Э.Н. Самохвалов, Г.И. Ревунков, Ю.Е. Гапанюк

Введение в проектирование и разработку приложений на языке программирования C#

Учебное пособие

BAUMANPRESS
Москва
ИЗДАТЕЛЬСТВО
МГТУ им. Н. Э. Баумана
2018

УДК 004.43
ББК 32.973.2
С17

Издание доступно в электронном виде на портале *ebooks.bmstu.ru* по адресу: http://ebooks.bmstu.ru/catalog/193/book1764.html

Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Рекомендовано Редакционно-издательским советом МГТУ им. Н.Э. Баумана в качестве учебного пособия

Самохвалов, Э. Н.

С17 Введение в проектирование и разработку приложений на языке программирования C# : учебное пособие / Э. Н. Самохвалов, Г. И. Ревунков, Ю. Е. Гапанюк. — Москва : Издательство МГТУ им. Н. Э. Баумана, 2018. — 244, [4] с. : ил.

ISBN 978-5-7038-4553-0

Представлены основы языка программирования C#. Рассмотрены среда исполнения .NET, конструкции языка C# и объектно-ориентированное программирование на нем, работа с коллекциями, файловой системой, рефлексией, а также параллельная обработка данных. Дано введение в технологию разработки оконных пользовательских интерфейсов Windows Forms.

Для студентов, изучающих информатику и вычислительную технику в МГТУ им. Н.Э. Баумана.

УДК 004.43
ББК 32.973.2

ISBN 978-5-7038-4553-0

Посвящается памяти любимого Учителя профессора Эдуарда Николаевича Самохвалова, который трудился в МГТУ им. Н.Э. Баумана более 50 лет и сделал очень много доброго для нашего вуза и кафедры «Системы обработки информации и управления»

Предисловие

Данное учебное пособие поможет ознакомиться с основами языка программирования C#. В нем изложены основы среды исполнения .NET, представлены конструкции языка C#, объектно-ориентированное программирование в C#, работа с коллекциями, файловой системой, рефлексией, параллельная обработка данных, а также дано введение в технологию разработки оконных пользовательских интерфейсов Windows Forms.

В пособии принят принцип обучения на основе примеров. Все предлагаемые технологии рассмотрены в виде **последовательности примеров**, архив с которыми расположен на сайте кафедры ИУ-5 (http://iu5.bmstu.ru). Архив содержит набор **пронумерованных примеров проектов, ссылки на их номера даны в тексте**.

В качестве среды разработки используется Visual Studio Community 2017. В соответствии с лицензионным соглашением на данный программный продукт «любое количество ваших пользователей могут использовать это программное обеспечение для разработки и тестирования приложений в рамках сетевого дистанционного или аудиторного обучения и образования, а также для проведения академических исследований» (условия лицензионного соглашения на использование программного обеспечения Microsoft Visual Studio Community 2017 представлены на https://www.visualstudio.com/ru/license-terms/mlt553321/).

Дисциплину «Базовые компоненты интернет-технологий» читают преподаватели кафедры «Системы обработки информации и управления» (ИУ-5) в третьем семестре. В ее рамках основной предмет изучения — язык программирования C#. Это обусловлено следующим:

• в двух первых семестрах студенты кафедры изучают языки программирования C и C++, поэтому к третьему они знакомы с синтаксисом языка C++ и основами объектно-ориентированного программирования на C++. Язык программирования C# имеет много общего с C++, поэтому его изучение непосредственно после C++ позволяет сконцентрироваться на сходствах и различиях между ними;

• в отличие от объектно-ориентированного языка программирования C++, считающегося классическим, C# содержит ряд конструкций, присущих объектно-ориентированным языкам следующего поколения, например таких, как делегаты и рефлексия;

• для языка C# реализовано большое количество прикладных технологий, применяющихся при создании промышленных приложений; Windows Forms и WPF для разработки оконного пользовательского интерфейса; LINQ и Entity Framework для обработки данных; ASP.NET MVC для создания веб-приложений. Использовать язык C++ можно только с ограниченным набором прикладных технологий, например с Windows Forms, разработка веб-приложений на основе ASP.NET MVC не предполагает применения языка C++;

• дисциплина «Разработка интернет-приложений», которую читают преподаватели кафедры «Системы обработки информации и управления» в пятом семестре, включает в себя изучение технологии ASP.NET MVC, для чего требуется знание языка C#.

