ПРОЕКТ ПРОГРАММЫ ДИСЦИПЛИНЫ

Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Новосибирский национальный исследовательский государственный университет»
(Новосибирский государственный университет, НГУ)

Факультет информационных технологий
Кафедра Общей информатики

Рабочая программа дисциплины
«Введение в C# и платформу .NET»

Направление подготовки
230100 «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА»

Квалификация (степень) выпускника
Бакалавр

Форма обучения
Очная

Новосибирск
2013

Программа дисциплины «Введение в C# и платформу .NET» составлена в соответствии с требованиями ФГОС ВПО к структуре и результатам освоения основных образовательных программ бакалавриата по профессиональному циклу по направлению подготовки «Информатика и вычислительная техника», а также задачами, стоящими перед Новосибирским государственным университетом по реализации Программы развития НГУ.

Автор:
Факторович С.Б.
Факультет информационных технологий
Кафедра Общей информатики

Разработка подготовлена в рамках реализации Программы развития НИУ НГУ
на 2009–2018 гг

1. Цели и задачи курса

Целью курса «Введение в C# и платформу .NET» является подготовка специалистов по разработке крупных программных систем с помощью платформы .NET и языка C#.

Данная цель в полной мере отвечает основной цели программы бакалавриата, а именно подготовке выпускников к решению следующих профессиональных задач:

- Проектирование программных и аппаратных средств (систем, устройств, деталей, программ, баз данных и т.п.) в соответствии с техническим заданием с использованием средств автоматизации проектирования.
- Применение современных инструментальных средств при разработке программного обеспечения.
- Применение Web-технологий при реализации удаленного доступа в системах клиент/сервер и распределенных вычислений.
- Использование стандартов и типовых методов контроля и оценки качества программной продукции.
- Освоение и применение современных программно-методических комплексов исследования и автоматизированного проектирования объектов профессиональной деятельности.

Кроме того, курс способствует развитию у студентов необходимых личностных качеств и формированию общекультурных и профессиональных компетенций в соответствии с ФГОС ВПО по данному направлению подготовки

Для достижения поставленных целей выделяются следующие **задачи курса**:

- Ознакомление студентов с языком C# и платформой .NET
- Изложение общепринятых практик разработки программного обеспечения с помощью языка C#
- Сравнение C# с другими языками программирования

2. Место курса в структуре образовательной программы

Дисциплина «Введение в C# и платформу .NET» входит в вариативную часть профессионального цикла образовательной программы бакалавриата направления подготовки 230100 «ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА».

Перед началом изучения дисциплины студент должен:

Знать:

- Основные парадигмы программирования
- Общие принципы ООП и ООАД

Уметь:

- Разрабатывать программы под ОС Windows

Владеть:

- По крайней мере одним объектно-ориентированным языком программирования (например, Java или C++)
- Средствами отладки кода

3. Компетенции учащегося, формируемые в результате освоения дисциплины

Изучение дисциплины направлено на формирование следующих общекультурных и профессиональных компетенций:

- готов к кооперации с коллегами, работе в коллективе (ОК-3);
- способен осваивать методики использования программных средств для решения практических задач (ПК-2);
- способен разрабатывать компоненты программных комплексов и баз данных, использовать современные инструментальные средства и технологии программирования (ПК-5);

В результате освоения дисциплины студент должен:

Знать:

- Преимущества и недостатки JIT-компилируемых языков
- Основные синтаксические особенности C#

- Принципы работы CLR и отличия CLR от других виртуальных машин
-

Уметь:

- Разрабатывать и поддерживать проекты на основе C#

Владеть:

- Средствами ООП в C#
- Средствами разработки и рефакторинга кода на C#
- Общепринятыми подходами к разработке коммерческих проектов с использованием C#

4. Структура и содержание дисциплины

Курс «Управление знаниями и ведение документации в IT-проектах» рассчитан на один семестр. Общая аудиторная нагрузка составляет 48 аудиторных часов. В течение семестра лекции и семинары чередуются: сразу после освещения какой-либо темы на лекциях проводятся семинары для закрепления этой темы. Кроме этого, курс включает в себя 96 часов самостоятельной работы студентов (чтение дополнительных материалов, выполнение домашних заданий, и пр.)

Общая трудоемкость курса составляет 4 зачетных единицы или 144 часа.

По завершению курса студентам выставляется дифференцированный зачет. Итоговая оценка за семестр определяется двумя параметрами: оценкой за практические задания и оценкой за реферат, сдаваемый в конце семестра.

