ГУАП

КАФЕДРА № 44

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, доцент |  |  |  | Н. В. Кучин |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| ПРОГРАММИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ПРОЦЕССОВ |
| по курсу: ОПЕРАЦИОННЫЕ СИСТЕМЫ |
|  |
|  |

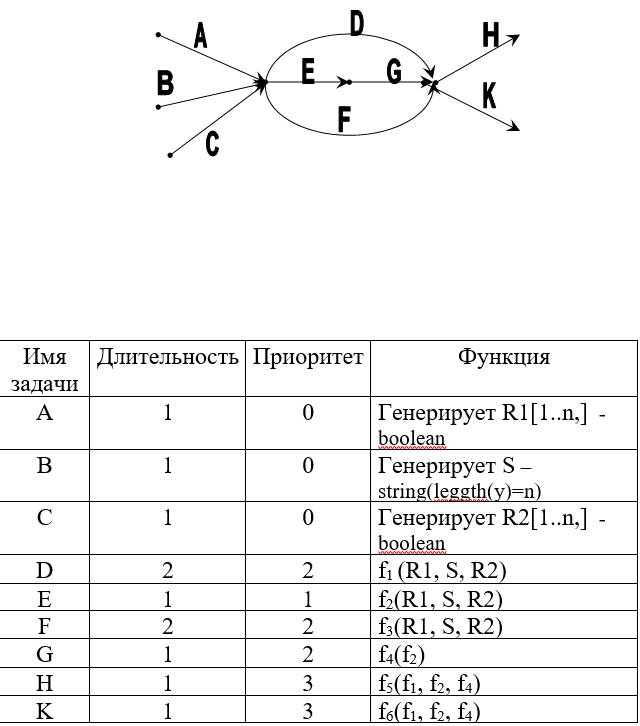
РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4941 |  |  |  | М.П. Леднёв |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

**Цель работы:** необходимо написать и отладить программу, которая реализует параллельное выполнение нескольких задач, каждая из которых решает некоторую заданную функцию.

**Вариант 17:**



**Код программы:**

Листинг 1. DataGridModel.cs

using System.ComponentModel;

namespace ThreadProgramming

{

public class DataGridModel : INotifyPropertyChanged // определяет структуру таблицы

{

string timestart;

string taskname;

string result;

string inittaskname;

string timeend;

public string TimeStart

{

get { return timestart; }

set { timestart = value;

OnPropertyChanged("TimeStart");

}

}

public string TaskName

{

get { return taskname; }

set

{

taskname = value;

OnPropertyChanged("TaskName");

}

}

public string Result

{

get { return result; }

set

{

result = value;

OnPropertyChanged("Result");

}

}

public string InitTaskName

{

get { return inittaskname; }

set

{

inittaskname= value;

OnPropertyChanged("InitTaskName");

}

}

public string TimeEnd

{

get { return timeend; }

set

{

timeend = value;

OnPropertyChanged("TimeEnd");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged(string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

Листинг 2. VMGrid.cs

using System.Collections.Generic;

using System.ComponentModel;

using System.Collections.ObjectModel;

namespace ThreadProgramming

{

class VMGrid: INotifyPropertyChanged

{

ObservableCollection<DataGridModel> data;

public ObservableCollection<DataGridModel> Data

{

get

{

return data;

}

set

{

data = value;

OnPropertyChanged("Data");

}

}

public VMGrid (List<DataGridModel> data)

{

Data = new ObservableCollection<DataGridModel>(data);

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged(string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

Листинг 3. Tworker.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading;

using System.ComponentModel;

using System.Windows.Controls;

using System.Windows;

namespace Thread17

{

//обертка для класса потока

public class Tworker

{

public delegate object func(object x);

public string name; //имя потока

DateTime StartTime;

public string initName; // имя потока, .который иницализировал

public bool flag = false; // флаг выполненности

public bool used = false; // проверялся ли флаг

public int delay; // задержка при выполнении функций

public func F1;

public object result;

public int index;

public ProgressBar bar; // элемент GUI

public DataGrid table;

public BackgroundWorker worker; // новый поток

public List<Tworker> ParallelTasks;

public List<Tworker> StartTasks; // потоки, которе надо будет стартануть

public List<DataGridModel> Data;

void DoTask(object sender, DoWorkEventArgs e) // выполнение вычислений. Выполняется в отдельном потоке