Цель преподавания дисциплины «Базовые компоненты интернет-технологий» — содействовать формированию у обучающихся следующих компетенций:

• способности разрабатывать и отлаживать компоненты аппаратно-программных комплексов с помощью современных автоматизированных средств проектирования (ПК-7);

• умению разрабатывать интерфейсы «человек — ЭВМ» (ПК-12).

В результате изучения дисциплины «Базовые компоненты интернет-технологий»:

студент должен знать:

• основы языка программирования C#;

уметь:

• разрабатывать консольные и оконные приложения с использованием C#;

• создавать интерфейсы человек — ЭВМ с применением технологии Windows Forms;

иметь навыки:

• разработки консольных и оконных приложений с использованием C#;

• разработки интерфейсов человек — ЭВМ с помощью Windows Forms.

Материал данного учебного пособия в целом соответствует содержанию дисциплины «Базовые компоненты интернет-технологий», которую читают в третьем семестре преподаватели кафедры «Системы обработки информации и управления».

В заключение еще раз отметим, что **архив с используемыми примерами** находится на сайте кафедры ИУ-5 по адресу: http://iu5.bmstu.ru/

Введение

Особенность современного состояния дел в области разработки интернет-приложений заключается в том, что используется много технологий.

На стороне веб-браузера — язык разметки HTML, технология каскадных таблиц стилей CSS, язык программирования JavaScript. На стороне веб-сервера — большое количество фреймворков (каркасов) для веб-разработки на различных языках программирования, из которых наиболее известны PHP, Python, Perl, C#, Java, JavaScript.

Для использования какого-либо из фреймворков необходимо знать соответствующий язык программирования. Сейчас в сообществе веб-разработчиков достаточно популярен фреймворк ASP.NET MVC, который предполагает знание языка C#.

Именно поэтому изучению языка C# посвящено данное издание.

В учебном пособии рассмотрены краткая характеристика среды исполнения .NET, базовые конструкции языка программирования C#, основы объектно-ориентированного программирования с использованием C#, работа с коллекциями, основы применения файловой системы, рефлексия, основы параллельной обработки данных, основы технологии Windows Forms.

Приведенный материал в целом соответствует содержанию дисциплины «Базовые компоненты интернет-технологий».

1. Краткая характеристика языка программирования C#

Современный объектно-ориентированный язык программирования общего назначения C# разработан компанией Microsoft для платформы .NET. С его помощью можно создавать консольные приложения, оконные приложения с использованием технологий Windows Forms и WPF, веб-приложения с применением технологий ASP.NET и ASP.NET MVC. Для обработки данных предназначены технологии LINQ и Entity Framework.

Первая версия языка C# появилась в начале 2000 г. Его создателем был Андерс Хейлсберг, который до работы над ним разработал компилятор с языка Паскаль и среду разработки Delphi. Это, безусловно, сказалось на C#, который, несмотря на синтаксис, унаследованный от C++, впитал в себя лучшие структурные черты Паскаля. В 2005 г. вышла его версия 2.0, включавшая в себя средства и библиотеки для разработки оконных приложений на основе технологии Windows Forms.

В 2008 г. была выпущена версия 3.5, содержавшая средства и библиотеки для разработки оконных приложений на основе технологии WPF. В нее была добавлена технология LINQ, представлявшая собой SQL-подобный язык запросов, встроенный в C#. Этот новаторский подход Андерса Хейлсберга до сих пор не имеет развитых аналогов в других языках программирования. Фактически, именно версия 3.5 сформировала основную концепцию языка, которая с тех пор не претерпела принципиальных изменений, хотя в его новые версии добавлено большое количество новых синтаксических конструкций и библиотек, особенно для параллельной и асинхронной обработки данных.

Язык C# продолжает активно развиваться. Из новых проектов следует отметить проект модульного компилятора Roslyn, предоставляющего разработчикам API для различных этапов компиляции, что облегчает создание новых языков программирования для платформы .NET.

