|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  федеральное государственное бюджетное образовательное  учреждение высшего образования  **«Национальный исследовательский университет «МЭИ»** |

|  |  |
| --- | --- |
| **Институт** | ИВТИ |
| **Кафедра** | ПМИИ |

|  |  |
| --- | --- |
|  |  |
| **Дисциплина: «Системное программирование»**  **Отчет по лабораторной работе №8**  **«Процессы и потоки»**  **Выполнил: студент группы А-13-22**  **Кокляева М.О.**  **Преподаватель: Меньшикова К. Г.**  **Москва, 2024** | |

**Выполнение работы.**

1. Программа должна выдать информацию о первичном (главном) потоке и создать еще три потока, которые стартуют одновременно (друг за другом).

Thread main\_th; //Главный поток

//Создание ещё 3-х потоков

Thread t1;

Thread t2;

Thread t3;

private void btn\_threads\_Click(object sender, EventArgs e)

{

main\_th = Thread.CurrentThread;

main\_th.Name = "main";

thread\_info.Text = "Имя домена: " + Thread.GetDomain().FriendlyName +

"\nИмя потока: " + main\_th.Name +

"\nСостояние потока: " + main\_th.ThreadState.ToString() +

"\nУровень приоритета: " + main\_th.Priority.ToString() +

"\nПоток существует: " + main\_th.IsAlive.ToString();

t1 = new Thread(random\_img1);

t2 = new Thread(random\_img2);

t3 = new Thread(show\_system\_info);

2.Два вторичных потока должны быть созданы на основе двух разных функций, каждая из которых реализует метод случайного представления фрагментов одной из двух картинок (каждая картинка режется на квадратики). Поток должен завершиться, если «соберет» свою картинку в правильном порядке (этот процесс должен занимать 2-3 минуты).

t1 = new Thread(random\_img1);

t2 = new Thread(random\_img2);

В первом потоке картинка нарезается (2\*2)

Во втором потоке картинка нарезается (6\*1)

public void random\_img1()

{

// Включаем замер времени

image\_1\_stopwatch.Start();

// Создаем список из структуры (битмап - номер ячейки в пазле)

List<Tuple<Bitmap, int>> list\_image\_parts = new List<Tuple<Bitmap, int>>();

// Загружаем картинку

Bitmap bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP8\WindowsFormsApp1\daman.png");

Graphics DC = image1.CreateGraphics();

// Номер ячейки

int cell\_number = 0;

// Цикл по двум строкам

for (int i = 0; i < 2; i++)

{

// Начальая и конечная у-координаты ячейки

int y\_begin = i \* (bitmap.Height / 2);

int y\_end = i \* (bitmap.Height / 2) + (bitmap.Height / 2);

// Начальая и конечная х-координаты ячейки

int x\_begin = 0;

int x\_end = (bitmap.Width / 2);

// Цикл по 2 столбцам

for (int j = 0; j <= 1; j++)

{

// Получаем прямоугольник (ячейку в матрице)

Rectangle rect = new Rectangle(x\_begin, y\_begin, x\_end - x\_begin, y\_end - y\_begin);

// Получаем регион битмапа по в этом прямоугольнике

Bitmap buffer = GetBitmapRegion(rect, bitmap);

// Составляем структуру битмап-номер ячейки

var image\_part = new Tuple<Bitmap, int>(buffer, cell\_number);

// Добавляем в список

list\_image\_parts.Add(image\_part);

x\_begin += (bitmap.Width / 2);

x\_end += (bitmap.Width / 2);

cell\_number++;

}

}

// Флаг, ассоциирующий собранность картинки

bool isAssembled = false;

while (!isAssembled)

{

isAssembled = true;

Random random = new Random();

// Перемешиваем ячейки

for (int i = 3; i >= 1; i--)

{

int num = random.Next(i + 1);

var rand\_image\_part = list\_image\_parts[num];

list\_image\_parts[num] = list\_image\_parts[i];

list\_image\_parts[i] = rand\_image\_part;

}

// Выводим перемешанные ячейки на экран

cell\_number = 0;

int fragmentWidth = bitmap.Width / 2;

int fragmentHeight = bitmap.Height / 2;

for (int i = 0; i < 2; i++)

{

int y\_begin = i \* fragmentHeight;

for (int j = 0; j < 2; j++)

{

int x\_begin = j \* fragmentWidth;

DC.DrawImage(list\_image\_parts[cell\_number].Item1, x\_begin, y\_begin, fragmentWidth, fragmentHeight); // Изменение здесь

cell\_number++;

}

}

// Проверяем собранность

for (int i = 0; i < 4; i++)

{

if (list\_image\_parts[i].Item2 != i)

{

isAssembled = false;

break;

}

}

// Если собралось - завершаем процедуру

if (isAssembled)

{

image\_1\_stopwatch.Stop();

image\_1\_timespan = image\_1\_stopwatch.Elapsed;

return;