№ п/п	Разделы курса	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)				Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)
			Лекции	Семинары	Консультации	Самостоятельная работа	Контрольные работы, проверка домашних зданий, проверка семестрового задания

	Введение. История языка, отличительные особенности, применение в индустрии. Основные языковые конструкции. Пример кросс-платформенного исполняемого файла и его примерный формат. Связка CLR, CIL, CTS и CLS. Сборка программ на C#: Visual Studio, использование csc из коммандной строки, Mono.	1	2	-	-	10	
	Синтаксис языка. Пространства имен (namespaces). Классы и методы, модификаторы const и static.	2	2	2	-	10	
	Ссылочные типы (reference types) и типы-значения (value types). Ссылочная семантика. Оператор == и метод object.Equals(). Передача параметров в функцию. Ключевые слова ref, out и params. Класс String.	3-4	2	2	-	10	
	Массивы, ключевое слово foreach. Перечисления (enums).	5	0	4	-	10	-
	Наследование. Интерфейсы,	6-7	2	4		10	

	абстрактные классы. Ключевые слова interface, abstract, virtual, sealed, override, new.						
	Перегрузка операторов. Арифметические операторы и индексеры (indexers). Ключевое слово foreach и метод object.GetEnumerator(). Операторы приведения типов, ключевые слова implicit и explicit.	8-9	2	2	-	10	Проверка домашнего задания
	Исключения (exceptions). Делегаты. Частичные (partial) типы и методы. Ключевое слово using и интерфейс IDisposable.	10-11	2	4	-	10	Проверка домашнего задания
	Потоки (threads). Примитивы синхронизации в стандартной библиотеке. Ключевое слово lock. Асинхронные делегаты.	12	2	4	-	16	
	Шаблонные типы (generics). Ключевые слова where и default. Пространство имен System.Collections.Generic. Часто используемые	13	2	4		10	

	структуры данных.						
	Прослушивание докладов студентов	14-16	-	6	-	0	Сдача и представление рефератов
							Дифференцированный зачет

4.1. Содержание курса

Тема 1. Введение.

Введение. История языка, отличительные особенности, применение в индустрии. Основные языковые конструкции. Пример кросс-платформенного исполняемого файла и его примерный формат. Связка CLR, CIL, CTS и CLS. Сборка программ на C#: Visual Studio, использование csc из коммандной строки, Mono.

Тема 2. Синтаксис C#

Синтаксис языка. Пространства имен (namespaces). Классы и методы, модификаторы const и static.

Тема 3. Ссылочные типы и типы-значения

Ссылочные типы (reference types) и типы-значения (value types). Ссылочная семантика. Оператор == и метод object.Equals(). Передача параметров в функцию. Ключевые слова ref, out и params. Класс String.

Тема 4. Массивы, коллекции

Массивы, ключевое слово foreach. Перечисления (enums).

Тема 5. Элементы ООП

Наследование. Интерфейсы, абстрактные классы. Ключевые слова interface, abstract, virtual, sealed, override, new.

Тема 6. Приведение типов, перегрузка операторов

Перегрузка операторов. Арифметические операторы и индексеры (indexers). Ключевое слово foreach и метод object.GetEnumerator(). Операторы приведения типов, ключевые слова implicit и explicit.

Тема 7. Прочие особенности языка

Исключения (exceptions). Делегаты. Частичные (partial) типы и методы. Ключевое слово using и интерфейс IDisposable.

Тема 8. Сборка мусора

Сборщик мусора (garbage collector) в .NET. Поколения. Финализаторы (finalizers) и метод object.Finalize().

Тема 9. Многопоточность

Потоки (threads). Примитивы синхронизации в стандартной библиотеке. Ключевое слово lock. Асинхронные делегаты.

Тема 10. Шаблонные типы

Шаблонные типы (generics). Ключевые слова where и default. Пространство имен System.Collections.Generic. Часто используемые структуры данных.

5. Образовательные технологии

При проведении курса широко используются активные и интерактивные формы проведения занятий, в том числе:

- коллективная работа,
- дискуссии,
- практические задания
- элементы дистанционной поддержки обучения.

Процесс обучения представлен как взаимодействие системы лекций с семинарской, практической и самостоятельной работой студентов.