{

/\*принимаем параметр-объект

затем необходимо вызвать функцию, котоаря что-то делает. Она делает. Мы после неё эмулируем постепенной визуализации процесса

в конце мы должны проверить все потоки-соседи на получение резльтата. Результат получаем как объект-объект

если потоки-соседи выполнились, то от лица этого потока запускаем новые. Иначе завершаемся

\*/

StartTime = DateTime.Now; //начали работать

result = F1((e.Argument as object));

for (int i = 0; i <= 100; i++) //эмуляция прогресса

{

Thread.Sleep(delay);

(sender as BackgroundWorker).ReportProgress(i);

}

flag = true; // мы по факту закончили

}

void ProgressChanged(object sender, ProgressChangedEventArgs e) // обновление GUI в основном потоке

{

bar.Value = e.ProgressPercentage;

}

void Complete(object sender, RunWorkerCompletedEventArgs e) // выполняется после выполнения DoTask

{

bool allDone = true;

List<object> results = new List<object>(); // резльтаты

//добавляем данные в конечную табличку

Data.Add(new DataGridModel { TaskName = name, InitTaskName = initName, TimeStart = StartTime.ToLongTimeString() + ":" + StartTime.Millisecond, TimeEnd = DateTime.Now.ToLongTimeString() + ":" + DateTime.Now.Millisecond });

if (ParallelTasks != null)

{

ParallelTasks.Add(this);

var tworkers = ParallelTasks.OrderBy(x => x.index); //сортируем согласно заданными индексам

foreach (Tworker task in tworkers)

{

lock (task) //блокируем общие переменные

{

if (task.used) // если уже использовано, то значит следующий поток запущен

{

allDone = false;

break;

}

allDone = allDone && task.flag;

results.Add(task.result);

}

}

if (allDone) // если все выполнены, то отмечаем, что мы проверили

{

foreach (Tworker task in ParallelTasks)

{

task.used = true;

}

}

}

else

{

results.Add(result);

}

if (StartTasks != null && StartTasks.Count != 0)

{

if (!allDone)

{

results.Clear();

}

else

{

foreach (Tworker task in StartTasks)

{

lock (task)

{

task.Start(results.ToArray(), name);//запускаем потоки

}

}

}

}

else

{

if (ParallelTasks == null || allDone)

{

//последний таск, а значит выводим результаты

MessageBox.Show("Done");

DataWindow window = new DataWindow(Data);

window.ShowDialog();

}

}

}

public void halfInit(List<Tworker> startTasks, List<Tworker> parallelTasks)

{

worker = new BackgroundWorker();

ParallelTasks = parallelTasks;

StartTasks = startTasks;

}

public void Start(object x, string TaskName) //метод запуска

{

initName = TaskName;

worker.WorkerReportsProgress = true;

worker.DoWork += DoTask;

worker.ProgressChanged += ProgressChanged;

worker.RunWorkerCompleted += Complete;

worker.RunWorkerAsync(x);

} //запуск потока

}

}

Листинг 5. DataWindow.xaml

<Window x:Class="Thread17.DataWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Thread17"

mc:Ignorable="d"

Title="DataWindow" Height="450" Width="800">

<Grid>

<Border>

<Border.Background>

<LinearGradientBrush StartPoint="0.5, 0" EndPoint="0.5, 1">

<GradientStop Color="LightGreen" Offset="0.7"/>

<GradientStop Color="Yellow" Offset="1.0"/>

</LinearGradientBrush>

</Border.Background>

</Border>

<DataGrid Margin="10"

AutoGenerateColumns="True"

ItemsSource="{Binding Data}"

Background="Transparent"/>

</Grid>

</Window>

Листинг 6. MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

namespace Thread17

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

Tworker TaskA; // потоки

Tworker TaskB;

Tworker TaskC;

Tworker TaskD;

Tworker TaskE;

Tworker TaskF;

Tworker TaskG;

Tworker TaskH;

Tworker TaskK;

List<DataGridModel> Data = new List<DataGridModel>();

public MainWindow()

{

InitializeComponent();

}

private void btnStart(object sender, RoutedEventArgs e)

{

//инициализируем все потоки

TaskA = new Tworker { name = "A", delay = 4, F1 = f1, bar = Abar, Data = Data, index = 0 };

TaskB = new Tworker { name = "B", delay = 11, F1 = f2, bar = Bbar, Data = Data, index = 1 };

TaskC = new Tworker { name = "C", delay = 7, F1 = f1, bar = Cbar, Data = Data, index = 2 };

TaskD = new Tworker { name = "D", delay = 12, F1 = f4, bar = Dbar, Data = Data, index = 0 };

TaskE = new Tworker { name = "E", delay = 5, F1 = f5, bar = Ebar, Data = Data, index = 0 };

TaskF = new Tworker { name = "F", delay = 6, F1 = f6, bar = Fbar, Data = Data, index = 2 };

TaskG = new Tworker { name = "G", delay = 11, F1 = f7, bar = Gbar, Data = Data, index = 1 };

TaskH = new Tworker { name = "H", delay = 10, F1 = f8, bar = Hbar, Data = Data, index = 0 };

TaskK = new Tworker { name = "K", delay = 8, F1 = f9, bar = Kbar, Data = Data, index = 0 };

TaskA.halfInit(new List<Tworker> { TaskD, TaskE, TaskF }, new List<Tworker> { TaskB, TaskC });

TaskB.halfInit(new List<Tworker> { TaskD, TaskE, TaskF }, new List<Tworker> { TaskA, TaskC });

TaskC.halfInit(new List<Tworker> { TaskD, TaskE, TaskF }, new List<Tworker> { TaskB, TaskA });

TaskD.halfInit(new List<Tworker> { TaskK, TaskH }, new List<Tworker> { TaskG, TaskF });

TaskE.halfInit(new List<Tworker> { TaskG }, null);

TaskF.halfInit(new List<Tworker> { TaskK, TaskH }, new List<Tworker> { TaskG, TaskD });

TaskG.halfInit(new List<Tworker> { TaskK, TaskH }, new List<Tworker> { TaskF, TaskD });

TaskH.halfInit(null, new List<Tworker> { TaskK });

TaskK.halfInit(null, new List<Tworker> { TaskH });

List<object> start = new List<object>();

start.Add(Convert.ToInt32(N.Text) as object);

TaskA.Start(start as object, "MAIN");

TaskB.Start(start as object, "MAIN");

TaskC.Start(start as object, "MAIN");

}

object f1(object x) // таск А C генерация массива

{

try

{

List<object> param = x as List<object>;

int N = Convert.ToInt32(param[0]);

Random rnd = new Random();

bool[] mas = new bool[N];

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

{

mas[i] = rnd.Next(1000)%2 ==0;

}

return (mas);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return null;

}

object f2(object x) // таск B

{

try

{

List<object> param = x as List<object>;

int N = Convert.ToInt32(param[0]);

Random rnd = new Random();

StringBuilder stringBuilder = new StringBuilder();

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

{

stringBuilder.Append(Convert.ToChar(rnd.Next(95)+32));

}

return (stringBuilder.ToString());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message + " B");

}

return null;

}

object f3(object x) // таск C

{

try

{

List<object> param = x as List<object>;

int N = Convert.ToInt32(param[0]);

Random rnd = new Random();

int[] mas = new int[N];

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

{

mas[i] = rnd.Next(1000);

}

return (mas as object);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return null;

}

object f4(object x) // таск D

{

try

{

var list = (x as object[]);

bool[] boolMas1 = list[0] as bool[];

string str = list[1] as string;

StringBuilder stringBuilder = new StringBuilder();

bool[] boolMas2 = list[2] as bool[];

int N = boolMas1.Length;

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

{

if (boolMas1[i])

{

if (boolMas2[i])

{

stringBuilder.Append(str[i] + 10);

}

else

{

stringBuilder.Append(str[i] + 5);

}

}

else

{

if (boolMas2[i])

{

stringBuilder.Append(str[i] + 2);

