МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ГАГАРИНА Ю.А.»
(СГТУ имени Гагарина Ю.А.)
Саратовский колледж машиностроения и энергетики

«СОГЛАСОВАНО»	«УТВЕРЖДАЮ»
Решением П(Ц)МК	Заместитель директора по УР
Протокол № 8	_____ С.В. Клюквина
от «28» июня_____2021г.	«28» _июня_____2021г.

Методические рекомендации для студентов по проведению лабораторных занятий по дисциплине

«МДК 01.01. Разработка программных модулей» для специальности 09.02.07 «Информационные системы и программирование»

Саратов 2020

Разработал преподаватель СКМ и Э ФГБОУ ВО «СГТУ имени Гагарина Ю.А» Пояркова Т.А.

Данные методические рекомендации предназначены для студентов 3 курса, специальности 09.02.07 «Информационные системы и программирование»

СОДЕРЖАНИЕ

Введение

Лабораторные занятия, как виды учебных занятий, направлены на экспериментальное подтверждение теоретических положений и формирование учебных и профессиональных практических умений и составляют важную часть теоретической и профессиональной практической подготовки. Семинар является видом практических занятий.

В процессе лабораторного занятия обучающиеся выполняют одно или несколько практических заданий под руководством преподавателя в соответствии с изучаемым содержанием учебного материала.

Выполнение обучающимися практических занятий проводится с целью:

- формирования практических умений в соответствии с требованиями к уровню подготовки обучающихся, установленными рабочей программой дисциплины по конкретным разделам/ темам дисциплины;
- обобщения, систематизации, углубления, закрепления полученных теоретических знаний;
- совершенствования умений применять полученные знания на практике, реализации единства интеллектуальной и практической деятельности;
- развития интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;
- выработки таких профессионально значимых качеств, как самостоятельность, ответственность, точность, творческая инициатива при решении поставленных задач при освоении общих компетенций.

Цель практических занятий

В результате освоения учебной дисциплины «МДК 01.01. Разработка программных модулей» обучающийся должен **уметь:**

- - Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.
- Оформлять документацию на программные средства.
- Оценка сложности алгоритма.
- Создавать программу по разработанному алгоритму как отдельный модуль;
- Оформлять документацию на программные средства.
- Осуществлять разработку кода программного модуля на языках низкого уровня и высокого уровней.
- Выполнять оптимизацию и рефакторинг программного кода.
- Осуществлять разработку кода программного модуля на современных языках программирования.
- Оформлять документацию на программные средства.

В результате освоения учебной дисциплины обучающийся должен **знать:**

- Основные этапы разработки программного обеспечения.
- Основные принципы технологии структурного и объектно-ориентированного программирования.
- Актуальная нормативно-правовая база в области документирования алгоритмов.
- Основные этапы разработки программного обеспечения.
- Основные принципы технологии структурного и объектно-ориентированного программирования.
- Основные этапы разработки программного обеспечения.

Методические рекомендации по проведению практических занятий

- Основные принципы технологии структурного и объектно-ориентированного программирования.

Изучение дисциплины направлено на формирование следующих компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОП 02.Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей.

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языках.

ОК 11. Планировать предпринимательскую деятельность в профессиональной сфере.

ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием

ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств

ПК 1.4 Выполнять тестирование программных модулей

ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода

ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ

Организация практического занятия

Лабораторное занятие должно проводиться в учебной лаборатории «Программирования и баз данных» оснащенная необходимым для реализации программы учебной дисциплины оборудованием.

В соответствии с требованиям ФГОС 09.02.07 «Информационные системы и программирование» реализация ППССЗ должна обеспечивать выполнение обучающимися и практических занятий, включая как обязательный компонент *Лабораторные занятия (с использованием персональных компьютеров).*

Выполнению практических занятий предшествует проверка знаний обучающихся - их теоретической готовности к выполнению задания.

Лабораторные занятия могут носить репродуктивный, частично-поисковый и поисковый характер.

Работы, носящие *репродуктивный характер,* отличаются тем, что при их проведении обучающиеся пользуются подробными инструкциями, в которых указаны: цель работы, пояснения (теория, основные характеристики), оборудование, аппаратура, материалы и их характеристики, порядок выполнения работы, таблицы, выводы (без формулировки), контрольные вопросы, учебная и специальная литература.

Работы, носящие *частично-поисковый характер*, отличаются тем, что при их проведении обучающиеся не пользуются подробными инструкциями, им не дан порядок выполнения необходимых действий, и они требуют от обучающихся самостоятельного подбора оборудования, выбора способов выполнения работы в инструктивной и справочной литературе и др.

Работы, носящие *поисковый характер*, характеризуются тем, что обучающиеся, опираясь на имеющиеся у них теоретические знания, должны решить новую для них проблему.

При планировании практических занятий необходимо находить оптимальное соотношение репродуктивных, частично-поисковых и поисковых работ, чтобы обеспечить высокий уровень интеллектуальной деятельности.

Формы организации обучающихся при проведении практических занятий - фронтальная, групповая и индивидуальная.

При *фронтальной форме* организации занятий все обучающиеся выполняют одновременно одну и ту же работу.

При *групповой форме* организации занятий одна и та же работа выполняется бригадами по 2 - 5 человек.

При *индивидуальной форме* организации занятий каждый обучающийся выполняет индивидуальное задание.

Для повышения эффективности проведения практических занятий рекомендуется:

- разработка сборников задач, заданий и упражнений;
- разработка контрольно-диагностических материалов для контроля за подготовленностью обучающихся к практическим занятиям, в том числе в форме педагогических тестовых материалов для автоматизированного контроля;
- подчинение методики проведения практических занятий ведущим дидактическим целям с соответствующими установками обучающимся;

- применение коллективных и групповых форм работы, максимальное использование индивидуальных форм с целью повышения ответственности каждого обучающегося за самостоятельное выполнение полного объема работ;
- проведение практических занятий на повышенном уровне трудности с включением в них заданий, связанных с выбором обучающимися условий выполнения работы, конкретизацией целей, самостоятельным отбором необходимого оборудования;
- подбор дополнительных задач и заданий для обучающихся, работающих в более быстром темпе, для эффективного использования времени, отводимого на Лабораторные занятия.

Структура лабораторного занятия

Лабораторное занятите № 1

Тема: Работа с файловыми потоками

Цели и задачи урока: Изучить возможность работы с файловыми потоками.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60х90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.
5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60х90 1/16. - (Среднее профессиональное образование).

Теоретические сведения

Файл – это набор данных, который хранится на внешнем запоминающем устройстве (например на жестком диске). Файл имеет имя и расширение. Расширение позволяет идентифицировать, какие данные и в каком формате хранятся в файле.

Под работой с файлами подразумевается:

- создание файлов;
- удаление файлов;
- чтение данных;
- запись данных;
- изменение параметров файла (имя, расширение…);
- другое.

В Си-шарп есть пространство имен **System.IO**, в котором реализованы все необходимые нам классы для работы с файлами. Чтобы подключить это пространство имен, необходимо в самом начале программы добавить строку using System.IO. Для использования кодировок еще добавим пространство using System.Text;

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

```
using System.Text;
using System.IO;
```
Как создать файл?

Для создания пустого файла, в классе **File** есть метод **Create()**. Он принимает один аргумент – путь. Ниже приведен пример создания пустого текстового файла new_file.txt на диске D:

```
static void Main(string[] args)
{
  File.Create("D:\\new_file.txt");
}
```

Если файл с таким именем уже существует, он будет переписан на новый пустой файл.

Метод **WriteAllText()** создает новый файл (если такого нет), либо открывает существующий и записывает текст, заменяя всё, что было в файле:

```
static void Main(string[] args)
{
  File. WriteAllText("D:\\new_file.txt", "текст");
}
```

Метод **AppendAllText()** работает, как и метод WriteAllText() за исключением того, что новый текст дописывается в конец файла, а не переписывает всё что было в файле:

```
static void Main(string[] args)
{
  File.AppendAllText("D:\\new_file.txt", "текст метода AppendAllText ()"); //допишет текст в конец файла
}
```

Как удалить файл?

Метод **Delete()** удаляет файл по указаному пути:

```
static void Main(string[] args)
{
  File.Delete("d:\\test.txt"); //удаление файла
}
```

Кроме того, чтобы читать/записывать данные в файл с Си-шарп можно использовать потоки.

Поток – это абстрактное представление данных (в байтах), которое облегчает работу с ними. В качестве источника данных может быть файл, устройство ввода-вывода, принтер.

Класс **Stream** является абстрактным базовым классом для всех потоковых классов в Си-шарп. Для работы с файлами нам понадобится класс **FileStream** (файловый поток).

FileStream - представляет поток, который позволяет выполнять операции чтения/записи в файл.

```
static void Main(string[] args)
{
  FileStream file = new FileStream("d:\\test.txt", FileMode.Open
, FileAccess.Read); //открывает файл только на чтение
}
```

Режимы открытия **FileMode**:
- *Append* – открывает файл (если существует) и переводит указатель в конец файла (данные будут дописываться в конец), или создает новый файл. Данный режим возможен только при режиме доступа FileAccess.Write.
- *Create* - создает новый файл(если существует – заменяет)
- *CreateNew* – создает новый файл (если существует – генерируется исключение)
- *Open* - открывает файл (если не существует – генерируется исключение)
- *OpenOrCreate* – открывает файл, либо создает новый, если его не существует
- *Truncate* – открывает файл, но все данные внутри файла затирает (если файла не существует – генерируется исключение)

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\file1.txt", FileMode.CreateNew); //создание нового файла
  FileStream file2 = new FileStream("d:\\file2.txt", FileMode.Open); //открытие существующего ф
айла
  FileStream file3 = new FileStream("d:\\file3.txt", FileMode.Append); //открытие файла на
дозапись в конец файла
}
```

Режим доступа **FileAccess**:

- *Read* – открытие файла только на чтение. При попытке записи генерируется исключение
- *Write* - открытие файла только на запись. При попытке чтения генерируется исключение
- *ReadWrite* - открытие файла на чтение и запись.

Чтение из файла

Для чтения данных из потока нам понадобится класс **StreamReader**. В нем реализовано множество методов для удобного считывания данных. Ниже приведена программа, которая выводит содержимое файла на экран:

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\test.txt", FileMode.Open); //создаем файловый поток
  StreamReader reader = new StreamReader(file1); // создаем «потоковый читатель» и связываем
его с файловым потоком
  Console.WriteLine(reader.ReadToEnd()); //считываем все данные с потока и выводим на экран
  reader.Close(); //закрываем поток
  Console.ReadLine();
}
```

Метод **ReadToEnd()** считывает все данные из файла. **ReadLine()** – считывает одну строку (указатель потока при этом переходит на новую строку, и при следующем вызове метода будет считана следующая строка).

Свойство **EndOfStream** указывает, находится ли текущая позиция в потоке в конце потока (достигнут ли конец файла). Возвращает *true* или *false*.

