Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

Факультет математики и технологий программирования
Кафедра вычислительной математики и программирования

СОГЛАСОВАНО

Заведующий кафедрой

_____ Д.С. Кузьменков

23.02 2022 г.

СОГЛАСОВАНО

Декан факультета

_____ С.П. Жогаль

23.02 2022 г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО ДИСЦИПЛИНЕ

«ИНСТРУМЕНТЫ И СРЕДСТВА ПРОГРАММИРОВАНИЯ»

для специальности
1-40 04 01 «Информатика и технологии программирования»

Составитель:
М.В. Москалева, старший преподаватель

Рассмотрено и утверждено
на заседании кафедры
вычислительной математики
и программирования

23.02.2022
Протокол №_8_

Рассмотрено и утверждено
на заседании научно - методического совета университета
02.03 2022 г., протокол № _3_

02 Содержание учебно-методического комплекса по дисциплине «Инструменты и средства программирования»

01 Титульный лист.
02 Содержание.
03 Пояснительная записка.
1 Теоретический раздел.
- 1.1 Перечень теоретического материала.
- 1.2 Материалы для обеспечения самостоятельной учебной работы студентов.

2 Практический раздел.
- 2.1 Перечень лабораторных работ.
- 2.2 Задания для лабораторных работ.

3 Контроль знаний.
- 3.1 Перечень вопросов к зачету.
- 3.2 Образец тестовых заданий по дисциплине.

4 Вспомогательный раздел.
- 4.1 Учебная программа дисциплины.
- 4.2 Перечень рекомендуемой литературы.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к учебно-методическому комплексу по дисциплине «Инструменты и средства программирования» для специальности 1-40 04 01 «Информатика и технологии программирования»

Учебно-методический комплекс дисциплины «Инструменты и средства программирования» составлен в соответствии с учебным планом учреждения образования для специальности 1-40 04 01 «Информатика и технологии программирования», утвержденным 29.08.2013 (регистрационный номер I-40-02-13), учебной программой учреждения высшего образования, утвержденной 07.06.2017 (регистрационный номер УД-13-2017-50/уч.).

Данные документы предусматривают изучение студентами теоретических и практических основ разработки приложений с применением современной платформы .Net. В них заложена задача формирования у будущих специалистов профессиональных и личностных компетенций в области современных информационных технологий.

Целью учебно-методического комплекса «Инструменты и средства программирования» является овладение студентами основами современных компьютерных технологий и программного обеспечения, изучение возможностей разработки приложений с применением современной платформы .NET на языке программирования C#.

Задачами учебно-методического комплекса являются:

– ознакомление студентов с основами проектирования приложений с использованием новых технологий и языков программирования;

– усвоение основных методов разработки алгоритмов;

– совершенствование умений и навыков программирования на языке C#;

– формирование умений и навыков по созданию веб-сайтов и веб-приложений с применением современной платформы ASP.NET Core.

Материал дисциплины и соответствующего учебно-методического комплекса базируется на ранее полученных студентами знаниях по таким дисциплинам, как «Основы алгоритмизации и программирования», «Языки программирования», «Современные платформы программирования».

В структурном отношении учебно-методический комплекс «Инструменты и средства программирования» включает в себя четыре раздела: теоретический, практический, раздел контроля знаний, вспомогательный.

Теоретический раздел содержит основные положения, выносимые на лекции, и включающий в себя 12 тем, предназначенных для аудиторной работы со студентами (лекции преподавателя – 22 часа) и самостоятельного изучения тем, вынесенных за рамки аудиторных часов (УСР – 10 часов). Посредством этих тем студенты смогут получить знания по основам

программирования на языке C#, использованию технологий ADO.Net и Entity Framework Core для работы с базой данных, разработке графических приложений с использованием WPF, разработке веб-сайтов и веб-приложений с использованием ASP.NET Core. Материал к лекциям позволяет также сориентировать студентов в сфере разработки приложений различного уровня сложности.

Практический раздел включает в себя в соответствии с учебным планом дисциплины 32 часа лабораторных занятий. Практический раздел состоит из перечня лабораторных работ и заданий для лабораторных работ. По каждой теме дается список основной и дополнительной литературы для самостоятельного изучения и более глубокой подготовки к лабораторным занятиям. По всем темам предполагается выполнение индивидуальных заданий.

Предполагаются различные формы работы со студентами на лабораторных занятиях: устный опрос, тестовые задания, групповая дискуссия, подготовка презентаций и другие. Указанные формы работы способствуют не только усвоению знаний и их репродуктивному воспроизведению, но и способности самостоятельно разрабатывать приложения на платформе .Net.

В *разделе контроля знаний* приводятся вопросы к зачету по дисциплине «Инструменты и средства программирования» для специальности 1-40 04 01 Информатика и технологии программирования и образец тестовых заданий, предназначенных для проверки уровня академических компетенций студентов по дисциплине. Они составлены в логической последовательности и охватывают практически все темы учебной дисциплины.

Вспомогательный раздел содержит необходимые элементы учебно-программной документации: учебную программу по дисциплине «Инструменты и средства программирования» учреждения образования с пояснительной запиской и содержанием учебного материала. Кроме того, в данном разделе имеется перечень рекомендуемой литературы.

В результате изучения дисциплины «Инструменты и средства программирования» и соответствующего учебно-методического комплекса:

Студент должен знать:

- язык программирования C#;
- технологию ADO.Net;
- технологию Entity Framework Core;
- технологию WPF;
- основы платформы ASP.NET Core

Студент должен уметь:

– программировать на языке программирования высокого уровня C#

– выполнять отладку и тестирование программ, написанных на языке программирования C#;

– работать с технологиями ADO.Net и Entity Framework Core;

– работать с технологией WPF;
– разрабатывать web-сайты и web-приложения на Asp.NET Core;
– тестировать web-приложения;
– создавать веб-сервис и интерфейс для работы с ним;
– создавать клиент-серверные приложения.

Студент должен владеть:

– умениями и навыками программирования на C#;
– умениями и навыками работы с технологией ADO.Net;
– умениями и навыками работы с технологией Entity Framework Core.
– умениями и навыками разработки графических приложений с использованием WPF.
– умениями и навыками разработки веб-приложений на ASP.NET Core.

Учебно-методический комплекс учебной дисциплины «Инструменты и средства программирования» адресуется студентам 3-го курса дневной формы обучения специальности 1-40 04 01 «Информатика и технологии программирования».

1.1 Перечень теоретического материала по дисциплине «Инструменты и средства программирования»

1. Основы программирования на языке C#

Язык C# и платформа .Net: .NET Framework, .NET Standard, .NET Core. Обзор стека .NET разработчика. Среда разработки Visual Studio. Типы данных. Средства ввода-вывода. Операции и выражения. Операторы. Массивы. Кортежи.

2. Классы. Объектно-ориентированное программирование

Понятие класса и объекта. Члены класса. Методы. Параметры методов. Типы значений и ссылочные типы. Структуры. Пространства имен. Модификаторы доступа. Свойства. Индексаторы.

Наследование. Преобразование типов. Перегрузка методов, операторов.

Виртуальные методы и свойства. Статические члены и модификатор static. Абстрактные классы. Класс System.Object и его методы.

3. Обобщения, интерфейсы, делегаты. Обработка исключений

Обобщенные типы. Ограничения обобщений. Наследование обобщенных типов.

Определение интерфейсов. Реализация интерфейсов в базовых и производных классах. Наследование интерфейсов. Интерфейсы в обобщениях.

Делегаты. Анонимные методы. Лямбды. События.

Исключения. Конструкция try catch finally.

4. Строки и работа с файловой системой

Строки. Класс System.String. Операции со строками. Класс StringBuilder. Регулярные выражения.

Работа с дисками, каталогами и файлами. Классы File и FileInfo. FileStream. Чтение и запись файла. Чтение и запись текстовых файлов. StreamReader и StreamWriter. Бинарные файлы. BinaryWriter и BinaryReader. Бинарная сериализация. BinaryFormatter.

5. Коллекции. Технология LINQ.

Абстрактные структуры данных. Пространство имен System.Collections. Необобщенные и обобщенные коллекции. Итераторы и оператор yield.

Основы LINQ. Фильтрация выборки и проекция. Сортировка. Работа с множествами. Агрегатные операции. Методы Skip и Take. Группировка. Соединение коллекций. Метод Join, GroupJoin и Zip. Методы All и Any.

6. Асинхронное программирование

Асинхронные методы, async и await. Возвращение результата из асинхронного метода. Последовательный и параллельный вызов асинхронных операций. Обработка ошибок в асинхронных методах. Отмена асинхронных

операций. Асинхронные стримы.

7. Понятие ORM. Обзор технологий ADO.NET, Entity Framework Core и Dapper

Понятие ORM.

Введение в ADO.NET. Работа с отсоединенной моделью через SqlCommand, SqlDataReader. Параметризация запросов. Выходные параметры запросов. Добавление и выполнение хранимых процедур. Работа с присоединенной моделью через SqlDataAdapter и DataSet.

Введение в Entity Framework Core. Создание моделей в Entity Framework Core. Отношения между моделями. Наследование. Запросы и LINQ to Entities. SQL в Entity Framework Core.

Технология Dapper.

8. Понятие паттерна проектирования. Принципы проектирования

Понятие паттерна, обзор архитектурных шаблонов: MVC, MVP, MVVM. Обзор паттернов singleton, repository, builder.

Принципы проектирования: SOLID, KISS.

9. Обзор технологий для создания графических приложений

Обзор технологии Windows Forms. Работа с формами. Контейнеры в Windows Forms. Элементы управления. Меню и панели инструментов.

Особенности платформы Windows Presentation Foundation WPF. Компоновка. Обзор элементов управления и их свойств. Привязка. Взаимодействие с базой данных. Работа с графикой. Анимация. Паттерн MVVM.

10. Основы ASP.NET Core

Введение ASP.NET Core. Класс Program. Класс Startup. Конвейер обработки запроса и middleware. Методы Use, Run, Map и MapWhen. IWebHostEnvironment и окружение. Конфигурация. Состояние приложения. Логгирование. Маршрутизация.

Введение в ASP.NET Core MVC. Контроллеры. Представления. Маршрутизация в ASP.NET Core MVC. Модели. HTML-хелперы и Tag-хелперы. Метаданные и валидация модели. Работа с базой данных и Entity Framework. Фильтры. Аутентификация и авторизация. Основы ASP.NET Core Identity.

11. Основы ASP.NET Core Web API

Спецификация REST, понятие RESTful. Введение в Web API. Запуск и конфигурация. IOptions. Протокол HTTP. Запросы в Web API. Валидация.

12. Основы юнит-тестирования

Введение в тестирование. NUnit и XUnit. Moq. TDD. Интеграционные тесты vs. Unit-тесты. PrivateObject.

1.2 Материалы для обеспечения самостоятельной учебной работы студентов по дисциплине «Инструменты и средства программирования»

Для самостоятельного изучения выделяются следующие темы дисциплины «Инструменты и средства программирования»:

1. Основы юнит-тестирования
2. Строки и работа с файловой системой
3. Асинхронное программирование
4. Понятие паттерна проектирования. Принципы проектирования
5. Основы ASP.NET Core

Тема: «Основы юнит-тестирования»

1. Введение в тестирование. Test-DrivenDevelopment

Одно из преимуществ разработки на платформе ASP.NET MVC предоставляют богатые возможности по тестированию веб-приложения. Можно самим выполнять тестирование тех или иных моментов вручную, а можно использовать специальные небольшие программы, которые называются юнит-тесты.

Юнит-тесты позволяют быстро и автоматически протестировать отдельные участки кода независимо от остальной части программы. При надлежащем составлении юнит-тесты вполне могут покрыть большую часть кода приложения.

Большинство юнит-тестов так или иначе имеют ряд следующих признаков:

Тестирование небольших участков кода ("юнитов")

При создании юнит-тестов выбираются небольшие участки кода, которые надо протестировать. Как правило, тестируемый участок должен быть меньше класса, а в большинстве случаев тестируется отдельный метод класса. Упор на небольшие участки позволяет довольно быстро писать простенькие тесты.

Однажды написанный код нередко читают многократно, поэтому важно писать понятный код. Особенно это важно в юнит-тестах, где в случае неудачи при тестировании разработчик должен быстро прочитать исходный код и понять в чем проблема и как ее исправить. А использование небольших участков кода значительно упрощает подобную работу.

Тестирование в изоляции от остального кода

При тестировании важно изолировать тестируемый код от остальной программы, с которой он взаимодействует, чтобы потом четко определить возможность ошибок именно в этом изолированном коде. Что упрощает и повышает контроль над отдельными компонентами программы.

Тестирование только общедоступных конечных точек

Всего лишь небольшие изменения в классе могут привести к неудаче многих юнит-тестов, поскольку реализация используемого класса изменилась. Поэтому при написании юнит-тестов ограничиваются только общедоступными конечными точками, что позволяет изолировать юнит-тесты от многих деталей внутренней реализации компонента. В итоге уменьшается вероятность, что изменения в классах могут привести к провалу юнит-тестов.

