Министерство образования и науки
Донецкой Народной Республики
ГОУ ВПО «Донецкий национальный университет»
Кафедра теории упругости и вычислительной математики

ПРАКТИЧЕСКИЙ КУРС СОВРЕМЕННЫХ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

для студентов направления подготовки
01.04.02 Прикладная математика и информатика

ДОНЕЦК 2016

УДК 004.9(07)
ББК 3973я73
У91

Рекомендовано к изданию Ученым советом
ГОУ ВПО «Донецкий национальный университет»
(протокол № 10 22.12.2016 г.)

Практический курс современных компьютерных технологий: учебно-методическое пособие / Сост.: Е.В. Авдюшина.– Донецк: ДонНУ, 2016. – 138 с.

Рецензенты:

Щепин Н.Н., кандидат физико-математических наук, доцент, ГОУ ВПО «Донецкий национальный университет»;

Машаров П.А., кандидат физико-математических наук, доцент, ГОУ ВПО «Донецкий национальный университет»

В учебно-методическом пособии изложена методика изучения курса «Современные компьютерные технологии»: структурирование учебного материала по темам, теоретический материал, задания к лабораторным работам и самостоятельной работе студентов, вопросы для подготовки в экзамену, образцы выполнения заданий по самостоятельной работе студентов и список литературы. Учебный материал содержит основные сведения о современных подходах разработки информационных систем, визуализации и обработки данных.

Учебно-методическое пособие предназначено для студентов специальности «Прикладная математика и информатика» и может быть использовано студентами других направления прикладной математики, информатики и информационных технологий.

УДК 004.9(07)
ББК 3973я73

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Учебная дисциплина «Современные компьютерные технологии» относится к вариативной части по выбору студента профессионального блока дисциплин подготовки студентов по направлению подготовки 01.04.02 Прикладная математика и информатика.

Содержание дисциплины является логическим продолжением дисциплин «Распределенная обработка данных в современных СУБД», «Распределенные информационные системы», и формирует основу для освоения дисциплин научно-исследовательская и Ассистенстская практики, а также может быть использована при написании магистерской диссертации.

Целью дисциплины является ознакомление магистрантов со специальными компьютерными технологиями, имеющими применение в области моделирования и создания специализированного программного обеспечения для решения прикладных задач в различных сферах жизнедеятельности.

Задачами являются формирование понимания студентами ключевых положений компьютерных технологий, структуры, связи с другими науками, понятий многоуровневое приложение, компоненты приложения, уровни данных, бизнес-логики и представления, целостного представления о видах информации, мировых информационных ресурсов, способах обработки информации.

Процесс изучения дисциплины направлен на формирование элементов следующих компетенций в соответствии с ГОС ВПО по данному направлению подготовки: способности к абстрактному мышлению, анализу, синтезу; готовности к саморазвитию, самореализации, использованию творческого потенциала; способности самостоятельно приобретать с помощью информационных технологий и использовать в практической деятельности новые знания и умения, в том числе в новых областях знаний, непосредственно не связанных со сферой деятельности, расширять и углублять свое научное мировоззрение; способности использовать и применять углубленные знания в

области прикладной математики и информатики; способности проводить научные исследования и получать новые научные и прикладные результаты самостоятельно и в составе научного коллектива; способности разрабатывать и применять математические методы, системное и прикладное программное обеспечение для решения задач научной и проектно-технологической деятельности; способности управлять проектами, планировать научно-исследовательскую деятельность, анализировать риски, управлять командой проекта; способности разрабатывать корпоративные стандарты и профили функциональной стандартизации приложений, систем, информационной инфраструктуры; способности разрабатывать аналитические обзоры состояния области прикладной математики и информационных технологий.

В результате изучения учебной дисциплины студент приобретает знания тенденций и направлений развития современных компьютерных технологий, программирования и разработки приложений; основных тенденций развития современных компьютерных технологий; существующих методов и стандартов управления проектами. Умениями, которые формирует дициплина являются выполнение различных математические расчеты с использованием современных компьютерных средств, работа с современными операционными системами и важнейшими прикладными программами обработки информации, представления информации, с базами данных, Интернет, разрабатка структуры приложения и его уровни, использую объектно-ориентированное программирование, приобретение навыков совместного использования различных языков программирования для создания приложений различных видов, разработка сетевые информационные приложения с использованием современных технологий программирования, применение принципов объектно-ориентированного программирования на языке C#, проектирования многоуровневую иерархию объектов, использования классов технологии ADO.Net для работы с базами данных и язык интегрированных запросов LINQ для работы с различными данными, создания приложений с использованием языка C# и Asp.Net, реализовывать аутентификацию пользователей веб-приложений. Также студент должен владеть техническими и программными

средствами, обеспечивающими применение компьютерных технологий, навыками практического применения современных информационных систем в различных сферах деятельности, методикой оценки эффективности проектов в информационных технологиях.

В рамках изучения дисциплины предусмотрены следующие формы организации учебного процесса: лекции, лабораторные занятия, самостоятельную работу студента.

Лекционные занятия предполагают овладение теоретическими основами дисциплины, лабораторные – для овладения методами разработки компьютерных систем.

Самостоятельная работа студентов предусматривает выполнение домашних заданий, подготовку к лабораторным занятиям, изучение учебно-методической литературы, составление конспектов, подготовку презентаций и докладов.