6. Оценочные средства для текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины и учебно-методическое обеспечение самостоятельной работы студентов

6.1. Методические указания по самостоятельной работе студентов

Курс предполагает следующие виды самостоятельной работы студентов:

1. Выполнение практических заданий, выданных преподавателем на семинарах
2. Подготовка рефератов
3. Самостоятельное изучение дополнительного материала по темам, озвученным преподавателем

Итоговая оценка за курс ставится с учетом оценки за практические задания и за представление реферата.

6.2. Критерии оценки практических заданий

Каждое практическое задание выполняется студентом самостоятельно и затем сдается преподавателю на семинаре.

По согласованию с преподавателем, студенты могут объединяться в группы для решения практических задач.

Каждому практическому заданию сопоставлено максимальное количество баллов, которое можно за него получить. Преподаватель оценивает сданное задание в соответствии с критериями его приема (см. ниже).

В данной таблице приводятся основные критерии оценки работы студента, а также процентный вес каждого критерия. При несоответствии сданного задания одному из критериев сумма полученных баллов уменьшается на указанный процент относительно максимально возможного количества баллов.

Критерий	Процентный коэффициент
1. Функциональная корректность решения (например, программа должна успешно проходить набор тестов, составленный преподавателем)	40%
2. Стабильность архитектуры решения и ее соответствие принципам ООД	30%
3. Следование рекомендациям к стилю кодирования и использованию паттернов (рекомендации выдаются преподавателем)	20%
4. Следование рекомендациям к оформлению и форматирования кода (рекомендации выдаются преподавателем)	10%
Итого, максимально за каждое задание	**100%**

Итоговая оценка за практические задания ставится по сумме набранных баллов за весь семестр (из расчета 4 практических заданий и максимальной суммы в 30 баллов):

- Менее 10 баллов: «неудовлетворительно»
- 10-14 баллов: «удовлетворительно»
- 15-24 баллов: «хорошо»
- 25-30 баллов: «отлично»

6.3. Рефераты

Финальная форма аттестации студентов по курсу — сдача и защита рефератов. Рефераты представляются студентами в конце семестра, после окончания серии лекций.

Темы рефератов

1. RTTI и рефлексия (reflection)
2. Сборки (assemblies) и GAC
3. Обфускация кода и рефлекторы (.NET reflectors).
4. Технология LINQ и привнесенные ей добавления в язык: лямбда-функции, анонимные типы, методы-расширения (extension methods), инициализаторы классов (class initializers)
5. Технология LINQ и ее связь с реляционными базами данных
6. Обзор ASP и ASP.NET
7. Обзор Silverlight
8. Обзор WPF
9. Обзор разработки под Windows Phone
10. Интеграция с unmanaged-кодом

6.4. Практические задания

Ниже приведены примеры практических заданий по курсу

1. Счетчик строк в проекте

Command-line утилита для подсчета строк кода в проекте, отфильтровывающая пустые строки и комментарии. Опционально — сделать архитектуру плагинов, позволяющую описывать структуру комментариев для произвольного языка.

Программа должна принимать параметром командной строки тип учитываемых файлов (например, *.cs). После запуска она должна обходить текущую директорию и все вложенные директории, искать в них файлы заданного типа и подсчитывать в них

количество осмысленных строк. Суммарное количество строк во всех учтенных файлах должно быть выведено на экран.

Количество баллов: 5

Ключевые слова: работа с файловой системой; для системы плагинов — рефлексия или наследование

2. JSON-сериализатор

Написать свой JSON-сериализатор для произвольных объектов.

Должна быть возможность корректно сериализовать, например, объект такого класса:

```
[Serializable]
class TestClass
{
    public int i;
    public string s;

    [NonSerializable]
    public string ignore; // это поле не должно сериализоваться

    public int[] arrayMember;

}
```

и получить на выходе что-нибудь вроде

```
{
    "i": 25,
    "s": "Hello world!",
    "arrayMember": [1, 2, 3, 4, 5]
}
```

Ваш сериализатор должен ориентироваться на атрибуты System.SerializableAttribute и System.NonSerializableAttribute.

Подробнее о формате JSON можно прочитать здесь: http://en.wikipedia.org/wiki/JSON

Количество баллов: 5

Ключевые слова: атрибуты, рефлексия

3. RSS2Email

Программа должна периодически проверять какой-нибудь RSS-поток и пересылать все новые записи в нём по какому-либо email-адресу. Опционально можно оформить программу в виде win32-сервиса.