Автоматизация тестов

Написание юнит-тестов для небольших участков кода ведет к тому, что количество этих юнит-тестов становится очень велико. И если процесс получения результатов и проведения тестов не автоматизирован, то это может привести к непроизводительному расходу рабочего времени и снижению производительности. Поэтому важно, чтобы результаты юнит-тестов представляли собой простое решение, означающее, пройден тест или нет. Для автоматизации процесса разработчики обычно обращаются к фреймворкам юнит-тестирования

Фрейморки тестирования

MS Test: фреймворк юнит-тестирования от компании Microsoft, который по умолчанию включен в VisualStudio (начиная с VS 2012 во все версии)

NUnit: портированныйфреймворк с JUnit для платформы .NET

xUnit.net: еще один фреймворк тестирования от создателей NUnit для платформы .NET

Разработка через тестирование (Test-Driven-Development)

Разработка через тестирование (TDD) процесс применения юнит-тестов, при котором сначала пишутся тесты, а потом уже программный код, достаточный для выполнения этих тестов.

Использование TDD позволяет снизить количество потенциальных багов в приложении. Создавая тесты перед написанием кода, мы тем самым описываем способ поведения будущих компонентов, не связывая себя при этом с конкретной реализацией этих тестируемых компонентов (тем более что реализация на момент создания теста еще не существует). Таким образом, тесты помогают оформить и описать API будущих компонентов.

Порядок написания кода при TTD довольно прост:

1. Пишем юнит-тест
2. Запускаем его и видим, что он завершился неудачей (программный код ведь еще не написан)
3. Пишем некоторое количество кода, достаточное для запуска теста
4. Снова запускаем тест и видим его результаты

Этот цикл повторяется снова и снова, пока не будет закончена работа над программным кодом. Так как большинство фреймворков юнит-тестирования помечают неудавшиеся тесты с красного цвета (например, выводится текст красного цвета), а удачный тест отмечается зеленым цветом (опять же выводится текст зеленого цвета), то данный цикл часто называют красным/зеленым циклом.

Интеграционные тесты

Даже если вы охватите юнит-тестами практически всю логику прило-

жения, все равно могут быть моменты, которые будут работать не так, как надо. Кроме того, сложно протестировать представления. Подобные вещи тестируются с помощью интеграционных тестов, выполняющихся на уровне веб-браузера. Для создания подобных тестов в VisualStudio 2013 в версиях Ultimate и Premium доступен такой тип проекта как Coded UI TestProject. Кроме того, доступны open-source решения для создания интеграционных тестов:

WatiN - эмулирует поведение пользователя в веб-браузере. Поддерживает в том числе сайты, использующие ajax

SeleniumHQ - набор инструментов для тестирования в браузерах на различных платформах

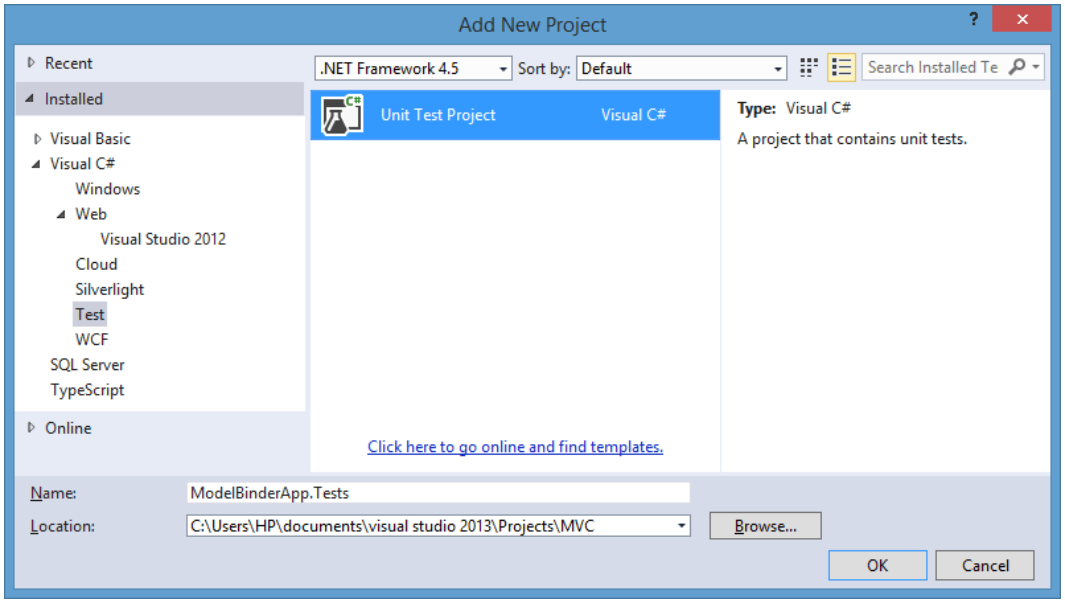
Нагрузочные тесты

Еще один вид тестов представляют нагрузочные тесты, которые призваны протестировать работу сайта в условиях высокой нагрузки. Подобные тесты позволяют выявить узкие места при работе с базой данных или при обращении к диску и ряд других проблем, которые сложно выявить другими способами.

VisualStudioUltimate также имеет инструментарий для создания нагрузочных тестов.

2 Создание проекта для юнит-тестов

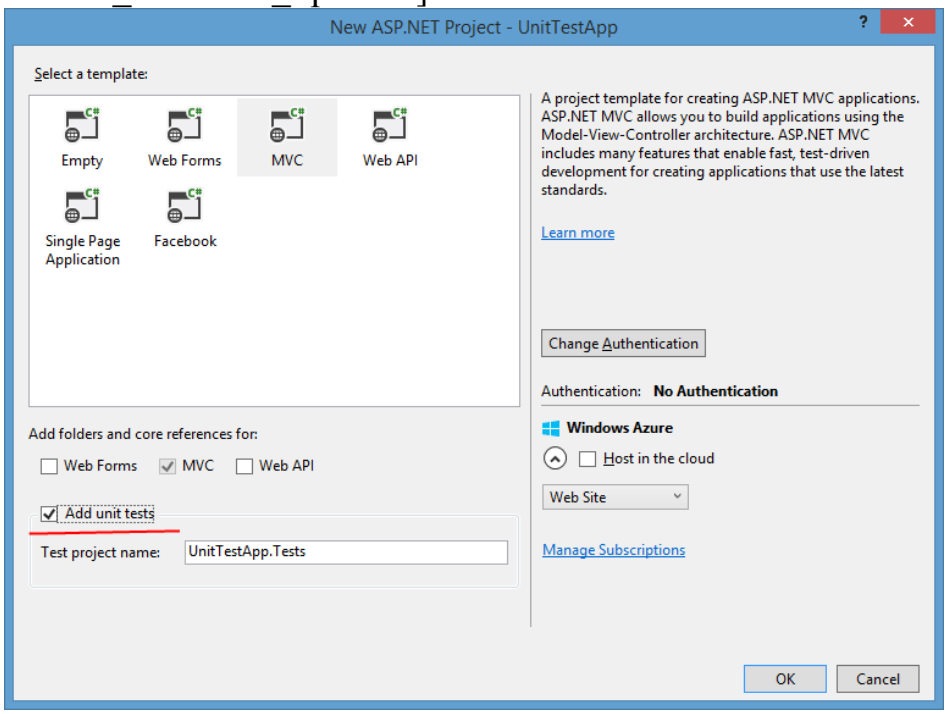
Посмотрим на примере, как создавать юнит-тесты. По умолчанию при создании проекта в любой версии VisualStudio 2013 для создаваемого проекта веб-приложения нам уже предлагается включить дополнительный проект с тестами с помощью опции Addunittests:



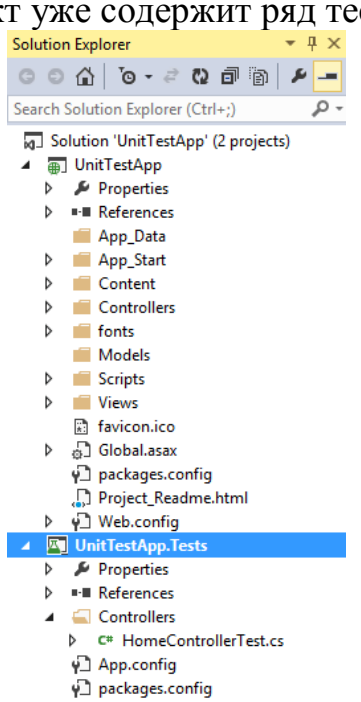
Отметим данный флажок и создадим новый проект.

Если вдруг нам надо добавить тесты к уже существующему проекту, то мы можем добавить в решение новый тип проекта UnitTestProject. Как правило, для проекта тестов название выбирается по следующей схе-

ме [название_главного_проекта].Tests:



При одновременном создании обоих проектов по умолчанию тестовый проект уже содержит ряд тестов:



В частности для контроллера HomeController в проекте тестов будет создан класс HomeControllerTest:

```
[TestClass]
public class HomeControllerTest
{
   [TestMethod]
   public void Index()
   {// Arrange
      HomeController controller = new HomeController();
```

```
        // Act
        ViewResult result = controller.Index() as ViewResult;

        // Assert
        Assert.IsNotNull(result);
    }

    [TestMethod]
    public void About()
    {
        // Arrange
        HomeController controller = new HomeController();

        // Act
        ViewResult result = controller.About() as ViewResult;

        // Assert
        Assert.AreEqual("Your application description page.", result.ViewBag.Message);
    }

    [TestMethod]
    public void Contact()
    {
        // Arrange
        HomeController controller = new HomeController();

        // Act
        ViewResult result = controller.Contact() as ViewResult;

        // Assert
        Assert.IsNotNull(result);
    }
}
```

Класс тестов имеет атрибут [TestClass], а каждый метод, проводящий тестирование, - атрибут [TestMethod].

Каждый метод содержит три логических части - Arrange, Act и Assert. Они помечены соответствующими комментариями. Вкратце рассмотрим, что они делают.

Секция Arrange выполняет начальную инициализацию контекста тестов, а именно создает объект контроллера: HomeControllercontroller = newHomeController()

Далее в секции Act выполняет само действие, которое надо протестировать, а именно генерация представления: ViewResultresult = controller.Index() asViewResult. Так как метод Index контроллера возвращает объект ActionResult, то его еще надо привести к объекту ViewResult.

Чтобы проверить на выполнение предыдущее действие, в секции Assert вызывается метод Assert.IsNotNull(result). Этот метод проверяет, был ли сгенерирован объект представления в предыдущей секции. Класс Assert еще имеет ряд методов, которые позволяют верифицировать результат.

Модель тестов Arrange-Act-Assert представляет просто особенность тестирования в VisualStudio, а целую парадигму тестирования:

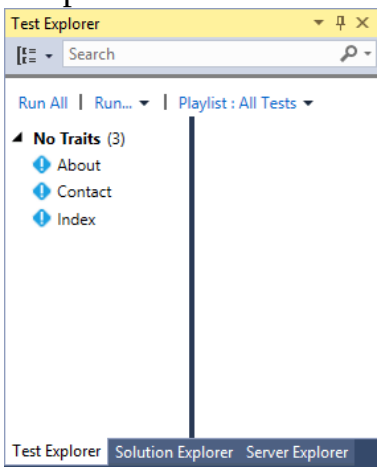
Arrange: подготовка среды, в которой выполняется код

Act: тестирование кода (обычно представляет одну строку кода)

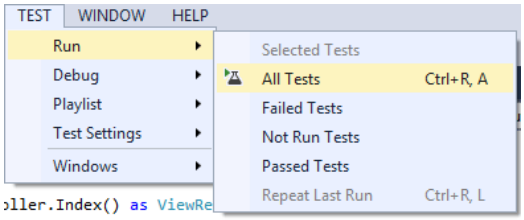
Assert: убеждаемся, что результат теста именно тот, что мы и ожидали

Этот простейший стандартный тест не охватывает всех возможных ошибок, например, были ли сгенерировано нужное представление. Он просто призван дать начальное понимание работы тестов.

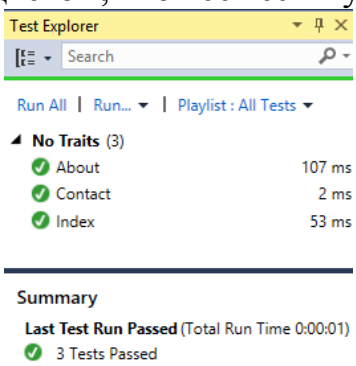
Теперь запустим тест на выполнение. Для этого перейдем в окно TestExplorer и нажмем в нем на кнопку RunAll:



Если оно по умолчанию не открыто, запустить тесты можно через меню Test->Run->AllTests:



Если все нормально, то обозреватель тестов сигнализирует нам зеленым цветом, что все тесты успешно пройдены:



Теперь создадим собственные тесты.