Запись в файл

Для записи данных в поток используется класс **StreamWriter**. Пример записи в файл:

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\test.txt", FileMode.Create); //создаем файловый поток
 StreamWriter writer = new StreamWriter(file1); //создаем «потоковый писатель» и связываем его с файловым потоком
  writer.Write("текст"); //записываем в файл
  writer.Close(); //закрываем поток. Не закрыв поток, в файл ничего не запишется
}
```

Метод **WriteLine()** записывает в файл построчно (то же самое, что и простая запись с помощью Write(), только в конце добавляется новая строка).

Нужно всегда помнить, что после работы с потоком, его нужно закрыть (освободить ресурсы), использовав метод **Close()**.

Кодировка, в которой будут считываться/записываться данные указывается при создании StreamReader/StreamWriter:

```
static void Main(string[] args)
{
  FileStream file1 = new FileStream("d:\\test.txt", FileMode.Open);
  StreamReader reader = new StreamReader(file1, Encoding.Unicode);
  StreamWriter writer = new StreamWriter(file1, Encoding.UTF8);
}
```

Кроме того, при использовании StreamReader и StreamWriter можно не создавать отдельно файловый поток FileStream, а сделать это сразу при создании StreamReader/StreamWriter:

```
static void Main(string[] args)
{
  StreamWriter writer = new StreamWriter("d:\\test.txt"); //указываем путь к файлу, а не поток
  writer.WriteLine("текст");
  writer.Close();
}
```

Как создать папку?

С помощью статического метода **CreateDirectory()** класса **Directory**:

```
static void Main(string[] args)
{
 Directory.CreateDirectory("d:\\new_folder");
}
```

Как удалить папку?

Для удаления папок используется метод **Delete()**:

```
static void Main(string[] args)
{
  Directory.Delete("d:\\new_folder"); //удаление пустой папки
}
```

Если папка не пустая, необходимо указать параметр рекурсивного удаления - true:

```
static void Main(string[] args)
{
  Directory.Delete("d:\\new_folder", true); //удаление папки, и всего, что внутри
}
```

Задание

Номер варианта	Задание
1	Дан файл *f*, компоненты которого являются целыми числами. Записать в файл *g*, компоненты файла *f*, исключив повторные вхождения чисел.
2	Дан файл *f*, компоненты которого являются действительными числами. Найти: 1. наибольшее из значений компонентов *f*; 2. наименьшее из значений компонентов с четными номерами; 3. наибольшее из значений модулей компонентов с нечетными номерами; 4. сумму наибольшего и наименьшего из значений компонентов файла *f*; 5. разность первой и последней компоненты файла *f*.
3	Дан символьный файл *f*. Подсчитать число вхождений в файл каждой из букв a, b, c, d, e, f. Результат вывести в файл *g* в виде таблицы с комментариями.
4	Дан файл *f*, компоненты которого являются целыми числами. Записать в файл *g* все четные числа исходного файла, в файл *h* – все нечетные. Порядок следования чисел сохраняется. Записать в файл *g* и *h* комментарии.
5	Дан текстовый файл, содержащий программу на языке С. Проверить эту программу на соответствие числа открывающих и закрывающих фигурных скобок.
6	Дан символьный файл *f*. Найти и записать в файл *g* самое длинное слово файла *f*, снабдив его комментарием.
7	Дан файл *f*, компоненты которого являются целыми числами. Получить в файле *g* все компоненты файла *f*: 1. являющиеся четными числами; 2. делящиеся на 3 и не делящиеся на 7; 3. являющиеся точными **квадратами**. Записать в файл *g* комментарий.
8	Дан файл *f*. Создать два файла, записав в первый из них все четные числа, расположив их в порядке возрастания, а во второй – все нечетные, расположив их в порядке убывания.
9	Дан текстовый файл *f*. Переформатировать исходный файл, разделяя его на строки так, чтобы каждая строка содержала столько символов, сколько содержит самая короткая строка исходного файла.
10	Дан файл *f*. Создать два файла, записав в первый из них среднее геометрическое всех четных чисел, а во второй – среднее арифметическое всех нечетных чисел.
11	Дан числовой файл *f*. Выбрать все значения, которые делятся нацело на 2 и 4, но не делятся на 6. Записать эти значения в файл *g*, а все остальные – в файл *h*.
12	Дан текстовый файл *f*. Определить, являются ли первые два символа цифрами и если да, то четно ли это число. Записать его в файл *g*, если оно четно и в *h* ,если оно нечетно.
13	Дан текстовый файл *f*. Создать новый файл *g* и переписать в него исходный файл в обратном порядке, разделив пробелами.
14	Дан текстовый файл *f*. Создать новый файл, включив в него только те строки исходного файла, которые содержат самый часто используемый символ исходного файла.
15	Дан текстовый файл, содержащий программу на языке С. Проверить эту программу на соответствие открывающих и закрывающих скобок разных типов - (), {}, [].

Задание второе: реализовать свой класс и перегрузить в нём операторы << и >> для работы со стандартными потоковыми классами. Использовать этот класс для решения задания вместе со стандартными потоковыми классами.

Номер варианта	Задание
1	Сформировать массив на диске, содержащий сведения о пациентах глазной клиники. Класс содержит поля: фамилия пациента, пол, возраст, место проживания (город), диагноз. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - количество иногородних, прибывших в поликлинику; - список пациентов старше X лет с диагнозом J.

2	Сформировать массив на диске, содержащий сведения о сотрудниках института. Класс содержит поля: фамилия работающего, название отдела, год рождения, стаж работы, должность, оклад. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - список сотрудников пенсионного возраста на сегодняшний день с указанием стажа работы; - средний стаж, работающих в отделе X.
3	Сформировать массив на диске, содержащий сведения об отправлении поездов дальнего следования с жд вокзала. Класс содержит поля: номер поезда, станция назначения, время отправления, время в пути, наличие билетов. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - время отравления поездов в город X во временном интервале от A до B часов; - наличие билетов на поезд с номером XXX.
4	Сформировать массив на диске, содержащий сведения о том, какие из пяти предлагаемых дисциплин по выбору желает изучать студент. Класс содержит поля: фамилия студента, индекс группы, пять дисциплин, средний балл успеваемости. Выбираемая дисциплина отмечается символом 1, иначе – пробелом. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран список студентов, желающих прослушать дисциплину X. Если число желающих превышает 4 человека, то отобрать студентов, имеющих более высокий средний балл успеваемости.
5	Сформировать массив на диске, содержащий сведения об игроках футбольной команды. Класс содержит поля: имена нападающих, число заброшенных ими шайб, число сделанных голевых передач, заработанное штрафное время. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран четырех лучших игроков по сумме очков (голы + передачи).
6	Сформировать массивна диске, содержащий сведения об ассортименте обуви в магазине фирмы. Класс содержит поля: артикул, наименование, количество, стоимость одной пары. Артикул начинается с буквы Д – для дамской обуви, М – для мужской, П – для детской. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - сведения о наличии и стоимости обуви артикула X; - ассортиментный список дамской обуви с указанием наименования и имеющегося в наличии числа пар каждой модели.
7	Сформировать массив на диске, содержащий сведения о наличии билетов на авиарейсы. Класс содержит поля: номер рейса, пункт назначения, время вылета, время прибытия, количество свободных мест в салоне. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - время вылета самолетов в город X; - наличие свободных мест на рейс в город X с временем отправления Y.
8	Сформировать массив на диске, содержащий сведения о личной коллекции книголюба. Класс содержит поля: шифр книги, автор, название, год издания, местоположение (номер стеллажа). Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - местоположение книги, автора X названия Y; - список книг автора Z, находящихся в коллекции; - число книг издания XX года, имеющихся в библиотеке.

9	Сформировать массив на диске, содержащий сведения о сдаче студентами сессии. Класс содержит поля: индекс группы, фамилия студента, оценки по пяти экзаменам. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - фамилии неуспевающих студентов с указанием индексов групп и количества задолженностей; - средний балл, полученный каждым студентом группы X, и всей группы в целом.
10	Сформировать массив на диске, содержащий сведения об ассортименте игрушек в магазине. Класс содержит поля: название игрушки, цена, количество, возрастные границы (2 - 5). Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - название игрушек, которые подходят детям от 1 до 3 лет; - стоимость самой дорогой игрушки и ее название; - название игрушки, которая по стоимости не превышает X руб. и подходит ребенку в возрасте от A до B лет. Значения A, B, X вводятся с клавиатуры.
11	Сформировать массив на диске, содержащий сведения о телефонах абонентов. Класс содержит поля: фамилия абонента, место жительства (название улицы, номер дома), год установки телефона. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - номер телефона по вводимой с клавиатуры фамилии абонента; - количество установленных телефонов с XXXX года; - список номеров телефонов, принадлежащих жильцам определенного дома и улицы. Номер года, название улицы и номер дома вводятся с клавиатуры.
12	Сформировать массив на диске, содержащий сведения о количестве изделий категорий A, B, C, собранных рабочим за месяц. Класс содержит поля: фамилия сборщика, наименование цеха, количество изделий по категориям, собранных рабочим за месяц. Считая заданными значения расценок SA, SB, SC за выполненную работу по сборке единицы изделия категорий A, B, C, выбрать необходимую информацию с диска и вывести на экран: - общее количество изделий категорий A, B, C, собранных рабочим цеха; средний размер заработной платы рабочих цеха X.
13	Сформировать массив на диске, содержащий сведения о количестве изделий, собранных сборщиками цеха за неделю. Класс содержит поля: фамилия сборщика, количество изделий, собранных им ежедневно в течение шестидневной недели, т.е. раздельно в понедельник, вторник и т.д. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - фамилию сборщика и общее количество деталей, собранных им за неделю; - фамилию сборщика собравшего наибольшее количество изделий, и день, когда он достиг наивысшей производительности труда.
14	Сформировать массив на диске, содержащий сведения о личной видеоколлекции. Класс содержит поля: шифр по IMDB, название студии, название фильма, год издания, категория, жанр. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - статистика по жанрам для студии Y; - список фильмов категории Z, находящихся в коллекции; - число фильмов XX года, имеющихся в коллекции.
15	Сформировать массив на диске, содержащий сведения о клиентах предприятия. Класс содержит поля: фамилия, имя, отчество, дата рождения, число покупок, потраченная сумма. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран: - список клиентов у которых на этой неделе день рождения; - список клиентов у которых покупок более чем на X; - средний размер покупки для клиентов пенсионного возраста (старше 60).

Лабораторное занятите № 2

Тема: Описание собственного класса на языке ООП

Цели и задачи урока: Изучить возможность обработки создания классов.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60х90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.
5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60х90 1/16. - (Среднее профессиональное образование).

Теоретические сведения

C# является полноценным объектно-ориентированным языком. Это значит, что программу на C# можно представить в виде взаимосвязанных взаимодействующих между собой объектов.

Описанием объекта является **класс**, а объект представляет экземпляр этого класса. Можно еще провести следующую аналогию. У нас у всех есть некоторое представление о человеке, у которого есть имя, возраст, какие-то другие характеристики. То есть некоторый шаблон - этот шаблон можно назвать классом. Конкретное воплощение этого шаблона может отличаться, например, одни люди имеют одно имя, другие - другое имя. И реально существующий человек (фактически экземпляр данного класса) будет представлять объект этого класса.

По умолчанию проект консольного приложения уже содержит один класс Program, с которого и начинается выполнение программы.

По сути класс представляет новый тип, который определяется пользователем. Класс определяется с помощью ключевого слова **class**:

```
class Person
{

}
```

Где определяется класс? Класс можно определять внутри пространства имен, вне пространства имен, внутри другого класса. Как правило, классы помещаются в отдельные файлы. Но в данном случае поместим новый класс в файле, где располагается класс Program. То есть файл Program.cs будет выглядеть следующим образом:

```
using System;

namespace HelloApp
{
    class Person
    {

    }
    class Program
    {
        static void Main(string[] args)
        {

        }
    }
}
```

Вся функциональность класса представлена его членами - полями (полями называются переменные класса), свойствами, методами, событиями. Например, определим в классе Person поля и метод:

```
using System;

namespace HelloApp
{
    class Person
    {
        public string name; // имя
        public int age = 18;     // возраст

        public void GetInfo()
        {
            Console.WriteLine($"Имя: {name}  Возраст: {age}");
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Person tom;
        }
    }
}
```

В данном случае класс Person представляет человека. Поле name хранит имя, а поле age - возраст человека. А метод GetInfo выводит все данные на консоль. Чтобы все данные были доступны вне класса Person переменные и метод определены с модификатором public. Поскольку поля фактически те же переменные, им можно присвоить начальные значения, как в случае выше, поле age инициализировано значением 18.

Так как класс представляет собой новый тип, то в программе мы можем определять переменные, которые представляют данный тип. Так, здесь в методе Main определена переменная tom, которая представляет класс Person. Но пока эта переменная не указывает ни на какой объект и по умолчанию она имеет значение **null**. Поэтому вначале необходимо создать объект класса Person.

Конструкторы

Кроме обычных методов в классах используются также и специальные методы, которые называются **конструкторами**. Конструкторы вызываются при создании нового объекта данного класса. Конструкторы выполняют инициализацию объекта.

Конструктор по умолчанию

Если в классе не определено ни одного конструктора, то для этого класса автоматически создается конструктор по умолчанию. Такой конструктор не имеет параметров и не имеет тела.

Выше класс Person не имеет никаких конструкторов. Поэтому для него автоматически создается конструктор по умолчанию. И мы можем использовать этот конструктор. В частности, создадим один объект класса Person:

```
class Person
{
    public string name; // имя
    public int age;     // возраст

    public void GetInfo()
    {
        Console.WriteLine($"Имя: {name}  Возраст: {age}");
    }
}
class Program
{
    static void Main(string[] args)
    {
        Person tom = new Person();
        tom.GetInfo();      // Имя: Возраст: 0

        tom.name = "Tom";
        tom.age = 34;
        tom.GetInfo();  // Имя: Tom Возраст: 34
        Console.ReadKey();
    }
}
```

Для создания объекта Person используется выражение new Person(). Оператор **new** выделяет память для объекта Person. И затем вызывается конструктор по

умолчанию, который не принимает никаких параметров. В итоге после выполнения данного выражения в памяти будет выделен участок, где будут храниться все данные объекта Person. А переменная tom получит ссылку на созданный объект.

Если конструктор не инициализирует значения переменных объекта, то они получают значения по умолчанию. Для переменных числовых типов это число 0, а для типа string и классов - это значение **null** (то есть фактически отсутствие значения).

После создания объекта мы можем обратиться к переменным объекта Person через переменную tom и установить или получить их значения, например, tom.name = "Tom";.

Консольный вывод данной программы:

```
Имя:          Возраст: 0
Имя: Tom  Возраст: 34
```

Создание конструкторов

Выше для инициализации объекта использовался конструктор по умолчанию. Однако мы сами можем определить свои конструкторы:

```
class Person
{
    public string name;
    public int age;

    public Person() { name = "Неизвестно"; age = 18; }      // 1 конструктор

    public Person(string n) { name = n; age = 18; }         // 2 конструктор

    public Person(string n, int a) { name = n; age = a; }   // 3 конструктор

    public void GetInfo()
    {
        Console.WriteLine($"Имя: {name}  Возраст: {age}");
    }
}
```

Теперь в классе определено три конструктора, каждый из которых принимает различное количество параметров и устанавливает значения полей класса. Используем эти конструкторы:

```
static void Main(string[] args)
{
    Person tom = new Person();          // вызов 1-ого конструктора без параметров
    Person bob = new Person("Bob");     //вызов 2-ого конструктора с одним параметром
    Person sam = new Person("Sam", 25); // вызов 3-его конструктора с двумя параметрами


    bob.GetInfo();          // Имя: Bob  Возраст: 18
    tom.GetInfo();          // Имя: Неизвестно  Возраст: 18
    sam.GetInfo();          // Имя: Sam  Возраст: 25
}
```

Консольный вывод данной программы:

```
Имя: Неизвестно  Возраст: 18
Имя: Bob  Возраст: 18
Имя: Sam  Возраст: 25
```

При этом если в классе определены конструкторы, то при создании объекта необходимо использовать один из этих конструкторов.

Стоит отметить, что начиная с версии C# 9.0 мы можем сократить вызов конструктора, убрав из него название типа:

```
Person tom = new ();           // аналогично new Person();
Person bob = new ("Bob");      // аналогично new Person("Bob");
Person sam = new ("Sam", 25);  // аналогично new Person("Sam", 25);
```

Ключевое слово this

Ключевое слово **this** представляет ссылку на текущий экземпляр класса. В каких ситуациях оно нам может пригодиться? В примере выше определены три конструктора. Все три конструктора выполняют однотипные действия - устанавливают значения полей name и age. Но этих повторяющихся действий могло быть больше. И мы можем не дублировать функциональность конструкторов, а просто обращаться из одного конструктора к другому через ключевое слово this, передавая нужные значения для параметров:

```
class Person
{
    public string name;
    public int age;

    public Person() : this("Неизвестно")
    {
    }
    public Person(string name) : this(name, 18)
    {
    }
    public Person(string name, int age)
    {
        this.name = name;
        this.age = age;
    }
    public void GetInfo()
    {
        Console.WriteLine($"Имя: {name}  Возраст: {age}");
    }
}
```

В данном случае первый конструктор вызывает второй, а второй конструктор вызывает третий. По количеству и типу параметров компилятор узнает, какой именно конструктор вызывается. Например, во втором конструкторе:

```
public Person(string name) : this(name, 18)
{
}
```

идет обращение к третьему конструктору, которому передаются два значения. Причем в начале будет выполняться именно третий конструктор, и только потом код второго конструктора.

Также стоит отметить, что в третьем конструкторе параметры называются также, как и поля класса.

```
public Person(string name, int age)
{
    this.name = name;
    this.age = age;
}
```

И чтобы разграничить параметры и поля класса, к полям класса обращение идет через ключевое слово this. Так, в выражении this.name = name; первая часть this.name означает, что name - это поле текущего класса, а не название параметра name. Если бы у нас параметры и поля назывались по-разному, то использовать слово this было бы необязательно. Также через ключевое слово this можно обращаться к любому полю или методу.

Инициализаторы объектов

Для инициализации объектов классов можно применять **инициализаторы**. Инициализаторы представляют передачу в фигурных скобках значений доступным полям и свойствам объекта:

```
Person tom = new Person { name = "Tom", age=31 };
tom.GetInfo();          // Имя: Tom  Возраст: 31
```

С помощью инициализатора объектов можно присваивать значения всем доступным полям и свойствам объекта в момент создания без явного вызова конструктора.

При использовании инициализаторов следует учитывать следующие моменты:

- С помощью инициализатора мы можем установить значения только доступных из внешнего кода полей и свойств объекта. Например, в примере выше поля name и age имеют модификатор доступа public, поэтому они доступны из любой части программы.
- Инициализатор выполняется после конструктора, поэтому если и в конструкторе, и в инициализаторе устанавливаются значения одних и тех же полей и свойств, то значения, устанавливаемые в конструкторе, заменяются значениями из инициализатора.

Задание

1. Создать класс с двумя переменными. Добавить функцию вывода на экран и функцию изменения этих переменных. Добавить функцию, которая находит сумму значений этих переменных, и функцию которая находит наибольшее значение из этих двух переменных.
2. Описать класс, реализующий десятичный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне. Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями. Счетчик имеет два метода: увеличения и уменьшения, — и свойство, позволяющее получить его текущее состояние. Написать программу, демонстрирующую все возможности класса.
3. Создать класс с двумя переменными. Добавить конструктор с входными параметрами. Добавить конструктор, инициализирующий члены класса по умолчанию. Добавить деструктор, выводящий на экран сообщение об удалении объекта.

4. Создать класс, содержащий динамический массив и количество элементов в нем. Добавить конструктор, который выделяет память под заданное количество элементов, и деструктор. Добавить методы, позволяющие заполнять массив случайными числами, переставлять в данном массиве элементы в случайном порядке, находить количество различных элементов в массиве, выводить массив на экран.
5. Составить описание класса для определения одномерных массивов строк фиксированной длины. Предусмотреть контроль выхода за пределы массива, возможность обращения к отдельным строкам массива по индексам, выполнения операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, а также вывод на экран элемента массива по заданному индексу и всего массива.
6. Составить описание класса многочленов от одной переменной, задаваемых степенью многочлена и массивом коэффициентов. Предусмотреть методы для вычисления значения многочлена для заданного аргумента, операции сложения, вычитания и умножения многочленов с получением нового объекта-многочлена, вывод на экран описания многочлена.
7. Построить три класса (базовый и 3 потомка), описывающих некоторых хищных животных (один из потомков), всеядных(второй потомок) и травоядных (третий потомок). Описать в базовом классе абстрактный метод для расчета количества и типа пищи, необходимого для пропитания животного в зоопарке.
   a) Упорядочить всю последовательность животных по убыванию количества пищи. При совпадении значений – упорядочивать данные по алфавиту по имени. Вывести идентификатор животного, имя, тип и количество потребляемой пищи для всех элементов списка.
   b) Вывести первые 5 имен животных из полученного в пункте a) списка.
   c) Вывести последние 3 идентификатора животных из полученного в пункте a) списка.
   d) Организовать запись и чтение коллекции в/из файл.
   e) Организовать обработку некорректного формата входного файла.
8. Описать класс «домашняя библиотека». Предусмотреть возможность работы с произвольным числом книг, поиска книги по какому-либо признаку (например, по автору или по году издания), добавления книг в библиотеку, удаления книг из нее, сортировки книг по разным полям.
9. Составить описание класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменение размеров, построение наименьшего прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью (пересечением) двух прямоугольников.
10. Создать класс для хранения комплексных чисел. Реализовать операции над комплексными числами: сложение, вычитание, умножение, деление, сопряжение, возведение в степень, извлечение корня. Предусмотреть возможность изменения формы записи комплексного числа: алгебраическая форма, тригонометрическая форма, экспоненциальная форма.
11. Составить описание класса для представления времени. Предусмотреть возможности установки времени и изменения его отдельных полей (час, минута, секунда) с проверкой допустимости вводимых значений. В случае недопустимых значений полей выбрасываются исключения. Создать методы изменения времени на заданное количество часов, минут и секунд.
12. Составить описание класса для вектора, заданного координатами его концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления

скалярного произведения двух векторов, длины вектора, косинуса угла между векторами.

13. Описать класс, представляющий треугольник. Предусмотреть методы для создания объектов, вычисления площади, периметра и точки пересечения медиан. Описать свойства для получения состояния объекта.
14. Создать абстрактный класс Figure с методами вычисления площади и периметра, а также методом, выводящим информацию о фигуре на экран. Создать производные классы: Rectangle (прямоугольник), Circle (круг), Triangle (треугольник) со своими методами вычисления площади и периметра.Создать массив n фигур и вывести полную информацию о фигурах на экран.
15. Класс **Покупатель**: Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета; Конструктор; Методы: установка значений атрибутов, получение значений атрибутов, вывод информации. Создать массив объектов данного класса. Вывести список покупателей в алфавитном порядке и список покупателей, у которых номер кредитной карточки находится в заданном диапазоне.

1. Класс **Абонент**: Идентификационный номер, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время междугородных и городских переговоров; Конструктор; Методы: установка значений атрибутов, получение значений атрибутов, вывод информации. Создать массив объектов данного класса. Вывести сведения относительно абонентов, у которых время городских переговоров превышает заданное. Сведения относительно абонентов, которые пользовались междугородной связью. Список абонентов в алфавитном порядке.

Лабораторное занятите № 3

Тема: Создание конструктора и деструктора

Цели и задачи урока: Изучить возможность реализации конструктора и деструктора.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня. 1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).

2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60x90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6

3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.

4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.

5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60х90 1/16. - (Среднее профессиональное образование).

Теоретические сведения

Конструкторы - это специальные методы, используемые при создании экземпляра класса. Конструктор не может ничего возвращать, поэтому нет необходимости определять для него возвращаемый тип. Обычно метод определяется следующим образом:

```
public string Describe()
```

Конструктор может быть определен так:

```
public Car()
```

В нашем примере для этой главы, у нас есть Класс автомобиля, с конструктором, который принимает строку в качестве аргумента. Конечно, конструктор также может быть перегружен, т. е. у нас может быть несколько конструкторов с одинаковым именем, но разными параметрами. Вот пример:

```
public Car()
{

}

public Car(string color)
{
   this.color = color;
}
```

Конструктор может вызвать другой конструктор, который пригодится в нескольких ситуациях. Вот пример:

```
Download sample code
public Car()
{
   Console.WriteLine("Constructor with no parameters called!");
}

public Car(string color) : this()
{
   this.color = color;
   Console.WriteLine("Constructor with color parameter called!");
}
```

Если вы запустите этот код, вы увидите, что конструктор без параметров вызывается первой. Это может использоваться для создания различных объектов для класса в конструкторе по умолчанию, который может быть вызван из других конструкторов из класса. Вы также можете это сделать в конструкторе, принимающем параметры. Вот простой пример:

```
Download sample code
public Car(string color) : this()
{
   this.color = color;
```

```
    Console.WriteLine("Constructor with color parameter called!");
}

public Car(string param1, string param2) : this(param1)
{

}
```

Если вы вызываете конструктор, который принимает 2 параметра, первый параметр будет использоваться для вызова конструктора, принимающего 1 параметр.

Деструкторы

Так как C# использует сборщик мусора, то, учитывая, что фреймворк освободит объекты, которые вы больше не используете, могут быть случаи, когда вам нужно сделать некоторую ручную очистку. Деструктор, метод, вызываемый после удаления объекта, может использоваться для очистки ресурсов, используемых объектом. Деструкторы не очень похожи на другие методы в C#. Вот пример деструктора для нашего класса автомобилей:

```
~Car()
{
   Console.WriteLine("Out..");
}
```

Задание

Для классов из предыдущей работы определить конструкторы и дкеструкторы

Лабораторное занятите № 4

Тема: Создание наследованных классов

Цели и задачи урока: Изучить возможность реализации наследуемых классов.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60х90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.

4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.
5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60x90 1/16. - (Среднее профессиональное образование).

Теоретические сведения

Наследование (inheritance) является одним из ключевых моментов ООП. Благодаря наследованию один класс может унаследовать функциональность другого класса.

Пусть у нас есть следующий класс Person, который описывает отдельного человека:

```
1     class Person
2     {
3         private string _name;
4
5         public string Name
6         {
7             get { return _name; }
8             set { _name = value; }
9         }
10        public void Display()
11        {
12            Console.WriteLine(Name);
13        }
14    }
```

Но вдруг нам потребовался класс, описывающий сотрудника предприятия - класс Employee. Поскольку этот класс будет реализовывать тот же функционал, что и класс Person, так как сотрудник - это также и человек, то было бы рационально сделать класс Employee производным (или наследником, или подклассом) от класса Person, который, в свою очередь, называется базовым классом или родителем (или суперклассом):

```
1     class Employee : Person
2     {
3
4     }
```

После двоеточия мы указываем базовый класс для данного класса. Для класса Employee базовым является Person, и поэтому класс Employee наследует все те же свойства, методы, поля, которые есть в классе Person. Единственное, что не передается при наследовании, это конструкторы базового класса.

Таким образом, наследование реализует отношение **is-a** (является), объект класса Employee также является объектом класса Person:

```
1    static void Main(string[] args)
2    {
3        Person p = new Person { Name = "Tom"};
4        p.Display();
5        p = new Employee { Name = "Sam" };
6        p.Display();
7        Console.Read();
8    }
```

И поскольку объект Employee является также и объектом Person, то мы можем так определить переменную: Person p = new Employee().

По умолчанию все классы наследуются от базового класса **Object**, даже если мы явным образом не устанавливаем наследование. Поэтому выше определенные классы Person и Employee кроме своих собственных методов, также будут иметь и методы класса Object: ToString(), Equals(), GetHashCode() и GetType().

Все классы по умолчанию могут наследоваться. Однако здесь есть ряд ограничений:

- Не поддерживается множественное наследование, класс может наследоваться только от одного класса.
- При создании производного класса надо учитывать тип доступа к базовому классу - тип доступа к производному классу должен быть таким же, как и у базового класса, или более строгим. То есть, если базовый класс у нас имеет тип доступа **internal**, то производный класс может иметь тип доступа **internal** или **private**, но не **public**.

  Однако следует также учитывать, что если базовый и производный класс находятся в разных сборках (проектах), то в этом случае производый класс может наследовать только от класса, который имеет модификатор public.

- Если класс объявлен с модификатором **sealed**, то от этого класса нельзя наследовать и создавать производные классы. Например, следующий класс не допускает создание наследников:

```
1    sealed class Admin
2    {
3    }
```

- Нельзя унаследовать класс от статического класса.

Задание

Для классов из предыдущей работы определить классы наследники

Лабораторное занятите № 5

Тема: Динамическое создание объектов

Цели и задачи урока: Изучить возможность динамического создания обектов.

Тип урока – практическое занятие.
Методы обучения – *репродуктивный*
Обеспечение занятия – персональный компьютер.
Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60x90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.
5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60x90 1/16. - (Среднее профессиональное образование).

Теоретические сведения

Динамические объекты предоставляют такие элементы, как свойства и методы, во время выполнения, а не во время компиляции. Это позволяет создавать объекты для работы со структурами, не соответствующими статическому типу или формату. Например, можно использовать динамический объект для ссылки на модель DOM HTML, которая может содержать любую комбинацию допустимых элементов и атрибутов разметки HTML.

Задание

Разработайте собственную задачу в соотвествии с вариантом.
Создание пользовательского динамического класса

1. Запустите Visual Studio.
2. Выберите **Создать новый проект**.
3. В диалоговом окне **Создание нового проекта** выберите C# или Visual Basic, выберите **Консольное приложение**, а затем нажмите кнопку **Далее**.
4. В диалоговом окне **Настройка нового проекта** введите значение DynamicSample для параметра **Имя проекта** и нажмите кнопку **Далее**.
5. В диалоговом окне **Дополнительные сведения** выберите значение **.NET 5.0 (текущая)** для параметра **Целевая платформа**, а затем нажмите кнопку **Создать**. Создается новый проект.
6. В **обозревателе решений** щелкните проект DynamicSample правой кнопкой мыши и выберите **Добавить** > **Класс**. В поле **Имя** введите ReadOnlyFile, а затем нажмите кнопку **Добавить**.
   Будет добавлен новый файл, содержащий класс ReadOnlyFile.
7. В верхней части файла *ReadOnlyFile.cs* или *ReadOnlyFile.vb* добавьте следующий код для импорта пространств имен System.IO и System.Dynamic.

```
using System.IO;
using System.Dynamic;
```

8. Пользовательский динамический объект использует перечисление для определения условия поиска. Перед оператором класса добавьте следующее определение перечисления.

```
public enum StringSearchOption
{
   StartsWith,
   Contains,
   EndsWith
}
```

9. Обновите оператор класса, чтобы он наследовал класс DynamicObject, как показано в следующем примере кода.

C#Копировать

```
class ReadOnlyFile : DynamicObject
```

10. Добавьте в класс ReadOnlyFile следующий код, чтобы задать закрытое поле для пути к файлу и конструктор для класса ReadOnlyFile.

```
// Store the path to the file and the initial line count value.
private string p_filePath;

// Public constructor. Verify that file exists and store the path in
// the private variable.
public ReadOnlyFile(string filePath)
{
   if (!File.Exists(filePath))
   {
      throw new Exception("File path does not exist.");
   }

   p_filePath = filePath;
}
```

11. Добавьте следующий метод GetPropertyValue в класс ReadOnlyFile.

Метод GetPropertyValue принимает в качестве входных данных условие поиска и возвращает строки текстового файла, соответствующие этому условию. Динамический метод, предоставленный классом ReadOnlyFile, вызывает метод GetPropertyValue для извлечения соответствующих результатов.

```
public List<string> GetPropertyValue(string propertyName,
                                      StringSearchOption StringSearchOption =
StringSearchOption.StartsWith,
                                      bool trimSpaces = true)
{
   StreamReader sr = null;
   List<string> results = new List<string>();
   string line = "";
   string testLine = "";

   try
   {
      sr = new StreamReader(p_filePath);
```

```
            while (!sr.EndOfStream)
            {
               line = sr.ReadLine();

               // Perform a case-insensitive search by using the specified search options.
               testLine = line.ToUpper();
               if (trimSpaces) { testLine = testLine.Trim(); }

               switch (StringSearchOption)
               {
                  case StringSearchOption.StartsWith:
                     if (testLine.StartsWith(propertyName.ToUpper())) { results.Add(line); }
                     break;
                  case StringSearchOption.Contains:
                     if (testLine.Contains(propertyName.ToUpper())) { results.Add(line); }
                     break;
                  case StringSearchOption.EndsWith:
                     if (testLine.EndsWith(propertyName.ToUpper())) { results.Add(line); }
                     break;
               }
            }
         }
         catch
         {
            // Trap any exception that occurs in reading the file and return null.
            results = null;
         }
         finally
         {
            if (sr != null) {sr.Close();}
         }

         return results;
      }
```

12. После метода GetPropertyValue добавьте следующий код, чтобы переопределить метод TryGetMember класса DynamicObject. Метод TryGetMember вызывается при запросе члена динамического класса без указания аргументов. Аргумент binder содержит сведения об элементе, на который дается ссылка, а аргумент result ссылается на результат, возвращенный для указанного элемента. Метод TryGetMember возвращает логическое значение true, если запрошенный элемент существует. В противном случае возвращается false.