Все инструменты для парсинга XML, отправки почты и осуществления HTTP-запросов есть в стандартной библиотеке .NET.

Подсказка: самый простой RSS-поток, который можно обновлять самому — лента ваших твитов на twitter.com. Еще можно попробовать Tumblr и LiveJournal.

Количество баллов: 10

Ключевые слова: XML, SMTP, HTTP, многопоточность

4. Task scheduling library

Написать собственный планировщик задач, наподобие библиотеки Quartz.NET (http://quartznet.sourceforge.net). Ваша библиотека должна уметь выполнять произвольные методы по расписанию (например, каждые N минут) и с задержкой (например, выполнить это метод ровно через M секунд). Опционально — добавить поддержку расписаний в стиле crontab (http://en.wikipedia.org/wiki/Cron#Examples).

Подсказка: интерфейс библиотеки может выглядеть следующим образом:

```
interface IJob
{
           void Execute(object argument);
}

public void ScheduleDelayedJob(IJob job, TimeSpan delay);
public void SchedulePeriodicJob(IJob job, TimeSpan period);
public void SchedulePeriodicJob(IJob job, string cronExpression);
```

Количество баллов: 10

Ключевые слова: многопоточность, межпоточное взаимодействие

6.5. Дополнительные варианты практических заданий

Задача №1: Hello World Revenge

Имеется база данных под управлением MS SQL 2008. В ней находится единственная таблица с полями (Id, Value). В таблице единственное значение (1, "Hello World Revenge").

Необходимо реализовать WCF-сервис, который получит доступ к этой БД через LINQ2SQL, вытянет эту строчку, отрендерит из неё картинку (обычный PNG, в котором на белом фоне будет написана строчка произвольно выбранным шрифтом), после чего вернет ее как поток.

Создать WPF-графическое приложение, в котором будет ссылка этот сервис, которое по нажатию кнопки будет обращаться к сервису, получать картинку и отображать ее на экране.

Командность: задача не командная

Задача №2: Склад документов

Некоторой организации понадобилась система работы с документами: заявлениями, квитанциями и прочими.

Основной use-case: пользователь может зайти на страничку, увидеть список документов, которые уже есть в системе, скачать каждый документ в виде PDF, картинки или HTML.

Второй основной use-case: пользователь может выбрать какой документ ему создать (из списка всех доступных документов), ввести данные необходимых полей и сохранить документ в системе, после чего он появится в списке и его можно будет скачать, как и все остальные.

Узкий момент в том, что заказчик еще сам не знает, какие документы будут в системе, так что архитектурно надо предусмотреть возможность быстро добавить поддержку еще одного документа.

Программисты решили сделать сначала прототип без авторизации пользователей и редактирования документов, а так же без дизайна. В качестве наиболее подходящего технологического стека были выбраны ASP.NET MVC, MongoDB и Stimulsoft Reports. Для фронтенда планируется использовать Twitter Bootstrap и jQuery.

Подумав, программисты решили отдать прототип данного проекта на реализацию студенту ФИТ, т.к. задача довольно простая.

Помогите этой команде программистов.

Ключевые слова: ASP.NET MVC, работа с документно-ориентированными БД, практика применения атрибутов, практика reflection и написания своих атрибутов, ASHX

Командность: может быть выполнена в команде

Задача №3: Заглушки

Некому брокеру очень хочется разработать dashboard для своих клиентов, где они смогут увидеть свои активы, котировки, посмотреть телефон консультанта, адрес отделения на карте, историю сделок за определенные периоды и прочие статистические данные. Брокер заботливо подготовил мокапы web-интерфейса, однако не спешит говорить, откуда разработчикам брать данные для отображения. Говорил что-то про web-сервисы, но обсуждение и согласование подвисло на стадии определения источников данных на стороне заказчика. Несмотря на это печальное обстоятельство, клиенту уже хочется видеть схематичный интерфейс (Twitter Bootstrap) с тестовыми данными. Менеджмент проекта принимает очевидное решение сделать слой доступа к данным на заглушках, возвращающих тестовые данные, в надежде позже просто подменить реализации. Делать это они решили посредством IoC-контейнера (Unity Application Block, при том обязательно на xml-конфигурации). Реализуйте пожелания менеджмента.

Ключевые слова: ASP.NET MVC, n-tier архитектура, разработка с использованием IoC, Unity

Командность: может быть выполнена в команде

6.6. Контрольные вопросы

В некоторых случаях (например, для уточнения или повышения оценки студента) преподаватель может предложить студенту ответить на несколько контрольных вопросов. Список таких вопросов приведен ниже.