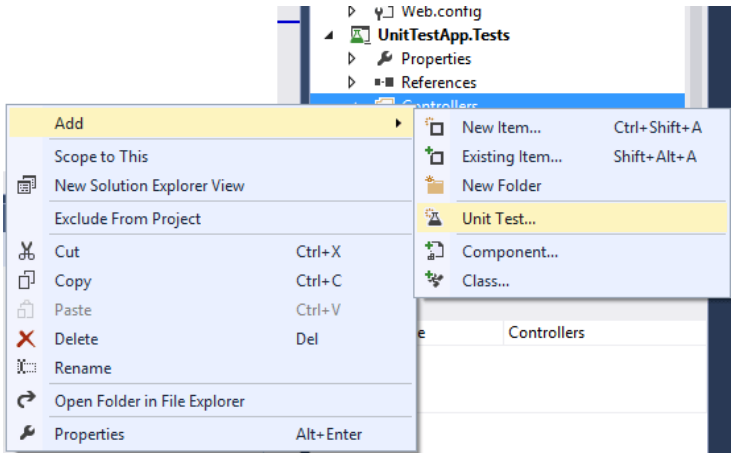
3 Создание юнит-тестов

Возьмем тот же проект из прошлый темы (либо создадим новый) и добавим в главный проект веб-приложения в папку Contollers новый контроллер StoreController:

```
public class StoreController : Controller
{
   publicActionResult Index()
   {
      ViewBag.Message = "Hello world!";
      return View();
   }
}
```

Контроллер имеет только один метод, который устанавливает свойство ViewBag.Message и генерирует объект ActionResult. А также добавим для метода Index представление.

Теперь перейдем к проекту тестов и добавим в него новый класс тестов. Для этого мы можем добавить либо стандартный класс, либо использовать специальный шаблон файлов. Для этого в проекте тестов нажмем правой кнопкой мыши на каталог Controllers и в появившемся контекстном меню выберем Add->UnitTest...:



По умолчанию добавляет класс UnitTest1. Во-первых, изменим название класса и файла на StoreControllerTest. Затем изменим следующим образом сам класс:

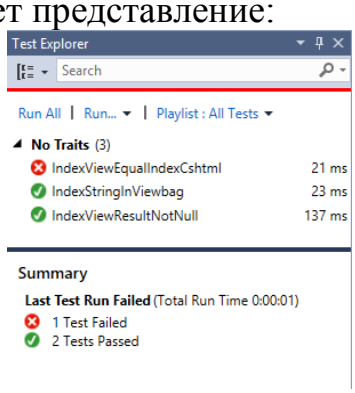
```
using System;
usingMicrosoft.VisualStudio.TestTools.UnitTesting;
usingUnitTestApp.Controllers;
usingSystem.Web.Mvc;

namespaceUnitTestApp.Tests.Controllers
{
   [TestClass]
   public class StoreControllerTest
   {
      [TestMethod]
      public void IndexViewResultNotNull()
      {
         StoreController controller = new StoreController();

         ViewResult result = controller.Index() as ViewResult;
```

```
            Assert.IsNotNull(result);
        }

        [TestMethod]
        public void IndexViewEqualIndexCshtml()
        {
            StoreController controller = new StoreController();

            ViewResult result = controller.Index() as ViewResult;

            Assert.AreEqual("Index", result.ViewName);
        }

        [TestMethod]
        public void IndexStringInViewbag()
        {
            StoreController controller = new StoreController();

            ViewResult result = controller.Index() as ViewResult;

            Assert.AreEqual("Hello world!", result.ViewBag.Message);
        }
    }
}
```

Метод IndexViewResultNotNull тестирует результат метода - возвращаемый объект ViewResult не должен иметь значение null. Метод IndexViewEqualIndexCshtml проверяет название вызываемого представления с помощью вызова Assert.AreEqual. А методIndexStringInViewbag проверяет значение строки в свойстве ViewBag.Message.

Хотя у нас только один метод в контроллере, для него мы создали три тестовых метода для теста каждого отдельного тестового действия. Подобная изоляция облегчает тестирования отдельных участков кода.

Перед запуском тестов перестроим главный проект. И запустим тесты. В этом случае мы увидим, что один тест не пройден - тот, который верифицирует представление:



Тест не пройден, потому что при вызове метода View нам надо явным образом указывать представление. Поэтому изменим метод Index в главном проекте следующим образом:

```
publicActionResult Index()
{
   ViewBag.Message = "Hello world!";
   return View("Index");
}
```

Снова запустим тесты. И теперь уже все тесты должны быть успешно пройдены.

Все три действия имеют одну и ту же секцию Arrange, и, возможно, было бы неплохо сразу установить все начальные настройки для всех методов. Для этого изменим в тестовом проекте класс StoreControllerTest следующим образом:

```
[TestClass]
public class StoreControllerTest
{
   privateStoreController controller;
   privateViewResult result;

   [TestInitialize]
   public void SetupContext()
   {
      controller = new StoreController();
      result = controller.Index() as ViewResult;
   }

   [TestMethod]
   public void IndexViewResultNotNull()
   {
      Assert.IsNotNull(result);
   }

   [TestMethod]
   public void IndexViewEqualIndexCshtml()
   {
      Assert.AreEqual("Index", result.ViewName);
   }

   [TestMethod]
   public void IndexStringInViewbag()
   {
      Assert.AreEqual("Hello world!", result.ViewBag.Message);
   }
}
```

Атрибут TestInitialize позволяет задать метод, который выполняет начальную инициализацию для каждого отдельного теста. Благодаря этому код сокращен, а в тестовых методах оставлены только части Assert. Однако подобный подход надо принимать с осторожностью, так как он осложняет возможности по изменению кода. В данном случае общий контекст очень прост, но если при изменении методов будет изменяться и их контекст, то придется вносить большие изменения во всех классах тестов, а не только в отдельный метод для тестов.

Класс Assert и тестирование результата

Класс Assert из пространства имен Microsoft.VisualStudio.TestTools.UnitTesting с помощью своих статических методов позволяет верифицировать результат выполнения некоторого действия. Ранее уже было рассмотрено несколько методов, в частности, метод Assert.IsNotNull(), проверяющий, не равен ли некоторый объект значению null. Кроме того, при тестировании нам доступен еще ряд методов:

AreEqual(objectexpected, objectactual): проверяет, равны ли оба объекта. Имеет различные перегруженные версии, позволяющие сравнивать различные типы объектов

AreEqual<T>(T expected, T actual): обобщенная версия предыдущего метода. Например, Assert.AreEqual<string>("Index", result.MasterName)

AreNotEqual(object expected, object actual): проверяет, неравнылиобаобъекта. Тест проходит успешно, если объекты не равны

AreNotEqual<T>(T expected, T actual): обобщенная версия предыдущего метода

AreSame(objectexpected, objectactual): проверяет, указывают ли оба объекта на один и тот же объект в памяти

AreNotSame(objectexpected, objectactual): проверяет, указывают ли оба объекта на разные объекты в памяти. Если они указывают на один и тот же объект, то тест заканчивается неудачно

Equals(objectobjA, objectobjB): проверяет на равенство оба объекта

IsFalse(boolcondition): проверяет, равно ли условие condition значению false

IsTrue(boolcondition): проверяет, равно ли условие condition значению true

IsNull(objectvalue): проверяет, имеет ли объект value значение null

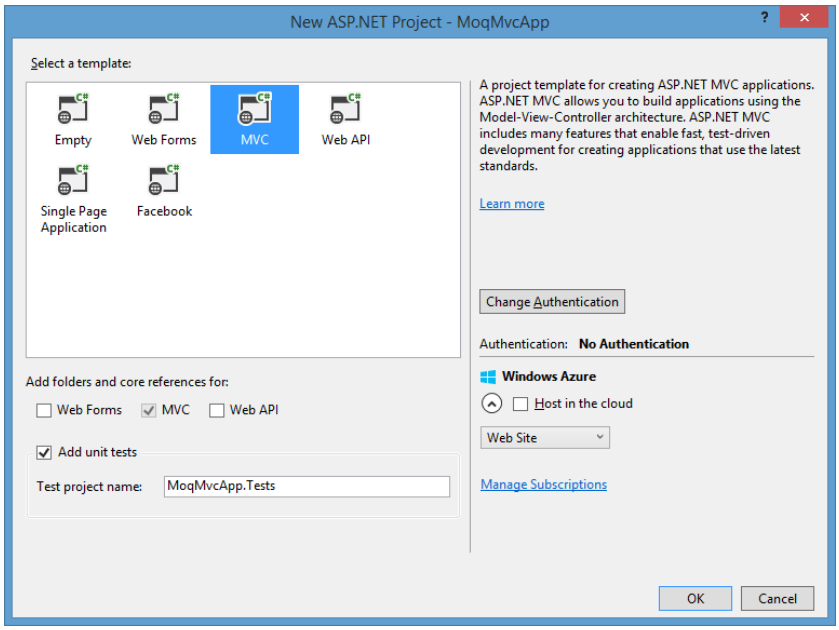
IsInstanceOfType(object value, Type expectedType): проверяет, представляетлиобъект value типexpectedType

Используя эти методы, мы можем проверить различные ситуации в своем приложении.

4 Слабосвязанные объекты и тестирование работы с БД

Большинство приложений так или иначе взаимодействуют с базой данных. Посмотрим, как мы можем протестировать данный аспект.

И вначале создадим новый проект, и вместе с ним проект тестов:



Затем определим в проекте модель Computer и класс контекста CompContext, с помощью которых идет взаимодействие с базой данных:

```
// модель
public class Computer
{
    public int Id { get; set; }
    public string Name { get; set; }
}
// контекст
public class CompContext : DbContext
{
    publicDbSet<Computer> Computers { get; set; }
}
```

Теперь в соответствии с принципами TDD сначала напишем тест, который будет проверять модель представления. В проекте тестов изменим класс HomeControllerTest следующим образом:

```
[TestClass]
public class HomeControllerTest
{
    [TestMethod]
    public void IndexViewModelNotNull()
    {
        // Arrange
        HomeController controller = new HomeController();

        // Act
        ViewResult result = controller.Index() as ViewResult;

        // Assert
        Assert.IsNotNull(result.Model);
    }
}
```

Здесь проверяем модель, которую передаем из базы данных в представление.

Запустим тест на выполнение и увидим, что он естественно заканчивается неудачей, так как в методе Index мы еще ничего не получаем из бд и не передаем в представление.

Теперь пишем минимальный код, в самом контроллере, который производит это действие. Для этого изменим контроллер HomeController следующим образом:

```
public class HomeController : Controller
{
   CompContextdb = new CompContext();
   publicActionResult Index()
   {
      return View(db.Computers);
   }
}
```

Перестроим проект и снова запустим тесты. Теперь зеленый цвет тестов сигнализирует об их успешном окончании. Однако тут есть одна проблема. Такой метод сложно тестировать в силу жесткой связи с базой данных. Если что-то случится с базой данных, и она окажется недоступна, тогда тест завершится неудачей. Но это не значит, что что-то не так в самом коде или что логика построена неправильно.

Поэтому таких ситуаций обычно избегают. И для этого вместо жесткой связи объектов используют слабосвязанные объекты(looselycoupledobjects).

Непосредственно в данном случае проблема решается перенесением уровня взаимодействия с бд в специальный класс, который представляет реализацию паттерна репозиторий.

Для этого добавим в главный проект новый интерфейс и класс, его реализующий:

```
public interface IRepository : IDisposable
{
   List<Computer>GetComputerList();
   Computer GetComputer(int id);
   void Create(Computer item);
   void Update(Computer item);
   void Delete(int id);
   void Save();
}

public class ComputerRepository : IRepository
{
   privateCompContextdb;
   publicComputerRepository()
   {
      this.db = new CompContext();
   }
   public List<Computer>GetComputerList()
   {
      returndb.Computers.ToList();
   }
   public Computer GetComputer(int id)
```

```
        {
            returndb.Computers.Find(id);
        }

        public void Create(Computer c)
        {
            db.Computers.Add(c);
        }

        public void Update(Computer c)
        {
            db.Entry(c).State = EntityState.Modified;
        }

        public void Delete(int id)
        {
            Computer c = db.Computers.Find(id);
            if(c!=null)
                db.Computers.Remove(c);
        }

        public void Save()
        {
            db.SaveChanges();
        }

        privatebool disposed = false;

        public virtual void Dispose(bool disposing)
        {
            if(!this.disposed)
            {
                if(disposing)
                {
                    db.Dispose();
                }
            }
            this.disposed = true;
        }

        public void Dispose()
        {
            Dispose(true);
            GC.SuppressFinalize(this);
        }
    }
```

Теперь в контроллере установим взаимодействие с базой данных через репозиторий:

```
    using System;
    using System.Collections.Generic;
    using System.Linq;
    usingSystem.Web;
```

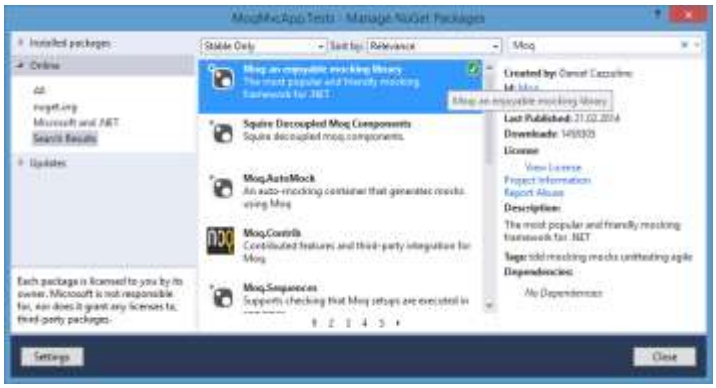
```
usingSystem.Web.Mvc;
usingMoqMvcApp.Models;

namespaceMoqMvcApp.Controllers
{
   public class HomeController : Controller
   {
      IRepository repo;

      publicHomeController(IRepository r)
      {
         repo = r;
      }
      publicHomeController()
      {
         repo = new ComputerRepository();
      }

      publicActionResult Index()
      {
         return View();
      }
      protected override void Dispose(bool disposing)
      {
         repo.Dispose();
         base.Dispose(disposing);
      }
   }
}
```

Теперь мы избавились от жесткой связи с базой данных. Но теперь нам надо будет в тестах сымитировать объект репозитория. И в этом нам поможет Moq-фреймворк.