```
// Implement the TryGetMember method of the DynamicObject class for dynamic member calls.
public override bool TryGetMember(GetMemberBinder binder,
                                  out object result)
{
   result = GetPropertyValue(binder.Name);
   return result == null ? false : true;
}
```

13. После метода TryGetMember добавьте следующий код, чтобы переопределить метод TryInvokeMember класса DynamicObject. Метод TryInvokeMember вызывается при запросе члена динамического класса с аргументами. Аргумент binder содержит сведения об элементе, на который дается ссылка, а аргумент result ссылается на результат, возвращенный для указанного элемента. Аргумент args содержит массив аргументов, передаваемых в элемент. Метод TryInvokeMember возвращает логическое значение true, если запрошенный элемент существует. В противном случае возвращается false. Пользовательская версия метода TryInvokeMember ожидает, что первый аргумент будет значением из перечисления StringSearchOption, заданного на предыдущем шаге. Метод TryInvokeMember ожидает, что второй аргумент будет логическим значением. Если один или оба элемента имеют допустимые значения, они передаются в метод GetPropertyValue для получения результатов.
C#Копировать

```
// Implement the TryInvokeMember method of the DynamicObject class for
// dynamic member calls that have arguments.
public override bool TryInvokeMember(InvokeMemberBinder binder,
                                     object[] args,
                                     out object result)
{
    StringSearchOption StringSearchOption = StringSearchOption.StartsWith;
    bool trimSpaces = true;

    try
    {
        if (args.Length > 0) { StringSearchOption = (StringSearchOption)args[0]; }
    }
    catch
    {
        throw new ArgumentException("StringSearchOption argument must be a
StringSearchOption enum value.");
    }

    try
    {
        if (args.Length > 1) { trimSpaces = (bool)args[1]; }
    }
    catch
    {
        throw new ArgumentException("trimSpaces argument must be a Boolean value.");
    }

    result = GetPropertyValue(binder.Name, StringSearchOption, trimSpaces);

    return result == null ? false : true;
}
```

14. Сохраните и закройте файл.

Создание примера текстового файла

1. В **обозревателе решений** щелкните проект DynamicSample правой кнопкой мыши и выберите **Добавить** > **Новый элемент**. На панели **Установленные шаблоны** выберите **Общие**, а затем шаблон **Текстовый файл**. В поле **Имя** оставьте имя

по умолчанию *TextFile1.txt* и нажмите кнопку **Добавить**. В проект добавится новый текстовый файл.

2. Скопируйте в файл *TextFile1.txt* следующий текст.

```
textКопировать
List of customers and suppliers

Supplier: Lucerne Publishing (https://www.lucernepublishing.com/)
Customer: Preston, Chris
Customer: Hines, Patrick
Customer: Cameron, Maria
Supplier: Graphic Design Institute (https://www.graphicdesigninstitute.com/)
Supplier: Fabrikam, Inc. (https://www.fabrikam.com/)
Customer: Seubert, Roxanne
Supplier: Proseware, Inc. (http://www.proseware.com/)
Customer: Adolphi, Stephan
Customer: Koch, Paul
```

3. Сохраните и закройте файл.

Создание примера приложения, в котором применяется пользовательский динамический объект

1. В **обозревателе решений** дважды щелкните файл *Program.vb*, если используется Visual Basic, или файл *Program.cs*, если используется Visual C#.
2. Добавьте следующий код в процедуру Main, чтобы создать экземпляр класса ReadOnlyFile для файла *TextFile1.txt*. В этом коде используется позднее связывание для вызова динамических элементов и извлечения строк текста, которые содержат строку "Customer".

```
dynamic rFile = new ReadOnlyFile(@"..\..\..\TextFile1.txt");
foreach (string line in rFile.Customer)
{
   Console.WriteLine(line);
}
Console.WriteLine("----------------------------");
foreach (string line in rFile.Customer(StringSearchOption.Contains, true))
{
   Console.WriteLine(line);
}
```

3. Сохраните файл и нажмите клавиши CTRL+F5 для сборки и запуска приложения.

Вызов библиотеки динамического языка

В следующем пошаговом руководстве создается проект, который осуществляет доступ к библиотеке, написанной на динамическом языке IronPython.

Создание пользовательского динамического класса

1. В Visual Studio выберите **Файл** > **Создать** > **Проект**.
2. В диалоговом окне **Создание нового проекта** выберите C# или Visual Basic, выберите **Консольное приложение**, а затем нажмите кнопку **Далее**.
3. В диалоговом окне **Настройка нового проекта** введите значение DynamicIronPythonSample для параметра **Имя проекта** и нажмите кнопку **Далее**.
4. В диалоговом окне **Дополнительные сведения** выберите значение **.NET 5.0 (текущая)** для параметра **Целевая платформа**, а затем нажмите кнопку **Создать**. Создается новый проект.
5. Установите пакет NuGet IronPython.

6. Если вы используете Visual Basic, отредактируйте файл *Program.vb*. Если вы используете Visual C#, отредактируйте файл *Program.cs*.
7. В верхней части файла добавьте следующий код для импорта пространств имен Microsoft.Scripting.Hosting и IronPython.Hosting из библиотек IronPython и пространства имен System.Linq.

```
using System.Linq;
using Microsoft.Scripting.Hosting;
using IronPython.Hosting;
```

8. В методе Main добавьте следующий код, чтобы создать объект Microsoft.Scripting.Hosting.ScriptRuntime, в котором будут размещены библиотеки IronPython. Объект ScriptRuntime загружает модуль библиотеки IronPython random.py.

C#Копировать

```
// Set the current directory to the IronPython libraries.
System.IO.Directory.SetCurrentDirectory(
   Environment.GetFolderPath(Environment.SpecialFolder.ProgramFiles) +
   @"\IronPython 2.7\Lib");

// Create an instance of the random.py IronPython library.
Console.WriteLine("Loading random.py");
ScriptRuntime py = Python.CreateRuntime();
dynamic random = py.UseFile("random.py");
Console.WriteLine("random.py loaded.");
```

9. После указания в коде необходимости загрузки модуля random.py добавьте следующий код, чтобы создать массив целых чисел. Массив передается методу shuffle модуля random.py, который произвольно сортирует значения в массиве.

```
// Initialize an enumerable set of integers.
int[] items = Enumerable.Range(1, 7).ToArray();

// Randomly shuffle the array of integers by using IronPython.
for (int i = 0; i < 5; i++)
{
   random.shuffle(items);
   foreach (int item in items)
   {
      Console.WriteLine(item);
   }
   Console.WriteLine("--------------------");
}
```

10. Сохраните файл и нажмите клавиши CTRL+F5 для сборки и запуска приложения.

Лабораторное занятите № 6

Тема: Использование виртуальных методов

Цели и задачи урока: Изучить возможность работы с виртуальными методоми.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)

4. Подведение итогов

Литература

1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60х90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.
5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60х90 1/16. - (Среднее профессиональное образование).

Теоретические сведения

При наследовании нередко возникает необходимость изменить в классе-наследнике функционал метода, который был унаследован от базового класса. В этом случае класс-наследник может переопределять методы и свойства базового класса.

Те методы и свойства, которые мы хотим сделать доступными для переопределения, в базовом классе помечается модификатором **virtual**. Такие методы и свойства называют виртуальными.

А чтобы переопределить метод в классе-наследнике, этот метод определяется с модификатором **override**. Переопределенный метод в классе-наследнике должен иметь тот же набор параметров, что и виртуальный метод в базовом классе.

Например, рассмотрим следующие классы:

```
class Person
{
   public string Name { get; set; }
   public Person(string name)
   {
      Name = name;
   }
   public virtual void Display()
   {
      Console.WriteLine(Name);
```

```
    }
}
class Employee : Person
{
    public string Company { get; set; }
    public Employee(string name, string company) : base(name)
    {
        Company = company;
    }
}
```

Здесь класс Person представляет человека. Класс Employee наследуется от Person и представляет сотрудника предприятия. Этот класс кроме унаследованного свойства Name имеет еще одно свойство - Company.

Чтобы сделать метод Display доступным для переопределения, этот метод определен с модификатором **virtual**. Поэтому мы можем переопределить этот метод, но можем и не переопределять. Допустим, нас устраивает реализация метода из базового класса. В этом случае объекты Employee будут использовать реализацию метода Display из класса Person:

```
static void Main(string[] args)
{
    Person p1 = new Person("Bill");
    p1.Display(); // вызов метода Display из класса Person

    Employee p2 = new Employee("Tom", "Microsoft");
    p2.Display(); // вызов метода Display из класса Person

    Console.ReadKey();}
```

Задание

Создайте три класса: Воин, Воин_в_легких_доспехах и Воин_в_тяжелых_доспехах. У всех них есть свойство – Количество_жизней, а также метод Получить_урон, который принимает в качестве аргумента значение получаемого урона. Реализуйте этот метод по-разному для всех типов, установив различные коэффициенты в зависимости от типа доспехов в формуле вычета здоровья.

Лабораторное занятите № 7

Тема: Создание модуля доступа к БД

Цели и задачи урока: Организация доступа к данным.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. 1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).

2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60x90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6

3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.

4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.

5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60x90 1/16. - (Среднее профессиональное образование).

Задание

1. Создайте базу данных в MS SQL Serwer
2. Создайте таблицу в базе данных
3. Теперь подключимся к ней из приложения. В любом проекте WPF, как и в ряде других типов проектов для .NET, по умолчанию есть файл конфигурации, который называется *app.config* и который имеет следующее содержимое:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6" />
    </startup>
</configuration>
```

4. Добавим в него строку подключения к бд, изменив файл следующим образом:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6" />
    </startup>
    <connectionStrings>
    <add name="DefaultConnection"
        connectionString="Data Source=.\SQLEXPRESS;Initial Catalog=mobiledb;Integrated Security=Tru
        providerName="System.Data.SqlClient"/>
    </connectionStrings>
</configuration>
```

Для определения всех подключений в программе в пределах узла <configuration> добавляется новый узел <connectionStrings>. В этом узле определяются строки подключения с помощью элемента <add>. Каждая строка подключения имеет название, определяемое с помощью атрибута name. В данном случае строка подключения называется "DefaultConnection". Название может быть произвольное.

Атрибут connectionString собственно хранит строку подключения. Он состоит из трех частей:

- Data Source=.\SQLEXPRESS: указывает на название сервера. По умолчанию для MS SQL Server Express используется ".\SQLEXPRESS"
- Initial Catalog=mobiledb: название базы данных. Так как база данных называется mobiledb, то соответственно здесь данное название и указываем
- Integrated Security=True: задает режим аутентификации

Так как мы будем подключаться к базе данных MS SQL Server, то соответственно мы будем использовать провайдер для SQL Server, функциональность которого заключена в пространстве имен System.Data.SqlClient.

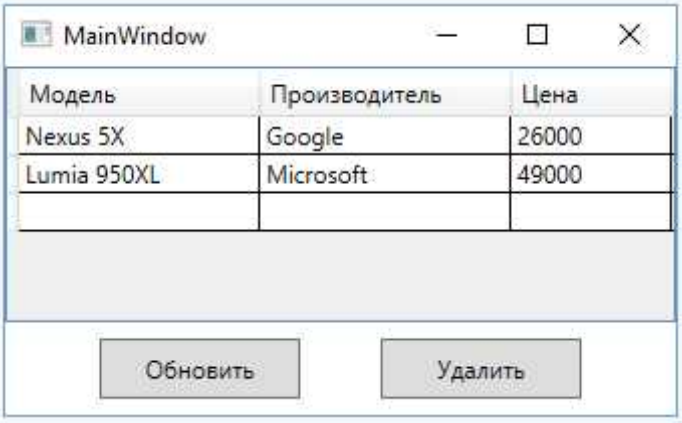
Далее определим код графического интерфейса в xaml:

```
<Window x:Class="DbApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:DbApp"
        mc:Ignorable="d"
        Title="MainWindow" Height="250" Width="350" Loaded="Window_Loaded">
    <Window.Resources>
        <Style TargetType="Button">
            <Setter Property="Margin" Value="20 8 20 8" />
            <Setter Property="Width" Value="100" />
            <Setter Property="Height" Value="30" />
        </Style>
    </Window.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>
        <DataGrid AutoGenerateColumns="False" x:Name="phonesGrid">
            <DataGrid.Columns>
                <DataGridTextColumn Binding="{Binding Title}" Header="Модель" Width="120"/>
                <DataGridTextColumn Binding="{Binding Company}" Header="Производитель" Width="12
                <DataGridTextColumn Binding="{Binding Price}" Header="Цена" Width="80"/>
            </DataGrid.Columns>
```

```
        </DataGrid>

        <StackPanel HorizontalAlignment="Center"  Grid.Row="1" Orientation="Horizontal">
            <Button x:Name="updateButton" Content="Обновить" Click="updateButton_Click" />
            <Button x:Name="deleteButton" Content="Удалить" Click="deleteButton_Click" />
        </StackPanel>
    </Grid>
</Window>
```

Здесь определен довольно простой интерфейс: датагрид для отображения данных, и две кнопки для обновления данных в бд и для удаления. В итоге приложение будет выглядеть следующим образом:



5. Теперь определим код подключения и все обработчики кнопок в файле кода c#:

```
System;
System.Windows;
System.Windows.Controls;
System.Data.SqlClient;
System.Data;
System.Configuration;

space DbApp

lic partial class MainWindow : Window

tring connectionString;
qlDataAdapter adapter;
ataTable phonesTable;

ublic MainWindow()

  InitializeComponent();
  // получаем строку подключения из app.config
  connectionString =
```

```
gurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;

)rivate void Window_Loaded(object sender, RoutedEventArgs e)

  string sql = "SELECT * FROM Phones";
  phonesTable = new DataTable();
  SqlConnection connection=null;
  try
  {
     connection = new SqlConnection(connectionString);
     SqlCommand command = new SqlCommand(sql, connection);
     adapter = new SqlDataAdapter(command);

     // установка команды на добавление для вызова хранимой процедуры
     adapter.InsertCommand = new SqlCommand("sp_InsertPhone", connection);
     adapter.InsertCommand.CommandType = CommandType.StoredProcedure;
     adapter.InsertCommand.Parameters.Add(new SqlParameter("@title", SqlDbType.NVarChar,
Title"));
     adapter.InsertCommand.Parameters.Add(new SqlParameter("@company",
)Type.NVarChar, 50, "Company"));
     adapter.InsertCommand.Parameters.Add(new SqlParameter("@price", SqlDbType.Int, 0,
e"));
     SqlParameter parameter = adapter.InsertCommand.Parameters.Add("@Id", SqlDbType.Int, 0,

     parameter.Direction = ParameterDirection.Output;

     connection.Open();
     adapter.Fill(phonesTable);
     phonesGrid.ItemsSource = phonesTable.DefaultView;
  }
  catch(Exception ex)
  {
     MessageBox.Show(ex.Message);
  }
  finally
  {
     if(connection!=null)
        connection.Close();
  }

)rivate void UpdateDB()

  SqlCommandBuilder comandbuilder = new SqlCommandBuilder(adapter);
  adapter.Update(phonesTable);
```

```
private void updateButton_Click(object sender, RoutedEventArgs e)

  UpdateDB();

private void deleteButton_Click(object sender, RoutedEventArgs e)

  if (phonesGrid.SelectedItems != null)
  {
     for (int i = 0; i < phonesGrid.SelectedItems.Count; i++)
     {
        DataRowView datarowView = phonesGrid.SelectedItems[i] as DataRowView;
        if (datarowView != null)
        {
           DataRow dataRow = (DataRow)datarowView.Row;
           dataRow.Delete();
        }
     }
  }
  UpdateDB();
   }
```

5. Вся работа с бд производится стандартными средствами ADO.NET и прежде всего классом SqlDataAdapter. Вначале мы получаем в конструкторе строку подключения, которая определена выше в файле *app.config*:

```
connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;
```

6. Чтобы задействовать эту функциональность, нам надо добавить в проект библиотеку **System.Configuration.dll**.

Далее в обработчике загрузки окна Window_Loaded создаем объект SqlDataAdapter:

```
adapter = new SqlDataAdapter(command);
```

7. В качестве команды для добавления объекта устанавливаем ссылку на хранимую процедуру:

```
adapter.InsertCommand = new SqlCommand("sp_InsertPhone", connection);
```

Получаем данные из БД и осуществляем привязку:

```
adapter.Fill(phonesTable);
phonesGrid.ItemsSource = phonesTable.DefaultView;
```

За обновление отвечает метод UpdateDB():

```
private void UpdateDB()
{
   SqlCommandBuilder comandbuilder = new SqlCommandBuilder(adapter);
   adapter.Update(phonesTable);
}
```

8. Чтобы обновить данные через SqlDataAdapter, нам нужна команда обновления, которую можно получить с помощью объекта SqlCommandBuilder. Для самого обновления вызывается метод adapter.Update().

Причем не важно, что мы делаем в программе - добавляем, редактируем или удаляем строки. Метод adapter.Update сделает все необходимые действия. Дело в том, что при загрузке данных в объект DataTable система отслеживает состояние загруженных строк. В методе adapter.Update() состояние строк используется для генерации нужных выражений языка SQL, чтобы выполнить обновление базы данных. Более подробно про обновление с помощью адаптеров данных можно почитать здесь: Обновление БД из DataSet вручную

В обработчике кнопки обновления просто вызывается этот метод UpdateDB, а в обработчике кнопки удаления предварительно удаляются все выделенные строки.

Таким образом, мы можем вводить в DataGrid новые данные, редактировать там же уже существующие, сделать множество изменений, и после этого нажать на кнопку обновления, и все эти изменения синхронизируются с базой данных.

Причем важно отметить действие хранимой процедуры - при добавлении нового объекта данные уходят на сервер, и процедура возвращает нам id добавленной записи. Этот id играет большую роль при генерации нужного sql-выражения, если мы захотим эту запись изменить или удалить. И если бы не хранимая процедура, то нам пришлось бы после добавления данных загружать заново всю таблицу в datagrid, только чтобы у новой добавленной записи был в datagrid id. И хранимая процедура избавляет нас от этой работы.

9. Также здесь мы могли бы выполнять обновление данных сразу после редактирования строки. Для этого нужно задействовать событие RowEditEnding элемента DataGrid:

```
public MainWindow()
{
   InitializeComponent();

   connectionString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString
   phonesGrid.RowEditEnding += PhonesGrid_RowEditEnding;
}

private void PhonesGrid_RowEditEnding(object sender, DataGridRowEditEndingEventArgs e)
{
   UpdateDB();
```

}

И если после окончания редактирования мы нажмем на Enter, то срабатает обработчик события RowEditEnding, который обновит базу данных.

Лабораторное занятите № 8

Тема: Создание запросов БД

Цели и задачи урока: Изучить возможность отображать результаты выборок из баз данных.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. 1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60x90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.
5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60x90 1/16. - (Среднее профессиональное образование).

Теоретические сведения

Задание

1. Добавить текчтовое поле для фильтрации даных по столбцу Модель
2. Добавить поле со списком для фильтрации по производителю

Лабораторное занятите № 9

Тема: Создание модуля вывода информации БД на печать

Цели и задачи урока: Изучить возможность печати значений из бд.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы

3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. 1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60х90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.
5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60х90 1/16. - (Среднее профессиональное образование).

Теоретические сведения

Вывод на печать различных бланков и форм документов востребован во многих программах.

Стандартным решением для данной задачи является использование специализированных генераторов отчётов, которые на основе заданного макета формируют заполненный данными документ для отправки на печать.

Современные генераторы отчётов обладают мощным и гибким функционалом, но к сожалению, они не всегда доступны. Например, вследствие того, что среда разработки не имеет в своём составе генератора отчётов или его функционал не отвечает требованиям проекта, а приобретение генераторов отчётов сторонних разработчиков связано с большими затратами или не оправдано так как их возможности будут использоваться лишь в незначительной степени.

Как поступить в подобной ситуации?

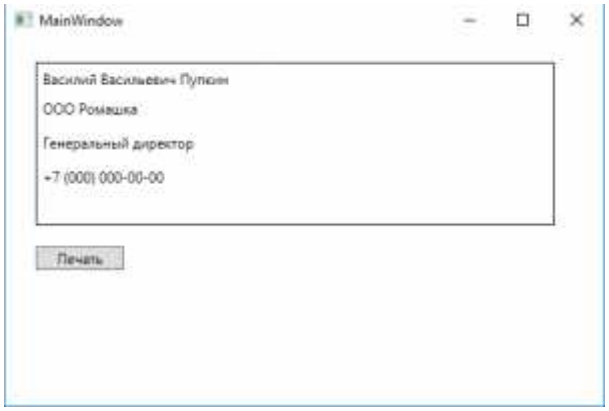
Ответ на этот вопрос во многом зависит от платформы на которой выполняется разработка. Но, в случае .NET писать свой генератор отчётов или скрепя сердце тратить огромные суммы на сторонние продукты не обязательно потому что в роли генератора отчётов вполне может выступить и сам фреймворк WPF.

Дело в том, что в WPF поддерживает вывод элементов XAML на печать в том виде, в котором они отображаются в работающей программе.

Задание

Создайте есть окно, в котором отображается макет визитной карточки и эту карточку требуется распечатать.

Внешний вид окна показан на скриншоте:



Код его разметки на XAML:

```
<Window
      xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
      xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
      xmlns:d=http://schemas.microsoft.com/expression/blend/2008
      xmlns:mc=http://schemas.openxmlformats.org/markup-compatibility/2006
      xmlns:local="clr-namespace:WpfApp1"
      xmlns:Forms="clr-namespace:System.Windows.Forms;assembly=System.Windows.Forms"
x:Class="WpfApp1.MainWindow"
      mc:Ignorable="d"
      Title="MainWindow" Height="350" Width="525">
   <Grid>
      <Border x:Name="card"  BorderBrush="Black" BorderThickness="1"
HorizontalAlignment="Left" Height="138" Margin="26,20,0,0" VerticalAlignment="Top"
Width="441">
         <Grid>
            <Label x:Name="FIO" Content="Василий Васильевич Пупкин"/>
            <Label x:Name="Company" Content="ООО Ромашка" Margin="0,25,0,82"/>
            <Label x:Name="Position" Content="Генеральный директор" Margin="0,54,0,53"/>
            <Label x:Name="Phone" Content="+7 (000) 000-00-00" Margin="0,83,0,24"/>
         </Grid>
      </Border>
      <Button x:Name="printBtn" Content="Печать" HorizontalAlignment="Left"
Margin="26,176,0,0" VerticalAlignment="Top" Width="75" Click="printBtn_Click"/>
   </Grid>
</Window>
```

В WPF можно напечатать как окно целиком, так и его отдельные элементы. Из представленного кода разметки следует, что визитная карточка представляет собой элемент Border со вложенными в него элементами. Поэтому именно его мы и будем выводить на печать.

Для печати содержимого окна используется обычный PrintDialog, а точнее его метод PrintVisual. Этот метод принимает два параметра: элемент, который нужно напечатать и название задания напечатать, которое будет отображаться в очереди печати Windows в виде строки (строка может быть пустой).

