МИНИСТЕРСТВО ОБРАЗОВАНИЯ ОРЕНБУРГСКОЙ ОБЛАСТИ
Государственное автономное профессиональное образовательное учреждение
«ОРЕНБУРГСКИЙ КОЛЛЕДЖ ЭКОНОМИКИ И ИНФОРМАТИКИ»
(ГАПОУ ОКЭИ)

Практическая работа

ОКЭИ 09.02.07 7026 07 У

Тема: «*Эмуляция критического состояния*»

Выполнил: *Гадиляев Ислам Ильгамович*

Оренбург 2026

Содержание

1 Цель работы 3
2 Скриншот структуры готового проекта 3
3 Листинг программы 3
4 Результаты работы программы 4

1 Цель работы

Цель работы: эмуляция непрерывного сохранения данных разными потоками в общей базе данных.

Задание: разработать приложение, реализующее одновременный доступ нескольких потоков к разделяемому ресурсу

2 Скриншот структуры готового проекта

На рисунке 1 представлен скриншот структуры готового проекта.

Рисунок 1 – Структура проекта

3 Листинг программы

Листинг файла Program.cs представлен ниже.

```
namespace pz_04;

class Program
{
    public static Database dbService = new Database();

    static void WorkerThreadMethodOne()
    {
        Console.WriteLine("Поток 1 начал");
        dbService.SaveData("Поток 1");
        Console.WriteLine("Поток 1 кончил \n");
    }

    static void WorkerThreadMethodTwo()
    {
        Console.WriteLine("Поток 2 начал");
        dbService.SaveData("Поток 2");
        Console.WriteLine("Поток 2 кончил \n");
    }
```

```
    static void Main(string[] args)
    {
        var threadOne = new Thread(new ThreadStart(WorkerThreadMethodOne));
        var threadTwo = new Thread(new ThreadStart(WorkerThreadMethodTwo));

        threadOne.Start();
        threadTwo.Start();
    }
}
```

Листинг файла Database.cs представлен ниже.

```
namespace pz_04;

public class Database
{
    public void SaveData(string text)
    {
        lock (this)
        {
            Console.WriteLine("Сохранение...");

            for (var i = 0; i <= 5; i++)
            {
                Console.WriteLine($"[{text}] запись {i}");
                Thread.Sleep(3000);
            }

            Console.WriteLine("Сохранение завершено");
        }
    }
}
```

4 Результаты работы программы

Результат выполнения программы показан на рисунке 2.

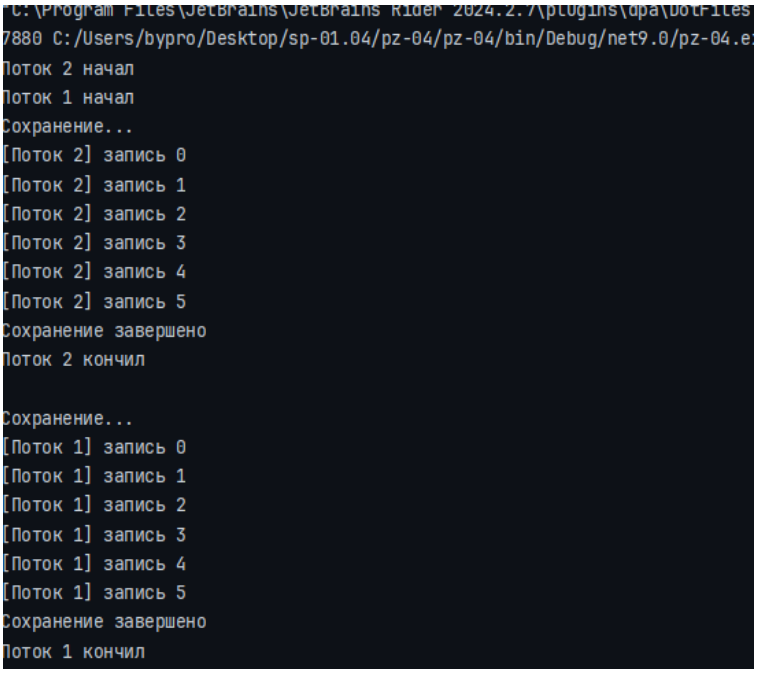

Рисунок 2 – Результат работы программы

Таким образом, с помощью lock можно ограничить доступ к ресурсу, сделав его строго последовательным, тем самым избегая race condition, и сохраняя ресурс в правильном состоянии. Если же убрать lock, как показано на рисунке 3, то доступ будет хаотичным, что, конечно, быстрей, но может порождать race condition, и данные могут оказаться неправильными в базе данных. Таким образом, использование критических секций необходимо для обеспечения корректной синхронизации потоков.

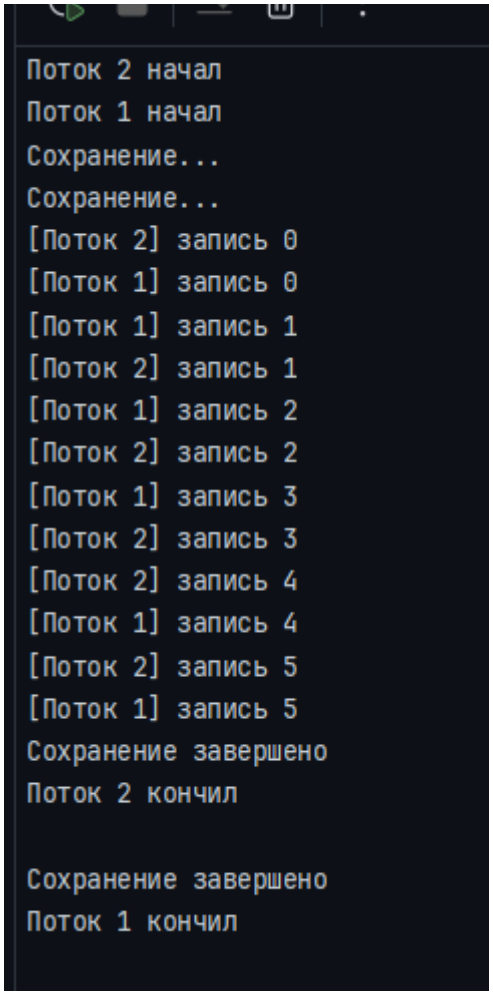

Рисунок 3 – Результат работы программы без критической секции