5 Фреймворк Moq

Кроме собственно фреймворков для создания и проведения юнит-тестов при тестировании часто бывают полезны такие фреймворки, которые позволяют имитировать или эмулировать какую-то функциональность или создавать мок-объекты. Подобных фреймворков существует множество, и одним из самых популярных является Moq.

Итак, возьмем проект из прошлой темы и подключим Moq через NuGet в проект тестов (а не в проект веб-приложения):



После этого в узел References проекта тестов будет добавлена библиотека Moq, содержащая всю основную функциональность.

Теперь изменим класс тестов следующим образом:

```
usingMoq;
.....

[TestClass]
public class HomeControllerTest
{
   [TestMethod]
   public void IndexViewModelNotNull()
   {
      // Arrange
      var mock = new Mock<IRepository>();
      mock.Setup(a =>a.GetComputerList()).Returns(new List<Computer>());
      HomeController controller = new HomeController(mock.Object);

      // Act
      ViewResult result = controller.Index() as ViewResult;

      // Assert
      Assert.IsNotNull(result.Model);
   }
}
```

Для доступа к функциональности Moq вначале подключается соответствующее пространство имен using Moq;.

Moq предназначен для имитации объектов. В данном случае имитируется функциональность репозитория. Для этого объект Mock типизируется соответствующим типом: varmock = newMock<IRepository>()

Затем выполняется настройка mock объекта с помощью метода Setup. Так как нам надо имитировать возвращение методом GetComputerList() набора объектов, то данный метод вызывается в методе Setup, а с помощью метода Returns определяем данный набор объектов.

Поскольку контроллер HomeController теперь в конструкторе принимает объект репозитория, то мы можем передать в конструктор мок-объект, который имитирует функциональность репозитория: HomeControllercontroller = newHomeController(mock.Object)

Теперь запустим тест. Он должен завершиться неудачей, так как у нас не передается в методе Index в представление никакой модели. Поэтому из-

меним метод Index:

```
publicActionResult Index()
{
   var model = repo.GetComputerList();
   returnView(model);
}
```

И если мы сейчас запустим тест, то он пройдет успешно.

Теперь для примера добавим в класс тестов еще один метод:

```
[TestMethod]
public void IndexViewBagMessage()
{
   // Arrange
   var mock = new Mock<IRepository>();
   mock.Setup(a =>a.GetComputerList()).Returns(new List<Computer>() { new Com-
puter()});
   HomeController controller = new HomeController(mock.Object);
   stringexpected = "В базе данных 1 объект";

   // Act
   ViewResult result = controller.Index() as ViewResult;
   string actual = result.ViewBag.Message as string;

   // Assert
   Assert.AreEqual(expected, actual);
}
```

Цель данного метода - проверить сообщение, передаваемое через ViewBag. Причем я хочу, чтобы сообщение передавалось, если в базе данных больше 0 объектов. Для этого список, возвращаемый методом GetComputerList(), инициализируется одним объектом. А метод Assert.AreEqual проверяет оба сообщения.

Запустим тест и увидим, что он завершился неудачей. Теперь нам надо изменить метод Index в главном проекте, чтобы он соответствовал тесту:

```
publicActionResult Index()
{
   var model = repo.GetComputerList();
   if (model.Count> 0)
      ViewBag.Message = String.Format("Вбазеданных {0} объект", model.Count);
   returnView(model);
}
```

Поскольку с помощью Moq мы имитировали в методе контроллера список с одним элементом, то если мы сейчас запустим тест, то он пройдет успешно, так как строка в тесте и значение ViewBag.Message будут совпадать.

6 Тестирование создания модели и переадресации

Рассмотрим на примере тестирование создания модели и переадресации. Во-первых, нам нужен минимальный код для запуска тестов. Для этого возьмем проект веб-приложения из прошлой темы и определим в нем стан-

дартный метод Create:

```
publicActionResult Create()
{
   return View();
}
[HttpPost]
publicActionResult Create(Computer c)
{
   returnView();
}
```

Это минимальный код, достаточный для написания тестов. Теперь в класс тестов HomeControllerTest добавим ряд методов:

```
[TestMethod]
public void CreatePostAction_ModelError()
{
   // arrange
   string expected = "Create";
   var mock = new Mock<IRepository>();
   Computer comp = new Computer();
   HomeController controller = new HomeController(mock.Object);
   controller.ModelState.AddModelError("Name", "Название модели не установлено");
   // act
   ViewResult result = controller.Create(comp) as ViewResult;
   // assert
   Assert.IsNotNull(result);
   Assert.AreEqual(expected, result.ViewName);
}
```

В данном методе проверяется механизм валидации. Через объект контроллера мы можем добавить ряд ошибок к свойству ModelState. И в случае ошибок, идет обращение к представлению Create.

Если мы запустим, то тест не сработает, так как никакой валидации в методе контроллера нет. Поэтому исправим метод Create:

```
[HttpPost]
public ActionResult Create(Computer c)
{
   if(ModelState.IsValid)
   {}
   return View("Create");
}
```

Запустим тест снова. Теперь он сработает успешно, и мы можем переходить к новому тесту. Добавим в класс тестов следующий метод:

```
[TestMethod]
public void CreatePostAction_RedirectToIndexView()
{
   // arrange
   string expected = "Index";
   var mock = new Mock<IRepository>();
   Computer comp = new Computer();
   HomeController controller = new HomeController(mock.Object);
```

```
        // act
        RedirectToRouteResult result = controller.Create(comp) as RedirectToRouteResult;

        // assert
        Assert.IsNotNull(result);
        Assert.AreEqual(expected, result.RouteValues["action"]);
    }
```

Тестирование переадресации делается также просто, как и тестирование представлений. Объект RedirectToRouteResult, который представляет переадресацию, имеет свойство RouteValues, которое позволяет получить данные маршрута - в данном случае получаем метод и сравниваем его с ожидаемым.

И поскольку данный тест опять завершится неудачно, снова изменим метод Create:

```
    [HttpPost]
    publicActionResult Create(Computer c)
    {
        if(ModelState.IsValid)
        {
            returnRedirectToAction("Index");
        }
        returnView("Create");
    }
```

Запустим еще раз тест и убедимся, что он завершился успешно. И теперь добавим третий метод тестов:

```
[TestMethod]
public void CreatePostAction_SaveModel()
{
    // arrange
    var mock = new Mock<IRepository>();
    Computer comp = new Computer();
    HomeController controller = new HomeController(mock.Object);
    // act
    RedirectToRouteResult result = controller.Create(comp) as RedirectToRouteResult;
    // assert
    mock.Verify(a =>a.Create(comp));
    mock.Verify(a =>a.Save());
}
```

В тестах мы легко можем взять результат тестируемых методов и сравнить этот результат с определенным значением, чтобы понять, правильно ли все работает. Но не все методы возвращают определенные значения. Некоторые методы возвращают тип void, однако и тоже необходимо тестировать. И чтобы протестировать подобные методы, в классе Mock определен метод Verify.

Его синтаксис прост: mock.Verify(a=>a.Method(parameter)), где mock - это объект Mock, а parameter - значение, передаваемое в качестве параметра в метод.

В данном случае вызывается метод Create, в который передается модель для сохранения, и метод Save.

Опять же тест не сработает, поэтому изменим метод Create следующим образом:

```
[HttpPost]
publicActionResult Create(Computer c)
{
   if(ModelState.IsValid)
   {
      repo.Create(c);
      repo.Save();
      returnRedirectToAction("Index");
   }
   returnView("Create");
}
```

Запустим тест и убедимся, что он завершится успешно.

Таким образом, мы можем протестировать сохранение модели и механизм переадресации.

2.1 Перечень лабораторных работ по дисциплине «Инструменты и средства программирования»

1 Операторы ветвления, циклы. Массивы.
2 Классы. Объектно-ориентированное программирование
3 Обобщения, интерфейсы, делегаты. Обработка исключений
4 Строки и работа с файловой системой
5 Коллекции. Технология LINQ.
6 Работа с технологией ADO.NET.
7 Создание WPF приложений с использованием Entity Framework Core.
8 Применение паттернов проектирования при разработке приложений.
9 Разработка веб-приложения на ASP.NET Core
10 Разработка Web API на ASP.NET Core
11 Разработка юнит-тестов для приложения.

2.2 Задания для лабораторных работ по дисциплине «Инструменты и средства программирования»

Лабораторная работа №1.

Задание для пункта 1 Функции. Разработать консольное приложение, вычисляющую значение функций (по вариантам) для x изменяющегося от xn до xk с шагом h. На печать выдавать в виде таблицы: аргумент x, значения f. Осуществить обработку исключительных ситуаций (проверка на корректность вводимых данных, и проверка если функция выходит за границы отрезка, представленного на изображении, для данного значения аргумента в таблице выдавать в качестве значения функции - не определена). На печать выдавать в виде таблицы.

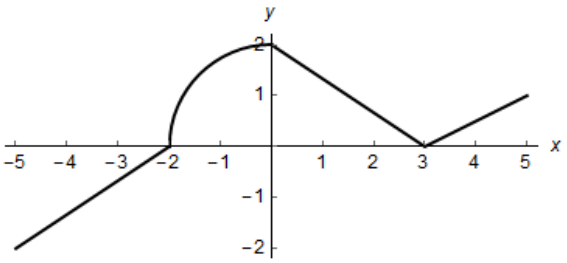
Задание 2.

Разработать консольное приложение с меню, состоящим из 3 пунктов:

1. Вектор 1
2. Вектор 2
3. Матрица

Каждый пункт меню должен решать соответствующую задачу (варианты смотреть ниже). Задачи должны быть реализованы в виде методов с передачей параметров: **по значению, по ссылке, использовать выходные параметры и параметры-массивы** (реализовать все виды передачи параметров). Данные методы не должны содержать ввода и вывода исходных данных (для необходимости создать необходимые методы для ввода и вывода значений). При реализации заданий по массивам (Вектор и Матрица) должен быть организован ввод массива, вывод исходного массива (если вектор – то в строчку, матрица – в матричном виде), вывод полученных результатов, обработку всех исключительных ситуаций и вывод соответствующих сообщений.

Вариант 1



1. Дан целочисленный вектор *A(n)*. Найти номер последнего максимального элемента среди положительных элементов, начиная с первого элемента, большего заданного числа *t*.
2. Дана квадратная матрица *A(nxn)*. Найти максимальный элемент матрицы среди элементов выше главной диагонали.
3. Дана квадратная матрица *A(nxn)*. Построить вектор *b*, где b_i, i=1,…,n – сумма положительных элементов *i*-ой строки массива.

Лабораторная работа №2.

Задание. Разработать систему классов соответственно тематике (по вариантам). В функции Main должно демонстрироваться использование всех разработанных элементов классов. **Определить в программе свой собственный интерфейс и применить его для одного или нескольких разработанных классов из задания. В данной системе классов должны быть реализованы следующие пункты:**

1. Использование наследования;
2. Использование абстрактных классов или членов класса;
3. Использование принципов инкапсуляции;
4. Использование переопределений методов/свойств;
5. Использование минимум 4 собственных классов;
6. Использование конструкторов классов с параметрами;
7. Использование обобщений;
8. Использование свойств;
9. Использование композиции классов.
10. Использование статических элементов или классов;
11. Корректное использование абстрактных классов (использовать их там, где это обусловлено параметрами системы);
12. Корректное использование модификаторов элементов класса (чтобы важные поля не были доступны для полного контроля извне, использование protected);
13. Использование свойств с логикой в get и/или set блоках.
14. Использование индексаторов;
15. Использование перегруженных операторов.
16. Использование методов расширения;
17. Использование наследования обобщений;
18. Использование агрегации классов;
19. Использование обобщенных методов;

Варианты тем по вариантам:

1 Автомобильный салон
2 Библиотека
3 Университет
4 Вокзал
5 Ветклиника
6 Животный мир
7 Мир растений

Лабораторная работа 3

Разработать консольное приложение, реализующие задачи:

Задание 1. Определить и использовать делегат на методы. В качестве методов взять подсчет функций f и g с передачей параметров. Описать метод для подсчета значений функции z (по вариантам), в качестве func передать в функцию либо функцию f (если с клавиатуры введена цифра 1) либо функцию g (если с клавиатуры введена цифра 2).

z=func(x,y)+2*func(x,y);	f(x,y)=5+3x-y	g(x,y)=7x-y

Задание 2. Дан вектор. Описать необходимые методы (по вариантам, см. ниже), которые в качестве параметров должны получать массив чисел и делегат, задающий условие для значений массива через лямбда-выражение. В зависимости от ввода с клавиатуры вызвать либо первый метод (если с клавиатуры введена цифра 1) либо второй метод (если с клавиатуры введена 2.)

