МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ДАГЕСТАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет Информатики и информационных технологий

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Кафедра информатики и информационных технологий

Образовательная программа

09.03.02 «Информационные системы и технологии»

Профиль подготовки
Общий профиль

Уровень высшего образования
Бакалавриат

Форма обучения
очная

Статус дисциплины: **вариативная по выбору**

Махачкала, 2020

Рабочая программа дисциплины «Современные технологии программирования» составлена в 2020 году в соответствии с требованиями ФГОС ВО по направлению подготовки 09.03.02 Информационные системы и технологии, уровень бакалавриат, утвержденного приказом Минобрнауки РФ от 12 марта 2015 г. №219.

Разработчик: Магомедова С.Р., старший преподаватель

Рабочая программа дисциплины одобрена:
на заседании кафедры ИиИТ_____от «13» 03 2020г., протокол № 8
Зав.кафедрой_____Ахмедов С.А.
(подпись)

на заседании Методической комиссии факультета ИиИТ
от «12» 03 2020г., протокол № 8
Председатель_____Ахмедова З.Х.
(подпись)

Рабочая программа дисциплины согласована с учебно-методическим управлением «26» 03 2020 г. _____
(подпись)

Аннотация рабочей программы дисциплины

Дисциплина «Современные технологии программирования» входит в базовую часть образовательной программы бакалавриата по направлению 09.03.02 «Информационные системы и технологии»

Дисциплина реализуется на факультете Информатики и ИТ кафедрой Информатики и ИТ.

Содержание дисциплины охватывает круг вопросов, связанных с изучением современных технологий и методов программирования, основных принципов объектно-ориентированного программирования, механизмов доступа к базам данных и работы с ними, приобретением практических навыков использования современных инструментальных средств для разработки, отладки и тестирования создаваемых прикладных программ.

Дисциплина нацелена на формирование следующих компетенций выпускника:, профессиональных – ПК-37.

Преподавание дисциплины предусматривает проведение следующих видов учебных занятий: *лекции, практические занятия, лабораторные занятия, самостоятельная работа.*

Рабочая программа дисциплины предусматривает проведение следующих видов контроля успеваемости в форме *контрольной работы или тестирования* и промежуточный контроль в форме экзамена.

Объем дисциплины 3 зачетных единиц, в том числе в академических часах по видам учебных занятий

Семестр		Учебные занятия							Форма промежуточной аттестации (зачет, дифференцированный зачет, экзамен
		в том числе							
	Всего	Контактная работа обучающихся с преподавателем						СРС, в том числе экзамен	
		Всего	из них						
			Лекции	Лабораторные занятия	Практические занятия	КСР	консультации		
7	108	54	18	18	18			54	экзамен

1. Цели освоения дисциплины

Целями освоения дисциплины «Современные технологии программирования» является подготовка к самостоятельной профессиональной работе, ознакомление с методами и технологиями программирования, умение ориентироваться во всем многообразии технологий программирования, умение применять практические навыки использования инструментальных и прикладных технологий в различных отраслях техники, экономики, управления и бизнеса.

2. Место дисциплины в структуре образовательной программы:

Программа дисциплины «Современные технологии программирования» разработана в соответствии с федеральным государственным стандартом высшего образования по направлению подготовки бакалавриата 09.03.02 «Информационные системы и технологии», утвержденным приказом Минобрнауки России от 12 марта 2015 г. № 219_, вступил в силу 30 марта 2015 г.

Дисциплина входит в базовую часть (БД.Б.3.6) образовательной программы бакалавриата по направлению 09.03.02 «Информационные системы и технологии» профиля «Информационные системы и технологии в образовании», изучается в 75 семестре. Объем дисциплины: 3 ЗЕ / 108 часов, в том числе 54 часов - контактная работа с преподавателем,54 часа - самостоятельная работа.

Для изучения данной учебной дисциплины необходимы следующие знания, умения и навыки, формируемые предшествующими дисциплинами:

Из курса «Алгоритмические языки и системы программирования»:

Знания: ядро языка программирования высокого уровня, его синтаксис и семантику; основы проектирования программ: типовые алгоритмы.

Умения: описывать разработанные программы посредством блок схем, тестировать и отлаживать разработанные программы; реализовывать на языке программирования высокого уровня типовые алгоритмы: табуляцию функций, формирование таблиц, нахождение сумм, среднего и т.п.; поиск экстремума, работу с датчиком случайных чисел, ввод и вывод одномерных и двумерных массивов, поиск элементов в массиве, обработку массивов с выводом таблиц, сортировку, ввод и вывод текстов, сравнение фрагментов текста, изменение фрагмента текста по определенному правилу, запись информации в файл, чтение информации из файла, поиск и изменение информации в файле по заданному условию.

Владения: приемами работы в среде программирования (составление, отладка и тестирование программ; разработка и использование интерфейсных объектов)

Из курса «Высокоуровневые методы информатики и программирования»:

Знания: основные подходы к разработке программ; основные методы программирования; принципы отношения между классами – наследование, универсализация и их роль в построении программных систем.

Умения: проводить декомпозицию; использовать средства разработки для создания и отладки программного обеспечения; использовать готовые программные решения.

Владения: приемами и методами проектирования программ, основанных на классах; приемами объектно-ориентированного анализа; приемами работы в современных средах программирования.

Перечень последующих учебных дисциплин, для которых необходимы знания, умения и владения, формируемые данной учебной дисциплиной:

- Web-программирование;
- Методы и средства проектирования информационных систем и технологий.

3. Компетенции обучающегося, формируемые в результате освоения дисциплины (перечень планируемых результатов обучения).

Дисциплина направлена на формирование компетенций ОК-1, ПК-4, ПК-20 и планируемых результатов обучения.

Код компетенции из ФГОС ВО	Наименование компетенции из ФГОС ВО	Планируемые результаты обучения
ПК-37	способностью выбирать и оценивать способ реализации информационных систем и устройств (программно-, аппаратно- или программно-аппаратно-) для решения поставленной задачи	Знает: технологии выбора и оценки способов реализации ИС Умеет: выбирать и оценивать существующие технологии разработки ИС для решения поставленной задачи Владеет: практическими навыками выбора и оценки современных технологий проектирования и разработки ИС для решения поставленной задачи

4. Объем, структура и содержание дисциплины.

4.1. Объем дисциплины составляет 3 зачетных единиц, 108 академических часов.

4.2. Структура дисциплины.

№ п/п	Разделы и темы дисциплины	Семестр	Неделя семестра	Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)				Самостоятельная работа	Формы текущего контроля успеваемости *(по неделям семестра)* Форма промежуточной аттестации *(по семестрам)*
				Лекции	Практические занятия	Лабораторные занятия	Контроль самост. раб.		
	Модуль 1. *Введение в объектно-ориентированное программирование на C#*								
1	ООП как подход к программированию	7	1	2	2			6	
2	Основные понятия ООП в C#: объекты, классы и методы	7	2-3	2	2	2		6	Выполнение и защита лаборат. работ 1-2
3	Понятие инкапсуляции, наследования, полиморфизма и его применение в C#	7	4-5	2	2	2			
	Итого по модулю 1:			6	4	4		12	Тестирование по мод
	Модуль 2. *Платформа .NET Framework и ее применение для ООП*								
5	Основные понятия платформы .NET Framework	7	6-7	2	2	2		6	Выполнение и защита лаборат. работ 3-6
6	Платформа ASP.NET. Система типизации в .NET	7	8-9	2	2	2		6	
7	Событийно управляемое программирование в .NET	7	10-11	2	2	2		6	
8	Компонентное программирование в .NET	7	12-13	2	2	2		6	
	Итого по модулю 2:			8	8	8		24	Тестирование по мод
	Модуль 3. *Технологии Веб программирования и ADO.NET*								
7	Введение в Web приложения. Создание приложений Web Forms	7	14-15	2	2	2		6	Выполнение и защита лаборат. работ 7-9
9	Работа с базами данных в .NET. Технология ADO.NET. Автономный и	7	16-17	2	2	2		6	

	подключенный уровни								
	Итого по модулю 3:			4	6	6		18	Тестирование по мод
	ИТОГО:			18	18	18		54	

4.3. Содержание дисциплины, структурированное по темам (разделам).

4.3.1. Содержание лекционных занятий по дисциплине

Модуль 1. *Введение в объектно-ориентированное программирование на C#*

Тема 1. ООП как подход к программированию

Введение в современные подходы к программированию. Современные подходы к программированию. Особенности декларативного подхода. Особенности процедурного подхода. Особенности функционального подхода. Основные понятия ООП. Абстракция, инкапсуляция, наследование и полиморфизм. Преимущества и недостатки ООП

Тема 2. Основные понятия ООП в C#: объекты, классы и методы

Интуитивные определения объектов, классов и методов. Соотношение понятий объекта и класса. Концептуальная модель для классов и объектов.

Классы, объекты, свойства и методы в языке C#. Классы и структуры в языке C#. Конструкторы и деструкторы классов в языке C#. Преимущества и недостатки объектных теорий.

Тема 3. Понятие инкапсуляции, наследования, полиморфизма и его применение в C#

Инкапсуляция в ООП. Примеры инкапсуляции в языке C#. Виды областей видимости объектов. Рекомендации по разграничению областей видимости. Преимущества инкапсуляции

Наследование в ООП. Отношение частичного порядка. Базовые и производные классы в C#. Иерархия классов в .NET. Отображение классов .NET в типы языка C#

Понятие полиморфизма в ООП. Примеры полиморфизма в C#. Виды полиморфизма. Абстрактные типы данных. Методы вызова процедур. Преимущества программирования с полиморфизмом. Абстрактные структуры данных, методы, свойства и индексаторы в C#. Интерфейсы в языке C# и их связь с абстрактными классами. Реализация интерфейсов на основе классов и структур

Модуль 2. *Платформа .NET Framework и ее применение для ООП*

Тема 4. Основные понятия платформы .NET Framework

.NET как концепция. .NET как вычислительная модель. .NET как технологическая платформа. .NET как инструментальное средство. Common Language Runtime и .NET Framework. Система типов Common Type System в .NET. Веб-сервисы в .NET.

Компонентное программирование в .NET. Сравнение компонентного программирования с ООП. Преимущества и недостатки .NET

Тема 5. Платформа ASP.NET. Система типизации в .NET

Неформальное и формальное определения типов. Преимущества теорий с типами. Классификация систем типизации. Система типов (Common Type System, CTS) в .NET. Базисные типы языков C# и их отображение в CTS. Пространства имен. Преобразования типов в .NET.

Структура Web-приложений. Структура .NET Framework. Компоненты Web-форм. Языки программирования в .NET Framework

Тема 6. Событийно управляемое программирование в .NET

Понятие события в программировании. Методы моделирования событий. Фреймы и функции как модели событий.

Делегаты в языке C#. Конструкторы для делегатов в языке C#. Делегаты с множественным вызовом в языке C#

События как особый вид делегатов. Исключения и их обработка в языке C#.

Тема 7. Компонентное программирование в .NET

Компонентный подход к программированию как расширение ООП. Обзор архитектурного решения .NET. Понятия сборки и манифеста в .NET

Пространства имен в .NET. Гетерогенное компонентное программирование в .NET.

Модуль 3. *Технологии Веб программирования и ADO.NET*

Тема 8. Введение в Web приложения. Создание приложений Web Forms

Типы Интернет-приложений. Принцип работы Web-приложений. Возможности и преимущества ASP. NET.

Создание проекта Web-приложения. Управление проектом при помощи IIS. Обработка событий eb-приложения. Обработка данных. Управление процессами.

Тема 9. Работа с базами данных в .NET. Технология ADO.NET. Автономный и подключенный уровни

Обзор ADO.NET. Архитектура ADO.NET. Создание базы данных. Генератор поставщиков данных. Подключение к базе данных. Команды

Вставка, удаление, обновление записей в базе данных. Хранимые процедуры. Транзакции баз данных.

Понятие автономного уровня ADO.NET. Основные свойства и методы класса DataSet. Работа с объектами DataColumn. Работа с объектами DataRown. Работа с объектами DataTable

4.3.2. Содержание лабораторно-практических занятий по дисциплине.

Модуль 1. *Введение в объектно-ориентированное программирование на C#*

Лабораторная работа № 1.

Задание 1. Разработать проект «Печать фотографий», который позволяет рассчитать стоимость печати фотографий.