Текущий контроль осуществляется путем написания самостоятельных и контрольных работ для проверки текущих знаний теории и практики, модульной контрольной работы по проверке знаний теоретических и практических положений.

Тема 1. Объектно-ориентированный язык программирования C#

1.1. Синтаксис языка программирования C#

Язык программирования C# появился сравнительно недавно. Он является наследником языков C и C++, которые признаны одними из самых удачных языков программирования. С другой стороны он имеет родственные связи и с языком Java.

Язык C# тесно связан со средой .Net Framework [1]. Она обеспечивает возможность совместного использования различных языков программирования, контроль безопасности и переносимости программ. Общим между языком и средой являются

– общеязыковая среда выполнения (Common Language Runtime – CLR);

– библиотека классов .Net.

Все программы, написанные на языке C# являются объектно-ориентированными. Для реализации этого используются инкапсуляция, наследование и полиморфизм.

Переменная – это именованная область памяти, для которой может быть установлено значение.

Все переменные должны быть объявлены до своего использования. Для этого используется оператор

тип имя_переменной1, имя_переменной2, имя_переменнойN;

где тип — это конкретный тип объявляемой переменной, а имя_переменной — имя самой переменной.

Можно также описать переменную с инициализацией

тип имя_переменной1=значение_переменной1, …имя_переменнойN;

В этом случае значение переменной должно совпадать с ее типом.

Оператор присваивания обозначается одиночным знаком равенства (=). Общая форма такая

имя_переменной = выражение;

Здесь имя_переменной должно быть совместимо с типом выражения.

Можно использовать также составные операторы присваивания для арифметических и логических операций: +=, -=, &=, * =, |=, /=, ^=.

Условный оператор часто называют тернарным, т.к. в нем участвуют три операнда. Ниже приведена общая форма этого оператора.

Выражение 1 ? Выражение2 : Выражение3;

Здесь Выражение1 должно относиться к типу bool, а Выражение2 и Выражение3 – к одному и тому же типу. Значение выражения ? определяется следующим образом.

Оператор условного перехода имеет различные формы

```
if(условие) оператор;
else оператор;
```

где условие – это некоторое условное выражение, а оператор – адресат операторов if и else. Оператор else не является обязательным. Адресатом обоих операторов, if и else, могут также служить блоки операторов. Ниже приведена общая форма оператора if, в котором используются блоки операторов.

```
if (условие)
{
последовательность операторов
}
else
{
последовательность операторов
}
```

Здесь условие представляет собой булево, т.е. логическое, выражение, принимающее одно из двух значений: "истина" или "ложь". Если условие истинно, то оператор выполняется. А если условие ложно, то выполнение программы происходит, минуя оператор.

Оператором множественного выбора в C# является оператор switch, который обеспечивает многонаправленное ветвление программы. Следовательно, этот оператор позволяет сделать выбор среди нескольких альтернативных вариантов дальнейшего выполнения программы. Ниже приведена общая форма оператора switch.

```
switch{выражение) {
case константа1 : последовательность операторов break;
case константа2: последовательность операторов break;
case константа3: последовательность операторов break;
default: последовательность операторов break;
}
```

Заданное выражение в операторе switch должно быть целочисленного типа (char, byte, short или int), перечислимого или же строкового. А выражения других типов, например с плавающей точкой, в операторе switch не допускаются. Зачастую выражение, управляющее оператором switch, просто сводится к одной переменной. Кроме того, константы выбора должны иметь тип, совместимый с типом выражения. В одном операторе switch не допускается наличие двух одинаковых по значению констант выбора.

Операторы цикла имеют 4 вида [8]. Оператор for

```
for (инициализация; условие; итерация) оператор;
```

В самой общей форме в части инициализация данного оператора задается начальное значение переменной управления циклом. Часть условие представляет собой булево выражение, проверяющее значение переменной управления циклом. Если результат проверки истинен, то цикл продолжается. Если же он ложен, то цикл завершается. В части итерация определяется порядок изменения переменной управления циклом на каждом шаге цикла, когда он повторяется.

Еще одним оператором цикла в C# является **оператор while**:

while (условие) оператор;

где оператор — это единственный оператор или же блок операторов, а условие означает конкретное условие управления циклом и может быть любым логическим выражением. В этом цикле оператор выполняется до тех пор, пока условие истинно. Как только условие становится ложным, управление программой передается строке кода, следующей непосредственно после цикла. Как и в цикле for, в цикле while проверяется условное выражение, указываемое в самом начале цикла.

Третьим оператором цикла в C# является **оператор do-while**. В отличие от операторов цикла for и while, в которых условие проверялось в самом начале цикла, в операторе do-while условие выполнения цикла проверяется в самом его конце. Это означает, что цикл do-while всегда выполняется хотя бы один раз. Ниже приведена общая форма оператора цикла do-while.

```
do {
операторы;
} while (условие);
```

При наличии лишь одного оператора фигурные скобки в данной форме записи необязательны.

Оператор цикла foreach служит для циклического обращения к элементам коллекции, которая представляет собой группу объектов, и имеет вид

```
foreach (ТипЭлемента коллекции ИмяПереременой in ИмяКоллекции)
    {операторы с ИмяПереременой }
```

В C# определено несколько видов коллекций, к числу которых относится массив.