Задание 3. Изменить задание 2, таким образом, чтобы реализация вызова метода происходила при помощи **события**. Также реализуйте проверку введённых данных от пользователя через конструкцию TryCatchFinally с использованием **собственного типа исключения.**

Лабораторная работа 4

Разработать консольное приложение (название проекта WorkFile) с меню, состоящим из 3 пунктов:

1. Строки
2. Текстовый файл
3. Бинарный файл

Каждый пункт меню должен решать соответствующую задачу (варианты смотреть ниже). Задачи должны быть реализованы в виде методов с передачей параметров (не должны содержать ввода и вывода).

При реализации заданий по обработке строк (1 и 2), одна должна обрабатывать текст введённый с клавиатуры и выводить результат на экран, а вторая должна считывать исходный текст из текстового файла и записывать результат в файл.

Вариант 1

1. Дан текст. Найти в строках первое по порядку слово с наибольшим числом вхождений в него буквы 'и'.
2. Дан текстовый файл. Вывести в обратном порядке все строки, т.е. 1-я строка выводится последней 2-я предпоследней и т.д.
3. Написать программу-загрузчик данных из бинарного формата в текст. На вход программа получает бинарный файл Var1.dat, предположительно, это база данных преподавателей.

   Свойства сущности Teacher:

Имя — Name (string);
Кафедра — Chair (string);
Номер телефона — PhoneNumber (string).

Ваша программа должна:

– Создать на рабочем столе директорию Teachers.

– Внутри раскидать всех преподавателей из файла по кафедрам (каждая кафедра-отдельный текстовый файл), в файле кафедры преподаватели перечислены построчно в формате "Имя, номер телефона".

Лабораторная работа 5

Задание: Разработать Windows-приложение, реализующее 2 задания:

1. задание – решить одну задачу по коллекциям:

На прямой гоночной трассе стоит N автомобилей, для каждого из которых известны начальное положение и скорость. Определить, сколько произойдет обгонов.

2. задание – использование технологии LINQ:

Описать класс Класс1 (с полями: поле1.1, поле1.2(числовое), поле1.3) и класс Класс2 (с полями поле2.1(должно совпадать с полем 1.3 из класса1), поле2.2, поле2.3). **Название классов и полей придумать самостоятельно, не оригинальные классы не засчитываются.** Создать 2 набора этих классов (по одному для каждого). Используя минимум 5 запросов LINQ и 5 методов расширения LINQ организовать следующую функциональность (продемонстрировать данные результирующих выборок для каждого пункта, т.е. на экране должно быть исходные данные и полученные):

1. произвести фильтрацию любого из списка по одному критерию
2. произвести фильтрацию по одному списку минимум для двух критериев, причем для одного из критериев значение должно запрашиваться у пользователя
3. выдать отсортированный список по одному из полей
4. получить размер выборки по какому-нибудь критерию (использовать метод Count)
5. используя методы Max(), Average(), Sum() выдать соответствующий результат по одному из полей
6. сформировать новую выборку, используя при этом оператор let
7. сформировать выборку из одного списка используя группировку по одному из полей
8. сформировать выборку из двух списков, использовать для группировки метод Join
9. сформировать выборку из двух списков, использовать для группировки метод GroupJoin
10. проверить какое-либо условие для одного из списков, используя метод All

11. проверить какое-либо условие для одного из списков, используя я метод Any

Лабораторная работа 6.

Общая часть задания: написать Windows-приложение (или WPF) для работы с базой данных по вариантам используя технологию ADO.Net (Варианты работы расположены ниже, предложенный набор полей и набор таблиц можно немного изменять, если это необходимо (например, в некоторых вариантах не написан ID, но естественно он подразумевается, либо если необходимо организовать связь многие-ко-многим можете создать дополнительную таблицу) и при отображении, редактировании, добавлении поле ID не должно быть видно). Реализовать все отношения вида один-к-одному, один-ко-многим и многие-ко-многим. Обработать все возможный ошибки ввода. Если будут использоваться хранимые процедуры, то их код также необходимо их вставить. Для работы с базой создайте и подключите свой репозиторий.

В программе должно быть предусмотрено:

1. ввод данных в БД, вывод, корректировка (для всех таблиц) для этого разработать соответствующие методы и вызвать их. При добавлении и изменении данных связанных таблиц предусмотреть загрузку связанных данных в Combobox и выборку из него значений.
2. поиск и фильтрация по одному и нескольким критериям для каждой таблицы
3. сортировка минимум по одному полю для каждой таблицы
4. реализация всех запросов (по вариантам)
5. формирование отчетов в Word, минимум 3:
    a. 1 отчет по всем данным любой таблицы
    b. 2 отчет по любому из запросов
    c. 3 отчет по группирующему запросу + подсчет итога (по вариантам)

Вариант 1

Предметная область БД: Видеотека

Таблицы:

Фильмы	Режиссеры	Кассовые сборы
ID Название фильма ID режиссёра Дата выхода фильма Страна, в которой выпущен фильм	ID ФИО Дата рождения Стаж	ID ID фильма Количество проданных билетов Дата продажи Стоимость одного билета

Запросы:

– Вывести информацию о фильмах, вышедших на экран в текущем и прошлом году.

– Вывести информацию о режиссёре, снимавших заданный фильм.

– Вывести информацию о фильмах (Название фильма, ФИО режиссёра, Дата выхода фильма, Количество проданных билетов), количество проданных билетов на которые больше заданного, отсортировать по убыванию количества проданных билетов

– Вывести информацию о режиссёрах, снявших как минимум N фильмов.

– Выдать список фильмов, с указанием даты продажи и суммарной стоимости проданных билетов на данную дату (для группирующего отчета подсчитать общую выручку по каждому фильму и по всем фильмам)

Лабораторная работа 7

Задание: написать WPF-приложение для работы с базой данных по вариантам используя технологию Entity Framework. Тематику работы взять из задания лаб. работы 6. Разработать модели таким образом, чтобы реализовать все отношения вида один-к-одному, один-ко-многим и многие-ко-многим (можно взять структуру бд из задания для контрольной работы и изменить ее необходимым образом, либо для данной тематики придумать свои таблицы). Для работы с базой создайте и подключите свой репозиторий.

В программе должно быть предусмотрено:

1. ввод данных в БД, вывод, корректировка для каждой таблицы.
2. поиск, сортировка минимум по одному полю (на выбор)
3. реализация минимум 3 запросов (можно своих или взять из задания по КР)

Лабораторная работа 8

Задание: К одной из разработанных программ по лабораторным применить паттерн проектирования на выбор.

Лабораторная работа 9

Разработать сайт на .Net, использую шаблон MVC:

1 Создать Бд минимум 3-5 таблиц (в примере таблицы Товар, Категория товара, Покупка), тематика по вариантам смотреть ниже, придумать свои поля для каждой таблицы (должна присутствовать связь один к многим (в примере на скриншотах это Цветок и Категория). Обязательно в таблице (хотя бы в одной) должны присутствовать: поле с ссылкой на картинку товара, числовое поле, поля чтобы выполнить пункт 4.

2 Главная страница сайта должна содержать меню, можно небольшой

текст и картинку в качестве фона.

3 При нажатии на пункт меню Каталог, должен отображаться список товаров (минимум 10 записей) в виде таблицы, с возможностью добавить товар в корзину или просмотреть детали. Добавить фильтрацию данных по нескольким критериям и сортировку по цене (по возрастанию и убыванию)

4 Добавить корзину. Реализовать:

5 возможность покупки товара, после добавления его в корзину

6 добавлять несколько товаров в корзину, просматривать их общую стоимость, удалять товары из корзины.

7 При переходе в корзину должна открываться форма для оформления покупки, использовать как встроенные, так и свои хелперы, минимум 5 разных). Добавить валидацию данных.

8 После заполнения формы, при нажатии на кнопку ОК переходим на представление, в котором выводится информация о данном заказе (использовать шаблонный хелпер)

9 На представлении Редактирование отображается список товаров с возможностями Просмотра деталей, Изменения, Удаления. А также ссылка для создания нового товара (при добавлении в примере нового цветка добавить выборку категории из списка, который подгружается из БД). Добавить фильтрацию данных по нескольким критериям (минимум по 3 критерия для двух любых таблиц)

10 Добавить авторизацию и регистрацию (возможность для редактирования должна быть только у администратора, у пользователя только просмотр товаров в Каталоге и покупка товара)

11 Оформление вашего сайта по вашему выбору, но чтобы этот выбор не совпадал с другими работами! (приветствуется использование стилизации с помощью Bootstrap)

Вариант 1 Магазин мороженного

Лабораторная работа 10

Разработать Web API на ASP.NET Core. Продемонстрировать работу запросов через Postman или другую программу на выбор.

Лабораторная работа 11

Разработать набор юнит-тестов для разработанного приложения из лабораторной работы № 9.

3.1 Перечень вопросов к зачету по дисциплине «Инструменты и средства программирования»

1 Язык C# и платформа .Net: .NET Framework, .NET Standard, .NET Core.

2 Обзор стека .NET разработчика. Среда разработки Visual Studio.

3 Типы данных. Средства ввода-вывода.

4 Операции и выражения. Операторы.

5 Массивы. Кортежи.

6 Понятие класса и объекта. Члены класса.

7 Методы. Параметры методов.

8 Типы значений и ссылочные типы. Структуры.

9 Пространства имен. Модификаторы доступа.

10 Свойства. Индексаторы.

11 Наследование.

12 Преобразование типов.

13 Перегрузка методов, операторов.

14 Виртуальные методы и свойства.

15 Статические члены и модификатор static. Абстрактные классы.

16 Обобщенные типы. Ограничения обобщений. Наследование обобщенных типов.

17 Определение интерфейсов. Реализация интерфейсов в базовых и производных классах. Наследование интерфейсов. Интерфейсы в обобщениях.

18 Делегаты. Анонимные методы. Лямбды. События.

19 Исключения. Конструкция try catch finally.

20 Строки. Класс System.String. Операции со строками.

21 Работа с дисками, каталогами и файлами. Классы File и FileInfo. FileStream.

22 Бинарные файлы. BinaryWriter и BinaryReader. Бинарная сериализация. BinaryFormatter.

23 Необобщенные и обобщенные коллекции.

24 Основы LINQ. Фильтрация выборки и проекция. Сортировка. Работа с множествами. Группировка. Соединение коллекций.

25 Технология ADO.NET

26 Технология Entity Framework Core

27 Понятие паттерна, обзор архитектурных шаблонов: MVC, MVP, MVVM. Обзор паттернов singleton, repository, builder.

28 Принципы проектирования: SOLID, KISS.

29 Обзор технологий для создания графических приложений

30 Введение ASP.NET Core. Класс Program. Класс Startup.

31 Конвейер обработки запроса и middleware. Методы Use, Run, Map и MapWhen.

32 IWebHostEnvironment и окружение. Конфигурация.

33 Состояние приложения. Логгирование.

34 Маршрутизация.

35 Введение в ASP.NET Core MVC. Контроллеры. Представления. Модели.

36 HTML-хелперы и Tag-хелперы.

37 Метаданные и валидация модели.

38 Работа с базой данных и Entity Framework.

39 Фильтры.

40 Аутентификация и авторизация.

41 Основы ASP.NET Core Identity.

42 Спецификация REST, понятие RESTful.

43 Введение в Web API. Запуск и конфигурация. IOptions. Протокол HTTP. Запросы в Web API.

44 Основы юнит-тестирования

3.2 Образец тестовых заданий по дисциплине «Инструменты и средства программирования»

1) **Что не относится к моделям технологии ASP.NET**
a) Web Forms
b) MVC
c) Web API
d) Razor Pages
e) Html
2) **Контроллер – это**
a) класс, описывающий учетную запись пользователя
b) класс, обеспечивающий начальную инициализацию
c) класс, описывающий логику используемых данных
d) класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных
e) визуальная часть или пользовательский интерфейс приложения
3) **Укажите неверное утверждение**
a) для рендеринга представлений в выходной поток используется метод View()
b) представления – это файлы с расширением .cshtml
c) при компиляции приложения на основе требуемого представления сначала генерируется класс на языке C#, а затем этот класс компилируется
d) представления формируют внешний вид приложения
e) представление является html страницей
4) **Модель – это**
a) класс, описывающий учетную запись пользователя
b) класс, обеспечивающий начальную инициализацию
c) класс, описывающий логику используемых данных
d) класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных
e) визуальная часть или пользовательский интерфейс приложения
5) **Представление – это**
a) класс, описывающий учетную запись пользователя
b) класс, обеспечивающий начальную инициализацию
c) класс, описывающий логику используемых данных
d) класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных
e) визуальная часть или пользовательский интерфейс приложения
6) **Какой тип шаблона представлений генерирует представление, которое отображает значение всех свойств модели**
a) Details
b) Empty
c) Create
d) Edit

e) List

7) Строго типизированное представление – это

a) представление, которое содержит @{Layout= “~/Views/Shared/_Layot.cshtml”}

b) представление, которое содержит директиву @model с указанием типа передаваемых данных

c) представления, которые можно встраивать в другие представления

d) представления, которые используются для рендеринга результатов AJAX-запросов

e) это представления, для рендеринга которых в выходной поток используется метод PartialView()

8) К встроенным html-хелперам не относится

a) Html.TextBox

b) Html.TextArea

c) Html.Image

d) Html.Password

e) Html.Hidden

9) Какой хелпер используется для создания выпадающего списка с множественным выбором

a) Html.TextBox

b) Html.TextArea

c) Html.Label

d) Html.DropDownList

e) Html.ListBox

10) Имеется следующий код HomeController:

```
public ActionRezult Index()
{ return View(); }
```

