фоновый: False
[Поток 10]
Фоновый: True
Приоритет: Normal
Искомое число: 65, найдено: False
Приоритет: Highest
Искомое число: 6, найдено: True
Приоритет: Normal
Искомое число: 47, найдено: False
Массив: [26, 37, 21, 97, 16, 72, 12, 96]
Массив: [44, 49, 59, 60, 93, 1, 36]
Массив: [16, 95, 22, 6, 70, 42, 47]
--------------------------------------------------

Основной поток завершён.
```

Порядок может отличаться. Фоновый поток может не успеть вывести результат, если основной завершится слишком быстро (в нашем случае — успевает благодаря Join для других потоков).

4.2. Полный листинг программы с комментариями

```
using System;
using System.Threading;

// Лабораторная работа №7. Типы потоков. Управление потоками
// Вариант 9

class DataItem
{
    public int[] Array { get; private set; }
    public int Target { get; private set; }

    public DataItem(int size, int target)
    {
        var rand = new Random();
```

```
        Array = new int[size];
        for (int i = 0; i < size; i++)
            Array[i] = rand.Next(1, 100);
        Target = target;
    }
}

class ThreadWorker
{
    public void ProcessData(object item)
    {
        var data = (DataItem)item;
        bool found = false;

        // Имитация долгой операции
        Thread.Sleep(2000);

        foreach (int val in data.Array)
        {
            if (val == data.Target)
            {
                found = true;
                break;
            }
        }

        Console.WriteLine($"[Поток {Thread.CurrentThread.ManagedThreadId}]");
        Console.WriteLine($"Фоновый: {Thread.CurrentThread.IsBackground}");
        Console.WriteLine($"Приоритет: {Thread.CurrentThread.Priority}");
        Console.WriteLine($"Искомое число: {data.Target}, найдено: {found}");
        Console.WriteLine($"Массив: [{string.Join(", ", data.Array)}]");
        Console.WriteLine(new string('-', 50));
    }
}

class Program
{
    static void Main()
    {
        Console.WriteLine("=== Лабораторная работа №7. Вариант 9 ===");
        Console.WriteLine("Типы потоков. Управление потоками\n");

        Random rand = new Random();

        // Создаём 3 потока: обычный, фоновый, высокого приоритета
        var items = new[]
        {
            new DataItem(rand.Next(5, 10), rand.Next(1, 100)),
            new DataItem(rand.Next(5, 10), rand.Next(1, 100)),
            new DataItem(rand.Next(5, 10), rand.Next(1, 100))
        };
```

```
        // Поток 1: обычный (не фоновый, нормальный приоритет)
        Thread thread1 = new Thread(new ThreadWorker().ProcessData);
        thread1.Name = "Обычный поток";

        // Поток 2: фоновый
        Thread thread2 = new Thread(new ThreadWorker().ProcessData);
        thread2.IsBackground = true;
        thread2.Name = "Фоновый поток";

        // Поток 3: высокий приоритет
        Thread thread3 = new Thread(new ThreadWorker().ProcessData);
        thread3.Priority = ThreadPriority.Highest;
        thread3.Name = "Поток высокого приоритета";

        // Запуск потоков
        thread1.Start(items[0]);
        thread2.Start(items[1]);
        thread3.Start(items[2]);

        // Ожидание завершения НЕфоновых потоков
        thread1.Join();
        thread3.Join();

        Console.WriteLine("\n Основной поток завершён.");
        // Фоновый поток завершится автоматически при завершении main
    }
}
```

5. Вывод

В ходе выполнения лабораторной работы №7 были:

- Реализованы три типа потоков: обычный, фоновый, высокоприоритетный;
- Продемонстрировано управление свойствами IsBackground и Priority;
- Показано различие в поведении фоновых и обычных потоков при завершении программы;
- Решена задача поиска числа в массиве в каждом потоке.