Нужно подчеркнуть, что Андерсу Хейлсбергу удалось не только решить сложную техническую задачу по созданию быстрого компилятора, но и, главное, создать концептуально удачный язык программирования, который подходит как для задач обучения, так и для практической разработки крупных проектов. Основные черты языка программирования C#:

• синтаксис, в целом унаследованный от C++ и Java;

• некоторые конструкции языка, унаследованные из Паскаля, но «обернутые» в синтаксис, подобный тому, что у C++;

• хорошая читаемость кода, что очень важно при разработке больших проектов;

• технология LINQ, облегчающая обработку сложных структур данных;

• большое количество библиотек, что полезно для решения различных задач.

Конечно, в рамках небольшого пособия можно рассмотреть только основы языка программирования C#. Для его углубленного изучения стоит использовать работы [1–3].

Следует также отметить, что среда исполнения .NET и язык программирования C# детально документированы в справочной системе MSDN (Microsoft Developer Network) — https://msdn.microsoft.com.

2. Среда исполнения .NET

2.1. Краткое описание среды исполнения .NET

Среда .NET представляет собой виртуальную среду исполнения программ, или, проще говоря, программно-реализованную модель микропроцессора, которая исполняет команды на специфическом машинном языке, который называется MSIL (Microsoft Intermediate Language) или иногда IL (Intermediate Language). Он подобен другим языкам машинных команд, например языку команд микропроцессора Intel.

Как для команд микропроцессора Intel существует язык ассемблера, так и для MSIL есть специфический язык ассемблера. Программу на нем можно получить путем дизассемблирования исполняемого файла .NET.

Контрольные вопросы

1. Что такое MSIL?
2. Что такое CLR?
3. Для чего используется сборщик мусора?
4. В чем различие между платформами .NET, Mono и .NET Core?
5. Опишите процесс компиляции, компоновки и исполнения программы для .NET-платформы.
6. В чем различие между процессами компиляции и компоновки для языков без использования среды исполнения и для .NET- платформы?
7. Что такое JIT-компиляция?
8. Почему машинный код, сформированный JIT-компилятором, будет в некоторых случаях выполняться быстрее, чем машинный код, сформированный обычным компилятором?

3. Основы языка C#

3.1. Организация типов данных в языке C#

Особенность типов данных языка C# заключается в том, что для всей среды исполнения .NET используется единая общая система типов — CTS (Common Type System), причем и в C#, и в Visual Basic .NET, а также в других языках .NET-платформы.

Все типы данных разделяются на типы-значения (value type) и ссылочные типы (reference type). Типы-значения хранятся в области памяти, доступ к которой организован в виде стека. К ним относятся базовые типы данных, перечисления (enum), структуры (struct).

Если переменная типа-значения передается как параметр в метод, то в стеке делается копия значения и передается в метод, а исходное значение не изменяется. В частности, такой порядок копирования значений удобен для примитивных типов.

Ссылочные типы хранятся в динамически распределяемой области памяти, которую принято называть кучей (heap). Для работы с ней используются ссылки, фактически представляющие собой указатели. Ссылочными типами являются классы, интерфейсы, массивы, делегаты.

```
            }
        }
        while (!flag);

        return resultDouble;
    }
  }
}
```

В этом примере следует обратить внимание на функцию ReadDouble, выполняющую ввод строки с клавиатуры. Также она пытается преобразовать строку в вещественное число. Если это невозможно, то строка вводится повторно до тех пор, пока преобразование строки в число не пройдет успешно.

Контрольные вопросы

1. Что такое CTS?
2. Что такое типы-значения?
3. Что такое ссылочные типы?
4. Какие целочисленные типы данных существуют в языке C#?
5. Какие типы данных с плавающей точкой есть в языке C#?
6. Какие символьные и строковые типы данных существуют в языке C#?
7. Что такое тип object?
8. Какими способами можно преобразовать значение строкового типа в значение числового типа?
9. Как объявляются одномерные массивы в языке C#?
10. Как объявляются прямоугольные и зубчатые многомерные массивы в языке C#? В чем разница между ними?
11. Какие средства консольного ввода-вывода существуют в языке C#?
12. Как работает форматированный вывод в консоль?
13. Что такое пространства имен и сборки, как они соотносятся друг с другом?
14. Как задаются и обрабатываются аргументы командной строки в консольном приложении?
15. Какие условные операторы существуют в языке C#?
16. Как реализованы операторы сопоставления с образцом в языке C#?
17. Какие операторы цикла существуют в языке C#?
18. Как работает цикл foreach?

