**C# как язык параллельного программирования.**

**Библиотека System.Threading.**

Курсовая работа

по дисциплине Алгоритмы и языки

параллельного программирования

студента физико-математического

факультета 4 курса группы “ВГ”

Котова Виталия Евгеньевича

Научный руководитель

Старший преподаватель кафедры информатика

Каменская Наталья Евгеньевна

# **ОГЛАВЛЕНИЕ**

[ОГЛАВЛЕНИЕ 2](#_Toc389734398)

[ВВЕДЕНИЕ 3](#_Toc389734399)

[ГЛАВА 1. ОСНОВЫ МНОГОПОТОЧНОЙ ОБРАБОТКИ 4](#_Toc389734400)

[1.1. Создание и запуск потока 4](#_Toc389734401)

[1.2. Определение момента окончания потока 5](#_Toc389734402)

[1.3. Передача аргумента потоку 6](#_Toc389734403)

[1.4. Приоритеты потоков 7](#_Toc389734404)

[1.5. Свойство IsBackground 8](#_Toc389734405)

[1.6. Прерывание потока 8](#_Toc389734406)

[ГЛАВА 2. СИНХРОНИЗАЦИИЯ ПОТОКОВ 10](#_Toc389734407)

[2.1. Понятие синхронизации 10](#_Toc389734408)

[2.2. Взаимоблокировка и состояние гонки 11](#_Toc389734409)

[ГЛАВА 3. Полная реализация решения задачи 13](#_Toc389734410)

[3.1. Создание класса Forest 13](#_Toc389734411)

[3.2. Класс LifeSaver 14](#_Toc389734412)

[3.3. Классы Father, Mother, GrandFather, GrandMother 14](#_Toc389734413)

[3.4. Класс Girl 15](#_Toc389734414)

[ЗАКЛЮЧЕНИЕ 16](#_Toc389734415)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc389734416)

# ВВЕДЕНИЕ

Среди многих свойств языка С# особое место принадлежит поддержке

1. ОСНОВЫ МНОГОПОТОЧНОЙ ОБРАБОТКИ

## Создание и запуск потока

Система многопоточной обработки основывается на классе Thread, который инкапсулирует поток исполнения. Класс Thread является герметичным, т.е. он не может наследоваться. В классе Thread определен ряд методов и свойств, предназначенных для управления потоками.

## Определение момента окончания потока

Нередко оказывается полезно знать, когда именно завершается поток. С этой целью можно, прежде всего, опросить доступное только для чтения свойство IsAlive, определяемое следующим образом.

public bool IsAlive { get; }

Свойство IsAlive возвращает логическое значение true, если поток, для которого оно вызывается, по-прежнему выполняется.

## Передача аргумента потоку

Первоначально в среде .NET Framework нельзя было передавать аргумент потоку, когда он начинался, поскольку у метода, служившего в качестве точки входа в поток, не могло быть параметров. Если же потоку требовалось передать какую-то информацию, то к этой цели приходилось идти различными обходными путями, например, использовать общую переменную. Но этот недостаток был впоследствии устранен, и теперь

## Приоритеты потоков

У каждого потока имеется свой приоритет, который отчасти определяет, насколько часто поток получает доступ к ЦП. Низкоприоритетные потоки получают доступ к ЦП реже, чем высокоприоритетные. Таким образом, в течение заданного промежутка времени низкоприоритетному потоку будет доступно меньше времени ЦП, чем высокоприоритетному. Как и следовало ожидать, время ЦП, получаемое потоком, оказывает определяющее влияние на характер его выполнения и взаимодействия с другими потоками, исполняемыми в настоящий момент в системе.