}

else

{

stringBuilder.Append(str[i]);

}

}

}

return (stringBuilder.ToString());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return null;

}

object f5(object x) // таск E

{

try

{

var list = (x as object[]);

bool[] boolMas1 = list[0] as bool[];

string str = list[1] as string;

StringBuilder stringBuilder = new StringBuilder();

bool[] boolMas2 = list[2] as bool[];

int N = boolMas1.Length;

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

{

if (boolMas1[i])

{

if (boolMas2[i])

{

stringBuilder.Append(str[i] + 10);

}

else

{

stringBuilder.Append(str[i] + 5);

}

}

else

{

if (boolMas2[i])

{

stringBuilder.Append(str[i] + 2);

}

else

{

stringBuilder.Append(str[i]);

}

}

}

return (stringBuilder.ToString());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return null;

}

object f6(object x) // таск F

{

try

{

var list = (x as object[]);

bool[] boolMas1 = list[0] as bool[];

string str = list[1] as string;

StringBuilder stringBuilder = new StringBuilder();

bool[] boolMas2 = list[2] as bool[];

int N = boolMas1.Length;

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

{

if (boolMas1[i])

{

if (boolMas2[i])

{

stringBuilder.Append(str[i] + 10);

}

else

{

stringBuilder.Append(str[i] + 5);

}

}

else

{

if (boolMas2[i])

{

stringBuilder.Append(str[i] + 2);

}

else

{

stringBuilder.Append(str[i]);

}

}

}

return (stringBuilder.ToString());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return null;

}

object f7(object x) // таск G

{

try

{

var list = (x as object[]);

string str = list[0] as string;

return (str.Reverse().ToString());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return null;

}

object f8(object x) // таск H

{

try

{

var list = (x as object[]);

string str = list[0] as string;

string str1 = list[1] as string;

StringBuilder stringBuilder = new StringBuilder();

string str2 = list[2] as string;

int N = str.Length;

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

{

stringBuilder.Append(str[i]);

stringBuilder.Append(str1[i]);

stringBuilder.Append(str2[i]);

}

return (stringBuilder.ToString());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return null;

}

object f9(object x) // таск K

{

try

{

var list = (x as object[]);

string str = list[0] as string;

string str1 = list[1] as string;

StringBuilder stringBuilder = new StringBuilder();

string str2 = list[2] as string;

stringBuilder.Append(str);

stringBuilder.Append(str1);

stringBuilder.Append(str2);

return (stringBuilder.ToString());

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

return null;

}

}

}

Листинг 7. MainWindow.xaml

<Window x:Class="ThreadProgramming.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:ThreadProgramming"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800" ResizeMode="NoResize">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="500"/>

</Grid.ColumnDefinitions>

<StackPanel>

<Image Source="scheme.png" Margin="5"/>

<Button Content="Запуск" Height="25" Width="70" Click="btnDoMagic"/>

</StackPanel>

<StackPanel Grid.Column="1"

x:Name="BarPanel"

Orientation="Vertical">

<StackPanel Margin="5" Orientation="Horizontal">

<Label >Task B</Label>

<ProgressBar x:Name="Bbar" Width="400"/>

</StackPanel>

<StackPanel Margin="5" Orientation="Horizontal">

<Label >Task C</Label>

<ProgressBar x:Name="Cbar" Width="400"/>

</StackPanel>

<StackPanel Margin="5" Orientation="Horizontal">

<Label >Task D</Label>

<ProgressBar x:Name="Dbar" Width="400"/>

</StackPanel>

<StackPanel Margin="5" Orientation="Horizontal">

<Label >Task E</Label>

<ProgressBar x:Name="Ebar" Width="400"/>

</StackPanel>

<StackPanel Margin="5" Orientation="Horizontal">

<Label >Task F</Label>

<ProgressBar x:Name="Fbar" Width="400"/>

</StackPanel>

<StackPanel Margin="5" Orientation="Horizontal">

<Label >Task G</Label>

<ProgressBar x:Name="Gbar" Width="400"/>

</StackPanel>

<StackPanel Margin="5" Orientation="Horizontal">

<Label >Task H</Label>

<ProgressBar x:Name="Hbar" Width="400"/>

</StackPanel>

<StackPanel Margin="5" Orientation="Horizontal">

<Label >Task K</Label>

<ProgressBar x:Name="Kbar" Width="400"/>

</StackPanel>

</StackPanel>

</Grid>

</Window>

Пример работы программы:

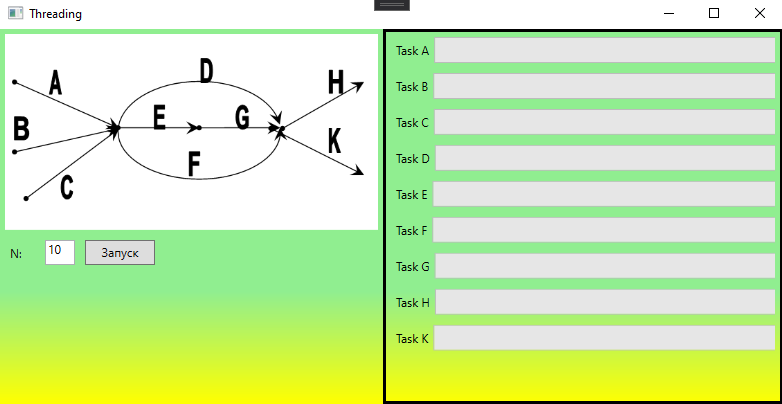


Рисунок 1. Начальное состояние

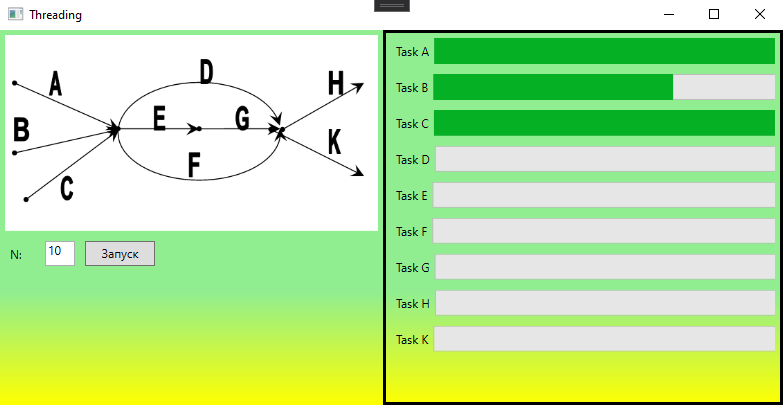


Рисунок 2. Во время выполнения

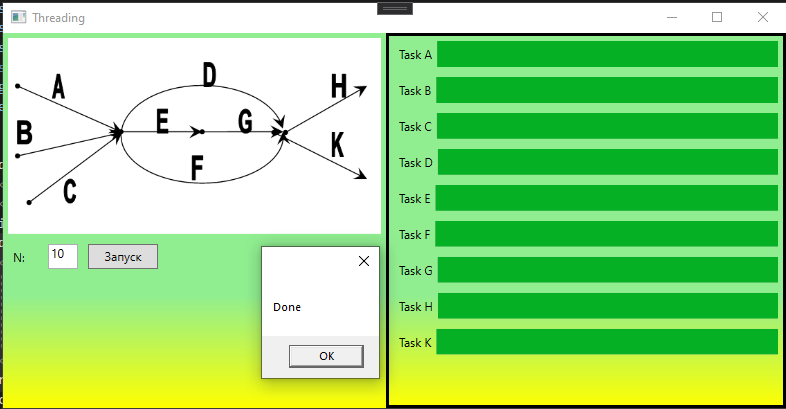


Рисунок 3. Выполнено

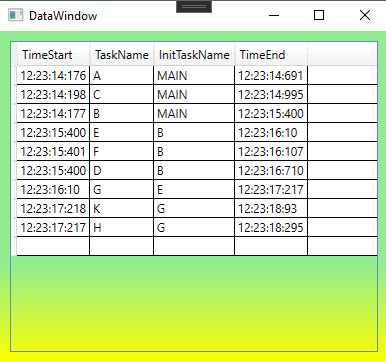


Рисунок 4. Таблица потоков