1. Какие из следующих выражений объявляют переменные значимых типов (value type)?

```
int x = 10;
string s = "строка";
object o = 20;
int y = new int();
int? z = null;
object o2 = y;
object o3 = new object();
```

2. Что выведет представленный ниже код?

```
static class Program
{
    static const int NaturalMin = 1;
    static bool IsNatural(int x)
    {
        Console.Write("IsNatural({0}) ", x);
        return x >= NaturalMin;
    }

    static readonly string EmptyString = "";
    static bool IsEmpty(string s)
    {
        Console.Write("IsEmpty({0}) ", s);
        return s == EmptyString;
    }

    static void Main(string[] args)
    {
        int x = 10;
        string s = "";
        Console.WriteLine(IsNatural(x) || IsEmpty(s));
    }
}
```

3. Что выведет представленный ниже код?

```
struct Sample
    {
        public int i;
    }
    class MyProgram
    {
        static void Main()
        {
            Sample x = new Sample();
            x.i = 10;
            fun(x);
            Console.Write(x.i + " ");
        }
        static void fun(Sample y)
        {
            y.i = 20;
            Console.Write(y.i + " ");
        }
    }
```

4. Что выведет представленный ниже код?

```
struct Sample
   {
       public int i;
   }
   class MyProgram
   {
       static void Main(string[] args)
       {
           Sample x = new Sample();
           x.i = 10;
           fun(ref x);
           Console.Write(x.i + " ");
       }
       public static void fun(ref Sample y)
       {
           y.i = 20;
           Console.Write(y.i + " ");
       }
   }
```

5. Что выведет представленный ниже код?

```
public class A
    {
        public A()
        {
            Console.WriteLine("1");
            Print();
        }

        public virtual void Print()
        {
            Console.WriteLine("2");
        }
    }

    public class B: A
    {
        public B()
        {
            Console.WriteLine("3");
            Print();
        }

        public new void Print()
        {
            Console.WriteLine("4");
        }
    }

    static class Program
    {
        static void Main(string[] args)
        {
            A b = new B();
            b.Print();
        }
    }
```

6. Оцените неэффективность приведенного кода. Сколько лишних объектов он создаст? Как бы вы переписали этот код?

```
static void Main(string[] args)
{
      string text = "Hello world!";

      string reverse = string.Empty;

      foreach (char c in text)
      {

          reverse = c + reverse; // concantenate the characters in a reverse mode

      }
      Console.WriteLine(reverse);
}
```

7. Что произойдет при компиляции и выполнении следующего кода?

```
public class Generic<T>
{
    public T Field;
    public void TestSub()
    {
        T i = Field + 1;
    }
}
class MyProgram
{
    static void Main(string[] args)
    {
        Generic<int> gen = new Generic<int>();
        gen.TestSub();
    }
}
```

8. Какие преимущества и недостатки вы видите в следующих пяти реализациях паттерна singleton?

```
public sealed class Singleton
{
    private static Singleton instance=null;

    private Singleton()
    {
    }

    public static Singleton Instance
    {
        get
        {
            if (instance==null)
            {
                instance = new Singleton();
            }
            return instance;
        }
    }
}
```

```
public sealed class Singleton
{
    private static Singleton instance = null;
    private static readonly object padlock = new object();

    Singleton()
    {
    }

    public static Singleton Instance
    {
        get
        {
            lock (padlock)
            {
                if (instance == null)
                {
                    instance = new Singleton();
                }
                return instance;
            }
        }
    }
}
```

```
public sealed class Singleton
{
    private static Singleton instance = null;
    private static readonly object padlock = new object();

    Singleton()
    {
    }

    public static Singleton Instance
    {
        get
        {
            if (instance == null)
            {
                lock (padlock)
                {
                    if (instance == null)
                    {
                        instance = new Singleton();
                    }
                }
            }
            return instance;
        }
    }
}
```

```
public sealed class Singleton
{
    private static readonly Singleton instance = new Singleton();

    static Singleton()
    {
    }

    private Singleton()
    {
    }

    public static Singleton Instance
    {
        get
        {
            return instance;
        }
    }
}
```

```
public sealed class Singleton
{
    private Singleton()
    {
    }

    public static Singleton Instance { get { return Nested.instance; } }

    private class Nested
    {
        static Nested()
        {
        }

        internal static readonly Singleton instance = new Singleton();
    }
}
```