Массивы [9]. Массив представляет собой совокупность переменных одного типа с общим для обращения к ним именем. В C# массивы могут быть как одномерными, так и многомерными. Главное преимущество массива — в организации данных таким образом, чтобы ими было проще манипулировать. Например, массивы позволяют организовать данные таким образом, чтобы

легко отсортировать их. Массивами в C# можно пользоваться практически так же, как и в других языках программирования. Тем не менее у них имеется одна особенность: они реализованы в виде объектов.

Реализация массивов в виде объектов дает ряд существенных преимуществ, и далеко не самым последним среди них является возможность утилизировать неиспользуемые массивы средствам "сборки мусора".

Одномерный массив представляет собой список связанных переменных. Для использования массивом необходимо объявить переменную, которая может обращаться к массиву и создать экземпляр массива, используя оператор new. Так, для объявления одномерного массива обычно применяется следующая общая форма:

```
тип[ ] имя_массива = new тип[размер] ;
```

где тип объявляет конкретный тип элемента массива. Тип элемента определяет тип данных каждого элемента, составляющего массив. Обратите внимание на квадратные скобки, которые сопровождают тип. Они указывают на то, что объявляется одномерный массив. А размер определяет число элементов массива.

Объявление массива можно разделить на два отдельных оператора.

```
тип[ ] имя_массива;
имя_массива = new тип[размер] ;
```

В данном случае переменная имя_массива не ссылается на какой-то определенный физический объект. Только после выполнения второго оператора эта переменная ссылается на массив.

Доступ к отдельному элементу массива осуществляется по индексу: Индекс обозначает положение элемента в массиве. В языке C# индекс первого элемента всех массивов оказывается нулевым.

Инициализировать массив можно различными способами:

– вручную, присваивая каждому элементу значение;

– при описании, например, для одномерного массива:

тип[] имя_массива = {vail, val2, val3, ..., valN} ;

где val1-valN обозначают первоначальные значения, которые присваиваются по очереди, слева направо и по порядку индексирования. Для хранения инициализаторов массива в C# автоматически распределяется достаточный объем памяти. А необходимость пользоваться оператором new явным образом отпадает сама собой.

– при описании уже созданной ссылки, используя оператор new

тип[] имя_массива = new тип [] { значение1, значение2, …, значениеN};

или в форме

тип[] имя_массива;

имя_массива = new тип [] { значение1, значение2, …, значениеN};

или

тип[] имя_массива = new тип [размер]

{значение1, значение2, …, значениеN};

Но размер должен совпадать с количеством значений в фигурных скобках.

Границы массива в C# строго соблюдаются. Если границы массива не достигаются или же превышаются, то возникает ошибка, исключительная ситуация типа IndexOutOf RangeException, связанная с выходом за пределы индексирования массива, и программа преждевременно завершится.

Можно создать также многомерные массивы. Многомерным называется такой массив, который отличается двумя или более измерениями, причем доступ к каждому элементу такого массива осуществляется с помощью определенной комбинации двух или более индексов. Для объявления многомерного массива используется форма

тип[, . . .,] имя_массива = new тип[размер1, размер2, . . . размерN] ;

Количество запятых в первых квадратных скобках равно N-1.

Для инициализации многомерного массива достаточно заключить в фигурные скобки список инициализаторов каждого его размера. Ниже в качестве примера приведена общая форма инициализации двумерного массива:

```
тип[,] имя_массива = {
{val, val, val, ..., val),
{val, val, val, ..., val),
{val, val, val, ..., val}
};
```

где val обозначает инициализирующее значение, а каждый внутренний блок – отдельный ряд. Первое значение в каждом ряду сохраняется на первой позиции в массиве, второе значение — на второй позиции и т.д. Обратите внимание на то, что блоки инициализаторов разделяются запятыми, а после завершающей эти блоки закрывающей фигурной скобки ставится точка с запятой.

С каждым массивом связано свойство Length, содержащее число элементов, из которых может состоять массив. Для обращения к этому свойству используется точка имя_массива.Length. Если массив является многомерным, то функция вернет число элементов по всем измерениям. Если нужно знать число элементов внутри измерения, можно использовать вместо этого метод GetLength(). Свойство Rank позволяет получить количество измерений массива.

Оператор цикла foreach для массивов. Оператор foreach служит для циклического обращения к элементам массива. Общая форма оператора имеет вид

```
foreach (ТипМассива имя_переменной_цикла in ИмяМассива) оператор;
```

Здесь тип обозначает тип элемиента массива, который может также обозначаться как var; имя_переменной_цикла обозначает имя переменной управления циклом, которая получает значение следующего элемента коллекции на каждом шаге выполнения цикла foreach.

1.2. Инкапсуляция и элементы класса на языке C#

Все программы, написанные на языке C# являются объектно-ориентированными. Для реализации этого используются инкапсуляция, наследование и полиморфизм [24].

Инкапсуляция - это техника программирования, объединяющая данные и код [12]. При этом код должен обеспечить обработку данных, их защиту от внешнего вмешательства и неправильного использования. Объединение данных и программного кода характеризуют объект.

Язык C# является строго объектно-ориентированным, поэтому в классе определяется данные и код, который будет оперировать с этими данными. Класс представляет собой шаблон, по которому определяется форма объекта. Класс, по существу, представляет собой ряд схематических описаний способа построения объекта. При этом очень важно подчеркнуть, что класс является логической абстракцией. Физическое представление класса появится в оперативной памяти лишь после того, как будет создан объект этого класса.

