# Цель и задачи лабораторной работы

**Цель:** освоить принципы создания и управления потоками.

**Задачи:**

1. Научиться создавать фоновые и приоритетные потоки;
2. Научиться работать с пулом потоков;
3. Научиться решать практические задачи с использованием пула потоков.

# Реализация индивидуального задания

Согласно варианту задания, требуется реализовать пользовательский тип делегата со следующей сигнатурой: ***Action<Func<int>, List<float>>***

Это означает, что делегат должен принимать два аргумента:

1. *Func<int>*
   1. Метод, принимающий *int* с непустым возвратом
2. *List<float>*
   1. Список значений с float

И ничего не возвращать.

Был объявлен делегат *CustomDelegate*. С использованием пула потоков. В алгоритме предусмотреть задержку алгоритма с использованием метода Thread.Sleep()

## Листинг программного кода

***namespace*** lab7

{

***class*** Program

    {

*/// <summary>*

*/// Класс Текста, Символа, который надо найти и Результата*

*/// </summary>*

**public** ***class*** StringSearchItem(**string** text, **char** symbol)

        {

**public** **string** Text { ***get***; ***set***; } **=** text;

**public** **char** SymbolToFind { ***get***; ***set***; } **=** symbol;

**public** **bool** Result { ***get***; ***set***; }

*// Метод, который будет выполняться в потоке пула*

**public** **void** SearchSymbol(**object** stateInfo)

            {

*// Имитация задержки для наглядности*

                Thread.Sleep(1000);

*// Логика поиска символа*

                Result **=** Text.Contains(SymbolToFind);

                Console.WriteLine($"[Процесс {Environment.CurrentManagedThreadId}]: Поиск символа '{SymbolToFind}' в строке \"{Text}\". Результат: {Result}");

            }

        }

**public** ***class*** CollectionManager

        {

**private** **readonly** List<StringSearchItem> \_items **=** [];

**public** **void** AddItem(StringSearchItem item)

            {

                \_items.Add(item);

            }

**public** **void** ProcessAll()

            {

                Console.WriteLine("Запуск обработки коллекции...");

*foreach* (***var*** item *in* \_items)

                {

*// Ставим в очередь метод каждого элемента на выполнение в пуле потоков*

                    ThreadPool.QueueUserWorkItem(item.SearchSymbol);

                }

*// Даем время на завершение фоновых операций (для консольного приложения)*

*// Thread.Sleep(2000);*

                Console.WriteLine("Обработка коллекции завершена.");

            }

        }

**public** **static** **void** Main(**string**[] args)

        {

***var*** manager **=** **new** CollectionManager();

*// Добавляем элементы в коллекцию*

            manager.AddItem(**new** StringSearchItem("Hello World", 'o'));

            manager.AddItem(**new** StringSearchItem("Parallel Processing", 'z'));

            manager.AddItem(**new** StringSearchItem("C# ThreadPool", '#'));

*// Запускаем обработку*

            manager.ProcessAll();

            Console.WriteLine("Завершение работы основного потока.");

            Console.ReadKey();

        }

    }

}

## Описание кода

**Ключевые механизмы**

**1. Класс StringSearchItem**

* **Инкапсулирует задачу** поиска символа в строке
* **Свойства:** текст, искомый символ, результат поиска
* **Метод SearchSymbol:** бизнес-логика выполнения в потоке

**2. Использование ThreadPool**

* **Автоматическое управление** потоками системой
* **Переиспользование потоков** вместо создания новых
* **Оптимизация ресурсов** - пул сам решает, когда создавать/уничтожать потоки

**3. CollectionManager**

* **Управление коллекцией** задач поиска
* **Массовая обработка** через пул потоков
* **Асинхронное выполнение** без блокировки основного потока

**Особенности выполнения:**

* **Потоки из пула** являются фоновыми (IsBackground = true)
* **Основной поток может завершиться** до окончания работы пула

## Результат работы программы

Запуск обработки коллекции...

Обработка коллекции завершена.

Завершение работы основного потока.

[Процесс 4]: Поиск символа 'o' в строке "Hello World". Результат: True

[Процесс 7]: Поиск символа '#' в строке "C# ThreadPool". Результат: True

[Процесс 6]: Поиск символа 'z' в строке "Parallel Processing". Результат: False

# Контрольные вопросы

1. **Какие типы потоков вы знаете? Чем они отличаются?**
   1. **Foreground потоки:**
      1. Завершаются только после выполнения работы
      2. Приложение не закрывается, пока есть работающие foreground потоки
      3. Создаются через new Thread()
   2. **Background потоки:**
      1. Автоматически завершаются при закрытии приложения
      2. Не препятствуют завершению работы программы
      3. Потоки из ThreadPool являются background
   3. **Главное отличие:** Влияние на время жизни приложения.
2. **Для чего применяются приоритеты потоков? Как задать приоритет потока?**
   1. Назначение:
      1. Определяют долю процессорного времени для потока
      2. Влияют на порядок выполнения в условиях конкуренции за ресурсы
   2. Установка приоритета:
      1. Thread thread = new Thread(MyMethod);  
         thread.Priority = ThreadPriority.<Priority>;
      2. Доступные значения:
         1. Highest - наивысший приоритет
         2. AboveNormal - выше нормального
         3. Normal - нормальный (по умолчанию)
         4. BelowNormal - ниже нормального
         5. Lowest - низший
3. **Что такое пул потоков? Какой метод запускает поток в пуле?**
   1. Что это:
      1. Коллекция заранее созданных потоков для выполнения задач
      2. Оптимизирует создание/уничтожение потоков
      3. Автоматически управляет количеством потоков
   2. Метод запуска:
      1. ThreadPool.QueueUserWorkItem(MyMethod);