## Лабораторная работа №15 Работа с потоками

### Теоретические сведения

#### Многопоточная программа. Класс Thread

Многопоточная программа состоит из двух или больше частей, которые могут выполняться одновременно. Каждая часть такой программы называется потоком (thread), и каждый поток определяет собственный путь выполнения инструкций. При выполнении программы каждому потоку процессор выделяет определенный квант времени.

Различают два вида многозадачности:

* + с ориентацией на процессы,
  + с ориентацией на потоки.
* **Процесс** – это программа, т.е. многозадачность, ориентированная на процессы, — это средство, позволяющее компьютеру выполнять две или больше программ одновременно.
* **Поток**— это управляемая единица выполняемого кода (функция).

Все процессы имеют по крайней мере один поток (м.б. больше), т.е. одна программа может выполнять сразу две и более задач.

Многопоточная система С# встроена в класс Thread, который инкапсулирует поток управления. Классы, которые поддерживают многопоточное программирование, определены в пространстве имен System.Threading. Класс Thread является sealed-классом, т.е. он не может иметь наследников. В классе Thread определен ряд методов и свойств для управления потоками.

##### Состояния потока

Поток создается за счет вызова метода Start() и переходит в состояние Unstarted.

В состояние Running поток переходит после того, как планировщик потоков операционной системы выберет его для выполнения.

С помощью метода Thread.Sleep() поток можно перевести в состояние WaitSleepJoin и при этом указать, через какой промежуток времени поток должен возобновить работу.

Чтобы остановить поток, необходимо вызвать метод Thread.Abort(). При вызове этого метода в соответствующем потоке генерируется исключение типа ThreadAbortException.

##### Алгоритм создания потоков

1. Создать метод, который будет точкой входа для нового потока.
2. Создать новый делегат ParametrizedThreadStart/ThreadStart, передав конструктору адрес метода, определенного на предыдущем шаге.
3. Создать объект Thread, передав в качестве аргумента конструктора ParametrizedThreadStart/ThreadStart.
4. Вызвать метод Thread.Start(). Это запустит поток на методе, который указан делегатом, созданным на втором шаге, как только это будет возможно.

##### Пример создания потока

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

#### Синхронизация

Синхронизация – это процесс координации потоков, если:

* двум потокам требуется один ресурс;
* выполнение второго потока зависит от результатов первого потока.

В этом случае необходимо выполнять синхронизацию потоков, которая заключается в согласовании их скоростей путем приостановки потока до наступления какого-то события, а потом его активизации при наступлении этого события.

**Критическая секция** – это часть программы, результат выполнения которой может непредсказуемо меняться, если переменные этой части программы изменяются другими потоками, в то время, когда выполнение этой части еще не завершено. Критическая секция определяется по отношению к определенным критическим данным, при несогласованном изменении которых и может возникнуть нежелательный эффект.

**Блокировка** - управление доступом к некоторому блоку кода в объекте.

На то время, когда объект заблокирован одним потоком, никакой другой поток не может получить доступ к заблокированному блоку кода. Когда поток снимет блокировку, объект станет доступным для использования другим потоком.

Инструкция lock гарантирует, что указанный блок кода, защищенный блокировкой для данного объекта, может быть использован только потоком, который получает эту блокировку.

Все другие потоки остаются заблокированными до тех пор, пока блокировка не будет снята. Снята она будет лишь при выходе из этого блока.

Пример блокировки Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

#### Класс Task

Класс Task описывает отдельную задачу, которая запускается асинхронно в одном из потоков (обычно в Main). Задача – это отдельная продолжительная операция (абстракция асинхронной операции). Библиотека параллельных задач TPL (Task Parallel Library) располагается в пространстве имен System.Threading.Tasks.

##### Пример создания задачи с использованием Task

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Метод Parallel.Invoke в качестве параметра принимает массив объектов Action, то есть мы можем передать в данный метод набор методов, которые будут вызываться при его выполнении. Количество методов может быть различным, можем передать либо название метода, либо лямбда-выражение.

#### Пример использования Parallel.Invoke

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

### Постановка задачи

Реализовать многопоточное приложение. Для каждого из вариантов необходимо реализовать возможность задавать приоритет каждого из порожденных потоков. Предусмотреть разрешение конфликтов при обращении к одному ресурсу. Все приложения должны иметь графический интерфейс.

**Варианты:**

|  |  |
| --- | --- |
| **Вариант** |  |
| **1** | Поиск указанной строки в указанном файле. Обработка одной строки в порожденном потоке. Общий ресурс – файл. |
| **2** | Умножение матрицы на вектор. Обработка одной строки матрицы - в порожденном потоке. Общий ресурс – матрица. |
| **3** | Поиск всех простых чисел (простым называется число, которое является своим наибольшим делителем) в указанном интервале чисел, разделенном на несколько диапазонов и запись найденных чисел в файл. Обработка каждого диапазона производится в порожденном потоке. Классический алгоритм Евклида определения наибольшего общего делителя двух целых чисел (x, y) может применяться при следующих условиях:   * оба числа x и y неотрицательные; * оба числа x и y отличны от нуля.   На каждом шаге алгоритма выполняются сравнения:   * если x == y, то ответ найден; * если x < y, то y заменяется значением y - x; * если x > y, то x заменяется значением x - y.   Общий ресурс – файл. |
| **4** | Warcraft. Заданное количество юнитов добывают золото равными порциями из одной шахты, задерживаясь в пути на случайное время, до ее истощения. Работа каждого юнита реализуется в порожденном процессе потоке. Общий ресурс – шахта. |
| **5** | Винни-Пух и пчелы. Заданное количество пчел добывают мед равными порциями, задерживаясь в пути на случайное время. Винни-Пух потребляет мед порциями заданной величины за заданное время и столько же времени может прожить без питания. Работа каждой пчелы реализуется в порожденном потоке. Общий ресурс – Винни-Пух. |
| **6** | Шарики. Координаты заданного количества шариков изменяются на случайную величину по вертикали и горизонтали. При выпадении шарика за нижнюю границу допустимой области шарик исчезает. Изменение координат каждого шарика в отдельном потоке. Общий ресурс – область. |
| **7** | Противостояние нескольких команд. Каждая команда увеличивается на случайное количество бойцов и убивает случайное количество бойцов участника. Борьба каждой команды реализуется в отдельном потоке. Бойцы для каждой команды выбираются из общего пула бойцов. Общий ресурс – пул бойцов. |
| **8** | Контрольная сумма. Для нескольких файлов (разного размера) требуется вычислить контрольную сумму (сумму кодов всех символов файла). Обработка каждого файла выполняется в отдельном потоке.  Также нужно вычислить общую контрольную сумму для всех файлов. |
| **9** | Авиаразведка. Создается условная карта в виде матрицы, размерность которой определяет размер карты, содержащей произвольное количество единиц (целей) в произвольных ячейках. Из произвольной точки карты стартуют несколько разведчиков (потоков), курсы которых выбираются так, чтобы покрыть максимальную площадь карты. Каждый разведчик фиксирует цели, чьи координаты совпадают с его координатами и по достижении границ карты сообщает количество обнаруженных целей. Общий ресурс – пул целей. |
| **10** | Бег с препятствиями. Создается условная карта трассы в виде матрицы, ширина которой соответствует количеству бегунов, а высота – фиксирована, содержащей произвольное количество единиц (препятствий) в произвольных ячейках. Стартующие бегуны (потоки) перемещаются по трассе и при встрече с препятствием задерживаются на фиксированное время. По достижении финиша бегуны сообщают свой номер. Общий ресурс – трасса с препятствиями. |
| **11** | Создать поток, который управляет перемещением кнопки/панели. Кнопка движется по оси Х. Создать 3 кнопки и устроить "тараканьи" бега среди них. |
| **12** | Параллельно отрисовать графики функций: y = sin(x), y = 4\*x2-2\*x – 22, y = ln(x2)/x3. |
| **13** | Однорукий бандит - 3 потока, генерирующие числа от 0 до 9. По нажатию кнопки потоки останавливаются и результат анализируется. При анализе использовать следующие комбинации (три одинаковых числа, два одинаковых числа, три единицы, три семерки, две единицы, имеется четверка) |
| **14** | Вычислить 10-ю степень двойки 1 - сложением, умножением и просто возведением в степень. |
| **15** | Создать игру, где будут 2-3 барана и волк. При совпадении координат волка с бараном баран исчезает. При совпадении координат баранов появляется новый баран. Все движутся хаотически. |
| **16** | Создать два потока. Первый ищет числа Фибоначчи (каждое последующее число равно сумме двух предыдущих чисел), второй простые числа. Результат работы каждого потока сохраняются в файл. После остановки потока – программа производит анализ файла, выводит их на экран, а также показывает количество найденных чисел Фибоначчи и простых чисел. |
| **17** | Создать два потока. Первый поток производит запись в файл случайных данных. Второй производит чтение данных из этого файла и вывод их на экран. Общий ресурс – файл. |
| **18** | Создать приложение выполняющее построение графика функции y = 23\*x2 – 33, с шагом x=0.01. Первый поток производит расчет данных функции и добавляет их в конец массива данных. Второй поток извлекает из массива данных значения и производит выводит функции на экран. Общий ресурс – массив. |
| **19** | Создать приложение выполняющее сортировку массива данных и визуального отображения процесса сортировки на экране. Первый поток производит сортировку по возрастанию, второй по убыванию. После каждого перемещения элемента результирующий производится графический вывод текущего состояния сортировки. Каждый поток работает с отдельным экземпляром массива данных. Состояние сортировки выводится в двух элементах. |
| **20** | Создать приложение выполняющее сортировку массива данных и визуального отображения процесса сортировки на экране. Первый массив производит сортировку по возрастанию, второй по убыванию, третий производит графический вывод массива на экран. Каждый поток работает с общим экземпляром массива данных. |
| **21** | Создать приложение выполняющее перемещение кнопки по форме. Первый поток производит перемещение кнопки согласно текущих значений приращения х, у. Второй поток производит расчет текущего значения приращения х,у. У пользователя должна быть возможность менять правило расчета х,у – выбрав из выпадающего списка определенный элемент. Выпадающий список содержит элементы – “не перемещаться”, "по прямой", "sin(x)", "cos(x)". |
| **22** | Создать приложение содержащих два порожденных потока. Первый поток производит добавление в элемент RichTextBox новой строки, второй поток удаление последней строки. При невозможности удаления строки поток должен приостанавливаться на время необходимое первому потоку для добавления новой строки. |
| **23** | Создать приложение, содержащее три порожденных потока. Первый поток формирует случайные числа и записывает их в файл. Второй поток выбирает из файла и выводит на экран числа кратные 2, второй поток выбирает из файла и выводит числа кратные 3 |
| **24** | Создать приложение, которое содержит два порожденных потока. Первый поток формирует случайные числа и записывает их в очередь. Второй поток удаляет элементы из очереди и выводит их на экран. Если очередь пустая, по второй поток должен приостановить свою работу на время, необходимое для добавление нового элемента. |
| **25** | Создать приложение, которое визуализирует алгоритмы бинарного и линейного поиска. Сортировка массива, бинарный и линейный поиск запускаются в разных потоках. |