_Layout.cshtml:

```
Привет:
@RenderBody()
```

Index.cshtml:

```
Солнышко!
```

Что будет выведено при обращении к контроллеру HomeController Action – Index, стандартного Layot установлен _Layot.cshtml?

a) Ошибка, из-за того что не был задан Layot во View – Index.cshtml

b) Привет:

c) Привет: Солнышко!

d) Солнышко!

e) Нет верных ответов

11) Как из строки браузера добраться до action - "UserDetails", контроллера "UserController" с передачей переменной Id=5? С учётом того что используется стандартный роутинг ASP.NET MVC.

a) /UserController/User/5

b) /UserController/UserDetails/?id=5

c) /User/UserDetails/5

d) /UserDetails/5

e) /User/UserDetails/?id=5
f) /UserController/UserDetails/5

12) Имеется следующий код HomeController:

```
public class HomeController : Controller
    {
        public ActionResult Index()
        {   return View();   }
        public PartialViewResult PagePart()
        {   return PartialView();   }
    }
```

Index.cshtml:

```
  @Html.Partial("PagePart")  //1
  @Html.RenderPartial("PagePart")  //2
  @Html.RenderAction("PagePart")  //3
```

PagePart.cshtml:

```
  Hello!
```

Укажите строку или строки с ошибками

a) 1,3
b) 2
c) 2,3
d) 3
e) 1,2

13) Свойство Name какого атрибута аннотации данных содержит строку, которая отображается вместо имени свойства

a) Display
b) HiddenInput
c) ScaffoldColumn
d) DataType
e) UIHint

14) Какой атрибут аннотации данных позволяет полностью скрыть необходимое поле от хелперов

a) Display
b) HiddenInput
c) ScaffoldColumn
d) DataType
e) UIHint

15) Применение какого атрибута валидации к свойству модели означает, что данное свойство должно быть обязательно установлено

a) HiddenInput
b) Required
c) RegularExpression
d) StringLength
e) Compare

16) Какой атрибут позволяет выполнять валидацию на стороне клиента с обратными вызовами на сервер

a) HiddenInput
b) Required
c) Remote
d) StringLength
e) Compare

17) Имеется следующий код HomeController:

```
public ActionRezult Index()
{ ViewBag.Word = “Зайчик”;
  ViewData[“Word”] = “Лиса”;
   return View(); }
```

Index.cshtml:

```
<div> Привет – @((string) ViewBag.Word) </ div>
```

Что будет выведено при обращении к /Home/Index?

a) Привет – Зайчик
b) Привет –
c) Привет – Лиса
d) Зайчик
e) Ошибка времени выполнения

18) Задана следующая установка маршрутов:

```
routes.MapRoute(
 name: "Default",
 url: "{controller}/{action}" );
```

Выберите запрос соответствующий данному маршруту

a) http://localhost/Home/Index
b) http://localhost/Home
c) http://localhost
d) http://localhost/Home/Index/1
e) http://localhost/Default

19) Имеется код: HomeController:

```
public ActionResult Index()
{ViewBag.Word = "МедведД";
 return View();   }

public ActionResult PagePart()
{ViewData["Word"] = "Панда";
 return View();  }
```

Index.cshtml:

```
@{ Html.RenderPartial("PagePart");}
 <div>Привет - @((string)ViewBag.Word)</div>
@{ Html.RenderAction("PagePart");}
 <div>Привет - @((string)ViewData["Word"])</div>
@{ ViewData["Word"] = "Ёжик";}
 <div>Привет - @((string)ViewData["Word"])</div>
```

PagePart.cshtml:

```
Привет - @ViewBag.Word
```

Что будет выведено при обращении к : /Home/Index?

a) Привет - Панда Привет - МедвеД Привет - Панда Привет - МедвеД Привет - Ёжик
b) Привет - МедвеД Привет - МедвеД Привет - МедвеД Привет - МедвеД Привет - МедвеД
c) Привет - МедвеД Привет - МедвеД Привет - Панда Привет - МедвеД Привет - Ёжик
d) Привет - Панда Привет - Панда Привет - Панда Привет - Панда Привет - Ёжик
e) Ошибка времени исполнения. Нельзя переопределять ViewData внутри View.

20) Для постоянной переадресации используется метод

a) RedirectPermanent
b) Redirect
c) RedirectToAction
d) RedirectToRoute
e) нет верных ответов

21) Выберите верные утверждения

a) При создании маршрута можно задавать только 3 параметра
b) Количество параметров в маршруте не ограничено
c) При определении маршрутов необходимо более специфичные определять первее более общих
d) При определении маршрутов необходимо более общие определять первее более специфичных
e) Для получения переданных значений параметров маршрутов можно воспользоваться объектом RouteData
f) Параметр {*catchall} используется для установления ограничения маршрута.

22) Определен следующий маршрут

```
routes.MapRoute(
 name: "Default",
 url: "{controller}/{action}/{id}/{*catchall}",
 defaults: new { controller = "Home", action = "Index" },
 constraints: new { controller = "^H.*", id = @"d{2}" }   );
```

Выберите запрос, который будет соответствовать данному маршруту

a) Home/Index/1
b) Home/Index/2
c) Home/Index/11
d) Book/Index/2
e) Home/Index

23) Выберите что относится к типам фильтров в MVC Framework:

a) Фильтр авторизации
b) Фильтр представлений
c) Фильтр исключений
d) Фильтр действий

e) Фильтр результатов

24) Выберите верные высказывание для архитектуры MVC

a) Одно представление может использоваться для нескольких моделей
b) Контроллер заполняет модель данными
c) Модель обращается к контроллеру за данными
d) Представление использует данные из модели
e) Одна модель может иметь несколько представлений

25) Какой класс результатов действий, производный от FileResult, пишет в ответ массив байтов

a) FilePathResult
b) FileContentResult
c) FileStreamResult
d) EmptyResult
e) ViewResult

26) Какие высказывания верны для ViewBag и ViewData?

a) ViewBag это объект, а ViewData это коллекция.
b) ViewBag это коллекция, а ViewData это объект.
c) Для хранения информации ViewBag использует ViewData.
d) Для хранения информации ViewData использует ViewBag.
e) Если обратиться к элементу ViewBag в представлении(View), который ранее не был задан в контроллере, то произойдёт ошибка компиляции.
f) Элементы ViewBag и ViewData существует в рамках одного запроса.
g) Элементы ViewBag и ViewData существует в рамках жизни экземпляра контроллера.

27) Представим что у нас есть HomeController с Action - Index(возвращает ActionResult - index.cshtml) и PagePart(возвращает PartialView - PagePart.cshtml). Соответственно имеются 2 View: PagePart.cshtml и Index.cshtml .

PagePart.cshtml:

```
Медвед!
```

Index.cshtml:

```
ПревеД - ***
```

Что необходимо вставить вместо * чтобы, при обращении к /Home/Index/, на экране вывелось - "ПревеД - Медвед!". Выберите все подходящие варианты.**

a) @{Html.Action("PagePart");}
b) @Html.RenderAction("PagePart")
c) @{ Html.Partial("PagePart"); }
d) @{ Html.RenderAction("PagePart"); }
e) @Html.Action("PagePart")
f) @Html.RenderPartial("PagePart")

g) @{ Html.RenderPartial("PagePart"); }
h) @Html.Partial("PagePart")

28) При каком условии публичный метод контроллера не является action'ом?

a) если этот метод помечен атрибутом NonAction
b) если этот метод является абстрактным
c) если этот метод возвращает тип не наследующий ActionResult
d) если этот метод является виртуальным
e) это невозможно

29) Выберите неверное утверждение (асинхронные методы)

a) await используется с методами, возвращающими объект ActionRezult
b) Чтобы обозначить метод как асинхронный, перед возвращаемым типом ставиться ключевое слово async
c) Ключевое слово await применяется в асинхронных методах, чтобы приостановить выполнение этого метода до тех пор, пока ожидаемая задача не завершится
d) await используется с методами, возвращающими объект Task
e) асинхронные методы предпочтительно использовать при запросах к БД, к внешнем сетевым ресурсам

Учреждение образования
«Гомельский государственный университет
имени Франциска Скорины»

УТВЕРЖДАЮ
Проректор по учебной работе
ГГУ имени Ф. Скорины
_____ И.В. Семченко
07.04.2017

Регистрационный № УД-13-2017-50/ уч.

ИНСТРУМЕНТЫ И СРЕДСТВА ПРОГРАММИРОВАНИЯ

Учебная программа учреждения высшего образования по учебной дисциплине для специальности
1-40 04 01 Информатика и технологии программирования

2017 г.

Учебная программа по дисциплине специализации составлена на основе требований образовательного стандарта высшего образования, ОСВО 1-40 04 01-2013. Высшее образование. Первая ступень. Специальность 1-40 04 01 Информатика и технологии программирования и учебного плана регистрационный № I 40-02-13. Дата утверждения 29.08.2013.

СОСТАВИТЕЛЬ:

П.В. Бычков – доцент кафедры вычислительной математики и программирования учреждения образования «Гомельский государственный университет имени Франциска Скорины», к.ф.-м.н.,

М.В. Москалева – старший преподаватель кафедры вычислительной математики и программирования учреждения образования «Гомельский государственный университет имени Франциска Скорины».

РЕКОМЕНДОВАНА К УТВЕРЖДЕНИЮ:

Кафедрой вычислительной математики и программирования учреждения образования «Гомельский государственный университет имени Франциска Скорины»

(протокол № 10 от _24_ __05__ 2017 г.);

Научно-методическим советом учреждения образования «Гомельский государственный университет имени Франциска Скорины»

(протокол № _8_ от ___07.06____ 2017 г.)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Необходимыми требованиями к современному специалисту в области программного обеспечения информационных технологий, прикладной математики и информатики являются:

– свободное владение программированием как средством использования ЭВМ,

– умение использовать существующие прикладные программы (текстовые и графические редакторы, базы данных, средства коммуникации, и другое программное обеспечение),

– использование современных средств программирования для создания оригинальных программных продуктов,

– знание и владение современными Internet-технологиями, умение создавать web-приложения.

Дисциплина специализации «Инструменты и средства программирования» предназначен для формирования прочных знаний и практических навыков в области программирования, применении информационных технологий и программного обеспечения, повышения эффективности использования компьютерной техники.

Целью дисциплины специализации «Инструменты и средства программирования» является овладение студентами основами современных компьютерных технологий и программного обеспечения, изучение возможностей разработки приложений с применением современной платформы .NET.

Задачами дисциплины специализации являются:

– ознакомление студентов с основами проектирования приложений с использованием новых технологий и языков программирования;

– усвоение современной компьютерной технологии для разработки приложений

– овладение навыками программирования на языке C#;

– формирование умений и навыков разработки алгоритмов решения различных задач и их реализации на языке программирования C#

– формирование умений и навыков работы с технологиями ADO.Net и Entity Framework Core.

– формирование умений и навыков по разработке веб-приложений с применением современной платформы ASP.NET Core.

Материал дисциплины базируется на ранее полученных студентами знаниях по таким дисциплинам, как «Основы алгоритмизации и программирования», «Конструирование программ и языки программирования».

В результате изучения дисциплины «Инструменты и средства программирования»:

Студент должен знать:

- язык программирования C#;
- технологию ADO.Net;
- технологию Entity Framework Core;
- технологию WPF;
- основы платформы ASP.NET Core

Студент должен уметь:

- программировать на языке программирования высокого уровня C#
- исследовать исходный и исполняемый код программ, написанных на языке программирования C#;
- выполнять отладку и тестирование программ, написанных на языке программирования C#;
- уметь работать с технологиями ADO.Net и Entity Framework Core;
- уметь работать с технологией WPF;
- уметь разрабатывать веб-приложения на ASP.NET Core.

Студент должен владеть:

- умениями и навыками программирования на C#;
- умениями и навыками работы с технологией ADO.Net;
- умениями и навыками работы с технологией Entity Framework Core.
- умениями и навыками разработки графических приложений с использованием WPF.
- умениями и навыками разработки веб-приложений на ASP.NET Core.

Требования к профессиональным компетенциям специалиста:

- ПК-1. Владеть современными технологиями анализа предметной области и разработки требований к создаваемым системам и программным средствам.
- ПК-2. Владеть современными технологиями проектирования сложных систем и программных средств.
- ПК-3. Проводить технико-экономическую оценку вариантов проекта.
- ПК-4. Программировать на профессиональном уровне с учетом ресурсов и возможностей конкретного компьютера, требований стандартов, ограничений проекта.
- ПК-5. Использовать автоматизированные средства разработки программных средств.
- ПК-6. Владеть современными технологиями тестирования, отладки, верификации, аттестации и оценки качества программных средств.
- ПК-7. Управлять процессами жизненного цикла программных средств.
- ПК-8. Владеть вопросами информационно-методического и нормативного правового обеспечения процессов развития информатизации общества.

- ПК-9. Владеть вопросами информационной безопасности.

- ПК-10. Владеть компьютерными методами сбора, хранения и обработки информации в сфере своей профессиональной деятельности.

- ПК-11. Владеть методами эффективной эксплуатации программных средств.

- ПК-12. Администрировать компьютерные системы и сети.

- ПК-13. Конфигурировать компьютерные системы и сети для конкретных задач определенного круга пользователей.