## Свойство IsBackground

В среде .NET Framework определены две разновидности потоков: приоритетный и фоновый. Единственное отличие между ними заключается в том, что процесс не завершится до тех пор, пока не окончится приоритетный поток, тогда как фоновые потоки завершаются автоматически по окончании

## Прерывание потока

Иногда поток полезно прервать до его нормального завершения. Например, отладчику может понадобиться прервать вышедший из-под контроля поток. После прерывания поток удаляется из системы и не может быть начат снова. Для прерывания потока до его нормального завершения служит метод Thread.Abort(). Ниже приведена простейшая форма этого/

1. СИНХРОНИЗАЦИИЯ ПОТОКОВ

## Понятие синхронизации

Когда используется несколько потоков, то иногда приходится координировать их действия. Процесс достижения такой координации называется синхронизацией. Самой распространенной причиной применения синхронизации служит необходимость разделять среди двух или более потоков общий ресурс, который может быть одновременно доступен только

## Взаимоблокировка и состояние гонки

При разработке многопоточных программ следует быть особенно внимательным, чтобы избежать взаимоблокировки и состояний гонок. Взаимоблокировка, как подразумевает само название, — это ситуация, в которой один поток ожидает определенных действий от другого потока, а другой поток, в свою очередь, ожидает чего-то от первого потока. В итоге оба потока приостанавливаются, ожидая друг друга, и ни один из них не выполняется.

1. Реализация решения задачи

## Создание класса Forest

Создадим класс Forest в котором будет реализован лес.

struct ForestPoint

{

public bool heardAyy; //истина если в точке леса

//слышна «Ауу»!

public int personID;

}

static class Forest

{

private static int \_size = 50;//размер леса

public static ForestPoint[,] forest =

new ForestPoint[\_size, \_size];

public static int Size

{

get { return \_size; }

}

public static void Print()

{

for (int i = 0; i < \_size; i++)

{

for (int j = 0; j < \_size; j++)

{

//выводим в консоль букву в

//соответствии c personID

}

}

}

public static void PrintChanges()

{

Thread.Sleep(30);

Console.Clear();

Forest.Print();

}

}

где метод Print() выводит в консоль состояние леса.

## Класс LifeSaver

В класс LifeSaver реализованы методы которые отвечают за поиск девочки, а также следующие поля: \_x, \_y – первоначальные координаты спасателя; \_speed – скорость передвижения спасателя по лесу; \_girl – ссылка на объект класса Girl; \_idLifeSaver – номер спасателя; searching – ссылка на объект класса Thread. Конструктор этого класса приведен ниже.

public LifeSaver(int id, int yStart, int xStart, Girl girl,

int speed)

Опишем методы класса LifeSaver.

private void Movement()

в этом методе осуществляется движение спасателя по лесу.

protected virtual void MovementAlgorithm(int speed

ref int y, ref int x)

реализует алгоритм движения спасателя. Алгоритм представляет собой хаотичное передвижение в пределах размера леса.

## Классы Father, Mother, GrandFather, GrandMother

Классы Father, Mother, GrandFather, GrandMother наследуют класс LifeSaver. В классах Mother и GrandMother перегружается метод MovementAlgorithm(), реализующий алгоритм движения спасателя.

## Класс Girl

В классе Girl реализован объект девочка. Класс имеет следующие поля: \_x, \_y – первоначальные координаты девочки; \_region – размер области где слышно девочку; movement – ссылка на объект класса Thread.

Класс Girl имеет следующий конструктор:

public Girl()

{

\_random = new Random();

\_region = 4;

\_x = \_random.Next(1, Forest.Size - 2);

\_y = \_random.Next(1, Forest.Size - 2);

lock (Forest.forest)

{

SetRegion(true);

Forest.PrintChanges();

}

movement = new Thread(MovementGirls);

movement.Start();

}

Реализации этого класса имеет следующие методы:

private void MovementGirls()

этот метод осуществляется движение девочки по лесу.

private void MoveGirl\_and\_Region(Move moveTo)

перемещает девочку и ее голос по лесу.

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе были рассмотрены средства библиотеки System.Threading на примере решения задачи, сформулированной во введении. Найти приложение реализующее решение задачи можно в [Disk]:\CourseWork\LostGirl\LostGirl\obj\x86\Debug\ LostGirl.exe.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рихтер Дж., CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#, 2012. – 928 с.
2. Троелсен, Э., Язык программирования C# 5.0 и платформа .NET 4.5 6-е издание, 2012. – 1311 с.
3. Шилдт, Г., C# 4.0 Полное руководство, 2011. – 1056 с.
4. Microsoft Developer Network [Электронный ресурс] Режим доступа: http://msdn.microsoft.com/ru-ru/library/system.threading.aspx

Дата доступа: 10.05.2014.