Данные содержатся в членах данных, определяемых классом, а код – в функциях-членах. Члены данных также называют полями, к ним относятся переменные экземпляра и статические переменные, а к функциям-членам – методы, конструкторы, деструкторы, индексаторы, события, операторы и свойства.

Форма определения простого класса, содержащая только переменные экземпляра и методы, следующая

```
class имя_класса {
// Объявление переменных экземпляра.
специф_доступа тип переменная1;
специф_доступа тип переменная2;
//...
специф_доступа тип переменнаяN;
// Объявление методов.
```

```
специф_доступа возвращаемый_тип метод1 (параметры) {
// тело метода
}
специф_доступа возвращаемый_тип метод2 (параметры). {
// тело метода
}
//...
специф_доступа возвращаемый_тип методN (параметры) {
// тело метода
}
}
```

Методы класса. Метод состоит из одного или нескольких операторов. В грамотно написанном коде C# каждый метод выполняет только одну функцию. У каждого метода имеется свое имя, по которому он вызывается. В общем, методу в качестве имени можно присвоить любой действительный идентификатор. Следует, однако, иметь в виду, что идентификатор Main () зарезервирован для метода, с которого начинается выполнение программы. Кроме того, в качестве имен методов нельзя использовать ключевые слова C#. Если в методе используется переменная экземпляра, определенная в его классе, то делается это непосредственно, без указания явной ссылки на объект и без помощи оператора-точки. Ведь метод всегда вызывается относительно некоторого объекта его класса. Как только вызов произойдет, объект становится известным. Поэтому объект не нужно указывать в методе еще раз.

Выход из метода происходит при достижении закрывающей фигурной скобки или по оператору return. Имеются две формы оператора return: одна – для методов типа void, т.е. тех методов, которые не возвращают значения, а другая – для методов, возвращающих конкретные значения. Для возврата значения из метода в вызывающую часть программы служит следующая форма оператора return:

```
return значение;
```

где значение — это конкретное возвращаемое значение.

При вызове метода ему можно передать одно или несколько значений. Значение, передаваемое методу, называется аргументом. А переменная, получающая аргумент, называется формальным параметром, или просто параметром. Параметры объявляются в скобках после имени метода. Синтаксис объявления параметров такой же, как и у переменных. А областью действия параметров является тело метода. За исключением особых случаев передачи аргументов методу, параметры действуют так же, как и любые другие переменные. Функции могут иметь несколько параметров.

Конструкторы [13]. Конструктор инициализирует объект при его создании. У конструктора такое же имя, как и у его класса. Класс – это аналог метода, но конструктор не имеет возвращаемого значения, указанного явно. Ниже приведена общая форма конструктора.

```
доступ имя_класса(список_параметров) {
// тело конструктора
}
```

Конструктор используется для задания первоначальных значений переменных экземпляра, определенных в классе, или же для выполнения любых других действий, которые требуются для создания полностью сформированного объекта. Кроме того, доступ обычно представляет собой модификатор доступа типа public, поскольку конструкторы чаще всего вызываются вне класса. А список_параметров может быть как пустым, так и состоящим из одного или более указываемых параметров.

У всех классов имеется конструктор, используемый по умолчанию и инициализирующий все переменные экземпляра их значениями по умолчанию. Для большинства типов данных значением по умолчанию является нулевое, для типа bool – значение false, а для ссылочных типов – пустое значение. Но если определить свой собственный конструктор, то конструктор по умолчанию больше не используется.

Если конструктор имеет параметры, то для создания объекта класса необходимо использовать форму

new имя_класса (список_аргументов)

где имя_класса обозначает имя класса, реализуемого в виде экземпляра его объекта.

А имя_класса с последующими скобками обозначает конструктор этого класса. Если в классе не определен его собственный конструктор, то в операторе new будет использован конструктор, предоставляемый в C# по умолчанию. Следовательно, оператор new может быть использован для создания объекта, относящегося к классу любого типа.

При использовании оператора new свободная память для создаваемых объектов динамически распределяется из доступной буферной области оперативной памяти. Свободно доступная память рано или поздно исчерпывается. Это может привести к неудачному выполнению оператора new из-за нехватки свободной памяти для создания требуемого объекта. Именно по этой причине одной из главных функций любой схемы динамического распределения памяти является освобождение свободной памяти от неиспользуемых объектов, чтобы сделать ее доступной для последующего перераспределения. Во многих языках программирования освобождение распределенной ранее памяти осуществляется вручную. Например, в C++ для этой цели служит оператор delete. Но в C# применяется другой, более надежный подход: "сборка мусора".

Система "сборки мусора" в C# освобождает память от лишних объектов автоматически, действуя незаметно и без всякого вмешательства со стороны программиста. Если ссылки на объект отсутствуют, то такой объект считается ненужным, и занимаемая им память в итоге освобождается и накапливается. Эта утилизированная память может быть затем распределена для других объектов. "Сборка мусора" происходит лишь время от времени по ходу выполнения, поэтому, нельзя заранее знать или предположить, когда именно произойдет "сборка мусора".

Для четко контроля действий при окончания жизни объекта может применяться деструктор. Он будет вызываться непосредственно перед окончательным уничтожением объекта системой "сборки мусора". Но деструкторы применяются только в редких случаях. Общая форма деструктора:

```
~имя_класса() {
// код деструктора
}
```

где имя_класса означает имя конкретного класса. У деструктора отсутствуют возвращаемый тип и передаваемые ему аргументы.