Научно-исследовательская и образовательная деятельность

-ПК-14. Принимать участие в научных исследованиях, связанных с разработкой новых или совершенствованием и развитием имеющихся программных средств. -ПК-15. Выполнять теоретические и экспериментальные исследования, различные виды моделирования автоматизируемых предметных областей.

- ПК-16. Выполнять оценку эффективности программных средств.

-ПК-17. Приобретать новые знания, используя современные информационные технологии.

-ПК-18. Повышать квалификацию своих подчиненных в области программного обеспечения информационных технологий.

-ПК-19. Организовывать и проводить обучение обслуживающего персонала и пользователей.

Организационно-управленческая деятельность

- ПК-20. Работать с юридической литературой и трудовым законодательством. -ПК-21. Организовывать работу малых коллективов исполнителей для достижения поставленных целей.

- ПК-22. Взаимодействовать со специалистами смежных профилей.

- ПК-23. Анализировать и оценивать собранные данные.

- ПК-24. Вести переговоры с другими заинтересованными участниками. - ПК-25. Готовить доклады, материалы к презентациям.

- ПК-26. Владеть современными средствами инфокоммуникаций.

Форма получения образования – дневная. Дисциплина изучается студентами 3 курса специальности 1-40 04 01 «Информатика и технологии программирования» в 6 семестре.

Общее количество часов – 120 (3 зач. ед.); аудиторное количество часов — 64, из них: лекции — 22, лабораторные занятия — 32, управляемая самостоятельная работа (УСР) — 10. Форма отчётности — зачет.

СОДЕРЖАНИЕ УЧЕБНОГО МАТЕРИАЛА

1. Основы программирования на языке C#

Язык C# и платформа .Net: .NET Framework, .NET Standard, .NET Core. Обзор стека .NET разработчика. Среда разработки Visual Studio. Типы данных. Средства ввода-вывода. Операции и выражения. Операторы. Массивы. Кортежи.

2. Классы. Объектно-ориентированное программирование

Понятие класса и объекта. Члены класса. Методы. Параметры методов. Типы значений и ссылочные типы. Структуры. Пространства имен. Модификаторы доступа. Свойства. Индексаторы.

Наследование. Преобразование типов. Перегрузка методов, операторов.

Виртуальные методы и свойства. Статические члены и модификатор static. Абстрактные классы. Класс System.Object и его методы.

3. Обобщения, интерфейсы, делегаты. Обработка исключений

Обобщенные типы. Ограничения обобщений. Наследование обобщенных типов.

Определение интерфейсов. Реализация интерфейсов в базовых и производных классах. Наследование интерфейсов. Интерфейсы в обобщениях.

Делегаты. Анонимные методы. Лямбды. События.

Исключения. Конструкция try catch finally.

4. Строки и работа с файловой системой

Строки. Класс System.String. Операции со строками. Класс StringBuilder. Регулярные выражения.

Работа с дисками, каталогами и файлами. Классы File и FileInfo. FileStream. Чтение и запись файла. Чтение и запись текстовых файлов. StreamReader и StreamWriter. Бинарные файлы. BinaryWriter и BinaryReader. Бинарная сериализация. BinaryFormatter.

5. Коллекции. Технология LINQ.

Абстрактные структуры данных. Пространство имен System.Collections. Необобщенные и обобщенные коллекции. Итераторы и оператор yield.

Основы LINQ. Фильтрация выборки и проекция. Сортировка. Работа с множествами. Агрегатные операции. Методы Skip и Take. Группировка. Соединение коллекций. Метод Join, GroupJoin и Zip. Методы All и Any.

6. Асинхронное программирование

Асинхронные методы, async и await. Возвращение результата из асинхронного метода. Последовательный и параллельный вызов асинхронных операций. Обработка ошибок в асинхронных методах. Отмена асинхронных операций. Асинхронные стримы.

7. Понятие ORM. Обзор технологий ADO.NET, Entity Framework Core и Dapper

Понятие ORM.

Введение в ADO.NET. Работа с отсоединенной моделью через SqlCommand, SqlDataReader. Параметризация запросов. Выходные параметры запросов. Добавление и выполнение хранимых процедур. Работа с присоединенной моделью через SqlDataAdapter и DataSet.

Введение в Entity Framework Core. Создание моделей в Entity Framework Core. Отношения между моделями. Наследование. Запросы и LINQ to Entities. SQL в Entity Framework Core.

Технология Dapper.

8. Понятие паттерна проектирования. Принципы проектирования

Понятие паттерна, обзор архитектурных шаблонов: MVC, MVP, MVVM. Обзор паттернов singleton, repository, builder.

Принципы проектирования: SOLID, KISS.

9. Обзор технологий для создания графических приложений

Обзор технологии Windows Forms. Работа с формами. Контейнеры в Windows Forms. Элементы управления. Меню и панели инструментов.

Особенности платформы Windows Presentation Foundation WPF. Компоновка. Обзор элементов управления и их свойств. Привязка. Взаимодействие с базой данных. Работа с графикой. Анимация. Паттерн MVVM.

10. Основы ASP.NET Core

Введение ASP.NET Core. Класс Program. Класс Startup. Конвейер обработки запроса и middleware. Методы Use, Run, Map и MapWhen. IWebHostEnvironment и окружение. Конфигурация. Состояние приложения. Логгирование. Маршрутизация.

Введение в ASP.NET Core MVC. Контроллеры. Представления. Маршрутизация в ASP.NET Core MVC. Модели. HTML-хелперы и Tag-хелперы. Метаданные и валидация модели. Работа с базой данных и Entity Framework. Фильтры. Аутентификация и авторизация. Основы ASP.NET Core Identity.

11. Основы ASP.NET Core Web API

Спецификация REST, понятие RESTful. Введение в Web API. Запуск и конфигурация. IOptions. Протокол HTTP. Запросы в Web API. Валидация.

12. Основы юнит-тестирования

Введение в тестирование. NUnit и XUnit. Moq. TDD. Интеграционные тесты vs. Unit-тесты. PrivateObject.

УЧЕБНО-МЕТОДИЧЕСКАЯ КАРТА ДИСЦИПЛИНЫ

Номер раздела, темы, занятия	Название раздела, темы, занятия; перечень изучаемых вопросов	Количество аудиторных часов				Материальное обеспечение занятия (наглядные, методические пособия и др.)	Литература	Формы контроля знаний
		лекции	практические (семинарские) занятия	лабораторные занятия	управляемая самостоятельная работа (УСР)			
1	2	3	4	5	6	7	8	9
1	***Основы программирования на языке C#*** 1. Язык C# и платформа .Net: .NET Framework, .NET Standard, .NET Core. Обзор стека .NET разработчика. 2. Среда разработки Visual Studio. 3. Типы данных. Средства ввода-вывода. 4. Операции и выражения. Операторы. 5. Массивы. Кортежи.	2		2		Цифровой проектор, УМК	[1] [2] [3]	Защита отчета по лаб. работам
2	***Классы. Объектно-ориентированное программирование*** 1. Понятие класса и объекта. Члены класса. 2. Методы. Параметры методов. 3. Типы значений и ссылочные типы. Структуры. 4. Пространства имен. Модификаторы доступа. 5. Свойства. Индексаторы. 6. Наследование. Преобразование типов. 7. Перегрузка методов, операторов. Виртуальные методы и свойства. 8. Статические члены и модификатор static. 9. Абстрактные классы. 10. Класс System.Object и его методы.	4		4		Цифровой проектор, УМК	[1] [2] [3]	Защита отчета по лаб. работам
3	***Обобщения, интерфейсы, делегаты. Обработка исключений*** 1. Обобщенные типы. Ограничения обобщений. Наследование обобщенных типов. 2. Определение интерфейсов. Реализация интерфейсов в базовых и производных классах. Наследование интерфейсов. Интерфейсы в обобщениях.	2		2		Цифровой проектор, УМК	[1] [3] [4]	Защита отчета по лаб. работам

Номер раздела, темы, занятия	Название раздела, темы, занятия; перечень изучаемых вопросов	Количество аудиторных часов				Материальное обеспечение занятия (наглядные, методические пособия и др.)	Литература	Формы контроля знаний
		лекции	практические (семинарские) занятия	лабораторные занятия	управляемая самостоятельная работа (УСР)			
	3. Делегаты. Анонимные методы. Лямбды. События. 4. Исключения. Конструкция try catch finally.							
4	***Строки и работа с файловой системой*** 1. Строки. Класс System.String. Операции со строками. Класс StringBuilder. Регулярные выражения. 2. Работа с дисками, каталогами, с файлами. Классы File и FileInfo. FileStream. Чтение и запись файла. 3. Чтение и запись текстовых файлов. StreamReader и StreamWriter. 4. Бинарные файлы. BinaryWriter и BinaryReader. Бинарная сериализация.			2	2	Цифровой проектор, УМК	[1] [2] [6]	Защита отчета по лаб. работам Групповая консультация
5	***Коллекции. Технология LINQ.*** 1. Абстрактные структуры данных. Пространство имен System.Collections. Необобщенные и обобщенные коллекции. Итераторы и оператор yield. 2. Основы LINQ. Фильтрация выборки и проекция. Сортировка. Работа с множествами. Агрегатные операции. Методы Skip и Take. Группировка. Соединение коллекций. Метод Join, GroupJoin и Zip. Методы All и Any.	2		2		Цифровой проектор, УМК	[5] [12] [13]	Защита отчета по лаб. работам
6	**Асинхронное программирование** 1. Асинхронные методы, async и await. 2. Возвращение результата из асинхронного метода. 3. Последовательный и параллельный вызов асинхронных операций. 4. Обработка ошибок в асинхронных методах. Отмена асинхронных операций. 5. Асинхронные стримы.				2	Цифровой проектор, УМК	[1] [2] [3]	Групповая консультация

Номер раздела, темы, занятия	Название раздела, темы, занятия; перечень изучаемых вопросов	Количество аудиторных часов				Материальное обеспечение занятия (наглядные, методические пособия и др.)	Литература	Формы контроля знаний
		лекции	практические (семинарские) занятия	лабораторные занятия	управляемая самостоятельная работа (УСР)			
7	**Понятие ORM. Обзор технологий ADO.NET, Entity Framework Core и Dapper** 1. Понятие ORM. 2. Введение в ADO.NET. Работа с отсоединенной моделью через SqlCommand, SqlDataReader. Параметризация запросов. Выходные параметры запросов. Добавление и выполнение хранимых процедур. Работа с присоединенной моделью через SqlDataAdapter и DataSet. 3. Введение в Entity Framework Core. Создание моделей в Entity Framework Core. Отношения между моделями. Наследование. Запросы и LINQ to Entities. SQL в Entity Framework Core. 4. Технология Dapper.	4		4		Цифровой проектор, УМК	[11] [12] [13]	Защита отчета по лаб. работам
8	**Понятие паттерна проектирования. Принципы проектирования** 1. Понятие паттерна, обзор архитектурных шаблонов: MVC, MVP, MVVM. 2. Обзор паттернов singleton, repository, builder 3. Принципы проектирования: SOLID, KISS.			2	2	Цифровой проектор, УМК	[2] [14]	Групповая консультация
9	**Обзор технологий для создания графических приложений** 1. Обзор технологии Windows Forms. Работа с формами. Контейнеры в Windows Forms. Элементы управления. Меню и панели инструментов. 2. Особенности платформы Windows Presentation Foundation WPF. Компоновка. Обзор элементов управления и их свойств. Привязка. Взаимодействие с базой данных. Работа с графикой. Анимация. Паттерн MVVM.	2		4		Цифровой проектор, УМК	[1] [5] [9]	Защита отчета по лаб. работам

Номер раздела, темы, занятия	Название раздела, темы, занятия; перечень изучаемых вопросов	Количество аудиторных часов				Материальное обеспечение занятия (наглядные, методические пособия и др.)	Литература	Формы контроля знаний
		лекции	практические (семинарские) занятия	лабораторные занятия	управляемая самостоятельная работа (УСР)			
10	**Основы ASP.NET Core** 1. Введение ASP.NET Core. Класс Program. Класс Startup. Конвейер обработки запроса и middleware. Методы Use, Run, Map и MapWhen. IWebHostEnvironment и окружение. Конфигурация. Логгирование. Маршрутизация. 2. Введение в ASP.NET Core MVC. Контроллеры. Представления. Маршрутизация в ASP.NET Core MVC. Модели. HTML-хелперы и Tag-хелперы. Метаданные и валидация модели. Работа с базой данных и Entity Framework. 3. Фильтры. 4. Аутентификация и авторизация. Основы ASP.NET Core Identity.	4		6	2	Цифровой проектор, УМК	[1] [15]	Защита отчета по лаб. работам
11	**Основы ASP.NET Core Web API** 1. Спецификация REST, понятие RESTful. 2. Введение в Web API. Запуск и конфигурация. IOptions. 3. Протокол HTTP. 4. Запросы в Web API. 5. Валидация.	2		2		Цифровой проектор, УМК	[1] [9]	Защита отчета по лаб. работам
12	**Основы юнит-тестирования** 1. Введение в тестирование. 2. NUnit и XUnit. Moq. 3. TDD. 4. Интеграционные тесты vs. Unit-тесты. 5. PrivateObject.			2	2	Цифровой проектор, УМК	[1] [5] [9]	Защита отчета по лаб. работам
	Итого по дисциплине	**22**		**32**	**10**			**Зачет**

ИНФОРМАЦИОННО-МЕТОДИЧЕСКАЯ ЧАСТЬ

Примерный перечень лабораторных работ

1 Операторы ветвления, циклы. Массивы.
2 Классы. Объектно-ориентированное программирование
3 Обобщения, интерфейсы, делегаты. Обработка исключений
4 Строки и работа с файловой системой
5 Коллекции. Технология LINQ.
6 Работа с технологией ADO.NET.
7 Создание WPF приложений с использованием Entity Framework Core.
8 Применение паттернов проектирования при разработке приложений.
9 Разработка веб-приложения на ASP.NET Core
10 Разработка Web API на ASP.NET Core
11 Разработка юнит-тестов для приложения.