```
PrintDialog dialog = new PrintDialog();
```

```
dialog.PrintVisual(this.card, "Визитная карточка");
```

Важно отметить, что при вызове метода PrintVisual диалоговое окно не отображается и задание на печать сразу же отправляется на принтер по умолчанию.

Для того чтобы получить возможность подтвердить или отказаться от печати или выбрать нужный принтер следует предварительно воспользоваться методом ShowDialog, как показано ниже.

```
PrintDialog dialog = new PrintDialog();
if (dialog.ShowDialog() == true)
{
   dialog.PrintVisual(this.card, "Визитная карточка");
}
```

Поддерживается вывод как на обычный, так и на виртуальный принтер. Ниже представлен скриншот результата печати визитной карточки из вышеприведённой программы в формат PDF.

Для печати вовсе не обязательно, чтобы элемент относился непосредственно к окну. Он может находиться на пользовательском элементе управления и т.д.

Также элемент, предназначенный для печати может содержать не только текстовые поля, но и любые другие элементы (изображения, графики и пр.) и при этом технология вывода на печать не изменится.

Поэтому используя XAML для создания макетов печатных форм, можно легко обойтись и без стороннего генератора отчётов реализовав требуемый функционал стандартными средствами.

Лабораторное занятите № 10

Тема: Установка и настройка CMS. Тестирование сайта.

Цели и задачи урока: Изучить возможность работы с cms.

Тип урока – практическое занятие.

Методы обучения – *репродуктивный*

Обеспечение занятия – персональный компьютер.

Ход урока

1. Объяснение нового материала
2. Выполнение практической работы
3. Закрепление пройденного материала (ответы на контрольные вопросы)
4. Подведение итогов

Литература

1. 1. Немцова Т.И., Голоса С.Ю., Тереньтев А.И. Программирование на языке высокого уровня: учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев / под ред. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2019. – 512 с.: ил. – (Профессиональное образование).
2. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: Учебное пособие. / Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. - 336 с.: 60х90 1/16. - (Среднее профессиональное образование) (Переплёт 7БЦ) ISBN 978-5-906818-41-6
3. Голицына О.Л., Попов И.И., Программирование на языках высокого уровня Голицына О.Л., Попов И.И., – М.: Форум 2020.
4. Михеева Е.В. Информационные технологии в профессиональной деятельности : учеб. пособие для студ. сред. проф. образования. – 8-е изд., стер. - М.: Издательский центр «Академия», 2019. - 384 с.
5. Федорова Г. Н. Разработка модулей программного обеспечения для компьютерных систем/ Федорова Г.Н. - М.:КУРС, НИЦ ИНФРА-М, 2020. -384 с.: 60х90 1/16. - (Среднее профессиональное образование).

Теоретические сведения

ЛВС (Локальная вычислительная сеть) - компьютерная сеть, покрывающая обычно относительно небольшую территорию или небольшую группу зданий (дом, офис, фирму, институт). Также существуют локальные сети, узлы которых разнесены географически на расстояния более 12 500 км (космические станции и орбитальные центры). Несмотря на такие расстояния, подобные сети всё равно относят к локальным.

Типы подключения в Oracle VM VirtualBox

Не подключен

В этом режиме, VirtualBox сообщает гостю что сетевой адаптер есть, но он не подключен – так как если бы Ethernet кабель не присоединен к карте. Таким образом возможно симулировать «выдергивание» кабеля из виртуальной сетевой карты и обрыв соединения,что может быть полезно для информирования гостевой ОС об отсутствии сетевого соединения, но возможности его настройки.

Network Address Translation (NAT)

Режим Network Address Translation (NAT) предоставляет наиболее простой способ доступа к внешней среде из виртуальной машины. Обычно, для него не требуется никаких настроек хоста и гостевой системы. Поэтому он является сетевым режимом, настраиваемым по умолчанию.

Виртуальная машина с сетевым интерфейсом в режиме NAT подключается к сети, также как реальный компьютер подключается к Internet через маршрутизатор. «Маршрутизатором» в данном случае выступает сетевой модуль VirtualBox, который обрабатывает сетевой трафик виртуальной машины. Недостаток режима NAT, как и в случае локальной сети за маршрутизатором, в том что виртуальная машина недоступна для внешней сети (internet); вы не можете обрабатывать сетевые запросы, пока не настроите переброс портов

Можно настроить сеть NAT, выбрав этот тип подключения в настройках сети, предварительно необходимо создать сеть NAT в Файл-Настройки-Сеть.

Сетевой мост

В данном режиме VirtualBox использует драйвер устройства на хост системе , который обрабатывает данные проходящие через физический сетевой интерфейс. Этот драйвер обычно называют «net filter». Он позволяет перехватывать VirtualBox пакеты из физической сети и изменять данные в них, а также создавать новые программные сетевые интерфейсы. Когда гость использует такой программный интерфейс, то выглядит это так как будто бы гостевая

система подключается к физической сети, хост система может посылать и принимать данные от гостевой. Это означает, что вы можете использовать физический интерфейс хоста в качестве маршрутизатора или шлюза между гостевой системой и вашей физической сетью.

Внутренняя сеть

Режим Внутренняя сеть похож на режим Сетевой мост в котором ВМ может связываться с внешним миром. Однако, «внешний мир» ограничен другими виртуальными машинами на том же хосте и которые подключены к той же внутренней сети.

Хотя с технической стороны, все что можно сделать используя режим внутренняя сеть , можно также сделать в режиме сетевого моста, но в режиме внутренней сети есть преимущества в безопасности. В режиме моста, весь трафик проходит через физический интерфейс хоста. Поэтому возможно подключение анализаторов пакетов (например, Wireshark) к интерфейсу хоста и сбор всего трафика проходящего через него. Если по каким либо причинам вам необходимо чтобы две и более виртуальные машины имели защищенное соединение, скрывая свои данные от хост системы и других пользователей, то режим сетевой мост не является лучшим выбором.

Внутренние сети создаются автоматически, т.е. не существует централизованной настройки . Каждая внутренняя сеть идентифицируется своим именем. При существовании более чем одной виртуальной сетевой карты с одним идентификатором внутренней сети, драйвер VirtualBox автоматически поддерживает соединение этих интерфейсов в одну сеть, выступая в роли сетевого коммутатора. VirtualBox поддерживает полностью Ethernet коммутацию, широковещательную и многоадресную рассылку сетевых пакетов и режим promiscuous mode.

Виртуальный адаптер хоста

Данный режим можно использовать для создания сетей из хоста и нескольких виртуальных машин, без использования физического сетевого интерфейса хоста. На хосте создается виртуальный сетевой интерфейс (подобный петлевому интерфейсу) , обеспечивающий соединения между хост системой и виртуальными машинами.

Данный режим можно рассматривать как гибрид режимов сетевого моста и внутренней сети: как и режиме моста виртуальные машины могут соединяться друг с другом и с хост системой, как будто бы они соединены через физический коммутатор. Как и в режиме внутренней сети, нет необходимости в предоставления физического сетевого интерфейса и виртуальные машины не могут общаться с внешней сетью хоста, т.к. они никак не связаны с физическим сетевым интерфейсом.

При использовании режима внутренней сети, VirtualBox создает новый программный интерфейс на хосте, который добавляется к списку существующих сетевых интерфейсов хоста. Другими словами, в режиме сетевого моста существующий физический интерфейс используется для подключения виртуальных машин, а в режиме «виртуальный адаптер хоста» создается новый «петлевой» интерфейс хоста. В режиме внутренней сети трафик между виртуальными машинами не «виден», а трафик «петлевого» интерфейса возможно перехватить.

Режим виртуального адаптера хоста удобно использовать для нескольких предварительно настроенных виртуальных систем, которые предназначены для совместного использования. Например, одна виртуальная машина представляет собой web сервер который использует вторую с сервером базы данных. Другой дополнительный, сетевой интерфейс (мост) может соединить web сервер с внешним миром для выдачи данных, но внешнему миру не будет доступа к серверу базы данных.

Универсальный драйвер

Редко используемый режим универсального сетевого интерфейса (Rarely used modes share the same generic network interface), позволяет выбирать пользователю драйвер, который может быть включен в VirtualBox или поставляться с пакетом расширений (extension pack).

Примечание

Подробнее о настройке сети в VirtualBox можно прочитать на сайте virtualbox.

Утилита ping

Команда PING используется для проверки состояния между двумя сетевыми соединениями. Эти соединения могут быть локальными, глобальными или подключения к интернету в целом. Команда PING посылает пакеты информации с определением IP-адреса, а затем измеряет время, которое потребуется, чтобы получить ответ от указанного компьютера или устройства.

ping [-LRUbdfnqrvVaAB] [-c количество] [-i интервал] [-l преднагрузка] [-p шаблон] [-s размер-пакета] [-t ttl] [-w ограничение-на-время-работы] [-F идентификатор-потока] [-I адрес] [-M указание] [-Q тип-обслуживания] [-S буфер-отправки] [-T параметр-временной-метки] [-W время-ожидания-ответа] [переход …] назначение

Используем команду:

`ping 192.168.58.103`, где 192.168.58.103 - ip адрес цели

Получим следующую информацию:

- IP адрес
- Количество отправленных байт
- Время, которое потребовалось для соединения в миллисекундах
- TTL – время жизни пакета данных в протоколе IP, предельно допустимое время его пребывания в системе.

Предупреждение

Необходимо нажать CTRL+C, чтобы остановить команду и показать результаты.

Чем больше круглое число соединения в миллисекундах, тем выше ожидания, что может свидетельствовать о сетевой проблеме между вашим компьютером и сервером. <ping> использует обязательные датаграммы ECHO_REQUEST протокола ICMP для получения по этому протоколу ответов ECHO_RESPONSE от хоста или шлюза. Датаграммы ECHO_REQUEST состоят из заголовков IP и ICMP, структуры данных struct timeval и произвольного числа не смысловых байтов для заполнения пакета.

CMS

Система управления содержимым (контентом) (англ. Content management system, CMS) — информационная система или компьютерная программа, используемая для обеспечения и организации совместного процесса создания, редактирования и управления контентом (то есть содержимым).

Основные функции CMS:

- Предоставление инструментов для создания содержимого, организация совместной работы над содержимым,
- Управление содержимым: хранение, контроль версий, соблюдение режима доступа, управление потоком документов и т. п.,
- Публикация содержимого,
- Представление информации в виде, удобном для навигации, поиска.
- В системе управления содержимым могут находиться самые различные данные: документы, фильмы, фотографии, номера телефонов, научные данные и так далее. Такая система часто используется для хранения, управления, пересмотра и публикации документации. Контроль версий является одним из основных её преимуществ, когда содержимое изменяется группой лиц.

WordPress

WordPress — система управления содержимым сайта с открытым исходным кодом, распространяемая под GNU GPL. Написана на PHP, в качестве сервера базы данных использует MySQL. Сфера применения — от блогов до достаточно сложных новостных ресурсов и интернет-магазинов. Встроенная система «тем» и «плагинов» вместе с удачной

архитектурой позволяет конструировать практически любые проекты. WordPress выпущен под лицензией GPL версии 2.

Требования:

- PHP version 5.6 or greater
- MySQL version 5.5 or greater

PHP (англ. PHP: Hypertext Preprocessor — «PHP: препроцессор гипертекста»; первоначально Personal Home Page Tools — «Инструменты для создания персональных веб-страниц»; произносится пи-эйч-пи) — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

Задание

- Установите на 4 виртуальные машины операционную систему Ubuntu Server. Условно назовем эти машины: Hacker, Server, WWW, DataBase.

- Настроите сеть. В настройках сети (По умолчанию Файл-> настройки-> сеть, на вкладке Виртуальные сети хоста) создаем 2 адаптера (По умолчанию 1 уже создан).
- Для каждого из них прописать разные IP-адреса 192.168.*.*, где * - любое число от 0 до 255 (Если хотите сделать все как на схеме, на первом оставьте значения по умолчанию - 192.168.56.1, а на втором 192.168.57.1). На уже имеющимся адаптере можете посмотреть настройки DHCP и, по аналогии, настроить DHCP для второго адаптера. DHCP-сервер будет выдавать всем подключенным в сеть машинам IP-адреса автоматически через определенные промежутки времени. В данной лабораторной работе настройка DHCP в VirtualBox никак не отразиться на ее выполнении, а наоборот, только упростит построение сети между виртуальными машинами.
- Теперь в настройках каждой из виртуальных машин выберите вкладку сеть.
- В машине Hacker создайте 2 адаптера. В первом, чтобы вы могли использовать интернет, выберите тип подключения NAT. Во втором - виртуальный адаптер хоста (если Вы делаете все по схеме выберите тот, где ip-адрес 192.168.56.*).
- На машине Server создайте 3 адаптера. Первый - чтобы использовать интернет. Второй - тот же виртуальный адаптер хоста, что и в машине Hacker. Третий - виртуальный адаптер хоста, но выбираете уже второй (если вы делаете все по схеме выбираете тот, где ip-адрес 192.168.57.*).
- На машинах WWW и DataBase создайте 2 адаптера. Первый - выход в интернет. Второй - виртуальный адаптер хоста (второй, тот где 196.168.57.*).

- Особенностью подключения типа виртуальный адаптер хоста, является то, что компьютер, на котором запущен VirtualBox так же доступен, что может помочь во второй части лабораторной работы.
- Настройка статического IP-адреса (если используется на DHCP). Настройка сети осуществляется с помощью создания виртуального адаптера хоста. При первичном запуске всех виртуальных машин необходимо внести изменения в файл /etc/network/interfaces.
- nano /etc/network/interfaces
- В нем необходимо прописать новое соединение в форме:
- auto eth1
- iface eth1 inet static
- address 192.168.58.103
- netmask 255.255.255.0
- gateway 192.168.58.102

Поле address — ip адрес, выделяющийся машине, netmask — маска сети, используемая для разграничения адреса сети и непосредственно машины в ней, а gateway — шлюз, на который пойдут отправляемые пакеты. В данной лабораторной работе, например, в поле gateway пишется ip-адрес виртуального адаптера хоста на сервере, соответствующего локальной сети. Поэтому gateway не нужно прописывать для сервера, если этот сервер сам не подключается к какому-либо другому серверу. Запись осуществляется с правами суперпользователя. После записи необходима перезагрузка системы.

Вы можете использовать те же ip-адреса, что указаны на схеме. Для машины Server в настройках пропишите два сетевых интерфейса eth1 и eth2, из-за двух виртуальных адаптеров хоста. Названия сетевых интерфейсов начинаются с eth0, и во всех машинах, настраиваемых нами, eth0 будет автоматически использован первым виртуальным адаптером, который мы используем для выхода в интернет.

- Чтобы изменения вступили в силу необходимо перезапустить систему.
- Просмотреть сетевые интерфейсы вы можете, использовав команду ifconfig
- После настройки, используйте команду ping и проверьте локальное соединение.
- Машина Hacker – «злоумышленник», который попытается просканировать нашу сеть. Для того, чтобы он смог это сделать, необходимо установить универсальное средство сканирования – Nmap:
- sudo apt-get install nmap
- Server – сервер, атаку на который совершает Hacker. Во второй части работы будем устанавливать на него CMS WordPress.
- WWW – web-сервер, с помощью которого осуществляется доступ к базе данных DataBase. Соответственно, как и для Server, устанавливаем web-сервер. Также необходимо установить PHP.
- sudo apt-get install php5 libapache2-mod-php5 php5-mysql

Далее, необходимо создать файл /var/www/index.php, в который прописывается следующий скрипт:

```
<?php
$link = mysql_connect('192.168.57.101', 'sit', 'sit');
if (!$link) {
die('Error: ' . mysql_error());
}
echo 'Ok';

mysql_close($link);
?>
```

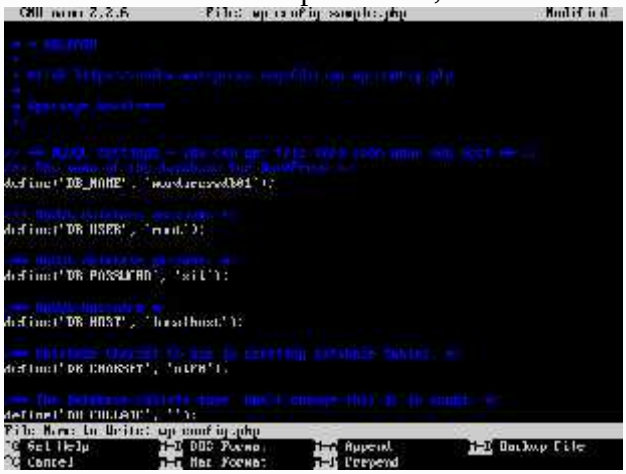
Этот скрипт определяет подключение WWW к DataBase, и если ввести в адресное поле браузера ip-адрес WWW /index.php, то при успешном подключении к базе данных будет выведено «OK».

- Настройка базы данных. На DataBase необходимо установить Mysql – универсальную систему управления базами данных.
- sudo apt-get install mysql-server
- Запускаем MySQL:
- sudo mysql -p
- Затем, в MySql необходимо добавить набор привилегий для пользователя
- (по умолчанию sit)
- ‘‘GRANT ALL PRIVILEGES ON *.* TO sit@localhost IDENTIFIED‘‘
- BY ’sit’ WITH GRANT OPTION
- ‘‘GRANT ALL PRIVILEGES ON *.* TO sit@"%" IDENTIFIED BY‘‘
- ’sit’ WITH GRANT OPTION
- Далее необходимо в файле /etc/mysql/my.cnf найти и закомментировать (поставить в начале строки символ #) строчку bind-address = 127.0.0.1.

Часть 2

- Установку CMS необходимо производить на виртуальную машину Server из первой части лабораторной работы.
- Для настройки CMS WordPress, виртуальная машина Server должна быть подключена по локальной сети с машиной с графической оболочкой.
- На виртуальную машину Server необходимо установить набор програмного обеспечения LAMP (Linux Apache MySQL PHP)
- sudo apt-get update
-
- sudo apt-get install tasksel
-
- sudo tasksel
- Выберите LAMP. Чтобы отметить выделите и нажмите пробел. И нажмите Enter. Задайте пароль root (Используйте *sit*)
- Установите phpmyadmin <sudo apt-get install phpmyadmin>. Выберите Apache, выберите Yes, введите пароль.
- На второй машине в браузере введите ip-адрес/phpmyadmin (Например 192.168.1.113/phpmyadmin)
- Введите имя пользователя и пароль. (Пользователь по умолчанию - root. Пароль - тот который вводили при установке phpmyadmin)
- Перейдите на вкладку Databases. Создайте Новую базу данных. В поле под надписью **Create Database** введите название базы данных (wordpressdb01) и нажмите **Create**.
- Вернитесь к виртуальной машине Server. Скачайте WordPress <wget http://wordpress.org/latest.tar.gz>
- Распакуйте скачанный архив в /var/www/html <sudo tar -xvpzf latest.tar.gz -C /var/www/html/>
- Теперь на второй машине введите 192.168.1.113/wordpress и нажимайте на кнопки, чтобы перейти к дальнейшим настройкам, пока не дойдете до настроек БД.
- В настройках введите имя БД - wordpressdb01, имя пользователя БД - root и пароль - sit. Нажимаете Submit.
- Вам выводят, каким должен быть быть файл wp-config.php в папке wordpress. И чтобы создать его и настроить, переходим к виртуальной машине Server.
- Перейдите в папку /var/www/html/wordpress

- Откройте в текстовом редакторе wp-config-sample.php <nano wp-config-samle.php>
- Измените настройки так, как показано на скриншоте:



- Сохраните под именем wp-cjnfig.php в той же папке. Когда будете нажимать CTRL-O и вас будут спрашивать имя сохраняемого файла, поменяйте на wp-config.php
- Возвращаемся ко второй машине и браузеру, нажимаем кнопки для дальнейшей установки.
- Заполняем настройки. Настройки потом можно будет поменять, неважно, что вы там в данный момент напишите, но запомните имя и пароль, вам их предстоит использовать сражу же после настройки, чтобы залогиниться.
- Нажимаете Install WordPress.
- Нажимаете Log In, вводите имя и пароль, которые вы только что задали.
- На этом установка CMS WordPress закончена.

Вопросы к лабораторной работе

1. Что такое ЛВС?
2. Как проверить есть ли локальное сетевое соединение?
3. Что такое ping?
4. Какую информацию можно получить командой ping?
5. Перечислите типы подключения в Oracle VM VirtualBox.
6. Как создать локальную сеть в VirtualBox?
7. Что такое NAT?
8. Какие ограничения есть у NAT?
9. Если в настройках двух виртуальных машин выбрать *Сетевой мост*, будут ли они подключены по локальной сети?
10. В чем отличия между типами подключений *Сетевой мост* и *Внутренняя сеть*?
11. Что такое PHP?
12. Что такое MySQL?
13. Для чего нужна маска сети?