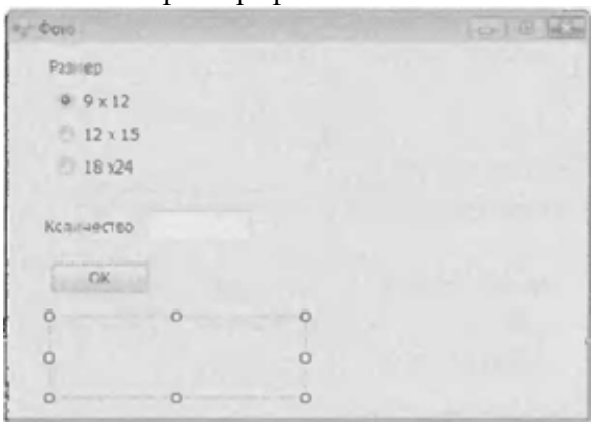
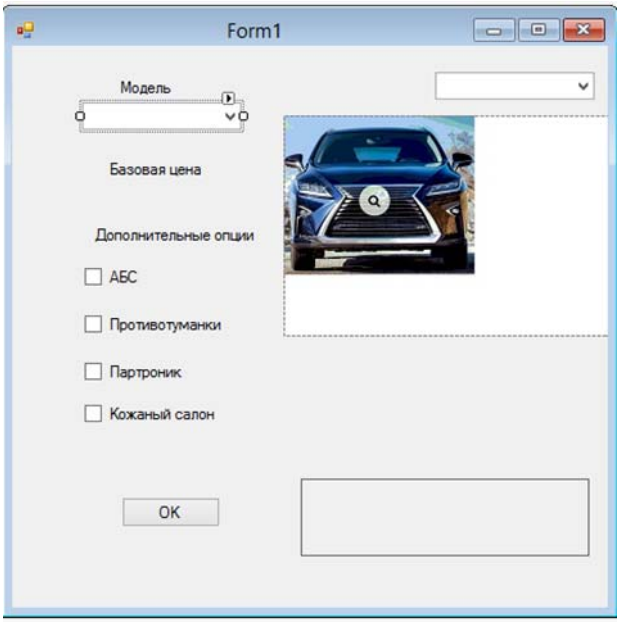


Рис. 1.4. Форма программы Фото

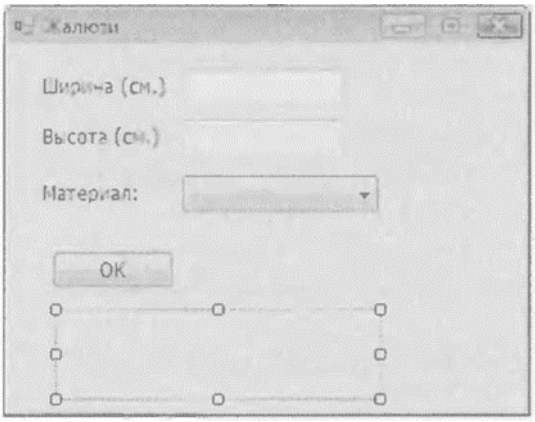
```
public Form1()
{
InitializeComponent();
// на стройка компонентов
radioButton1.Checked = true;
```

```
buttonl.Enabled = false;
}
// щелчок на кнопке OK
private void button1_Click(object sender, EventArgs e)
{
}
private void textBox1_TextChanged(object sender, EventArgs e)
{
if  (textBox1.Text.Length == 0)
button1.Enabled = false;
else
button1.Enabled = true;
label2.Text = "";
}
// щелчок на radioButton
private void radioButton1 Click(object sender, EventArgs e)
label2.Text=   "";
// установить курсор в поле Количество
textBox1.Focus();
}
}
```

Задание 2. Разработать проект **«Комплектация автомобиля»** позволяет рассчитать стоимость автомобиля в зависимости от выбранной комплектации. Отображение картинки обеспечивает компонент PictureBox.

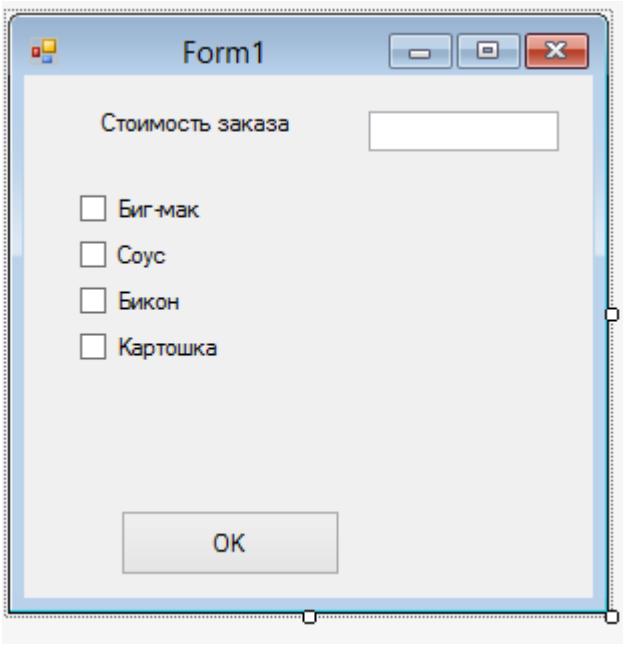


Задание 3. Разработать проект **«Жалюзи»**



Лабораторная работа 2.

Задание 1. Разработать проект Windows Form **Кафе,** ее форма приведена на рис, которая демонстрирует использование компонента CheckBox.



Форма программы **Кафе**

Задание 2. Разработка проекта «Любимые напитки»

Цель работы

Написать программу демонстрирующую работу с объектами ListBox.

Спецификация программы:

На пользовательской форме должны быть расположены 3 списка. Один из списков содержит названия напитков. Пользователь может переносить элементы из этого списка в списки «Любимые» и «Нелюбимые» и обратно. При этом перемещаемый элемент должен удалятся из списка-источника. То есть, например, перенос элемента «Чай» из общего списка в список «Любимые» происходит в следующем порядке: он добавляется в список «Любимые» и удаляется из общего списка. Таким образом общее количество элементов всех трех списков остается постоянным. Перенос элементов между списками должен осуществляться по нажатию на соответствующие кнопки, либо мышью (система Drag and Drop).

Система Drag and Drop позволяет напрямую перетаскивать объекты между разными источниками, например, из одного списка в другой. «Перетаскивание» представляет собой нажатие и удерживание левой кнопки мыши на объекте и дальнейшее его перемещение за курсором в желаемую область.

Списки «Любимые» и «Нелюбимые» должны сохраняться в текстовые файлы.

Для создания формы использовать компоненты:
Label – для подписей
ListBox – для вывода списков
Button – для инициирования действий
Рекомендуемая компоновка формы программы представлена на рисунке.

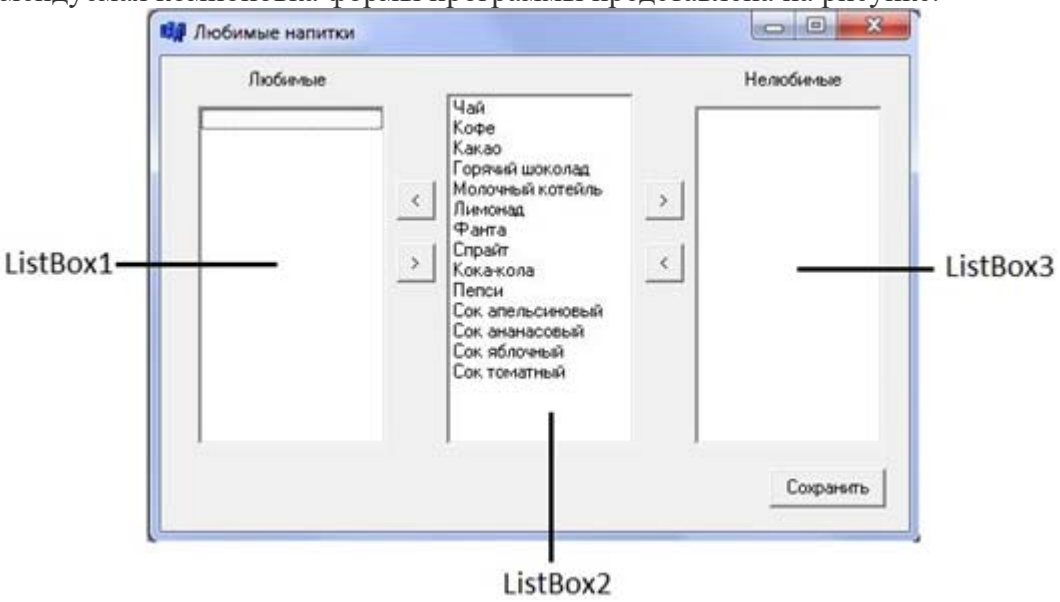


Рисунок. Рекомендуемая компоновка формы

Модуль 2. ***Платформа .NET Framework и ее применение для ООП***

Лабораторная работа № 3. Простые классы

Цель работы: Изучение правил разработки классов.

1. Постановка задачи

Создать новый проект и сохранить под под названием ***LabRab9.***

Имеются сведения об автомобилях и их владельцах:

- *Государственный регистрационный номер.*
- *Model.*
- *Color.*
- *FIO владельца.*

Требуется разработать приложение, позволяющее вводить указанные данные, хранить введенную информацию и удалять сведения об автомобиле и его владельце.

Интерфей спользователя

Список госномеров хранится в объекте ***comboBox1***. Поля ввода используются соответственно ***textBox1*** – ***textBox4***, кнопки – соответственно ***button1*** – ***button3***.

Постройте интерфейс пользователя как показано на рисунке.

Сценарий работы пользователя и программы

Пользователь видит на экране только список госномеров и кнопку **«УДАЛИТЬ»** – объекты группы данных не показываются. При выборе госномера из списка раскрываются данные. При вводе нового госномера также открываются поля данных, но они пустые.

Пользователь вводит новые значения (или изменяет имеющиеся) и сохраняет изменения (или отказывается от сохранения изменений). Данные о каждом Автомобиле –это отдельный объект. Сохраненные значения записываются в массив объектов, а в списке отображаются только госномера.

Госномер можно удалить из списка – в этом случае из массива удаляются и данные об объекте.

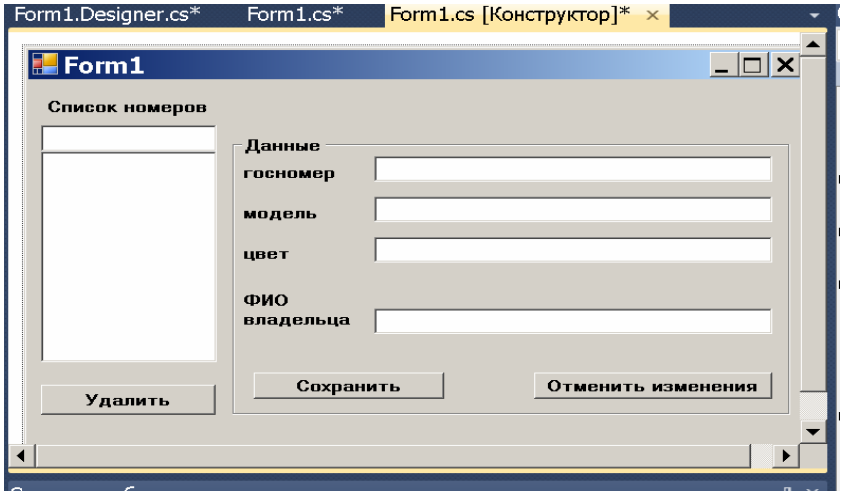


Рис. 20. Окно задачи № 5

2. Ввод и сохранение данных в массиве

Разработка некоторых элементов класса данных

Решение любой задачи начинается с описания используемых данных. Сведения о каждом транспортном средстве содержат четыре показателя, Автомобилей может быть любое количество. Фактически речь идёт о новом типе данных, содержащем в себе все перечисленные показатели.

Нужно оформить новый тип данных как класс ***Avto***.

Поместить заготовку описания этого класса в модуль ***Form1*** в пространство имён ***LabRab9*** после класса ***Form1***:

```
public class Avto
{ }
```

В составе класса **Gosnomer, Model, Color и FIO** определить как строки:

```
public class Avto
{
string Gosnomer; string Model; string Color; string FIO;
}
```

Далее необходимо описать конструктор объектов. Его назначение: при создании объекта дать исходные значения его полям. В классе описаны четыре поля, при создании объекта их инициализировать пустыми значениями. Тело конструктора будет содержать 4 оператора присваивания. Окончательно (фрагмент):

```
string  FIO; public Avto()
{ Gosnomer = ""; Model = ""; Color = "";  FIO = ""; }
```

Обработка ввода *Gosnomer*

Логично представить себе, что пользователь при первом запуске приложения захочет ввести данные о первом Автомобиле. Он попробует набрать в верхней строке комбинированного списка госномер Это событие – набор с клавиатуры. Точнее, нажатие символов на клавиатуре, ***KeyPress***. Такое событие связано с объектом ***ComboBox***.

Нужно создать обработчик и подключить его к событию.

Надо описать алгоритм, который будет работать при наборе госномера. В соответствии с сущностью события ***KeyPress*** обработчик будет выполняться каждый раз при любом нажатии на любую клавишу. Но госномер будет набран целиком только после нажатия клавиши ***«ENTER»***. Значит, все остальные нажатия метод должен проигнорировать. Но как определить, что нажата именно клавиша ***«ENTER»***?

Ответ на этот вопрос станет очевиден, если знать смысл параметров метода. Параметр ***sender*** – это источник события, т. е. объект ***comboBox1,*** а параметр ***e*** – это контейнер, содержащий информацию о событии. Выражение ***e.KeyChar*** позволяет узнать символ нажатой клавиши. Проверку можно осуществить так:

if (e.KeyChar == (char) Keys.Enter) { }

Вставьте эту конструкцию в обработчик. Системное перечисление Keys включает в себя все символы клавиатуры, в том числе и Enter. Только при сравнении его надо преобразовать в символ. Между фигурными скобками можно записать операторы, которые будут работать только при завершении набора госномера – все остальные клавиши никак не будут «замечены» обработчиком.