Деструктор не вызывается, например, в тот момент, когда переменная, содержащая ссылку на объект, оказывается за пределами области действия этого объекта. Это означает, что заранее нельзя знать, когда именно следует вызывать деструктор. Кроме того, программа может завершиться до того, как произойдет "сборка мусора", а следовательно, деструктор может быть вообще не вызван.

Ключевое слово this. Когда метод вызывается, ему автоматически передается ссылка на вызывающий объект, т.е. тот объект, для которого вызывается данный метод. Эта ссылка обозначается ключевым словом this. Следовательно, ключевое слово this обозначает именно тот объект, по ссылке на который действует вызываемый метод.

Управление доступом к членам класса. Ограничение доступа к членам класса является основополагающим этапом объектно-ориентированного программирования, поскольку позволяет исключить неверное использование объекта. Для соблюдения принципов инкапсуляции правильно реализованный класс образует некий "черный ящик", которым можно пользоваться, но внутренний механизм его действия закрыт для вмешательства извне.

Управление доступом в языке C# организуется с помощью четырех модификаторов доступа: public, private, protected и internal. Для обычных классов основные модификаторы доступа public и private. Модификатор protected применяется только в тех случаях, которые связаны с наследованием.

Модификатор internal служит в основном для сборки, которая в широком смысле означает в C# разворачиваемую программу или библиотеку.

Модификатор protected internal - доступен всем из данной сборки, а также типам, производным от данного, т. е. перед нами что-то вроде объединения модификаторов доступа protected и internal.

Член класса со спецификатором public, становится доступным из любого другого кода в программе, включая и методы, определенные в других классах. Когда же член класса обозначается спецификатором private, он может быть доступен только другим членам этого класса. Модификатор private является значение по умолчанию, т.е. его указывать необязательно.

Правильная организация закрытого и открытого доступа может быть достигнута, если соблюдать ряд общих принципов:

• члены, используемые только в классе, должны быть закрытыми;

• данные экземпляра, не выходящие за определенные пределы значений, должны быть закрытыми, а при организации доступа к ним с помощью открытых методов следует выполнять проверку диапазона представления чисел;

• если изменение члена приводит к последствиям, распространяющимся за пределы области действия самого члена, т.е. оказывает влияние на другие аспекты объекта, то этот член должен быть закрытым, а доступ к нему – контролируемым;

• члены, способные нанести вред объекту, если они используются неправильно, должны быть закрытыми. Доступ к этим членам следует организовать с помощью открытых методов, исключающих неправильное их использование;

• методы, получающие и устанавливающие значения закрытых данных, должны быть открытыми;

• переменные экземпляра допускается делать открытыми лишь в том случае, если нет никаких оснований для того, чтобы они были закрытыми.

Объект обладает свойствами. В некоторых объектных языках свойства - это просто переменные, которые принадлежат классу. В C# свойства

отличаются от переменных и являются отдельной структурой данных. Пусть в классе описаны переменные и к ним невозможно получить доступ извне, если по умолчанию (отсутствуют модификаторы доступа), переменные и методы создаются закрытыми и извне недоступны.

Переменные должны оставаться закрытыми (private или protected), а вот чтобы сделать их открытыми, нужно объявить их свойства следующим образом:

```
public тип ИмяСвойства
{
get return имяПеременной;
set width = value; }
```

Свойства создаются для того, чтобы они были открытыми, поэтому объявление начинается с модификатора доступа public. После этого идет тип данных и имя.

В .NET принято именовать переменные с маленькой буквы, а соответствующие свойства - с большой. Некоторые любят начинать переменные с символа подчеркивания, это тоже нормально, главное - именовать одинаково. В отличие от объявления переменной, тут нет в конце имени точки с запятой, а открываются фигурные скобки, внутри которых нужно реализовать два аксессора (accessor): get и set.

Аксессоры позволяют указать доступ к свойству на чтение (get) и запись (set). После каждого аксессора в фигурных скобках указываются также действия свойства. Для get нужно в фигурных скобках выполнить оператор return и после него указать, какое значение должно возвращать свойство. Значение свойства находится в виртуальной переменной value, определенной внутри фигурных скобок после ключевого слова set и имеет такой же тип данных, как и свойство. Получается, что свойство просто является оберткой для переменной объекта.

А что, если у вас множество переменных, которые не нуждаются в защите, и им надо всего лишь создать обертку? Писать столь большое количество кода достаточно накладно. Тут можно поступить одним из двух

способов: прибегнуть к рефакторингу или применить сокращенный метод объявления свойств.

В общем виде доступ к свойству выглядит следующим образом:

ИмяОбъекта.Свойство

Одно важное замечание - слева от точки указывается именно имя объекта, а не класса, т. е. имя определенного экземпляра класса.

Способы передачи аргументов методу [15]. Существует два способа передачи аргументов методу. Первым способом является вызов по значению. В этом случае значение аргумента копируется в формальный параметр метода. Следовательно, изменения, вносимые в параметр метода, не оказывают никакого влияния на аргумент, используемый для вызова. А вторым способом передачи аргумента является вызов по ссылке. В данном случае параметру метода передается ссылка на аргумент, а не значение аргумента. В методе эта ссылка используется для доступа к конкретному аргументу, указываемому при вызове. Это означает, что изменения, вносимые в параметр, будут оказывать влияние на аргумент, используемый для вызова метода.