}

}

}

public void random\_img2()

{

// Включаем замер времени

image\_2\_stopwatch.Start();

// Создаем список из структуры (битмап - номер ячейки в пазле)

List<Tuple<Bitmap, int>> list\_image\_parts = new List<Tuple<Bitmap, int>>();

// Загружаем картинку

Bitmap bitmap = new Bitmap(@"C:\Users\gagag\Desktop\3 курс\Системное программирование 2024\Лабы\SP8\WindowsFormsApp1\binto.png");

Graphics DC = image2.CreateGraphics();

// Номер ячейки

int cell\_number = 0;

// Цикл по 6 строкам

for (int i = 0; i < 6; i++)

{

// Начальая и конечная х-координаты ячейки

int x\_begin = 0;

int x\_end = bitmap.Width;

// Начальая и конечная у-координаты ячейки

int y\_begin = i \* (bitmap.Height / 6);

int y\_end = i \* (bitmap.Height / 6) + (bitmap.Height / 6);

// Получаем прямоугольник (ячейку в матрице)

Rectangle rect = new Rectangle(x\_begin, y\_begin, x\_end -

x\_begin, y\_end - y\_begin);

// Получаем регион битмапа в этом прямоугольнике

Bitmap buffer = GetBitmapRegion(rect, bitmap);

// Составляем структуру битмап-номер ячейки

var image\_part = new Tuple<Bitmap, int>(buffer,

cell\_number);

// Добавляем в список

list\_image\_parts.Add(image\_part);

y\_begin += (bitmap.Height / 6);

y\_end += (bitmap.Height / 6);

cell\_number++;

}

// Флаг, ассоциирующий собранность картинки

bool isAssembled = false;

while (!isAssembled)

{

isAssembled = true;

Random random = new Random();

// Перемешиваем ячейки

for (int i = 5; i >= 1; i--)

{

int num = random.Next(i + 1);

var rand\_image\_part = list\_image\_parts[num];

list\_image\_parts[num] = list\_image\_parts[i];

list\_image\_parts[i] = rand\_image\_part;

}

// Выводим перемешанные ячейки на экран

cell\_number = 0;

for (int i = 0; i < 6; i++)

{

// Начальая и конечная х-координаты ячейки

int x\_begin = 0;

int x\_end = bitmap.Width;

// Начальая и конечная у-координаты ячейки

int y\_begin = i \* (bitmap.Height / 6);

int y\_end = i \* (bitmap.Height / 6) + (bitmap.Height /

6);

DC.DrawImage(list\_image\_parts[cell\_number].Item1,

x\_begin, y\_begin, x\_end - x\_begin, y\_end - y\_begin);

x\_begin += (bitmap.Width / 6);

x\_end += (bitmap.Width / 6);

cell\_number++;

}

// Проверяем собранность

for (int i = 0; i < 6; i++)

{

if (list\_image\_parts[i].Item2 != i)

{

isAssembled = false;

break;

}

}

// Если собралось - завершаем процедуру

if (isAssembled)

{

image\_2\_stopwatch.Stop();

image\_2\_timespan = image\_2\_stopwatch.Elapsed;

return;

}

}

}

3.Приложение в процессе «сбора» картинки должно выводить на экран фрагменты картинок (в выбранном в потоке порядке).

// Выводим перемешанные ячейки на экран 1-ый поток

cell\_number = 0;

int fragmentWidth = bitmap.Width / 2;

int fragmentHeight = bitmap.Height / 2;

for (int i = 0; i < 2; i++)

{

int y\_begin = i \* fragmentHeight;

for (int j = 0; j < 2; j++)

{

int x\_begin = j \* fragmentWidth;

DC.DrawImage(list\_image\_parts[cell\_number].Item1, x\_begin, y\_begin, fragmentWidth, fragmentHeight); // Изменение здесь

cell\_number++;

}

}

// Проверяем собранность

for (int i = 0; i < 4; i++)

{

if (list\_image\_parts[i].Item2 != i)

{

isAssembled = false;

break;

}

}

// Если собралось - завершаем процедуру

if (isAssembled)

{

image\_1\_stopwatch.Stop();

image\_1\_timespan = image\_1\_stopwatch.Elapsed;

return;

}

}

// Выводим перемешанные ячейки на экран 2-й поток

cell\_number = 0;

for (int i = 0; i < 6; i++)

{

// Начальая и конечная х-координаты ячейки

int x\_begin = 0;

int x\_end = bitmap.Width;

// Начальая и конечная у-координаты ячейки

int y\_begin = i \* (bitmap.Height / 6);

int y\_end = i \* (bitmap.Height / 6) + (bitmap.Height /

6);

DC.DrawImage(list\_image\_parts[cell\_number].Item1,

x\_begin, y\_begin, x\_end - x\_begin, y\_end - y\_begin);

x\_begin += (bitmap.Width / 6);

x\_end += (bitmap.Width / 6);

cell\_number++;