Задание для самостоятельного выполнения

*Убедитесь, что обработчик реагирует на нажатие клавиши **«ENTER»**. Вставьте между фигурными скобками оператор выдачи на экран какого либо сообщения (через messageBox) и запустите программу. Наберите в поле ввода что-нибудь и нажмите **«ENTER»**. Убедившись – удалите оператор выдачи сообщения.*

Теперь определим, что делать, когда нажата клавиша ***«ENTER»***. Очевидно, нам придётся отобразить невидимые объекты окна – группу данных.

Задание для самостоятельного выполнения

Добавьте оператор, делающий группу объектов видимой.

Ввод данных

Теперь можно попробовать набирать данные в поля ввода. В общем-то всё нормально. Но! Ведь мы уже набрали Gosnomer – почему же приходится набирать его ещё раз?

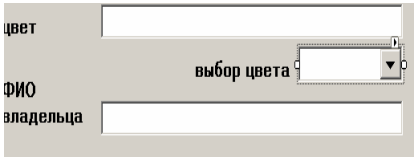
Задание для самостоятельного выполнения

Добавьте ещё один оператор. Как только группа объектов стала видимой, перенесите в первое поле ввода значение набранного госномера.

Есть и еще одно «но», значительно более существенное. Цвет лучше выбирать из какого-то списка, а не набирать. Пользователь вряд ли ошибётся с набором госномера и модели, а вот с цветом – запросто. Значит, придется запретить набор, т. е. установить у этого поля свойство ***Enable=false.***

Для правильного выбора цвета модифицируйте вид формы, как показано на рисунке.

добавьте между строками цвета и ФИО объект ***Label*** с текстом «Выбор цвета» и



объект ***comboBox2***, в котором нужно создать список цветов. У последнего объекта установите запрет набора цвета – это свойство ***DropDownStyle=DropDownList***.

Для заполнения списка цветов нужно в пространстве имен ***LabRab9*** между классами ***Form1*** и ***Avto*** тип данных – перечисление ***Color***. Включите в него шесть значений-констант:

public enum Color { неопределенный, белый, красный, фиолетовый, черный, серый, зелёный }

Затем нужно создать в comboBox2 список цветов на основе перечисления Color. Значения этого перечисления перепишем в коллекцию объекта comboBox2 при запуске программы – перед открытием окна.

В обработчик события **Form1.Load** вставить код:

int i;

for (i = 0; i < 6; i++) comboBox2.Items.Add((Color)i);

Нужно написать обработчик события ***SelectedIndexChanged*** двойным кликом по ***comboBox2***, которое выбранное значение цвета из свойства ***comboBox2.Text*** поместит в ***textBox3***. создаем обработчик этого события. В тело обработчика добавить строку:

textBox3.Text = comboBox2.Text;

Нужно также в приложении предусмотреть, чтобы при наборе данных пользователь мог перемещаться от поля к полю клавишей ***«TAB»***.

Для этого в модуле ***Form1.Designer.cs*** в секциях-описаниях каждого объекта есть для оператора, определяющий порядок обхода объектов окна с помощью клавиши ***«TAB»*** нужно установить индексы (TabIndex) следующим образом:

0 – comboBox1,

1 – textBox1,
2 – textBox2,
3 – comboBox2,
4 – textBox4,
5 – кнопка button2,
6 – кнопка button3,
7 – кнопка button1.

Остальные не играют роли. Теперь запустить программу и проверить работу клавишей ***«TAB»***. Курсор должен перемещаться в указанном порядке.

Обработчики нажатия на кнопки:

Кнопка отмены изменений

При отмене изменений (или отказе от сохранения набранных данных) надо сделать:

- очистить поля ***textBox1*** – ***textBox4***;
- убрать с экрана (скрыть) блок ввода данных;
- очистить поле ввода в объекте ***comboBox1***.

Задание для самостоятельного выполнения

- *Внесите необходимые изменения в программу.*

Кнопка сохранения результатов ввода данных

Введенные данные должны сохраниться в элементе массива данных Массив должен хранить все данные о каждом автомобиле. Соответственно, элементы массива должны быть типа ***Avto***. Определить хранилище на 1000 Автомобилей. Для объявления использовать оператор:

```
Avto [] md = new Avto[1000];
```

Нужно его записать в самом начале класса ***Form1***. Это гарантирует доступ к нему из любого метода этого класса, т. е. в пределах окна. Там же создать переменную-счетчик, которая будет содержать количество заполненных элементов массива:

```
int count=0;
```

Нужно учесть, что при объявлении массива в памяти выделено место только под его элементы, а они являются ссылками на объекты – при создании массива эти ссылки будут равны значению ***null***.

Поэтому нужно предусмотреть выделение памяти под каждый объект также при открытии окна с помощью оператора:

```
 for (i = 0; i < 1000; i++) // создание объектов
{
md[i] = new Avto();
}
```

Можно было бы введенные значения записать в поля первого свободного элемента массива. Однако, просто так этого сделать не удастся: поля объекта являются закрытыми. Для этого нужно включить в класс ***Avto*** метод, позволяющий обновить значения полей объекта:

```
public void Obnov(string g, string m, string c, string fio)
{ Gosnomer = g; Model = m; Color = c;  FIO = fio; }
```

Этот метод – динамический. Он может быть вызван для работы с объектом массива md. В обработчике нажатия на кнопку «Сохранить» нужно указать:

```
md[count]. Obnov(textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text);
count++;
```

После увеличения значения ***count*** следующий набор данных запишется уже в следующий по порядку объект.

Задание для самостоятельного выполнения

Добавьте в обработчик button2 очистку полей ввода данных и сокрытие группы объектов набора данных, как это сделано в обработчике button3.

Не пишите повторно операторы, которые уже есть в обработчике для button3. Воспользуйтесь уже имеющимся методом.

Отображение списка Госномеров

Но если с отменой набора все очевидно, то с сохранением не ясно: то ли сохраняются данные, то ли нет. В списке новый госномер не отображается.

Список госномеров – это список в коллекции ***comboBox1***, здесь нужно использовать массив объектов **md**

Первое поле каждого объекта – это госномер.

Задание для самостоятельного выполнения

В классе Avto создайте динамический метод, позволяющий получить Gosnomer объекта Avto.

В обработчике кнопки ***«СОХРАНИТЬ»*** *очистите коллекцию Items объекта comboBox1.*

Просмотрите массив md в цикле, выберите из каждого объекта поле Gosnomer с помощью созданного ранее метода и добавьте Gosnomer в коллекцию Items объекта comboBox1. Алгоритм следует оформить в виде метода ***CreateLisTGosn****. Метод вызвать в обработчике нажатием кнопки* ***«СОХРАНИТЬ»****.*

Обработка выбора Госномера из списка

Первая особенность – это другое событие: не набор, а выбор значения из списка. Имя события – ***SelectedIndexChanged***. Обработчик этого события можно получить двойным кликом по объекту. Внутрь обработчика надо записать операторы, извлекающие значения из выбранного объекта и записывающие их в поля ввода ***textBox1***. А затем – отобразить группу.

Для заполнения поля ввода госномера ***textBox1*** сам объект не нужен. Ведь после выбора госномера он автоматически отображается в поле ввода объекта ***comboBox1*** (свойство ***Text***), откуда его можно перекинуть в поле ***textBox1***. Вместе с оператором отображения группы это выглядит так:

```
textBox1.Text = comboBox1.Text;
groupBox1.Visible = true;
```

Две проблемы:

- получить номер выбранного объекта ***Avto***;
- извлечь из объекта значения модели, цвета и ФИО.

Свойство ***SelectedIndex*** объекта ***comboBox1*** содержит номер выбранной строки списка. Этот номер совпадает с индексом объекта в массиве. Выбрать номер можно оператором:

```
int nom=comboBox1.SelectedIndex;
```

Прямой доступ к полям ***Model***, ***Color***, ***FIO*** запрещен, поэтому извлечь значения сразу не получается.

Нужно расширить функциональность класса ***Avto***, т.е. добавить в него описания свойств, которые позволят получить значения закрытых полей. Итак:

```
public string mod { get { return Model; } }
public string col  { get { return Color; } }
public string  fiov { get { return  FIO; } }
```

Нужно воспользоваться этими свойствами для получения значений:

```
textBox2.Text = md[nom].mod;
textBox3.Text = md[nom].col;
textBox4.Text = md[nom]. fiov;
```

Задание для самостоятельного выполнения

Наберите и сохраните два разных набора значений. Затем последовательно отобразите их, выбрав госномера. Обратите внимание, что в одном из случаев значение в поле цвета противоречит значению в списке цветов – там осталось последнее

набиравшееся значение. Сделайте так, чтобы значение цвета в поле comboBox2 и textBox3 были одинаковыми.

Теперь выберите какое-либо значение госномера и внесите изменение в данные. Например, замените FIO и сохраните. В списке появится двойной номер, и это правильно: не установлен контроль на повторную запись. Измененные данные должны записываться в тот же объект, если госномер остался прежним. Это значит, что алгоритм в обработчике ***button2*** должен различать, новый это госномер или существующий, т. е. в массив пишется как бы новый элемент.

Задания для самостоятельного решения

1. *Внесите изменения в алгоритм обработчика кнопки button2, не допускающие двойных записей.*
2. *Разрешите пользователю набирать новые цвета и обеспечьте их сохранение в списке. Двойных записей в список цветов не делать.*
3. *Создайте обработчик кнопки **«УДАЛИТЬ»** и составьте алгоритм работы. Не забудьте, что удалять надо не только госномер из списка, но и данные из массива.*

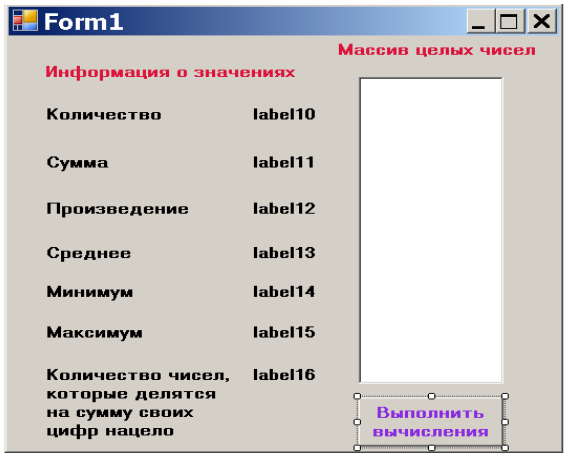
Лабораторная работа № 4. Типовые алгоритмы обработки массива

Цель работы: Изучение типовых алгоритмов обработки массивов.

Создать новый проект с именем ***Lab_rab6***. В объекте ***textBox1*** установить многострочный режим (свойство ***MultiLine*** = ***true***). В объектах label с 10 по 16 будут отображаться значения в соответствии с надписями в объектах.

В объект ***textBox1*** пользователь будет набирать значения элементов массива. Кнопка обеспечивает запуск алгоритма вычислений.

Порядок работы пользователя: сначала он вводит значения в массив, затем нажимает кнопку и получает результаты сразу всех вычислений.



Окно формы

Все характеристики массива (сумма, среднее и прочие) вычисляются по собственным алгоритмам. Все алгоритмы должны быть записаны внутри обработчика.

До нажатия кнопки данные хранятся в объекте ***textBox1***. Для хранения значений этот объект имеет коллекцию строк ***Lines.*** Это обычный массив типа ***string***. Сначала преобразовать элементы этого массива из строковых значений в численные и сохранить их значения в специально созданном целочисленном массиве. Будем считать, что пользователь задает в одной строке только одно число.

```
int[] m = new int[1000];
```

Вычислить количество чисел (строк) в коллекции:

```
int n = textBox1.Lines.Length;
```

В переменной k хранить количество значений в массиве:

```
int k = 0;
Цикл, формирующий массив m
for (i = 0; i < n; i++)
{   m[k] = Convert.ToInt32(textBox1.Lines[i]); k++;  }
```

Задания для самостоятельной работы

1. Обработайте исключение. Если в строке некорректное значение (не цифра или строка пуста), то по такой строке ничего делать не надо – никакое значение в массив не включается.

2. Запишите алгоритм выдачи на экран количества чисел в массиве.

3. Запишите алгоритм выдачи на экран суммы элементов массива.

4. Запишите алгоритм выдачи на экран произведения элементов массива.

Запишите алгоритм выдачи на экран среднего арифметического элементов массива.

5. Запишите алгоритм выдачи на экран минимального элемента массива.

6. Запишите алгоритм выдачи на экран максимального элемента массива.

7. Запишите алгоритм подсчета количества чисел, которые делятся нацело на сумму своих цифр.

8. Доработайте программу. Выдайте на экран номера строк, которые имеют ошибки ввода и не включены в вычисления. Вывод выполнить одним окном MessageBox.

Указание. *Сохраните номера строк в отдельном массиве.*

Лабораторная работа № 5. Работа с массивами случайных чисел

Цель работы: Изучение методов классов Random и Math.

Задание для самостоятельноговыполнения *Создайте окно следующего вида:*

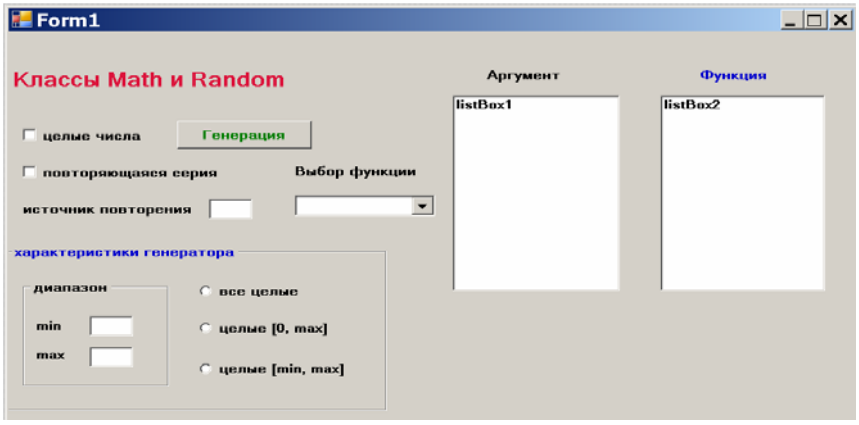
1. Два объекта ListBox:
- listBox1 – для аргументов функции;
- listBox2 – для значений функции.

2. Три объекта textbox:
- textBox1 – поле для источника повторения;
- textBox2 – поле для минимума;
- textBox3 – поле для максимума.

3. Объект comboBox1 – для выбора функции.
4. Два объекта checkbox:
- checkBox1 – для генерации повторяющейся серии чисел;
- checkBox2 – для генерации целых чисел.

5. Три объекта radioButton:
- radioButton1 – все целые;
- radioButton2 – целые в диапазоне [0, max];
- radioButton3 – целые в диапазоне [min, max].

6. Объект groupBox1 – группирует характеристики генератора случайных чисел.
7. Кнопка button1 – кнопка с надписью «Генерация».
8. Объект label3 – это надпись «Функции».
9. Объект label2 – это надпись *«источник повторения».*



Все остальные надписи – это объекты label под любыми номерами.

Окно формы

Предлагается реализовать следующий сценарий.

Пользователю нужна программа для изучения различных функций класса Math. Функцию он сможет выбрать из списка в объекте **comboBox1**. Убедитесь, что пока список пустой. При выборе функции надпись «Функция» должна замениться на имя выбранной функции.

Аргументы для вычисления функции будут размещаться в объекте **listBox1**, а результаты вычисления – в объекте **listBox2**. Вычисления должны производиться сразу же, как только пользователь выберет функцию.

Аргументы формируются в списке объекта **listBox1** при нажатии кнопки **«Генерация».** Генерируемые значения – это случайные числа. Всего их 10 штук.

Какие будут сгенерированы значения, зависит от настроек. Это могут быть вещественные числа в интервале от 0 до 1 (по умолчанию). Можно установить флажок – тогда будут генерироваться целые числа. Генерируемые серии чисел могут быть разными (по умолчанию), а при установке флажка повторения серии

– каждый раз одинаковыми для каждого заданного числа (источника повторений).

В случае целых чисел возможна генерация:

- из всех возможных положительных целых чисел;
- из отрезка от 0 до максимального значения;
- из отрезка от минимального до максимального – в зависимости от выбранного переключателя.

Значения минимума и максимума должны задаваться только в нужном случае.

Желательно, чтобы в каждый момент времени выполнения программы в окне отображалось бы только то, что необходимо. Например, если не установлен флажок целых чисел, то и все настройки для целых отображаться не должны. То же и с повторением серии: если флажок не установлен, то и источник отображаться не должен.

По условию задачи при запуске программы флажок «целые числа» не установлен. Значит, не должно отображаться ничего из настроек, то есть все объекты, заключенные в ***groupBox1*** должны быть невидимы. Найдём в свойствах объекта ***groupBox1*** свойство ***Visible*** и установим в нём значение ***false***.

<u>Задание для самостоятельного выполнения</u>

При запуске программы флажок повторения серии также не установлен. Сделайте, чтобы при запуске программы поле источника повторений и надпись не отображались на экране.

При установке флажка **«целые числа»** устанавливается режим генерации целых чисел. При этом должны показаться все режимы настройки, заключенные в объекте ***groupBox1***. И наоборот, когда флажок снят, режимы настройки вновь должны стать невидимыми. Данные функции реализуйте в обработчике события ***Click*** объекта ***checkBox2***:

Нужно добавить в обработчик код: если флажок установлен, то сделать группу объектов видимой, а если не установлен, сделать группу невидимой:

```
private void checkBox2_CheckedChanged(object sender, EventArgs e)
{

if (checkBox2.Checked == true)
        { groupBox2.Visible = true; }
else groupBox2.Visible = false;
}
```

Задание для самостоятельного выполнения

При установке флажка «повторяющаяся серия» поле источника и надпись должны отображаться. Наоборот, при снятии флажка оба объекта должны стать невидимыми. Запрограммируйте эту логику и проверьте работу.

При заполнении объекта ***comboBox1*** включите в список имена стандартных математических функций класса ***Math.***

Например, такие:

- **Геометрические:** Atan (арктангенс), Cos (косинус), Exp (экспонента), Log (натуральный логарифм), Sin (синус), Tan (тангенс).
- **Алгебраические:** Floor (округление), Ceiling (округление), Pow(x,3) (возведение числа в третью степень), Sqrt (квадратный корень), ABS() абсолютная величина.

Занесение в список ***comboBox1*** осуществляется через свойство ***Items***.

При выборе имени функции это выбранное имя должно заменить собой надпись **«Функция».**

Задание для самостоятельного выполнения

Замените надпись «Функция» при выборе имени из списка.

Указания.

*1. Выбранное имя функции – в свойстве **Text** объекта comboBox. Событие, которое наступает при выборе имени – **SelectedIndexChanged.***

2. Создайте соответствующий метод-обработчик – для создания заготовки достаточен двойной клик на объекте comboBox1.

Далее нужно создать обработчик события нажатия на кнопку. В обработчике нужно реализовать разные способы генерации последовательности чисел – их должно быть 10 штук.

Желательно использовать два генератора – два объекта класса ***Random***.

Один – для случая неповторяющейся последовательности.

Другой – для повторяющейся.

Для определения какой нужен генератор, используйте флажок ***checkBox1***.

В зависимости от того выбран он или нет, будут две ветви алгоритма. В ветвь **(if)** должна соответствовать неповторяющейся последовательности. Здесь должен быть создан простой объект класса ***Random***:

```
Random G = new Random();
```

Ветвь **(else)** должна соответствовать повторяющейся последовательности. Здесь должен быть создан сложный объект, учитывающий заданное значение в источнике повторения ***textBox3.Text***.

Однако, это значение типа ***string***, его надо преобразовать в целое число, а это возможно только тогда, когда это цифровое выражение. Следовательно, необходимо еще обработать исключение: если значение не задано или задано не цифрой, то взять в качестве источника любое число. Например, ноль.

```
Random G;
if (checkBox1.Checked == false)
        G = new Random();
else
{
int n;
try
{ n = Convert.ToInt32(textBox1.Text); }
catch {
        n = 0;
}
        G = new Random(n);
}
```

Теперь можно доставать из генератора случайные числа, используя метод ***Next*** или метод ***NextDouble.*** Каким методом – зависит от состояния флажка «целые числа». Если он не выбран, то используем ***NextDouble***, а если выбран, то ***Next***.

Если последовательность целая, то надо еще учесть, каков интервал выбора: все числа, от нуля до максимума или от минимума до максимума. А это зависит от того, какой из переключателей **(*radioButton1*, *radioButton2*, *radioButton3*)** выбран.

При выборе двух последних переключателей должны быть выбраны допустимые (цифровые) значения минимума и максимума. И весь этот выбор – в цикле. Каждое полученное случайное число надо разместить в ***listBox1*** как отдельную строку.

Цикл и ветвление в зависимости от состояния флажка:

```
for (n = 0; n < 10; n++)
{
if (checkBox2.Checked == false)
{ }
else
{ }
}
```

В секцию «if» добавьте операторы:

```
double d = G.NextDouble(); listBox1.Items.Add(d.ToString());
```

Первый оператор обеспечивает получение случайного вещественного числа. Второй с помощью метода Add добавляет это число в коллекцию объекта, предварительно преобразовав в строку.

Задания длясамостоятельного выполнения

1. Проверьте работу без флажка – генерация выполняется. А с флажком не выполняется – ещё не написан алгоритм.

2. Перезапустите программу. Нажмите на кнопку. Затем нажмите ещё раз. Должны быть новые значения, но старые тоже остались. Внесите соответствующие изменения в код.

3. Проверьте работу при установленном флажке повтора серии.

Действительно ли серия повторяется при повторном нажатии на кнопку? Убедитесь, что при разных значениях источника повтора получаются разные последовательности.

Запустите программу, установите флажок «целые числа». Отобразятся характеристики генератора для случая целых чисел. Обратите внимание: невыбран ниодин ив переключателей.

Задание для самостоятельного выполнения

Будем считать, что при выборе флажка надо установить значение «выбрано» в переключатель «все целые» (программно поставить точку). Это достигается

изменением свойства Checked данного переключателя. Внесите нужный оператор в программу.

Для формирования блока **else** обработчика нажатия на кнопку нужно добавить следующие операторы:

```
int m;
if (radioButton1.Checked == true)
{ }
        if (radioButton2.Checked == true)
{ }
        if (radioButton3.Checked == true)
{ }
```

Переменная ***m*** предназначена для случайного целого числа. Каждый блок предназначен для своего переключателя: если он выбран, то будет обработка. Из генератора берется целое число из всего диапазона целых положительных чисел:

m = G.Next();

Во втором блоке надо сначала вычислить максимальное значение. Надо учесть, что оно может быть с ошибкой или не задано – в этом случае можно взять какое-то значение как максимально возможное, например, 100:

```
int max;
try {
        max = Convert.ToInt32(textBox3.Text);
}
catch
{
        max = 100;
}
        m = G.Next(max);
```

Задания для самостоятельного выполнения

1. Доработайте третий блок. Здесь метод Next используется с двумя параметрами – минимальное и максимальное значение. Не забудьте их также проверить на цифровое значение.

2. Запишите сразу после третьего блока оператор, добавляющий полученное целое число в список listBox1.

Вычисление значений функции

Требуется алгоритм вычисления значений функции. При выборе функции в обработчик добавить блок вычислений. Фактически это разные варианты вычислений, но надо ещё распознать, какой именно вариант выбран – какая функция.

Для распознавания варианта используйте порядок их расположения в списке. Предпочтительнее в воспользоваться оператором ***switch***. Значения отправляются в ***listBox2***.

Добавьте в обработчик следующий код:

```
int i;
double d=0;
double r=0;
for (i = 0; i < 10; i++)
{
        d = Convert.ToDouble(listBox1.Items[i]);
switch (comboBox1.SelectedIndex)
{
    case 0: r= Math.Atan(d); break;
    …

}
```

```
        listBox2.Items.Add(r.ToString());
    }
```

Задания для самостоятельного выполнения

1. Проверьте работу программы для функции Atan.

2. При повторном выборе функции старые значения не исчезают. Внесите исправление.

3. После генерации новых аргументов в окне результатов остаются старые значения. Обеспечьте очистку окна результатов при генерации новых аргументов.

Сейчас вещественные аргументы и результаты сохраняются с большим числом знаков после запятой. Отформатируйте вывод значений: аргументы – для вещественных два знака после запятой, результаты – четыре.

4. Доработайте программу. Обеспечьте вычисление остальных функций.

Лабораторная работа № 6. *Файлы произвольного доступа*

Цель работы: Изучение правил организации файлов произвольного доступа.

Продолжительность работы: 8 часов.

Постановка задачи

Требуется организовать простую базу данных. База хранит сведения о жителях города: FIO, Пол, Дата рождения, Улица, Дом,

Номер квартиры, Общая площадь жилья. Необходимо обеспечить: ввод новых данных, составление списка данных по известным значениям (полные сведения или частичные), отображение полных сведений по элементу из списка.

Обсуждение проблемы

На первый взгляд задача похожа на задачу № 5. Также можно

организовать сериализуемый класс. Введённые данные хранить в файле, который в начале работы программы считывать в массив, а после закрытия окна – записывать данные из массива в файл. Однако, здесь объем данных может быть таким, что файл целиком просто не уместится в массиве – представим себе, что кроме перечисленных сведений есть еще паспортные данные, сведения об образовании и т. д. Да и количество жителей само по себе велико. Следовательно, подобный способ организации программы не годится. Вместе с тем, разнотипность данных всё-таки заставит нас использовать класс. И сериализацию нам придётся использовать – иначе объект на диск не записать. А в остальном – решение будет иным.

Сценарийработыпользователя

*Мы не станем отображать на экране список всех FIO. Изначально на экране отобразим только две кнопки: **«ВВОД ДАННЫХ»***

*и **«ПОИСК ДАННЫХ»**. Рассмотрим оба случая.*

*1. При нажатии на кнопку **«ВВОД ДАННЫХ»** появляются поля ввода данных для набора и сохранения новых данных. Вместе с полями появляются кнопки **«СОХРАНИТЬ»**, **«ОТМЕНИТЬ»** и*

***«ВЫБРАТЬ»**. Кнопка **«ВЫБРАТЬ»** обеспечивает выбор пола:*

*М или Ж. Должны быть заполнены все поля ввода. При нажатии кнопки **«СОХРАНИТЬ»** проверяется правильность заполнения полей (всё ли заполнено) и данные сохраняются в файле «База» – выполняется запись в конец файла. Пользователь информируется о записи сообщением на экран: «Сведения добавлены». При нажатии кнопки **«ОТМЕНИТЬ»** ничего не происходит, поля ввода скрываются, и вновь на экране отображаются две исходные кнопки.*

*2. При нажатии на кнопку **«ПОИСК ДАННЫХ»** отображаются все поля для набора данных, кроме номера квартиры и площади жилья. Отображаются также кнопки **«ПОИСК»**, **«ВЫБРАТЬ»** и **«ОТМЕНИТЬ».** В поля пользователь может набрать значения в каждое поле или только в некоторые. Введенные значения являются ключами поиска*

*информации в базе данных. При нажатии кнопки **«ОТМЕНИТЬ»** ничего не происходит, поля ввода скрываются, и вновь на экране две исходные кнопки.*

*3. При нажатии на кнопку **«ПОИСК»** выполняется поиск записей в файле «База». После завершения поиска на экране отображается список найденных данных в виде FIO. Список может быть и пустым, если при поиске не найдено ни одной записи. Вместе со списком отображается кнопка **«ЗАКРЫТЬ»**. Теперь пользователь может выбрать любой элемент списка, и на экране отображается полная информация.*

*При нажатии на кнопку **«ЗАКРЫТЬ»** поля ввода скрываются, и вновь на экране – две исходные кнопки.*

Интерфей спользователя

Интерфейс представлен на рис. 22. Все надписи – это объекты ***Label***.

Поля ввода – это объекты ***TextBox***. Для отображения списка найденных записей используется объект ***comboBox1***.

Надпись на кнопке ***«BUTTON3»*** изначально не сформирована. Надпись появится после выбора варианта работы: при вводе данных – надпись «Сохранить», при поиске – «Поиск». Порядок появления объектов описан в сценарии.

Заданиядлясамостоятельноговыполнения

1. В визуальном режиме установите невидимость всех объектов, кроме двух верхних кнопок.

*2. Оформите метод Видим Ввод, в котором установите видимость всех надписей, полей ввода и кнопок **«BUTTON3»,***

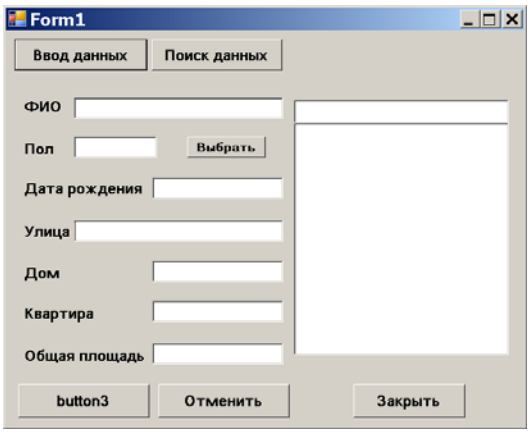


Рис. 22. Интерфейс пользователя

***«ВЫБРАТЬ»** и **«ОТМЕНИТЬ»**. В этом же методе установите пустые значения всех полей ввода.*

*3. В обработчике кнопки **«ВВОД ДАННЫХ»** запишите оператор вызова метода ВидимВвод. В button3 сформируйте надпись «Сохранить». Скройте кнопку **«ПОИСК ДАННЫХ»**, чтобы ее случайно не нажал пользователь.*

4. Оформите метод СкрытьВвод, отменяющий видимость объектов (действия, противоположные методу ВидимВвод).

*5. В обработчике кнопки **«ОТМЕНИТЬ»** запишите оператор вызова метода СкрытьВвод.*

*6. В обработчике кнопки **«ВЫБРАТЬ»** обеспечьте выбор пола. Значение М или Ж запишите в textBox2.*

7. Оформите метод Набрано, в котором выполняется контроль полноты набора данных. Метод возвращает true, если все поля заполнены, и false в противном случае.

*8. В обработчике кнопки **«СОХРАНИТЬ»** запишите оператор вызова метода Набрано. Если метод возвращает false, то выдайте сообщение о неполном наборе на экран и завершите методобработчик (return).*

Оформите метод Дата, в котором проверьте правильность заполнения поля Дата

рождения. Предполагается, что оно должно быть заполнено в формате: ЧЧ.ММ.ГГГГ. И число, и месяц, и год – целые числа. Соответствие числа наименованию месяца не проверять (31 июня допускается). Метод возвращает true, если формат верен, и значения численные. В противном случае – false. Кроме того, при правильном формате метод формирует три выходных целых параметра: день, месяц и год. Значения надо сохранить в переменных.

*9. В обработчике кнопки **«СОХРАНИТЬ»** обеспечьте вызов метода Дата. Если он возвращает false, то сообщить об этом на экран и обработчик завершить.*

*10. В обработчике кнопки **«СОХРАНИТЬ»** проверить правильность набора площади (численное значение). При неверном наборе сообщить об этом на экран, обработчик завершить. При правильном значении сохранить значение площади в переменной.*

Теперь после всех контролей продолжим разработку алгоритма кнопки ***«СОХРАНИТЬ»***. Если все поля заполнены, то надо вывести новые данные в файл. Но файла у нас пока нет!

Определение формата файла

Определимся с форматом данных в файле. Проще всего описать новый класс объектов с полями, соответствующими вводимым данным.

Задания для самостоятельного выполнения

1. Создайте новый класс Житель с полями FIO (строка), Пол (символ), ДатаРождения (целочисленный массив), Улица (строка), Дом (строка), Квартира (строка), Площадь (вещественное число). Не забудьте, что мы будем выводить данные в файл – нужна сериализация. Да и пространства имён – тоже.

2. Включите в класс конструктор объектов, заполняющий поля объекта нулевыми (с точки зрения типа) значениями.

3. Включите в класс метод Заполнить, который заполняет поля объекта Житель.

Теперь можно в обработчике кнопки ***«СОХРАНИТЬ»*** объявить переменную-объект, создать этот объект с помощью конструктора и заполнить его поля значениями (примерно так):

```
Житель T = new Житель();
T.Заполнить(textBox1.Text,textBox2.Text[0],c,m,g,textBox4.Text,
textBox5.Text,textBox6.Text,p);
```

Здесь: c, m, g, p – это число, месяц, год рождения и площадь (получены вами ранее). Объект сформирован и готов к выводу в файл.

Создание файла и вывод данных

Теперь описываем поток, открываем файл для вывода данных в конец файла и выводим данные:

```
FileStream F = new FileStream("База", FileMode.Append,
FileAccess.Write);
BinaryFormatter W = new BinaryFormatter(); W.Serialize(F, T);
F.Close(); СкрытьВвод();
```

Поиск данных Задания для самостоятельного выполнения

*Создайте метод ВидимПоиск, в котором устанавливается видимость всех надписей и полей ввода (кроме номера и площади квартиры). Отобразите кнопки **«BUTTON3»**, **«ВЫБРАТЬ»** и **«ОТМЕНИТЬ»**. В этом же методе установите пустые значения всех полей ввода.*

*1. В обработчике кнопки **«ПОИСК ДАННЫХ»** запишите оператор вызова метода ВидимПоиск. В button3 сформируйте надпись «Поиск».*

*Скройте кнопку **«ВВОД ДАННЫХ»**, чтобы случайно пользователь не смог ее нажать.*

Теперь нам надо определиться с алгоритмом поиска. Дело в том, что поиск мы заказали на ту же кнопку, которая использовалась

для сохранения данных – мы просто её переименовали. Если нажать на эту кнопку, то будет выполняться алгоритм сохранения данных – ведь обработчик события настроен именно на это. Нам нужен новый обработчик события, выполняющий поиск, но сработать он должен от той же кнопки.

Создадим вручную обработчик с похожим именем:

private void button3_Click_1(object sender, EventArgs e)

{ }

В обработчике кнопки ***«ПОИСК ДАННЫХ»*** сразу после переименования кнопки ***«BUTTON3»*** подменим обработчик события:

this.button3.Click -= new System.EventHandler(this.button3_Click);
this.button3.Click += new System.EventHandler(this.button3_Click_1);

Первый оператор операцией -= отключает прежний обработчик, второй подключает новый обработчик. Теперь в обработчике ***button3_Click_1*** можно реализовывать алгоритм поиска данных.

Задания для самостоятельного выполнения

Оформите метод НабранПоиск, в котором выполняется контроль набора данных для поиска. Метод возвращает true, если есть данные хотя бы в одном поле, и false в противном случае.

*1. В обработчике кнопки **«ПОИСК»** запишите оператор вызова метода НабранПоиск. Если метод возвращает false, то выдайте на экран сообщение об отсутствии ключей поиска и завершите метод-обработчик (return).*

Организуем поиск данных. По условию задачи, каждая запись файла, имеющая требуемые ключи, должна отображаться в списке

на экране – для этого у нас есть объект ***comboBox1***. Представим себе, что такой список получен, и пользователь решил посмотреть данные, выбрав какую-то фамилию из списка. Данные надо будет вновь доставать из файла – ведь при поиске мы не сохраняли полные сведения нигде. Значит, в процессе поиска надо сохранять адрес данных в файле: номер байта, с которого начинаются нужные данные. Создадим в классе ***Form1*** массив:

long[] Адрес = new long[1000];

Массив ***Адрес*** является индексным массивом. У нас будет соответствие: FIO – в списке объекта ***comboBox1***, а номер байта начала данных в файле – в соответствующем элементе массива ***Адрес***.

Теперь надо организовать доступ к файлу. Переменную-поток надо разместить так, чтобы к ней был доступ не только при поиске, но и при выборе FIO – а это уже будет другой метод. Значит, поток следует создать как поле класса, сразу после объявления массива:

FileStream БД;

Откроем файл в обработчике ***button3_Click_1***:

FileStream БД = new FileStream("База", FileMode.OpenOrCreate,
FileAccess.ReadWrite);

А в конце обработчика (чтобы не забыть) сразу же запишем оператор закрытия потока: **БД.Close();**

Переходим к поиску данных в файле – все действия в том же обработчике.

Задания для самостоятельного выполнения

1. Очистите список в объекте comboBox1.

2. Создайте объект R класса BinaryFormatter для выполнения десериализации объектов и служебный объект Ж класса Житель.

*3. Сделайте видимым объект comboBox и кнопку **«ЗАКРЫТЬ»**.*

Запишите цикл считывания данных из файла (просмотр от начала до конца – по аналогии с предыдущей лабораторной работой):

```
long i=0;
while (i < БД.Length)
{ Ж = (Житель)R.Deserialize(БД);
// далее будет алгоритм сравнения ключей i = БД.Position;
```

}

После того, как в объект ***Ж*** считаны данные, можно сравнивать ключи. Но до полей объекта невозможнодобраться – они закрытые.

Задания для самостоятельного выполнения

Создайте в классе Житель свойства для доступа ко всем закрытым полям. Имена свойств составьте из имени поля с добавлением знака «подчёркивание». Например: FIO_. Поскольку пол и дата будут использоваться для сравнения, свойства должны возвращать их в виде строки.

1. Создайте метод Сравнение, в котором будет обеспечиваться сравнение заданных полей с полями введенного объекта. Метод имеет один входной параметр – объект класса Житель. Метод возвращает true, если ключи совпадают.

Примечание. Сравнивать надо только заданные ключи!

Далее запишем в обработчике операторы проверки результатов сравнения и сохранения сведений:

```
if (Сравнение(Ж)==true)
{ // далее сохранение сведений о записи в списке и в массиве
Адрес[comboBox1.Items.Count] = i; comboBox1.Items.Add(Ж. FIO_);
}
```

Задания для самостоятельного выполнения

1. Проверьте работу программы. Наберите несколько разных сведений, сохраните их и выполните поиск по разным ключам. Всё должно работать!

*2. Обработайте нажатие на кнопку **«ЗАКРЫТЬ»**. Объект comboBox1 и сама кнопка должны стать невидимыми.*

Создайте обработчик события ***выбор из списка***. Вставьте в него такой алгоритм:

```
if (comboBox1.SelectedIndex >= 0)
{
FileStream БД = new FileStream("База", FileMode.
OpenOrCreate, FileAccess.ReadWrite); BinaryFormatter R = new BinaryFormatter();
Житель Ж;
БД.Position=Адрес[comboBox1.SelectedIndex]; // *** Ж=(Житель)
R.Deserialize(БД); БД.Close();
ВидимВвод();
textBox1.Text = Ж. FIO_; textBox2.Text = Ж.Пол_; textBox3.Text = Ж.Дата_;
textBox4.Text = Ж.Улица_;
textBox5.Text = Ж.Дом_; textBox6.Text = Ж.Квартира_; textBox7.Text =
Ж.Площадь_.ToString();
}
```

Обратите внимание на оператор, отмеченный тремя звёздочками.

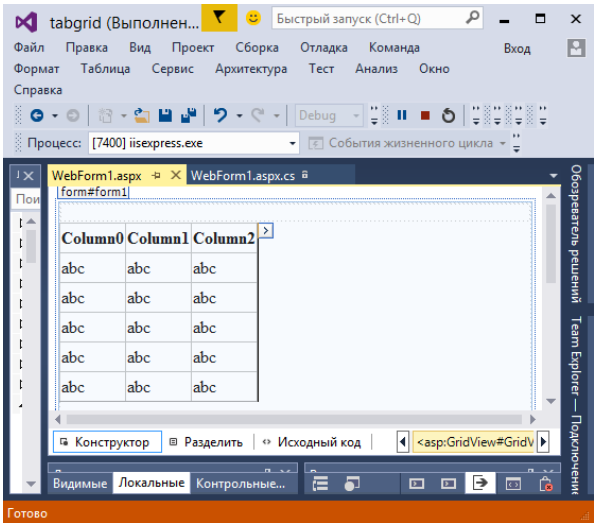
Он устанавливает файл в позицию начала нужного объекта, после чего объект считывается и с помощью свойств раскладывается по объектам ***TextBox***.

Задания для самостоятельного выполнения

*В алгоритме кнопки **«ОТМЕНИТЬ»** восстановите отображение кнопок **«ВВОД ДАННЫХ»** и **«ПОИСК ДАННЫХ»**.*

*Сейчас после выполнения поиска данных невозможно вернуться к вводу данных – ведь мы переключили кнопку **«BUTTON3»** на другой обработчик. Устраните дефект.*

Задание 7.
Отображение табличных данных в веб-форме с помощью *GridView*



Задание: Разработать проект отображения строковых массивов в виде таблицы на веб форме с помощью элемента управления **GridView** (Просмотр/обзор сетки данных в таблице) и объекта **DataTable.**

Таблица телефонов представлена в виде двух строковых массивов. Требуется написать приложение для вывода в веб-форму этих массивов в виде таблицы.

Для этого:

1.создать новый проект шаблона **Web приложение ASP.NET,** в поле **Имя** указать имя **TabGridweb**.

2. добавить к проекту веб-форму (в меню **Проект** выбрать команду **Добавить новый элемент** и в открывшемся окне выбрать шаблон **Веб форма**

3. перетащить на форму из **Панели элементов** (из подраздела **Данные)** элемент управления **GridView.**

Комментарии к программному коду:

Название страницы:**"Вывод таблицы в веб-форму";**

Содержимое первой строки:

String[] Имена = **{"Ахмед - раб", "Ахмед - дом", "Приемная ДГУ", "Кафедра ИИТ", "Далгатов-моб", "Шихнебиев Эмин"};**

Содержимое второй строки:

String[] Тлф = **{"8722-68-51-22", "8722-62-45-72", "8722-68-23-26", "8722-67-59-10", "906-482-66-55","989-665-14-41"};**

Создание обычной таблицы выполняется командой:

var Таблица = new System.Data.DataTable();

Заполнение шапки"шапки" таблицы вывода

```
Таблица.Columns.Add("ИМЕНА");
Таблица.Columns.Add("НОМЕРА ТЕЛЕФОНОВ");
```

Заполнение ячеек таблицы

```
for (var i = 0; i <= 5; i++)
   Таблица.Rows.Add(Имена[i], Тлф[i]);
```

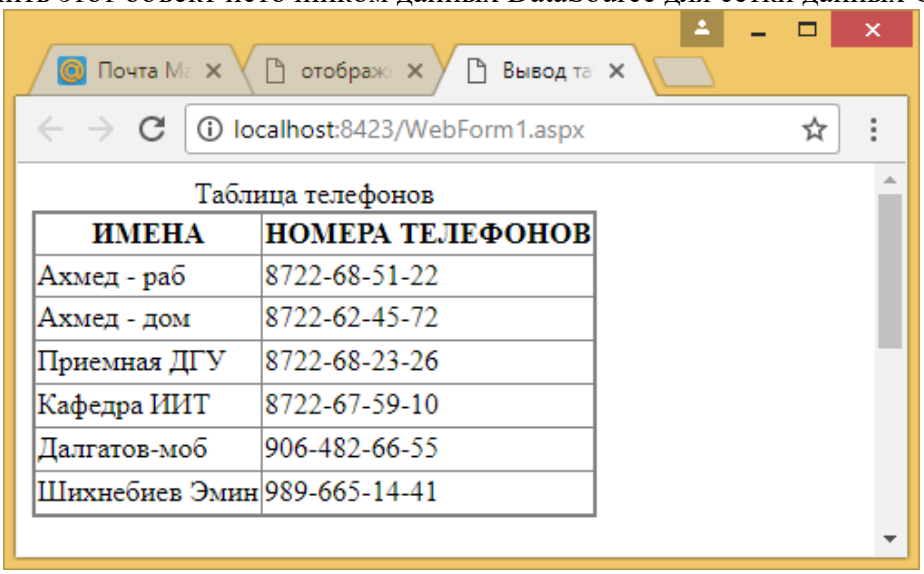
Отображение таблицы в компоненте GridView:

```
GridView1.Caption = "Таблица телефонов";
GridView1.BorderWidth = Unit.Pixel(2);
GridView1.BorderColor = System.Drawing.Color.Gray;
GridView1.DataSource = Таблица;
GridView1.DataBind();
```

Вначале нужно инициализировать два строковых массива: массив имен и массив телефонов

с помощью этих массивов заполнить ячейки объекта класса DataTable.

3. назначить этот объект источником данных DataSource для сетки данных GridView.

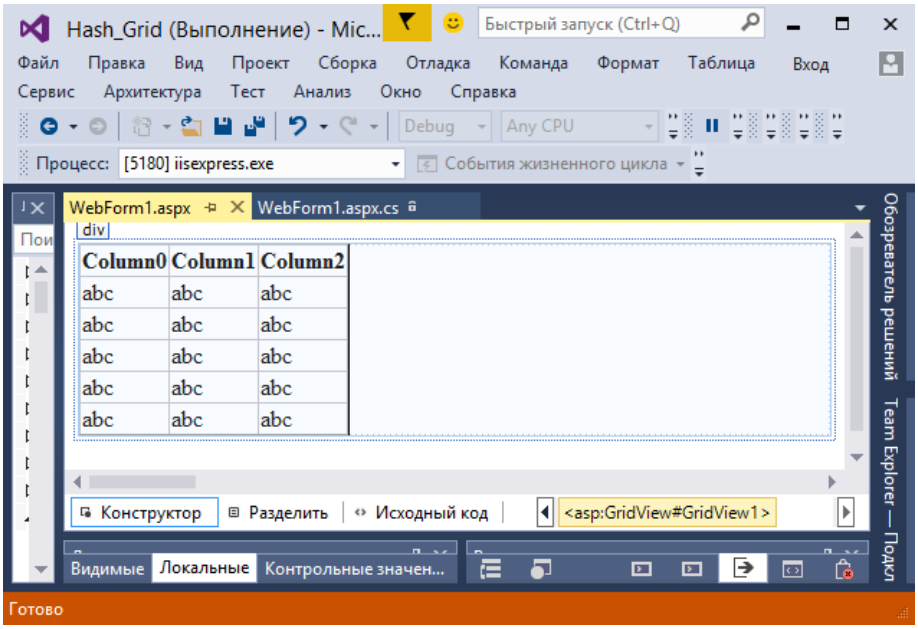


Лабораторная работа № 8. Разработка веб приложения отображения хэш-таблицы в веб-форме

Пояснение: *Хэш-таблица*- это таблица из двух столбцов, один из них содержит ключи, а второй - значения. То есть каждая строка в этой таблице образует пару "ключ - значение". Имея ключ в хэш-таблице, можно быстро найти значение. Хэш-таблицу можно назвать таблицей соответствий. Простейшим примером хэш-таблицы является таблица телефонов, которая участвовала в *задании 7*, однако там мы программировали ее просто как два массива. Если эти два массива поместить в хэш-таблицу, то ключом в данном случае было бы ФИО, а значением - номер телефона.

При этом программирование поиска значения по ключу оказалось бы тривиальной задачей, операция добавления и удаления пары также упрощается, поскольку хэш-таблица- это объект, который содержит соответствующие эти и другие методы. В реальной жизни много разнообразных примеров представления данных в виде хэш-таблицы. Например, таблица, где расширения файлов (txt, jpg, mdb, xls) являются ключами, а соответствующими значениями - программы, которые открывают файлы с такими расширениями (Notepad.exe, Pbrush.exe, MSAccess.exe, Excel.exe). Примерами хэш-таблиц являются англо-русский и другие словари, а также база доменных имен, которая доменному имени сопоставляет IP-адрес.

Вид формы



Задание: Разработать проект решения задачи: сопоставить в хэш-таблице государства в качестве ключей, а их столицы - в качестве значений.

Используя элемент управления **GridView,** вывести эту хэш-таблицу на веб-страницу.

Для этого:

1.создать новый проект шаблона **Web приложение ASP.NET,** в поле **Имя** указать имя **Hash_Grid**.

2. добавить к проекту веб-форму (в меню **Проект** выбрать команду **Добавить новый элемент** и в открывшемся окне выбрать шаблон **Веб форма**

3. перетащить на форму из **Панели элементов** (из подраздела **Данные)** элемент управления **GridView.**

1. При обработке события загрузки веб-страницы нужно создать объект класса **Hash_table** и заполнить его парами: **"ключ - значение"**. Хэш-таблицу **"ключ - значение"** можно заполнить парами, допустимы разные формы записи:

1. через присваивание
2. посредством метода **Add.**

В качестве ключа указать Страну, в качестве значения – **Столицу**.

Создать вспомогательный объект **Таблица** класса **DataTable**, в который выгружаются данные из хэш-таблицы оператором цикла **foreach** Хэш-таблица имеет структуру типа **DictionaryEntry**, которая позволяет перемещаться по рядам в цикле и, таким образом, получить все пары из хэш-таблицы.

В цикле выполнить заполнение объекта класса **DataTable**.

Для **GridView1** указать в качестве источника данных заполненный объект **DataTable**.

Комментарии к программному коду:

Название страницы: **"отображение на веб форме хэш-таблицы";**

{

Код создания объекта Хеш:

```
var Хэш = new System.Collections.Hashtable();
        // Заполнение хэш-таблицы:
        // Можно добавлять записи
```

Добавлять записи в таблицу в формате "ключ-значение" можно разными способами:

1. Хэш["Белорусия"] = "Минск";

2. **Хэш.Add("Россия", "Москва");**

Нужно добавить до 15 записей

Создание обычной таблицы выполняется командой:
var Таблица = new System.Data.DataTable();

Заполнение шапки"шапки" таблицы вывода
Таблица.Columns.Add("ГОСУДАРСТВА");
Таблица.Columns.Add("СТОЛИЦЫ");
Цикл выгрузки пары "ключ-значение"из хеш-таблицы в обычную таблицу по рядам:
foreach (System.Collections.DictionaryEntry ОднаПара in Хэш) //структура DictionaryEntry определяет пару ключ-значение

Таблица.Rows.Add(ОднаПара.Key, ОднаПара.Value);

Отображение таблицы в компоненте GridView:
GridView1.Caption = "Таблица государств"; // Заголовок таблицы
GridView1.BorderWidth = Unit.Pixel(2);
GridView1.BorderColor = System.Drawing.Color.Gray;
GridView1.DataSource = Таблица;
GridView1.DataBind();

Лабораторная работа № 9. Базовые технологии доступа к БД

Цель работы: Изучить основные способы работы с наборами данных. Получить навыки проектирования несложных фактографических систем.

Указания к выполнению лабораторной работы

Ранее уже говорилось, что *наборы данных* представляют собой группы записей, переданных из базы данных в приложения для просмотра и редактирования. Каждый набор данных инкапсулирован в специальном компоненте доступа к данным. Основные свойства и методы базового класса наборов данных (**TDataSet)** уже были рассмотрены нами ранее. Кроме того, мы познакомились с основными компонентами наборов данных, взаимодействующих с базами данных посредством ADO (**ADODataTable, ADODataQuery, ADOConnection** и др.). Компоненты наборов данных предоставляют разработчику довольно широкие возможности для эффективной обработки данных. Рассмотрим некоторые из них:

- **Навигация по набору данных**. Осуществляется с помощью уже рассматривавшихся методов перемещения *курсора* (указателя текущей записи) по набору данных: **First()**, **Last()**, **Next()**, **Prior()**, **MoveBy(int)**. Например, для последовательного перебора записей, начиная с текущей позиции до конца набора данных компонента **ADODataTable1**, можно использовать следующий цикл перебора:

```
ADODataTable1->Open()
...
while (!ADODataTable1->Eof)
{
      ...   // выполняются некоторые действия
      ADODataTable1->Next();
}
```

Интересным способом навигации является применение закладок. Подобно тому, как в книге можно заложить закладкой нужную страницу и впоследствии быстро к ней вернуться, также и в наборах данных можно осуществить аналогичные действия для отдельной записи. Для этого используются следующие методы набора данных:

- **void* GetBookmark()** – создает и возвращает закладку для текущей записи набора данных;
- **GotoBookmark(void *bookmark)** – перемещает курсор на запись, которая соответствует параметру bookmark (закладке).

- **Сортировка записей** является важной и довольно сложной проблемой. Можно выделить три способа определения нужного порядка следования записей:
  - *Индексирование полей таблиц БД*. Этот способ является возможным при использовании компонентов набора данных **ADODataTable**. При проектировании базы данных для каждой таблицы можно задать одно или несколько *индексированных полей*, позволяющих упорядочивать записи по значениям этого поля по возрастанию, либо по убыванию. После этого записи набора данных можно упорядочивать по любому индексированному полю, присвоив его наименование свойству IndexName. Рассмотрим, **например**, таблицу городов из базы данных, построенной для ИС «Телефонный справочник» (см. лабораторные работы № 3 и 4). Изначально список записей в наборе данных будет соответствовать действительному порядку размещения записей в таблице БД, поэтому при последовательном просмотре набора данных в программе получим:

```
Москва
Новосибирск
Санкт-Петербург
Бердск
```

Используя **конструктор таблицы** в Microsoft Access, сделаем поле city_name указанной таблицы индексированным. После этого, в качестве свойства IndexName для табличного набора данных можно указывать не только поле первичного ключа city_id, но и поле city_name. В последнем случае последовательный просмотр набора данных в программе даст упорядоченный список наименований городов:

```
Бердск
Москва
Новосибирск
Санкт-Петербург
```

  - *Определение порядка следования записей при построении SQL-запроса*. Этот способ используется при работе с компонентами **ADODataQuery** и их аналогами. Способы упорядочения записей в SQL-запросах на выборку данных будут подробно рассмотрены в курсе «Базы Данных».
  - Сортировка записей внутренними средствами компонента набора данных. Данный способ может быть использован не всегда, поскольку его применение ограничивается как свойствами компонента набора данных (который может не поддерживать внутреннюю сортировку записей), так и свойствами базы данных, с которой взаимодействует приложение. Компонент **ADODataTable** имеют свойство **IndexFieldNames**, которому можно присваивать список наименований полей, по которым будет производиться упорядочение. Поля в списке должны быть только полями данных (вычисляимые и поля синхронного просмотра не допускаются); одно поле отделяется от другого через точку с запятой. Сортировка набора данных при непустом значении свойства **IndexFieldNames** выполняется следующим образом: сначала записи упорядочиваются по значению первого поля, затем, если эти значения совпадают, – то по значениям второго поля и так далее. Список полей может состоять из единственного поля. **Например**, можно сортировать записи рассмотренной выше таблицы городов по значению кода города. Для этого достаточно присвоить свойству **IndexFieldNames** его наименование: (city_code). При этом поле city_code не обязательно должно быть индексированным. Список записей при последовательном просмотре набора данных в этом случае примет вид:

```
Новосибирск
Бердск
Москва
Санкт-Петербург
```

- **Поиск записей** в наборе данных может выполняться двумя способами.

- *Для позиционирования на нужной записи* используются методы **Locate** и **Lookup**:

Метод **Locate** ищет первую запись, удовлетворяющую критерию поиска, и, если такая запись найдена, делает ее текущей. Параметр **KeyFields** должен содержать список наименований одного или нескольких полей, по которым ведется поиск. В случае нескольких полей их названия разделяются точкой с запятой. Критерии поиска задаются в вариантном массиве **KeyValues** так, что *i*-тое значение в этом массиве автоматически ставится в соответствие *i*-тому полю в **KeyFields**. Параметр Options позволяет включить дополнительные необязательные режимы поиска. Метод Locate возвращает значение **true**, если запись найдена и установлена в качестве текущей, и **false** – в противном случае.

Метод **Lookup**, также как и метод **Locate**, находит запись, удовлетворяющую заданным условиям поиска, но не делает ее текущей, а возвращает значения некоторых полей этой записи. Кроме параметров **KeyFields** и **KeyValues**, смысл которых совпадает с одноименными параметрами метода **Locate**, функция метода **Lookup** содержит параметр **ResultFields**, который определяет список наименований полей, значения которых будут возвращены. Результатом метода является вариантный массив значений, причем *i*-тое значение в этом массиве соответствует *i*-тому полю списка **ResultFields**. Если запись, удовлетворяющая условиям поиска, не обнаружена, то возвращается пустое значение (*Null*()) Метод **Lookup** целесообразно использовать в тех случаях, когда изменять текущую запись нежелательно.

- *Для фильтрации записей* по определенному критерию используются свойства **Filter**, **Filtered** и **FilterOptions**, а также событие **OnFilterRecord**.

Свойство **Filter** позволяет задать *критерий фильтрации*. В этом случае набор данных будет отфильтрован, как только его свойство **Filtered** станет равным **true**. Синтаксис описания критериев фильтрации предоставляет разработчику довольно широкие возможности, в частности:

- Использование наименований полей и константных значений;
- Использование операций сравнения (<,>,=,>=,<=,<>);
- Использование арифметических операций;
- Использование логических операций (NOT, AND, OR);
- Использование спецсимвола “*” для фильтрации по частичному соответствию.

Чтобы реализовать более сложные алгоритмы фильтрации набора данных можно использовать событие **OnFilterRecord**. Оно возникает всякий раз, когда значение свойства **Filtered** устанавливается в true. Обработчик этого события имеет два параметра: указатель на фильтруемый набор данных (приведенный к базовому классу **TDataset**) и переменную **Accept**, в которую в результате выполнения обработчика должно быть помещено либо значение **true**, ели запись удовлетворяет условиям фильтра, либо значение **false** в противном случае. Следует помнить, что при включении фильтра указанное событие вызывается *для каждой записи*, поэтому для больших наборов данных такая фильтрация может выполняться довольно долго.

Задания к лабораторной работе

1. В соответствии с вариантом задания спроектировать и реализовать простейшую фактографическую информационную систему.

2. При проектировании БД рекомендуется использовать результаты выполнения лабораторных работ 4 и 5 курса «Информационные системы».
3. При реализации программного приложения для работы с БД выполнить следующие требования:
    а. Разработку программного приложения начать с проектирования пользовательского интерфейса и подготовки прототипа приложения.
    б. Развитие проекта программного приложения осуществлять в соответствии с принципами итеративной разработки.
    в. Обеспечить возможность добавления, редактирования и удаления записей во все основные таблицы БД;
    г. Обязательно использовать компонент TDBLookupComboBox для редактирования и/или определения условий поиска информации.
    д. Обязательно использовать поля синхронного просмотра и вычисляемые поля;
    е. Обязательно использовать один из способов упорядочения набора данных;
    ж. Обязательно использовать один из способов поиска информации в наборе данных.
4. В отчет о выполнении лабораторной работы включить:
    а. Модель данных (ER-диаграмму) ИС;
    б. Перечень и содержание выполненных итераций по разработке программного приложения;
    в. Результаты тестирования программы;
    г. Выводы по проделанной работе.

Варианты заданий

1. *ИС пункта проката*. Таблица предметов содержит название предмета, количество предметов в пункте проката, стоимость одних суток проката. Таблица выданных предметов содержит наименование предмета, фамилию арендатора, дату выдачи, количество суток, на которое произведена выдача, ожидаемую дату возврата, реальную дату возврата, сумму оплаты.
   Обязательные требования к ИС :
    а. Возможность получения информации о выданных предметах с дополнительным полем, в котором рассчитывается сумма оплаты за прокат. Учесть, что при выдаче предмета более, чем на десять дней, арендатору должна быть предоставлена скидка в 10%.
    б. Возможность получения списка всех выданных предметов;
    в. Возможность получения списка всех предметов, которые не были сданы в срок;

2. *ИС кадров предприятия*. Основная таблица содержит фамилию, отдел, оклад, дату рождения, дату приема, дату увольнения (если работает – пустое значение). Справочная таблица отделов содержит название отдела, фамилию начальника.
   Обязательные требования к ИС :
    а. Возможность получения информации о сотрудниках и их заработной плате. Учесть, что заработная плата сотрудников, работающих в организации более 5 лет, должна быть увеличена на 20% по сравнению с окладом.
    б. Возможность получения информации об отделах с дополнительными вычисляемыми полями, в которых указывается численность отдела и общий фонд заработной платы (с надбавками).

3. *ИС склада*. Имеется справочник клиентов и справочник товаров. Справочник товаров содержит наименование, количество, цену. Справочник клиентов содержит имя и

адрес клиента. Таблица фактур содержит наименование товара, приобретенное количество, наименование покупателя, дату приобретения, дату оплаты.
Обязательные требования к ИС :

а. Возможность получения информации по каждому товару с указанием стоимости каждой покупки;
б. Возможность получения информации по каждому клиенту с указанием списка неоплаченных им покупок и их количества;
в. Возможность получения списка всех неоплаченных фактур с указанием общей суммы к оплате.

4. *ИС телефонной станции*. Справочник абонентов содержит номер телефона, фамилию, адрес абонента. Справочник городов содержит коды городов, названия городов и зону. Тарифный справочник определяет стоимость одной минуты разговора в зависимости от зоны. Основная таблица содержит дату переговоров, время переговоров, дату оплаты (если отсутствует, то разговор не оплачен), номер телефона абонента, код города, продолжительность разговора.
Обязательные требования к ИС :
   а. Возможность получения информации о междугородних переговорах с указанием стоимости звонка. Учесть, что если звонок был произведен в ночное время (от 22:00 до 06:00), абоненту предоставляется скидка в 50%;
   б. Возможность получения списка городов из самой дорогой тарифной зоны.

5. *БД радиостанции*. Справочники звукооператоров и ведущих содержат фамилии, нормированный оклад (за 20 часов в эфире в месяц), дату принятия на работу. Основная таблица содержит название передачи, дату и время выхода в эфир, продолжительность в часах, фамилии звукооператора и ведущей.
Обязательные требования к ИС :
   а. Возможность получения информации о выходах сотрудников радиостанции в эфир в течение текущего месяца (с указанием оплаты за каждый выход). Учесть, что расчет оплаты производится по вычисленному в соответствии с окладом часовому тарифу с 50% надбавкой за работу в ночные часы (22:00-8:00) и с 10% надбавкой за стаж работы более 5 лет.
   б. Возможность получения информации о передачах, которые выходили в заданное время заданного дня недели.

6. *ИС библиотеки*. Основный справочник книг содержит шифр книги, фамилию автора, название книги, код УДК, стоимость, количество экземпляров Каталог читателей содержит номер билета, фамилию читателя, домашний адрес. Основная таблица выданных книг содержит шифр книги, номер билета, дату выдачи, дату фактического возврата (если книга на руках, то пустое значение), дату обязательного возврата (2 недели от даты выдачи).
Обязательные требования к ИС:
   а. Возможность получения информации о выданных книгах с указанием количества просроченных дней и суммы штрафа. Учесть, что начисление штрафа начинается с 3-го дня просрочки и составляет 5% от стоимости книги.
   б. Возможность получения информации о количестве выданных и количестве просроченных книг для некоторого читателя.

7. *ИС деканата*. Справочник «Студенты» содержит номер студенческого билета, ФИО студента, номер группы. Справочник «Дисциплины» - шифр и название предмета.

Таблица «Успеваемость» содержит номер студенческого билета, название дисциплины и полученные оценки.
Обязательные требования к ИС :

а. Возможность получения информации о дисциплинах, которые успешно сданы определенным студентом;
б. Возможность получения информации о дисциплинах с указанием тех, по которым у студентов есть долги;
в. Возможность получения информации о списке и количестве студентов определенной группы;

8. *ИС страховой компании*. Справочник видов страхования содержит наименование предметов страхования (недвижимость, транспорт), возможный срок страхования и процент страховой суммы от стоимости предмета. Основная таблица содержит фамилию страхователя, название предмета, оценочную стоимость, код вида страховки, дату страхования, дату наступления страхового случая (если он произошел), ежемесячный взнос, дата внесения последнего платежа.
Обязательные требования к ИС :
   а. Возможность получения информации о количестве действующих договоров страховки определенного вида.
   б. Возможность получения информации о договорах страхования с указанием полной информации о виде страховки и размере страховой суммы;
   в. выдача информации о договорах страхования, по которым наступил страховой случай.

9. *ИС турагентства.* Справочник туров содержит информацию о курорте, программе тура, базовой стоимости тура. Таблица заездов содержит информацию о возможных датах заезда для каждого из тура, а также цену путевки для тура с указанной датой как процент от базовой стоимости тура. Журнал заказов содержит фамилию, имя, отчество клиента, код тура, код заезда.
Обязательные требования к ИС :
   а. Возможность получения информации о турах, в которые можно отправиться в ближайшие 2 недели, – с указанием стоимости путевки;
   б. Возможность получения информации о списке и количестве путешественников, отправляющихся в определенный заезд;
   в. Возможность получения информации о путешественниках, которые отправлялись более чем в один тур.

10. *ИС банка.* Справочник вкладов содержит информацию о наименовании вклада, минимальном размере, сроке действия договора, процентной ставке по вкладу. Каталог клиентов содержит информацию о вкладчике (ФИО, паспортные данные, адрес), тип вклада, сумму вклада, дату заключения договора.
Обязательные требования к ИС :
   а. Возможность получения информации о клиентах, срок действия договора с которыми истек – с указанием суммы процентов и общей суммы возврата.
   б. Возможность получения информации о вкладчиках, заключивших договор в течение последнего месяца, упорядочив список по убыванию суммы.
   в. Возможность получения информации о денежных поступлениях и выплатах банка за последний месяц с указанием итогового сальдо.

11. *ИС супермаркета.* Справочник товаров содержит информацию о коде товара (штрих-код), его наименовании, цене, критическом количестве оставшихся товаров. Справочник операций содержит сведения о коде, наименовании и знаке операции: «+»

– поступление товара, «–» – расход товара (например, совершение покупки), «0» – прочие операции. Журнал операций содержит сведения о типе операции, коде и количестве товара, участвующего в операции, номере обосновывающего документа (например, кассового чека).

Обязательные требования к ИС :

а. Возможность получения информации о товарах, запасы которых необходимо пополнить (фактическое количество в магазине меньше критического)
б. Возможность получения информации об операциях, связанных с расходом любого выбранного товара с указанием стоимости каждой операции;
в. Возможность получения информации о самой крупной покупке любого выбранного товара за день;
г. Возможность получения информации о самой крупной покупке, совершенной одним клиентом.

Контрольные вопросы

1. Понятие набора данных.
2. Класс **TDataSet** и производные компоненты наборов данных.
3. Навигация по набору данных.
4. Способы упорядочения записей в наборе данных.
5. Поиск данных методом **Locate**. Необязательные режимы поиска.
6. Зависит ли результат выполнения метода **Locate** от способа упорядочения записей. Ответ обосновать.
7. Поиск данных методом **Lookup**.
8. Фильтрация данных. Синтаксис строки критерия фильтрации.
9. Свойство **FilterOption**.
10. Фильтрация данных с использованием события **OnFilterRecord**.

5. **Образовательные технологии**

Основными образовательными технологиями проведения курса «Технологии программирования» являются:

- Лекции, сопровождаемые компьютерными презентациями;
- лабораторные работы, в рамках которых составляются и тестируются программы, иллюстрирующие теоретический материал лекций;
- самостоятельная работа студентов, включающая усвоение теоретического материала, поиск дополнительного материала и эффективных способов выполнения заданий, завершение выполнения лабораторных работ; оформление и подготовка к защите лабораторных работ, подготовка к текущему контролю знаний и к итоговому экзамену;
- разработанные индивидуальные задания для самостоятельной работы;
- рейтинговая технология контроля учебной деятельности студентов для обеспечения их ритмичной работы в течение семестра
- консультирование студентов по вопросам учебного материала и выполнения курсового заданий.

6. Учебно-методическое обеспечение самостоятельной работы студентов

Таблица – Технологическая карта самостоятельной работы студента

№	Темы дисциплины	Задания для самостоятельной работы	Трудоёмкость задания, часы	Перечень учебно-методического обеспечения
1.	Проблемы разработки	Выполнить тест №1	2	Работа с

	сложных программных систем			источниками 2, 3.
2.	Жизненный цикл программного обеспечения	Выполнить тест №2	*2*	Работа с источниками 1, 3, 5.
3.	Унифицированный процесс разработки программного обеспечения	Выполнить тест №3	*2*	Использовать источники 1, 4, 5 и Интернет-ресурсы.
4.	Экстремальное программирование	Выполнить тест №4	4	Использовать источники 4, 6 и Интернет-ресурсы
5.	Анализ предметной области	Выполнить тест №5	4	Использовать источники 1,4, 5 и Интернет-ресурсы
6.	Качество программного обеспечения	Выполнить тест №6	4	Использовать источники 3,4, 5 и Интернет-ресурсы
7.	Подготовка к экзамену. Сдача экзамена.		36	Все темы курса
	Итого:		**54**	

Контроль результатов освоения дисциплины

Текущий контроль успеваемости осуществляется путем оценки результатов выполнения заданий лабораторных, самостоятельной работ, посещения лекций.

Промежуточная аттестация осуществляется в форме экзамена, который выставляется по результатам проверки выполнения тестов и заданий.

Оценочные средства результатов освоения дисциплины, критерии оценки выполнения заданий представлены в разделе «Фонды оценочных средств для проведения промежуточной аттестации» и фонде оценочных средств образовательной программы.

7. Фонд оценочных средств для проведения текущего контроля успеваемости, промежуточной аттестации по итогам освоения дисциплины

7.1. Перечень компетенций с указанием этапов их формирования в процессе освоения образовательной программы.

Перечень компетенций с указанием этапов их формирования приведен в описании образовательной программы.

Код компетенции из ФГОС ВО	Наименование компетенции из ФГОС ВО	Планируемые результаты обучения	Процедура освоения
ПК-37	способностью выбирать и оценивать способ реализации информационных систем и устройств (программно-, аппаратно- или программно-аппаратно-)	Знает: технологии выбора и оценки способов реализации ИС Умеет: выбирать и оценивать существующие технологии разработки ИС для решения поставленной задачи	Письменный опрос Круглый стол

	для решения поставленной задачи	Владеет: практическими навыками выбора и оценки современных технологий проектирования и разработки ИС для решения поставленной задачи	

7.2. Типовые контрольные задания

1. Задание

Признаки "небольшой" программы

-)решение четко поставленной, несущественной для практической деятельности задачи

-)решение одной или нескольких значимых для пользователей задач, не имеющих четкой постановки

-)периодически требуется доработка программы с появлением новых версий

-)для выполнения своих задач программа должна взаимодействовать с другими программами

-)есть существенная необходимость в документировании программы

2. Задание

Признаки "небольшой" программы

-)отсутствует необходимость в документировании программы

-)низкая производительность приносит существенный ущерб

-)для выполнения своих задач программа должна взаимодействовать с другими программами

-)в разработку вовлечено большое количество людей

3. Задание

Признаки сложной программной системы (программного комплекса)

-)для выполнения своих задач программа должна взаимодействовать с другими программами

-)в разработку вовлечено большое количество людей

-)неправильная работа программы наносит ощутимый ущерб

-)отсутствует необходимость в оптимизации производительности программы

4. Задание

Свойства сложных программных систем

-)низкая производительность приносит существенный ущерб

-)требуется документация для обучения пользователей

-)отсутствие проектной документации

-)ущерб от неправильной работы программы незначителен

5. Задание

Свойства сложных программных комплексов

-)удобство в использовании программы носит существенный характер

-)наличие проектной документации

-)в разработке участвует один человек

-)система решает одну четко поставленную задачу

6. Задание

Виды документации, требуемой для эксплуатации и развития программной системы

-)пользовательская

-)технический

-)инженерный

-)технологический

8. Задание

Системная инженерия изучает следующие аспекты создания программно-аппаратных систем …

-)разработка программно-аппаратных систем

-)эксплуатация программно-аппаратных систем

-)интеграция программной и аппаратной составляющих

-)разработка аппаратных устройств

9. Задание

Аспекты организации экономически эффективной работы

-)организация совместной работы коллектива разработчиков

-)учет требований к пользовательским свойствам программы

-)учет квалификации пользователя при проектировании пользовательских интерфейсов

-)создание безошибочно работающего программного продукта

10. Задание

Объективные причины отсутствия "безошибочно работающих" сложных программных систем

-)противоречие требований друг другу

-)изменение требований с течением времени

-)сложность поиска ошибок в коде программы

-)сложность исправления найденных ошибок

11. Задание

Сложные программные системы с точки зрения наличия в них ошибок условно делятся на …

-)достаточно качественные

-)недостаточно качественные

-)правильные

-)неправильные

12. Задание

Основные проблемы разработки сложных программных систем связаны с нахождением разумного компромисса между затратами на разработку и … ее результата

Правильные варианты ответа: качеством; качество;

13. Задание

К наиболее важным ресурсам при оценке затрат на программу относятся …

-)время выполнения проекта

-)бюджет проекта

-)персонал

-)стоимость оборудования

14. Задание

Функциональные возможности, надежность, гибкость, удобство внесения изменений являются аспектами … программной системы

Правильные варианты ответа: качества; качество;

15. Задание

К процессам создания программных систем относятся понятия …

-)жизненный цикл

-)качество

-)процесс разработки

-)разработка документации

7.3. Методические материалы, определяющие процедуру оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций.

Общий результат выводится как интегральная оценка, складывающая из текущего

контроля - 30% и промежуточного контроля - 70%.

Текущий контроль по дисциплине включает:

- посещение занятий - _0__ баллов,
- участие на практических занятиях - 20 баллов,
- выполнение лабораторных заданий – 60 баллов,
- выполнение домашних (аудиторных) контрольных работ – 20 баллов.

Промежуточный контроль по дисциплине включает:

- устный опрос - 30 баллов,
- письменная контрольная работа - 30 баллов,
- тестирование - 40 баллов.

8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины.

а) основная литература

1. Савич, Уолтер. Программирование на C++ : [Перевод] / Савич, Уолтер. - СПб. и др. : Питер: Питер принт, 2004. - 779 с. ; 24 см. - ISBN 5-94723-582-X : 380-00..
2. Программирование на языке C++ : 1 модуль: метод. пособие / [сост. З.Х.Ахмедова]; М-во образования и науки РФ, Дагест. гос. ун-т. - Махачкала : Изд-во ДГУ, 2010. - 34 с..

б) дополнительная литература

1. Эндрю Троелсен. Язык программирования C# 7 и платформы. NET и NET Core. Пер. с англ. - М.: "Вильямс", 2018. 1328с.
2. Эндрю Троелсен. Язык Программирования C# 5.0 и платформа .NET 4.5. Пер. с англ. - М.: "Вильямс", 2015. 1312с.
3. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#. Пер. с англ. - М.: «Русская Редакция» ; СПб. : Питер , 2007. 656 стр.
4. Акчурин Э.А. Программирование на языке C# в MS Visual Studio .Net или SharpDevelop. Учебное пособие. Самара, ИУНЛ. ПГУТИ, 2011, 150 с.
5. Нэш. C# 2010. Ускоренный курс для профессионалов. М: ИД Вильямс, 2010. 592с.
6. Макки А. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов. Пер. с англ. - М.: "Вильямс", 2010. 412с.
7. Нейгел К. и др. C# 2008 и платформа .Net 3.5 для профессионалов. – М. Диалектика, 2009, 1392 с.
8. Макаров А. и др. CIL и системное программирование в Microsoft.NET: – М. : Интернет-УИТ, 2006. 328 с.

9. **Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины.**

1. ЭБС IPRbooks: http://www.iprbookshop/ru/

2. Электронно-библиотечная система «Университетская библиотека онлайн»(архив):www.biblioclub.ru

3. Единое окно доступа к образовательным ресурсам. http://window.edu.ru/

4. http://www.microsoft.com/msf

5. http://www.uml.org

6. http://www.wikipedia.org

10. Методические указания для обучающихся по освоению дисциплины.

Критерии и показатели сформированности компетенций

Степень (уровень) сформированности компетенций на этапе изучения дисциплины «Современные технологии программирования» оценивается по следующим критериям: мотивационно-ценностный, когнитивный, операционно-деятельностный. Показателями

критериев являются результаты обучения по дисциплине (дескрипторы) таблицы 1. Инструментарий, этапы измерения показателей и критериев компетенции представлены в таблице.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Этапы контроля: раздел 2 (самостоятельная работа), раздел 3 (самостоятельная работа), раздел 4 (самостоятельная работа), раздел 5 (самостоятельная работа), раздел 6 (самостоятельная работа), раздел 7 (самостоятельная работа), экзамен.

Время на выполнение: 60 мин.

Метод оценивания: автоматизированный

Критерии оценки результатов выполнения: менее 50% правильных ответов - неудовлетворительно, менее 65% - удовлетворительно, менее 86% хорошо, 86% и более – отлично.

11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень программного обеспечения и информационных справочных систем.

Информационные технологии

Образовательный процесс осуществляется с применением локальных и распределенных информационных технологий (таблица 4, 5).

Таблица 4 – Локальные информационные технологии

Группа программных средств	Наименование программного продукта
Офисные программы	Microsoft Office
	Libre Office
Распознавание текста и речи	ABBYY FineReader 2010
Средства разработки	MicroSoft Visual Studio 2015 MicroSoft SQL Server 2012
Методические указания и материалы по видам занятий	Акчурин Э., Ильин А. Программирование на языке C#. ЛР в ИСР Visual C# 2010 Express или SharpDevelop. . Самара, ИУНЛ. ПГУТИ, 2011, 114 с.

Таблица 5 – Распределенные информационные технологии

Группа	Наименование
Система тестирования	Система сетевого компьютерного тестирования ДГУ www.ts.icc.dgu.ru
Библиотеки и образовательные ресурсы	Электронная библиотека ДГУ http://www.elib.dgu.ru
	Кафедральные сайты ДГУ http://cafedra.dgu.ru
	Сайте электронных образовательных ресурсов ДГУ http://eor.dgu.ru
Система электронного обучения	Сервер электронного обучения moodle http://moodle.dgu.ru

12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине.

Таблица 6 – Материально-техническая база

Помещения для осуществления образовательного процесса	Перечень основного оборудования (с указанием кол-ва посадочных мест)	Адрес (местоположение)

Аудитории для проведения лекционных занятий		
Лекционные аудитории	Интерактивная доска, ноутбук; проектор. Количество посадочных мест – 30.	Ауд. 3-14, 4-16, 2-10, учебный корпус № 8, г.Махачкала, ул. Джержинского, 12.
Аудитории для проведения лабораторных занятий, контроля успеваемости		
Компьютерный класс	Компьютеры с выходом в Интернет и доступом в электронную информационно-образовательную среду вуза. Количество посадочных мест – 15.	Компьютерный зал № 2 учебный корпус № 3, г.Махачкала, ул. Джержинского, 12.
Помещения для самостоятельной работы		
Компьютерные классы	Компьютеры с выходом в Интернет и доступом в электронную информационно-образовательную среду вуза. Количество посадочных мест – 15	Компьютерный зал № 1, учебный корпус № 3, г. Махачкала, ул. Джержинского, 12.
Читальный зал библиотеки ДГУ	Компьютеры с выходом в Интернет и доступом в электронную информационно-образовательную среду вуза. Количество посадочных мест – 30.	Электронный читальный зал научной библиотеки ДГУ, г. Махачкала, ул. Батырая, 4