По умолчанию в C# используется вызов по значению.

При передаче методу ссылки на объект сама ссылка по-прежнему передается по значению. Следовательно, создается копия ссылки, а изменения, вносимые в параметр, не оказывают никакого влияния на аргумент. Все изменения, происходящие с объектом, на который ссылается параметр, окажут влияние на тот объект, на который ссылается аргумент. Попытаемся выяснить причины подобного влияния.

Перегрузка методов. В C# допускается совместное использование одного и того же имени двумя или более методами одного и того же класса, при условии, что их параметры объявляются по-разному. В этом случае говорят, что методы перегружаются, а сам процесс называется перегрузкой методов. Перегрузка методов относится к одному из способов реализации полиморфизма в C#.

Для правильной реализации перегрузки функций необходимо соблюдать условие: тип или число параметров у каждого метода должны быть разными.

Два метода могут также отличались типами возвращаемых значений, но этого недостаточно для перегрузки. Они должны также отличаться типами или числом своих параметров. Когда вызывается перегружаемый метод, то выполняется тот его вариант, параметры которого соответствуют (по типу и числу) передаваемым аргументам.

Перегрузка методов поддерживает свойство полиморфизма, поскольку именно таким способом в C# реализуется главный принцип полиморфизма: один интерфейс –множество методов. Для того чтобы стало понятнее, как это делается, обратимся к конкретному примеру. В языках программирования, не поддерживающих перегрузку методов, каждому методу должно быть присвоено уникальное имя. Допустим, что требуется функция, определяющая абсолютное значение. В языках, не поддерживающих перегрузку методов, обычно приходится создавать три или более вариантов такой функции с несколько отличающимися, но все же разными именами. Главная ценность перегрузки заключается в том, что она обеспечивает доступ к связанным вместе методам по общему имени.

Конструктор, как и функция, может быть перегружен. Подходящий конструктор вызывается каждый раз, исходя из аргументов, указываемых при выполнении оператора new. Перегрузка конструктора класса предоставляет пользователю этого класса дополнительные преимущества в конструировании объектов. Одна из самых распространенных причин для перегрузки конструкторов заключается в необходимости предоставить возможность одним объектам инициализировать другие.

Когда приходится работать с перегружаемыми конструкторами, то иногда очень полезно предоставить возможность одному конструктору вызывать другой. В C# это дается с помощью ключевого слова this. Ниже приведена общая форма такого вызова.

```
имя_конструктора{список_параметров1) :
    this (список_параметров2) {
/1 ... Тело конструктора, которое может быть пустым.
```

```
}
```

В исходном конструкторе сначала выполняется перегружаемый конструктор, список параметров которого соответствует критерию список_параметров2, а затем все остальные операторы, если таковые имеются в исходном конструкторе.

```
public Point(double x, double y)
{ this.x = x; this.y = y; }
public Point():this(0,0)
{    }
public Point(Point p):this(p.x, p.y)
{      }
```

Метод Main (). Главный метод имеет различные перегруженные формы, которые позволяют учитывать особенности приложения. Он может быть без параметра

```
static void Main()
static int Main()
static void Main(string[ ] args)
static int Main(string[ ] args)
```

По описанию видно, что метод Main может не возвращать (тип void) и возвращать (тип int) значение; получать (string[ ] args) и не получать аргументы. Как правило, значение, возвращаемое методом Main (), указывает на нормальное завершение программы или на аварийное ее завершение из-за сложившихся ненормальных условий выполнения. Условно нулевое возвращаемое значение обычно указывает на нормальное завершение программы, а все остальные значения обозначают тип возникшей ошибки.

Применение ключевого слова static. Чаще всего доступ к членам класса осуществляется через имя созданного объекта. Для создания члена класса, к которому можно обращаться без ссылки на конкретный экземпляр объекта, его объявления ключевое слово static. С помощью ключевого слова static можно объявлять как переменные, так и методы. Наиболее характерным примером

члена типа static служит метод Main (), который объявляется таковым потому, что он должен вызываться операционной системой в самом начале выполняемой программы.

1.3. Парадигмы ООП – наследование и полиморфизм

Наследование является одним из трех основополагающих принципов объектно-ориентированного программирования, поскольку оно допускает создание иерархических классификаций. Благодаря наследованию можно создать общий класс, в котором определяются характерные особенности, присущие множеству связанных элементов. От этого класса могут затем наследовать другие, более специализированные классы, добавляя свои индивидуальные особенности. Классическое наследование позволяет строить новые определения классов, расширяющие функциональность существующих классов.

Существующий класс, который будет служить основой для нового класса, называется базовым или родительским классом. Назначение базового класса состоит в определении всех общих данных и членов для классов, которые расширяют его. Расширяющие классы формально называются производными или дочерними классами.

Следовательно, производный класс представляет собой специализированный вариант базового класса. Он наследует все переменные, методы, свойства и индексаторы, определяемые в базовом классе, добавляя к ним свои собственные элементы.

Поддержка наследования в C# состоит в том, что в объявление одного класса разрешается вводить другой класс. Для этого при объявлении производного класса указывается базовый класс

```
class производный_класс : базовый_класс {
//  тело класса
}
```

Для любого производного класса можно указать **только один базовый** класс. В C# не предусмотрено наследование нескольких базовых классов в одном производном классе. Тем не менее можно создать иерархию наследования, в которой производный класс становится базовым для другого производного класса. Главное преимущество наследования заключается в следующем: как только будет создан базовый класс, в котором определены общие для множества объектов атрибуты, он может быть использован для создания любого числа более конкретных производных классов. А в каждом производном классе может быть точно выстроена своя собственная классификация.

Наследование класса не отменяет ограничения, накладываемые на доступ к закрытым членам класса. Поэтому если в производный класс и входят все члены его базового класса, в нем все равно оказываются недоступными те члены базового класса, которые являются закрытыми.

Для преодоления данного ограничения в C# предусмотрены разные способы. Один из них состоит в использовании защищенных (protected) членов класса, а второй - в применении открытых свойств для доступа к закрытым данным.

Ключевое слово языка C# base применяется [17]

- для вызова конструктора базового класса;

- для доступа к члену базового класса, скрывающегося за членом производного класса.

С помощью формы расширенного объявления конструктора производного класса и ключевого слова base в производном классе может быть вызван конструктор, определенный в его базовом классе. Ниже приведена общая форма этого расширенного объявления:

```
конструктор_производного_класса(список_параметров) :
        base (список_аргументов) {
// тело конструктора
```

}

где список_аргументов обозначает любые аргументы, необходимые конструктору в базовом классе.

С помощью ключевого слова base можно вызвать конструктор любой формы, определяемой в базовом классе, причем выполняться будет лишь тот конструктор, параметры которого соответствуют переданным аргументам.

Виртуальные методы и их переопределение [23]. Виртуальным называется такой метод, который объявляется как virtual в базовом классе. Виртуальный метод отличается тем, что он может быть переопределен в одном или нескольких производных классах. Следовательно, у каждого производного класса может быть свой вариант виртуального метода. Кроме того, виртуальные методы интересны тем, что именно происходит при их вызове по ссылке на базовый класс. В этом случае средствами языка C# определяется именно тот вариант виртуального метода, который следует вызывать, исходя из типа объекта, к которому происходит обращение по ссылке, причем это делается во время выполнения. Поэтому при ссылке на разные типы объектов выполняются разные варианты виртуального метода.

Метод объявляется как виртуальный в базовом классе с помощью ключевого слова virtual, указываемого перед его именем. Когда же виртуальный метод переопределяется в производном классе, то для этого используется модификаторы override или new. Кроме того, виртуальный метод не может быть объявлен как static или abstract.

Переопределение методов — это еще один способ воплотить в C# главный принцип полиморфизма: один интерфейс — множество методов.

Предотвращение наследования. В C# имеется возможность предотвратить наследование класса с помощью ключевого слова sealed.

Для того чтобы предотвратить наследование класса, достаточно указать ключевое слово sealed перед определением класса. Как и следовало ожидать, класс не допускается объявлять одновременно как abstract и sealed, поскольку

сам абстрактный класс реализован не полностью и опирается в этом отношении на свои производные классы, обеспечивающие полную реализацию.

Интерфейсы [22]. Конечно, абстрактные классы и методы приносят известную пользу, но положенный в их основу принцип может быть развит далее. В C# предусмотрено разделение интерфейса класса и его реализации с помощью ключевого слова interface.

Для реализации интерфейса в классе должны быть предоставлены тела (т.е. конкретные реализации) методов, описанных в этом интерфейсе. Каждому классу предоставляется полная свобода для определения деталей своей собственной реализации интерфейса. Следовательно, один и тот же интерфейс может быть реализован в двух классах по-разному. Интерфейсы объявляются с помощью ключевого слова interface:

```
interface имя{
возвращаемый_тип имя_метода1 (список_параметров) ;
возвраща емый_ тип имя_ме то да 2 (список_параме тров) ;
// ...
возвращаемый_тип имя_методаЫ{список_параметров) ;
}
```

где имя — это конкретное имя интерфейса. В объявлении методов интерфейса используются только их возвращаемый_тип и сигнатура. Они, по существу, являются абстрактными методами. Как пояснялось выше, в интерфейсе не может быть никакой реализации. Поэтому все методы интерфейса должны быть реализованы в каждом классе, включающем в себя этот интерфейс. В самом же интерфейсе методы неявно считаются открытыми, поэтому доступ к ним не нужно указывать явно.

Как только интерфейс будет определен, он может быть реализован в одном или нескольких классах. Для реализации интерфейса достаточно указать его имя после имени класса, аналогично базовому классу. Ниже приведена общая форма реализации интерфейса в классе.

```
class имя_класса : имя_интерфейса1,
    имя_интерфейса2, …, имя_интерфейсаN {
```

```
// тело класса
}
```

где имя_интерфейса1, …, имя_интерфейсаN — это конкретное имя реализуемого интерфейса.

1.4. Вопросы для подготовки к контролю знаний

1. С каких символов начинается однострочный комментарий в C#?

а) #

б) \\

в) //

г) &

2. В какие символы заключаются многострочные комментарии?

д) /* и *\

е) */ и /*

ж) /- и -/

з) /* и */

3. sbyte, byte, short, ushort, int, uint, long, ulong, float, double, char все это является ... ?

и) единицами компиляции

к) структурированными типами

л) элементарными типами

м) типы-делегаты

5. Модификатор неуправляемого кода?

н) Static

о) Public

п) New

р) Extern

6. Использование одного имени или идентификатора для метода внутри одной иерархии класса таким образом, чтобы для разных классов этой иерархии этот метод реализовывал различные операции это - ...?