19. Как работает механизм обработки исключений в языке C#?
20. В чем особенность порядка расположения блоков catch в операторе обработки исключений?
21. Как работают ref-параметры в языке C#?
22. Как работают out-параметры в языке C#?
23. Как передать в метод переменное количество параметров?
24. В чем разница между операторами return и yield return?
25. Как применяются XML-комментарии в языке C#?
26. Как используются директивы препроцессора в языке C#?

4. Основы объектно-ориентированного программирования в языке C#

Если языки C++ и Java довольно схожи в плане синтаксиса основных конструкций, то в плане объектно-ориентированного программирования (ООП) они довольно сильно различаются.

Подход к ООП в языке C# в целом гораздо ближе к тому, что в Java, чем к тому, что в C++. Как и в Java, в языке C# нет множественного наследования классов, оно реализуется с помощью интерфейсов.

Ниже представлены основы ООП в языке C# на базе фрагментов **примера 3**, которые рассматриваются далее в этом разделе.

4.1. Объявление класса и его элементов

Классы в языке C# объявляются с использованием ключевого слова class. Здесь более детально рассмотрен базовый класс примера — BaseClass:

```
/// <summary>
/// Базовый класс
/// </summary>
class BaseClass
{
    private int i;

    //Конструктор
    public BaseClass(int param) { this.i = param; }
```

```
            Console.ReadLine();
        }
    }
}
```

В консоль будет выведено:

```
Прямоугольник площадью 20
Квадрат площадью 25
Круг площадью 78,5398163397448
```

Контрольные вопросы

1. Как объявить конструктор класса в языке C#?
2. Как объявить метод класса в языке C#?
3. Какие области видимости существуют в языке C#?
4. Что такое свойства и для чего они используются?
5. Что такое опорная переменная свойства?
6. Как используются get-аксессор и set-аксессор свойства?
7. Как задаются области видимости для свойств и аксессоров?
8. Приведите пример задания автоопределяемого свойства.
9. Приведите пример задания вычисляемого свойства.
10. Как объявить деструктор класса в языке C#?
11. Как объявить статические элементы класса в языке C#?
12. Как используется конструкция using static в языке C#?
13. Как реализуется наследование класса от класса?
14. Как из конструктора класса вызвать конструктор базового класса?
15. Как из конструктора класса вызвать другой конструктор текущего класса?
16. Как переопределить виртуальный метод?
17. В чем разница между ключевыми словами override и new при переопределении виртуального метода?
18. В чем сходства и различия между виртуальными и абстрактными методами?
19. Что такое абстрактный класс?
20. Как в Visual Studio сгенерировать заглушки методов при наследовании от абстрактного класса?
21. Что такое интерфейсы и для чего они используются в языке C#?
22. В чем сходства и различия между интерфейсами и абстрактными классами?

23. Можно ли в языке C# унаследовать класс от нескольких классов? От нескольких интерфейсов? Почему?

24. Как реализуется наследование класса от класса и интерфейсов?

25. Являются ли методы, унаследованные от интерфейсов, виртуальными?

26. Как в Visual Studio сгенерировать заглушки методов при наследовании от интерфейса?

27. В чем разница между «реализацией интерфейса» и «явной реализацией интерфейса»?

28. Что такое методы расширения и как они реализуются в языке C#?

29. Что такое частичные классы и как они реализуются в языке C#?

30. Как создать диаграмму классов в Visual Studio?

5. Расширенные возможности объектно-ориентированного программирования в языке C#

Помимо базовых возможностей ООП, связанных с созданием классов и интерфейсов и их наследованием, в языке C# существует большое количество расширенных возможностей, и некоторые из них являются уникальными особенностями языка C#.

5.1. Перечисления

Перечисление (enumeration, enum) — это множество целых чисел, за каждым из которых закреплена строковая константа. Перечисление позволяет перечислить множество значений какого-либо свойства.