}

// Проверяем собранность

for (int i = 0; i < 6; i++)

{

if (list\_image\_parts[i].Item2 != i)

{

isAssembled = false;

break;

}

}

// Если собралось - завершаем процедуру

if (isAssembled)

{

image\_2\_stopwatch.Stop();

image\_2\_timespan = image\_2\_stopwatch.Elapsed;

return;

}

}

4.Третий поток должен собрать информацию о компьютерной системе **(не менее 10-ти параметров из трех источников: 1) функции Windows, 2) реестр, 3) программное окружение)**. Для демонстрации этой информации на экране поток должен подготовить **картинку в графическом виде** (в виде рекламы или презентации) **и ждать** завершения двух предыдущих потоков (информация о системе должна быть представлена пользователю, когда произойдет завершение двух потоков собирающих картинки)

t3 = new Thread(show\_system\_info);

public void show\_system\_info()

{

image\_3\_stopwatch.Start();

var info = new StringBuilder();

// 1. Из функций Windows:

info.AppendFormat("Из функций Windows\n");

info.AppendFormat($"Пользователь: {Environment.UserName} \n");

info.AppendFormat($"Путь: {Environment.CurrentDirectory}\n");

info.AppendFormat("Версия Windows: {0}\n",Environment.OSVersion);

info.AppendFormat("Число процессоров: {0}\n", Environment.ProcessorCount);

info.AppendFormat("Использованная память: {0}\n", Environment.WorkingSet);

// 2. Из реестра:

using (RegistryKey key = Registry.LocalMachine.OpenSubKey(@"SOFTWARE\Microsoft\Windows NT\CurrentVersion"))

{

///object installDateObj = key.GetValue("InstallDate");

///long installDateTicks = 0; // Инициализация

///DateTime installDate = new DateTime(1601, 1, 1, 0, 0, 0, DateTimeKind.Utc).AddSeconds(installDateTicks);

info.AppendFormat("\nИз реестра\n");

info.AppendFormat($"Системный каталог: {key.GetValue("SystemRoot")}\n");

info.AppendFormat($"Название продукта: {key.GetValue("ProductName")}\n");

}

// 3. Из программного окружения (WMI):

using (var searcher = new ManagementObjectSearcher("SELECT \* FROM Win32\_ComputerSystem"))

{

foreach (ManagementObject obj in searcher.Get())

{

info.AppendFormat("\nИз Программного окружения\n");

info.AppendFormat($"Производитель: {obj["Manufacturer"]}\n");

info.AppendFormat($"Модель: {obj["Model"]} \n");

info.AppendFormat($"Тип системы: {obj["SystemType"]}\n");

}

}

t1.Join();

t2.Join();

Graphics DC = present.CreateGraphics();

DC.DrawString(info.ToString(), new Font("Franklin Gothic Medium Cond", 8),Brushes.Black, 3, 3);

image\_3\_stopwatch.Stop();

image\_3\_timespan = image\_3\_stopwatch.Elapsed;

}

5.Программа должна также предоставить пользователю информации о времени, в течении которого выполнялись потоки.

// значение может быть присвоено только раз, во время объявления

// Секудомеры для двух потоков

readonly Stopwatch image\_1\_stopwatch = new Stopwatch();

readonly Stopwatch image\_2\_stopwatch = new Stopwatch();

readonly Stopwatch image\_3\_stopwatch = new Stopwatch();

//ромежуток времени

TimeSpan image\_1\_timespan;

TimeSpan image\_2\_timespan;

TimeSpan image\_3\_timespan;

В событии кнопки:

label1.Text = "Поток 1: " +String.Format("{0:00}:{1:00}:{2:00}.{3:00}",

image\_1\_timespan.Hours,

image\_1\_timespan.Minutes, image\_1\_timespan.Seconds,

image\_1\_timespan.Milliseconds);

label2.Text = "Поток 2: " + String.Format("{0:00}:{1:00}:{2:00}.{3:00}",

image\_2\_timespan.Hours,

image\_2\_timespan.Minutes, image\_1\_timespan.Seconds,

image\_2\_timespan.Milliseconds);

label3.Text = "Поток 3: " + String.Format("{0:00}:{1:00}:{2:00}.{3:00}",

image\_3\_timespan.Hours,

image\_3\_timespan.Minutes, image\_1\_timespan.Seconds,image\_3\_timespan.Milliseconds);

В функциях:

В начале:

image\_1\_stopwatch.Start();

image\_2\_stopwatch.Start();

image\_3\_stopwatch.Start();

В конце:

image\_1\_stopwatch.Stop();

image\_1\_timespan = image\_1\_stopwatch.Elapsed;

image\_2\_stopwatch.Stop();

image\_2\_timespan = image\_1\_stopwatch.Elapsed;

image\_3\_stopwatch.Stop();

image\_3\_timespan = image\_1\_stopwatch.Elapsed;

Результат работы программы:

