Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Объектно-ориентированные технологии программирования и стандарты проектирования (ООТПиСП)

ОТЧЕТ

по лабораторной работе №1

Тема работы: Потоки

Выполнил

студент: гр. 951003 Антонов М.А.

Проверил: Деменковец Д.В.

Минск 2021

# Исходный код программы

**Задание 1:**

**Program.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace lab1\_queue

{

class Program

{

static void Main(string[] args)

{

int threadsNumber, taskNumber;

Console.WriteLine("Enter number of threads:");

threadsNumber = Convert.ToInt32(Console.ReadLine());

if (threadsNumber <= 0)

{

Console.WriteLine("Number of threads must be

more than 0. Enter number of threads:");

threadsNumber = Convert.ToInt32(Console.Read

Line());

}

Console.WriteLine("Enter number of tasks:");

taskNumber = Convert.ToInt32(Console.ReadLine());

if (taskNumber < 0)

{

Console.WriteLine("Number of threads must be positive or 0. Enter number of tasks:");

threadsNumber = Convert.ToInt32(Console.Read

Line());

}

TaskQueue taskQueue = new TaskQueue(threadsNumber);

for (int i = 0; i < taskNumber; i++)

{

taskQueue.EnqueuTask(MyTask);

}

Console.ReadLine();

taskQueue.FinishThreads();

void MyTask()

{

Console.WriteLine("Thread №{0} completed task",

Thread.CurrentThread.Name);

}

}

}

}

**TaskQueue.cs:**

using System;

using System.Collections.Generic;

using System.Collections.Concurrent;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Threading;

namespace lab1\_queue

{

class TaskQueue

{

public delegate void TaskDelegate();

private List<Thread> ThreadsPool = new List<Thread>();

ConcurrentQueue<TaskDelegate> TaskList= new Concur

rentQueue<TaskDelegate>();

private List<Thread> pool = new List<Thread>();

public TaskQueue (int threadsNumber)

{

for (int i = 0; i < threadsNumber; i++)

{

Thread thread = new Thread(TaskProc);

thread.Name = (i + 1).ToString();

thread.IsBackground = true;

ThreadsPool.Add(thread);

thread.Start();

}

}

public void FinishThreads()

{

foreach (Thread t in pool)

{

t.IsBackground = false;

}

}

private void TaskProc()

{

while (Thread.CurrentThread.IsBackground)

{

if (TaskList.TryDequeue(out TaskDelegate task)

&& task != null)

task();

}

}

public void EnqueuTask(TaskDelegate task)

{

TaskList.Enqueue(task);

}

}

}

**Задание 2:**

**Program.cs:**

using System;

using System.IO;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

namespace fileThreads

{

class Program

{

static void Main(string[] args)

{

string source, destination;

int threadsNumber;

string[] fileNames;

if (args.Length == 3)

{

source = args[0];

destination = args[1];

threadsNumber = Convert.ToInt32(args[2]);

}

else

{

source = @"D:\BSUIR\5sem\СПП\Copy\_source";

destination = @"D:\BSUIR\5sem\СПП\Copy\_destina

tion";

threadsNumber = 5;

}

TaskQueue task = new TaskQueue(threadsNumber, desti

nation);

fileNames = Directory.GetFiles(source);

foreach (var file in fileNames)

{

task.EnqueueTask(file);

}

while (task.TaskList.Count > 0) { }

task.FinishThreads();

while (task.CheckActiveThreads()) { }

Console.WriteLine($"\nNumber of copied files:

{TaskQueue.NumberOfCopiedFiles}");

Console.ReadLine();

}

}

}

**TaskQueue.cs:**

using System;

using System.Collections.Generic;

using System.Collections.Concurrent;

using System.Linq;

using System.Text;

using System.Threading;

using System.IO;

namespace fileThreads

{

class TaskQueue

{

public delegate void TaskDelegate();

private List<Thread> ThreadsPool = new List<Thread>();

private static List<bool> ActiveThreads = new

List<bool>();

public ConcurrentQueue<string> TaskList = new Concur

rentQueue<string>();

public string destination;

public static int NumberOfCopiedFiles = 0;

public TaskQueue(int threadsNumber, string dest)

{

for (int i = 0; i < threadsNumber; i++)

{

Thread thread = new Thread(TaskProc);

destination = dest;

thread.Name = i.ToString();

thread.IsBackground = true;

ActiveThreads.Add(false);

ThreadsPool.Add(thread);

thread.Start();

}

}

public void EnqueueTask(string task)

{

TaskList.Enqueue(task);

}

public void FinishThreads()

{

foreach (Thread t in ThreadsPool)

{

t.IsBackground = false;

}

}

private void TaskProc()

{

ActiveThreads[Convert.ToInt32(Thread.Cur

rentThread.Name)] = true;

while (Thread.CurrentThread.IsBackground)

{

if (TaskList.TryDequeue(out string fileName))

CopyFile(fileName, destination);

}

ActiveThreads[Convert.ToInt32(Thread.Cur

rentThread.Name)] = false;

}

private void CopyFile(string fileName, string dest)

{

Console.WriteLine($"COPY {fileName} TO {dest} WITH

THREAD №{Convert.ToInt32(Thread.CurrentThread.Name) + 1}");

FileInfo fileInfo = new FileInfo(fileName);

fileInfo.CopyTo(Path.Combine(dest, fileInfo.Name),

true);

NumberOfCopiedFiles++;

}

public bool CheckActiveThreads()

{

return ActiveThreads.Contains(true);

}

}

}