В классическом языке C аналогом перечислений были объявления констант с помощью директивы препроцессора #define. Достаточно быстро программисты на языке C стали объявлять константы с «иерархическими» именами, в имя константы входили и наименование перечисления, и наименование конкретного значения. В дальнейшем это привело к появлению перечислений, использующихся в языках C++, Java и C#.

Контрольные вопросы

1. Что такое перечисление?
2. Как получить значение перечисления на основе строки?
3. Как реализуется перегрузка операторов в языке C#?
4. Что такое индексатор?
5. Как создать обобщенный класс в языке C#?
6. Как создать обобщенный метод в языке C#?
7. Что такое делегат?
8. Что такое неявная типизация и как она используется для делегатов?
9. Как передать в метод параметр типа делегат?
10. Что такое лямбда-выражения и как они используются?
11. Что такое замыкания?
12. Как реализованы локальные функции в языке C#?
13. Что такое члены класса, основанные на выражениях?
14. Как используется строковая интерполяция в языке C#?
15. Как применяется обобщенный делегат Func?
16. Как используется обобщенный делегат Action?
17. Что такое групповой делегат?
18. Как реализуется механизм событий в языке C#?

6. Работа с коллекциями

Коллекции — важная часть любого языка программирования. В языке C# существует развитая система коллекций. В данном разделе сначала рассмотрены примеры использования стандартных коллекций, а затем примеры создания нестандартных коллекций.

6.1. Стандартные коллекции

Подходы к работе с коллекциями в языках C++, Java и C# в целом совпадают. Все коллекции можно разделить на две большие категории — обобщенные (шаблонизированные в C++) и необобщенные.

Использование обобщенной коллекции предполагает, что в нее помещаются элементы обобщенного типа, следовательно, все эле-

Метод Push просто вызывает метод Add для добавления данных в конец списка. Метод Pop в готовом виде в списке не реализован. Он возвращает последний элемент списка, а затем удаляет его оттуда. Реализация метода зависит от количества элементов в списке. Если в списке их не содержится, то возвращается значение по умолчанию для обобщенного типа default(T). Если в списке один элемент, то он возвращается, а список переводится в состояние пустого списка. Если список включает в себя более одного элемента, то возвращается и удаляется из списка последний элемент, а предпоследний устанавливается в качестве последнего.

Пример использования класса SimpleStack

```
SimpleStack<Figure> stack = new SimpleStack<Figure>();
//добавление данных в стек
stack.Push(rect);
stack.Push(square);
stack.Push(circle);
//чтение данных из стека
while (stack.Count > 0)
{
     Figure f = stack.Pop();
     Console.WriteLine(f);
}
```

Результаты вывода в консоль:

```
Стек
Круг площадью 78,5398163397448
Квадрат площадью 25
Прямоугольник площадью 20
```

В соответствии с ожидаемым алгоритмом работы метод Pop извлекает элементы из вершины стека в обратном порядке.

Контрольные вопросы

1. В чем сходство и различие между обобщенными и необобщенными коллекциями?
2. Как выполняется работа с обобщенным списком?
3. Как происходит работа с необобщенным списком?
4. Как осуществляется работа с обобщенным стеком?
5. Как проводится работа с обобщенной очередью?
6. Как ведется работа с обобщенным словарем?
7. Как выполняется работа с кортежем?

8. В чем особенности работы с новым синтаксисом кортежей в версии 7.0 языка C#?

9. Как реализуется сортировка коллекций?

10. Как используется интерфейс IComparable при сортировке коллекций?

11. Как можно реализовать класс разреженной матрицы на основе класса словаря?

12. Как можно реализовать классы списка и стека без применения стандартных коллекций?

7. Работа с файловой системой

Платформа .NET и язык C# предоставляют широкие возможности для работы с файловой системой. В данном разделе рассматриваются некоторые из этих возможностей на основе фрагментов **примера 12**.

7.1. Получение данных о файлах и каталогах

Наиболее простой способ получения информации о файловой системе — использование статического класса Directory. Если класс является статическим, то предполагается, что все его свойства и методы также статические.

Класс Directory, объявленный в пространстве имен System.IO, позволяет вести работу с каталогами: создание (метод CreateDirectory), удаление (метод Delete), перемещение (метод Move) каталогов, получение списка файлов и подкаталогов.

Пример вывода списка файлов и каталогов в корне диска C

```
string catalogName = @"c:\";
Console.WriteLine("\nСписок файлов каталога " + catalogName);
string[] files = Directory.GetFiles(catalogName);
foreach (string file in files)
{
     Console.WriteLine(file);
}

Console.WriteLine("\nСписок подкаталогов каталога " +
                  catalogName);
string[] dirs = Directory.GetDirectories(catalogName);
```

Результаты вывода в консоль:

```
Сериализация/десериализация в формате XML с атрибутами:
До сериализации:
str1=строка1 xml int1=3333 double1=333,33
После десериализации:
str1=строка1 xml int1=0 double1=333,33
```

Как и ожидалось, после десериализации свойство int1 не восстановлено (по умолчанию имеет значение, равное нулю), так как данное свойство в классе данных помечено атрибутом XmlIgnore.

Контрольные вопросы

1. Как получить список файлов для заданного каталога?
2. Как получить список подкаталогов для заданного каталога?
3. Как выполнить чтение текстового файла в виде единой строки?
4. Как записать текстовый файл в виде единой строки?
5. Как реализовать чтение текстового файла в виде массива строк?
6. Как записать текстовый файл в виде массива строк?
7. Как выполнить сериализацию и десериализацию объекта в бинарный файл?
8. Как осуществить сериализацию и десериализацию объекта в XML-файл?
9. Как управлять сериализацией и десериализацией объекта в XML-файл с использованием атрибутов?

8. Рефлексия

В данном разделе рассматриваются основы механизма рефлексии, которая, с одной стороны, включает достаточно большое количество сведений [1], с другой — очень широко применяется в современном языке C#. Выше в тексте рефлексия использовалась в большом количестве примеров:

для определения типа данных при работе с необобщенными коллекциями (работа с типами данных);

для сортировки коллекций при проверке того, что тип реализует интерфейс IComparable (работа с типами данных);

заявляют, что самоотображаемость планируется реализовать в отдаленной перспективе.

Стоит отметить, что самоотображаемость может быть реализована поверх языка с использованием библиотек для работы с AST (Abstract Syntax Tree — абстрактное синтаксическое дерево). AST дает возможность представить текст программы в виде дерева, в котором внутренние вершины соответствуют операторам языка программирования или функциям, а листья — операндам. Таким образом, AST позволяет представить текст программы в виде структуры данных (дерева) и фактически реализует принцип самоотображаемости. В отличие от получения команд из откомпилированной сборки с использованием рефлексии, AST-дерево строится по исходному тексту программу. В среде .NET работа с AST-деревом для прикладных программистов реализуется в рамках проекта модульного компилятора Roslyn.

В настоящее время наиболее известным языком, имеющим поддержку самоотображаемости, является Lisp, у которого существует большое количество диалектов и который исторически был одним из первых языков программирования. В Lisp программа представляется в виде иерархического списка (программа фактически и есть AST), и каждая программа может быть обработана другой программой. На платформе .NET реализован один из современных диалектов Lisp — clojure.

К самоотображаемым языкам также относятся Elixir (синтаксис которого похож на Ruby, но при этом реализована самоотображаемость в стиле Lisp) и Prolog (специализированный язык логического программирования).

Самоотображаемость на уровне команд языка ближе к интерпретируемым языкам, которые интерпретируются и выполняются на уровне исходного кода. Изначально Lisp был интерпретируемым языком. Самоотображаемость для компилируемого языка требует разработки сложного специализированного компилятора. В настоящее время проводятся активные работы по разработке самоотображаемых языков на основе интерпретируемого языка JavaScript.

Контрольные вопросы

1. Что такое рефлексия?
2. Как реализуется работа со сборками?

3. Как получить детальную информацию о типах данных с использованием рефлексии?

4. Как выполняются динамические действия с объектами классов?

5. Как обеспечивается работа с атрибутами?

6. Как можно просмотреть содержимое откомпилированной сборки?

7. Что такое самоотображаемость в языках программирования?

9. Параллельная обработка данных

В любом языке программирования параллельное программирование традиционно считается наиболее сложной темой. При параллельной обработке могут возникать сложные и плохо отлаживаемые ошибки, связанные с одновременной записью в один и тот же набор данных, одновременным обращением к ресурсам и др. Такие сложные случаи здесь не приведены.

В этом разделе рассмотрен наиболее простой вариант параллельной обработки данных, когда в однородном массиве данных необходимо провести поиск. В этом случае можно разделить массив данных на несколько фрагментов и вести поиск параллельно.

Параллельная обработка данных рассматривается в данном разделе на основе фрагментов **примера 15**.

В языке C# существуют три вида классов для параллельного запуска потоков:

Thread (поток),

ThreadPool (пул потоков),

Task (задача).

Класс Thread является исторически первым средством для многопоточной работы в языке C#. Он представляет собой практически полную копию такого же класса в языке Java. Основная проблема класса Thread в языке C# — медленный запуск потока, на его создание и запуск уходит довольно много времени.

Для решения этой проблемы был разработан класс ThreadPool, содержащий пул уже запущенных потоков, которые при запуске начинают выполняться без задержки. Проблема класса ThreadPool — не очень удобный для прикладного программиста набор методов для работы с потоками.

Поскольку метод DelayAsync объявлен как асинхронный, он будет запущен и управление сразу будет передано следующей команде — Console.WriteLine.

Далее в методе DelayAsync будет вызван метод GetTaskAsync. Он реализует задержку в 3 с и возвращает строковый результат, выводимый после нее.

Таким образом, команда Console.WriteLine будет действительно выполнена до завершения асинхронного метода DelayAsync, и сообщения в консоль будут выведены в следующем порядке:

```
Перед вызовом DelayAsync
После задержки в 3 секунды.
```

Контрольные вопросы

1. Как реализовать запуск параллельных потоков на основе класса Thread?
2. Как реализовать запуск параллельных потоков на основе класса Task?
3. Как получить результат выполнения потока с использованием класса Task?
4. Как реализовать ожидание завершения массива потоков с помощью класса Task?
5. Как измерить время выполнения программы с использованием класса Stopwatch?
6. Как применяются конструкции async и await?

10. Реализация алгоритма поиска с опечатками

Данное учебное пособие предназначено прежде всего для того, чтобы помочь в изучении языка программирования C#, и в нем не ставится цель изучения алгоритмов.

Но в качестве учебной задачи здесь будет рассмотрен один достаточно сложный алгоритм, довольно часто используемый при разработке информационных систем. Это алгоритм поиска в текстовой строке с опечатками.

Практически любая крупная информационная система содержит такие данные, как фамилии, имена, отчества, географические

Контрольные вопросы

1. Что такое расстояние Левенштейна?
2. Что такое расстояние Дамерау — Левенштейна?
3. Объясните алгоритм Вагнера — Фишера для вычисления расстояния Дамерау — Левенштейна.

11. Основы разработки пользовательского интерфейса с использованием технологии Windows Forms

В предыдущих разделах были рассмотрены примеры создания консольных приложений. В данном разделе приведены основы создания оконных приложений на основе фрагментов **примера 17**.

11.1. Создание проекта

Для создания оконного приложения необходимо реализовать в Visual Studio новый проект типа «приложение Windows Forms» (рис. 28).

После этого открывается не окно кода, как было ранее в случае консольных приложений, а редактор макета формы, и автоматически создается форма Form1.

При нажатии на вкладку «Панель элементов», расположенную в левой части окна, открывается панель, содержащая список тех элементов, которые можно перетащить на форму мышью (рис. 29).

Все элементы делятся на визуальные и невизуальные. Визуальные отображаются на форме в виде элементов управления. Это такие элементы, как кнопка (Button), поле ввода (TextBox), текстовая надпись (Label) и др.

Невизуальные элементы также можно перетащить на форму, однако они отображаются в специальной области редактора макета формы. Примером такого элемента является таймер (Timer).

Если какая-либо из панелей не видна в текущей настройке Visual Studio, то ее можно включить в пункте меню «Вид» (в англоязычной версии — View).

Контрольные вопросы

1. Как создать приложение Windows Forms?
2. Как используется элемент Button?
3. Как применяется элемент TextBox?
4. Как используется элемент Label?
5. Как применяется элемент Timer?
6. Как задать форму, которая открывается при запуске приложения?
7. Как открыть модальное окно?
8. Как открыть немодальное окно?
9. Как закрыть окно и приложение?
10. Для чего используются события FormClosed и FormClosing?

12. Пример многопоточного поиска в текстовом файле с использованием технологии Windows Forms

В данном разделе будет реализован комплексный пример, который обобщает предыдущие примеры.

Пример, реализованный с использованием технологии Windows Forms, выполняет следующие действия:

• запрашивает имя текстового файла для поиска с использованием стандартного диалога открытия файла;

• вводит слово для поиска;

• реализует четкий (без учета опечаток) поиск слова в файле; найденное слово, которое может встретиться в файле несколько раз, выводится в результирующий список;

• реализует нечеткий (на основе расстояния Дамерау — Левенштейна) многопоточный поиск в файле, причем вводится максимальное расстояние для нечеткого поиска, на которое искомое слово может отличаться от слова в файле. Также вводится количество потоков, на которое разделяется массив слов исходного файла. Найденное слово, которое может встретиться в файле несколько раз, выводится в результирующий список с указанием номера потока и вычисленного расстояния Дамерау — Левенштейна;

```
<li>языков(расстояние=2 поток=0)</li>
<li>Язык(расстояние=0 поток=1)</li>
<li>языка(расстояние=1 поток=1)</li>
<li>языкам(расстояние=2 поток=2)</li>
<li>языки(расстояние=1 поток=2)</li>
<li>Языка(расстояние=1 поток=2)</li>
<li>языках(расстояние=2 поток=4)</li>
<li>языку:(расстояние=2 поток=5)</li>
<li>языке(расстояние=1 поток=5)</li>
<li>"Язык(расстояние=1 поток=7)</li>
<li>языком(расстояние=2 поток=7)</li>
<li>Языки(расстояние=1 поток=8)</li>
<li>СЯзыки(расстояние=2 поток=8)</li>
</ul>
</td>
</tr>
</table>
</body>
</html>
```

Контрольные вопросы

1. Как используется класс OpenFileDialog?
2. Как применяется класс SaveFileDialog?
3. Как используется класс MessageBox?
4. Как используется элемент ListBox?
5. Как сформировать отчет с помощью класса StringBuilder и сохранить его в виде текстового файла?

Литература

1. *Нейгел К.*, *Ивьен Б.*, *Глинн Д.*, *Уотсон К.* C# 5.0 и платформа .NET 4.5 для профессионалов. М.: ООО «И.Д. Вильямс», 2014. 1440 с.

2. *Шилдт Г.* C# 4.0: полное руководство. М.: ООО «И.Д. Вильямс», 2013. 1056 с.

3. *Павловская Т.А.* C#. Программирование на языке высокого уровня: учебник для вузов. СПб.: Питер, 2015. 432 с.

4. *Левенштейн В.И.* Двоичные коды с исправлением выпадений, вставок и замещений символов // Доклады Академии наук СССР. 1965. № 4. С. 845–848.

Содержание

Учебное издание

Самохвалов Эдуард Николаевич
Ревунков Георгий Иванович
Гапанюк Юрий Евгеньевич

Введение в проектирование и разработку приложений на языке программирования C#

Художник *Я.М. Асинкритова*
Корректор *О.В. Новикова*
Компьютерная графика *О.В. Левашовой*
Компьютерная верстка *Н.Ф. Бердавцевой*

Оригинал-макет подготовлен
в Издательстве МГТУ им. Н.Э. Баумана.

В оформлении использованы шрифты
Студии Артемия Лебедева.

Подписано в печать 25.12.2017. Формат 60×90/16.
Усл. печ. л. 15,5. Изд. № 543-2015.
Тираж 100 экз. Заказ

Издательство МГТУ им. Н.Э. Баумана.
105005, Москва, 2-я Бауманская ул., д. 5, стр. 1.
press@bmstu.ru
www.baumanpress.ru

Отпечатано в типографии МГТУ им. Н.Э. Баумана.
105005, Москва, 2-я Бауманская ул., д. 5, стр. 1.
baumanprint@gmail.com