Рекомендуемые формы контроля знаний

1. Лабораторные работы
2. Тестирование

Рекомендуемая литература

Основная

1. Албахари, Джозеф. C# 7.0. Справочник. Полное описание языка: Пер. с англ. / Джозеф Албахари, Бен Албахари. – СпБ.: ООО “Альфакнига”, 2018. – 1024 с.

2. Троелсен, Эндрю. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. : Пер. с англ./ Эндрю Троелсен, Филипп Джепикс. – СПб.: ООО “Диалектика”, 2018. – 1328 с.

3. Прайс, Марк Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов, 3-е изд. / Марк Дж. Прайс. – СПб.: Питер, 2018. – 640 с.

4. Рихтер, Джеффри. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# / Джеффри Рихтер. – М.: Питер, 2016. – 896 с.

5. Шумаков, П. ADO.NET и создание приложений баз данных в среде Microsoft Visual Studio.NET / П. Шумаков. – М.: Диалог-МИФИ, 2004. – 528 с.

6. Мак-Дональд, М., Шпушта М. Microsoft ASP.NET 2.0 с примерами на C# 2005 для профессионалов / М. Мак-Дональд, М. Шпушта : Пер. с англ. – М.: ОДО «И.Д. Вильямс», 2006. – 1408с.: ил.

7. Эспозито, Д. Знакомство с Microsoft ASP.NET 2.0 / Д. Эспозито : Пер. с англ. – М.: Издательство «Русская Редакция»; СПб.: Питер, 2006. – 512 с.: ил.

8. Эспозито, Д. Microsoft ASP.NET 2.0. Углубленное изучение / Д. Эспозито : Пер. с англ. – М.: Издательство «Русская Редакция»; СПб.: Питер, 2007. – 592 с.: ил.

9. Троелсен, Э. C # и платформа .NET. Библиотека программиста / Э. Троелсен. – СПб.: Питер, 2004. – 796 с.: ил.

10. Уоткинз, Д. Программирование на платформе .NET. / Д. Уоткинз, М. Хаммонд, Б. Эйбрамз : Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 368 с.: ил.

11. Волоха, А.В. Microsoft SQL Server 2005. Новые возможности / А.В. Волоха. – СПб.: Питер, 2006. – 304 с.: ил.

12. Эвери, Дж. Microsoft ASP.NET. Конфигурирование и настройка / Дж. Эвери. – М.: Эком, 2005.

13. Бучек, Г. ASP.NET. Учебный курс (с CD-ROM) / Г. Бучек. – СПб.: Питер, 2002. – 512 с.

14. Андерсон, Р. ASP.NET для профессионалов. В 2-х томах / Р. Андерсон. – М.: Лори, 2005.

15. Оньон, Ф. Основы ASP.NET с примерами на C# / Ф. Оньон- М.: Издательский дом «Вильямс», 2003.

16. Макдональд, М. ASP.NET. Наиболее полное руководство / М. Макдональд. – СПб.: издательство «BHV-СПб»,2003. – 992 с.

17. Гаевский, А.Ю. Самоучитель по созданию Web-страниц: HTML, JavaScript и Dynamic HTML / А.Ю. Гаевский, В.А. Романовский – К.: А.С.К., 2002. – 472 с.

18. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма [и др.] – СПб.: Питер, 2001. – 368 с.: ил.

19. Хабибуллин, И. Ш. Разработка Web-служб средствами Java / И. Ш. Хабибуллин.– СПб.: издательство «BHV-СПб», 2003. – 400 с: ил.

20. Родичев, Ю.А. Нормативная база и стандарты в области информационной безопасности» / Ю.А. Родичев. – СПб: Питер, 2017. – 254 с.

Дополнительная

21. David, S. Platt Introducing Microsoft.NET. Third Edition / S. David. – Microsoft Press, 2003. – 329 p.

22. Вилдермьюс, Ш. Практическое использование ADO.NET. Доступ к данным в Internet / Ш. Вилдермьюс. – М.: Издательский дом "Вильямс", 2003. – 288 с.: ил.

23. Мак-Манус, Дж. П. Создание приложений ASP.NET, XML и ADO.NET в среде Visual Basic.NET / Дж. П. Мак-Манус, К. Кинсмен. - М.: Издательский дом "Вильямс", 2003.

24. Бошемин, Б. Основы ADO.NET / Б. Бошемин. - М.: Издательский дом "Вильямс", 2003.

Методические рекомендации по организации и выполнению УСР по дисциплине «Инструменты и средства программирования»

Для самостоятельного изучения выделяются следующие темы дисциплины «Инструменты и средства программирования»:

1. Строки и работа с файловой системой
2. Асинхронное программирование
3. Понятие паттерна проектирования. Принципы проектирования
4. Основы ASP.NET Core
5. Основы юнит-тестирования

Самостоятельное изучение данных тем преследует следующие цели:

– изучение информации о работе со строками и файловой системой на языке C#;

– изучение информации об асинхронном программировании;

– изучение информации о паттернах и основных принципах проектирования;

– изучение информации о ASP.NET Core;

– изучение информации об основах юнит-тестирования

– овладение навыками практической работы по асинхронному программированию;

– овладение навыками практической работы по созданию приложений ASP.NET Core;

– овладение навыками практической работы по созданию юнит-тестов для приложений.

Учебная программа УСР

1) Тема «*Строки и работа с файловой системой*» – 2 часа

Цели: 1) овладеть знаниями по данной теме, овладение навыками практической работы; 2) формирование умений и навыков работы со строками и файловой системой.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – защита лабораторной работы, тестирование в составе теста по окончанию курса «Инструменты и средства программирования».

Учебно-методическое обеспечение – электронный вариант текстов лекций, а также [1], [2], [11], [13], [15], [17] из списка рекомендуемой литературы.

2) Тема «*Асинхронное программирование*» – 2 часа

Цели: 1) овладеть знаниями по данной теме, овладение навыками практической работы; 2) формирование умений и навыков по работе с потоками; 4) формирование умений и навыков асинхронного программирования.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – тестирование в составе теста по окончанию курса «Инструменты и средства программирования».

Учебно-методическое обеспечение – электронный вариант текстов лекций, а также [3], [5], [7], [8], [9], [10] из списка рекомендуемой литературы.

3) Тема «Понятие паттерна проектирования. Принципы проектирования» – 2 часа

Цели: 1) овладеть знаниями по данной теме, овладение навыками практической работы; 2) формирование умений и навыков применения паттернов проектирования при разработке приложений.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – защита лабораторной работы, тестирование в составе теста по окончанию курса «Инструменты и средства программирования».

Учебно-методическое обеспечение – электронный вариант текстов лекций, а также [1], [5], [11], [10], [12], [14] из списка рекомендуемой литературы.

4) Тема «Основы ASP.NET Core» – 2 часа

Цели: 1) овладеть знаниями по данной теме, овладение навыками практической работы; 2) формирование умений и навыков по работе с конвейером запросов, по созданию своих компонент middleware.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – тестирование в составе теста по окончанию курса «Инструменты и средства программирования».

Учебно-методическое обеспечение – электронный вариант текстов лекций, а также [1], [2], [6], [7], [15], [17] из списка рекомендуемой литературы.

5) Тема «Основы юнит-тестирования» – 2 часа

Цели: 1) овладеть знаниями по данной теме, овладение навыками практической работы; 2) формирование умений и навыков по разработке тестов для приложений.

Форма выполнения заданий – индивидуальная.

Форма контроля выполнения заданий – защита лабораторной работы, тестирование в составе теста по окончанию курса «Инструменты и средства программирования».

Учебно-методическое обеспечение – электронный вариант текстов лекций, а также [1], [4], [5], [7], [13], [15] из списка рекомендуемой литературы.

4.2 Перечень рекомендуемой литературы по дисциплине «Инструменты и средства программирования»

Основная литература

1. Албахари, Джозеф. C# 7.0. Справочник. Полное описание языка: Пер. с англ. / Джозеф Албахари, Бен Албахари. – СпБ.: ООО “Альфакнига”, 2018. – 1024 с.
2. Троелсен, Эндрю. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. : Пер. с англ./ Эндрю Троелсен, Филипп Джепикс. – СПб.: ООО “Диалектика”, 2018. – 1328 с.
3. Прайс, Марк Дж. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов, 3-е изд. / Марк Дж. Прайс. – СПб.: Питер, 2018. – 640 с.
4. Рихтер, Джеффри. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# / Джеффри Рихтер. – М.: Питер, 2016. – 896 с.
5. Столбовский, Д. Н. Основы разработки Web-приложений на ASP.NET / Д.Н. Столбовский. – М.: Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2014. – 304 с.
6. Фримен, Адам ASP.NET 4.5 с примерами на C# 5.0 для профессионалов / Адам Фримен. – М.: Вильямс, 2014. – 112 с.
7. Фримен, Адам ASP.NET MVC 5 с примерами на C# 5.0 для профессионалов / Адам Фримен. – М.: Вильямс, 2015. – 736 с.
8. Чедвик, Джесс ASP.NET MVC 4. Разработка реальных веб-приложений с помощью ASP.NET MVC / Джесс Чедвик, Тодд Снайдер, Хришикеш Панда. – М.: Вильямс, 2013. – 432 с.
9. Мак-Дональд, М., Шпушта М. Microsoft ASP.NET 2.0 с примерами на C# 2005 для профессионалов / М. Мак-Дональд, М. Шпушта : Пер. с англ. – М.: ОДО «И.Д. Вильямс», 2006. – 1408с.: ил.
10. Эспозито, Д. Знакомство с Microsoft ASP.NET 2.0 / Д. Эспозито : Пер. с англ. – М.: Издательство «Русская Редакция»; СПб.: Питер, 2006. – 512 с.: ил.
11. Эспозито, Д. Microsoft ASP.NET 2.0. Углубленное изучение / Д. Эспозито : Пер. с англ. – М.: Издательство «Русская Редакция»; СПб.: Питер, 2007. – 592 с.: ил.
12. Троелсен, Э. C # и платформа .NET. Библиотека программиста / Э. Троелсен. – СПб.: Питер, 2004. – 796 с.: ил.
13. Уоткинз, Д. Программирование на платформе .NET. / Д. Уоткинз, М. Хаммонд, Б. Эйбрамз : Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 368 с.: ил.
14. Волоха, А.В. Microsoft SQL Server 2005. Новые возможности / А.В. Волоха. – СПб.: Питер, 2006. – 304 с.: ил.
15. Эвери, Дж. Microsoft ASP.NET. Конфигурирование и настройка / Дж. Эвери. – М.: Эком, 2005.

16. Бучек, Г. ASP.NET. Учебный курс (с CD-ROM) / Г. Бучек. – СПб.: Питер, 2002. – 512 с.

17. Андерсон, Р. ASP.NET для профессионалов. В 2-х томах / Р. Андерсон. – М.: Лори, 2005.

18. Оньон, Ф. Основы ASP.NET с примерами на C# / Ф. Оньон- М.: Издательский дом «Вильямс», 2003.

19. Макдональд, М. ASP.NET. Наиболее полное руководство / М. Макдональд. – СПб.: издательство «BHV-СПб»,2003. – 992 с.

20. Гаевский, А.Ю. Самоучитель по созданию Web-страниц: HTML, JavaScript и Dynamic HTML / А.Ю. Гаевский, В.А. Романовский – К.: А.С.К., 2002. – 472 с.

21. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма [и др.] – СПб.: Питер, 2001. – 368 с.: ил.

22. Хабибуллин, И. Ш. Разработка Web-служб средствами Java / И. Ш. Хабибуллин.– СПб.: издательство «BHV-СПб», 2003. – 400 с: ил.

23. Родичев, Ю.А. Нормативная база и стандарты в области информационной безопасности» / Ю.А. Родичев. – СПб: Питер, 2017. – 254 с.

Дополнительная литература

24. David, S. Platt Introducing Microsoft.NET. Third Edition / S. David. – Microsoft Press, 2003. – 329 p.

25. Вилдермьюс, Ш. Практическое использование ADO.NET. Доступ к данным в Internet / Ш. Вилдермьюс. – М.: Издательский дом "Вильямс", 2003. – 288 с.: ил.

26. Мак-Манус, Дж. П. Создание приложений ASP.NET, XML и ADO.NET в среде Visual Basic.NET / Дж. П. Мак-Манус, К. Кинсмен. - М.: Издательский дом "Вильямс", 2003.

27. Бошемин, Б. Основы ADO.NET / Б. Бошемин. - М.: Издательский дом "Вильямс", 2003.