9. Не компилируя этот код, скажите, вызовет ли он ошибки или warning-и при компиляции. Если да, то какие именно?

```
using System;
    namespace Polymorphism
    {
        class A
        {
              public void Foo() { Console.WriteLine("A::Foo()"); }
        }

        class B : A
        {
              public void Foo() { Console.WriteLine("B::Foo()"); }
        }

        class Test
        {
            static void Main(string[] args)
            {
                A a;
                B b;

                a = new A();
                b = new B();
                a.Foo();  // output --> "A::Foo()"
                b.Foo();  // output --> "B::Foo()"

                a = new B();
                a.Foo();  // output --> "A::Foo()"
            }
        }
    }
```

10. Сколько байт памяти занимает объект класса Derived?

```
namespace Test
{
    class Baseclass
    {
        private int i;
        protected int j;
        public int k;
    }
    class Derived: Baseclass
    {
        private int x;
        protected int y;
        public int z;
    }
    class MyProgram
    {
        static void Main (string[ ] args)
        {
            Derived d = new Derived();
        }
    }
}
```

11. Какие ошибки вы видите в этом коде?

```
switch (s) {
   case "a":
      try {
         HitA();
      }
      catch (Exception e) {
         throw e;
      }
      break;
   case "b":
      try {
         HitB();
      }
      catch (Exception e) {
         throw e;
      }
      break;
}
```

12. Какие из следующих участков кода открывают файл для записи?

```
File.Open("somefile.txt", FileMode.Create);
```

```
File.Open("somefile.txt", FileMode.Create, FileAccess.Write);
```

```
File.Open("somefile.txt", FileMode.Create, FileAccess.Read);
```

```
FileInfo file = new FileInfo("somefile.txt");
file.Open(FileMode.Create);
```

13. Какие из следующих участков кода запускает сборку, скачанную из Интернета?

```
object [] hostEvidence = {new Zone(SecurityZone.Internet)};
Evidence e = new Evidence(hostEvidence, null);
AppDomain d = AppDomain.CreateDomain("MyDomain", e);
d.ExecuteAssembly("Assembly.exe");
```

```
object [] hostEvidence = {new Zone(SecurityZone.Internet)};
AppDomain d = AppDomain.CreateDomain("MyDomain");
Evidence e = new Evidence(hostEvidence, null);
d.Evidence = e;
d.ExecuteAssembly("Assembly.exe");
```

```
AppDomain myDomain = AppDomain.CreateDomain("MyDomain");
myDomain.ExecuteAssembly("Assembly.exe", new Zone(SecurityZone.Internet));
```

```
Evidence e = new Evidence();
e.AddHost(new Zone(SecurityZone.Internet));
AppDomain myDomain = AppDomain.CreateDomain("MyDomain");
myDomain.ExecuteAssembly("Assembly.exe", e);
```

7. Учебно-методическое и информационное обеспечение дисциплины

Основная литература:

1. Inside C#, Microsoft Press, 2002.
2. Troelsen, Andrew. Pro C# 2008 and the .NET 3.5 Platform, Fourth Edition, Apress, 2008.
3. Д. Рихтер. Программирование на платформе .NET Framework 2.0 на языке C#, СПб.: Питер, 2007

Дополнительная литература

1. Troelsen, Andrew. Pro C# 4.0 and the .NET 4.5 Platform, 6th Edition, Apress, 2012
2. Skeet, John. C# in Depth. Manning Publications, 2013
3. Albahari, Joseph. C# 5.0 in a Nutshell: The Definitive Reference. O'Reilly Media, 2012

8. Материально-техническое обеспечение курса

Для лекций:

- Аудиторный класс
- Доступ в Интернет
- Ноутбук, мультимедиа-проектор, экран для показа презентаций

- Программное обеспечение: Microsoft PowerPoint (или PowerPoint Viewer), Adobe Reader

Для семинаров:

- Компьютерный класс
- Доступ в Интернет
- Мультимедиа-проектор, экран для показа презентаций
- Программное обеспечение на компьютере преподавателя: Microsoft PowerPoint (или PowerPoint Viewer), Adobe Reader
- Программное обеспечение на компьютерах студентов: Microsoft Visual Studio 2010.

Рецензент (ы) _____

Программа одобрена на заседании Методической комиссии ФИТ

от _____ года, протокол № _____.