а) Пространство имен

б) Наследование

в) Инкапсуляция

г) Полиморфизм

8. Создание новых классов, которые строятся на базе структур данных и методов уже существующих классов (базовых) - это ...?

а) Полиморфизм

б) Наследование

в) Инкапсуляция

г) Пространство имен

9. Совмещение структур данных с функциями (методами), манипулирующими этими данными это - ...?

а) Наследование

б) Полиморфизм

в) Инкапсуляция

г) Пространство имен

10. Объект, инкапсулирующий ссылку на метод (аналог указатель указателя на функцию) это - ...?

а) Делегат

б) Полиморфизм

в) Виртуальный класс

г) Абстрактный класс

11. Базовый класс, который не предполагает создания экземпляров - это ...?

а) Абстрактный класс

б) Виртуальный класс

в) Родительский класс

г) Делегат

12. Какой синтаксис используется для указания класса родителя в C#?

а) class ChildClass :: ParentClass

б) class ChildClass : ParentClass

в) class ChildClass = ParentClass

г) class ChildClass == ParentClass

13. Какие типы можно использовать в предложении foreach?

а) Методы;

б) Переменные;

в) Массивы, коллекции;

г) Модификаторы.

14. Один или несколько файлов, содержащий логический набор функциональности (код и другие данные, связанные с кодом) это - ...?

а) Коллекция

б) Подборка

в) Сборка

г) Метод

15. Кому доступны переменные с модификатором protected на уровне класса?

а) Только родительскому классу

б) Любому классу

в) Любому классу-наследнику

г) Никому

16. Что такое assembly? Что такое приватные и совместные сборки?

17. Какие простые типы данныхопределены в C#?

18. Что такое и для чего используется пространство имён?

19. Для чего предназначен оператор цикла for?

20. Что вам известно об операторе цикла while?

21. Массивы. Какие виды массивов вы знаете?

22. Спецификаторы доступа?

23. Опишите методы и способы их определения.

24. Конструкторы: виды, доступ.

25. Деструкторы и их функция.

26. Что вам известно о свойствах?

27. Перегрузка операторов и их назначение.

28. Что представляет собой событие?

29. Способы передачи параметров при вызове метода.

30. Типы передачи параметров в функцию.

1.5. Практические задания

Задание 1. Задания на основные элементы синтаксиса языка.

Методические рекомендации.

1. Изучите теоретический материал, изложенный в лекциях, в литературных источниках и информационных ресурсах по тематике индивидуального задания.

2. Разработайте алгоритм выполнения задания.

3. Запишите алгоритм на языке программирования C#.

4. Подготовьте отчет о выполнении задания.

Вариант № 1

1. Составить программу вычисления и вывода значения z

$$z=\begin{cases}\dfrac{3x+5y}{x^2+y^2}, & x^2+y^2\ge 1,\\ 7x^3+2y-x, & x^2+y^2<1,\ x\ge 0,\\ 3y^3-5y+x, & x^2+y^2<1,\ x<0;\end{cases}$$

2. Составить программу вычисления и вывода элементов массива y, которые вычисляются по формуле $y_k=\frac{1}{4}\ln\frac{1+k}{k}+\frac{1}{2}tgk\ \left(k=\overline{1,\ n}\right)$. Найти и вывести максимальный элемент массива

Вариант № 2

1. Составить программу вычисления и вывода значения z

$$y=\begin{cases}\dfrac{\sin x+2x}{a^3+3x^2}+2x, & x<-1,\\ 2ab+a^3, & |x|\le 1,\\ \dfrac{\sin x+2x}{a^3+3x^2}+1, & x>1;\end{cases}$$

2. Составить программу вычисления и вывода элементов массива y, которые вычисляются по формуле $y_i=e^{\cos i}\cos(\sin i)\ \left(i=\overline{1,3}\right)$. Найти и вывести минимальный элемент массива.

Вариант №3

1. Составить программу вычисления и вывода значения z

$$y=\begin{cases}x^2+1+\cos x+x \ \ при\ \ k=5,\ m=7,\\ x^2+1+\cos x \ \text{ при }\ k=5,\ m\ne 7,\\ x^2+1+24z \ \text{ при }\ k\ne 5;\end{cases}$$

2. Составить программу вычисления и вывода элементов массива y, которые вычисляются по формуле y, которые вычисляются по формуле $y_p=\left(1+2p^2\right)e^{p^2}\ \left(p=\overline{1,n}\right)$. Найти и вывести максимальный по модулю элемент массива.

Вариант №4

1. Составить программу вычисления и вывода значения z

$$y=\begin{cases}\dfrac{ax+b}{|ax|+5}+ax, & x<-1,\\ ax+b^2, & -1\le x\le 3.5,\\ \dfrac{ax+b}{|ax|+5}-a^2x, & x>3.5;\end{cases}$$

2. Составить программу вычисления и вывода элементов массива y, которые вычисляются по формуле $y_k=\sin(2-k)\ \left(k=\overline{1,n}\right)$. Найти и вывести максимальный элемент массива

Вариант № 5

1. Составить программу вычисления и вывода значения z

$$y=\begin{cases} x+\sin x, \quad x=1 \text{ или } x=10, \\ x\sqrt{x}, \text{ если } 1<x<9, \\ 2x^2 \text{ для остальных } x; \end{cases}$$

2. Составить программу вычисления и вывода элементов массива y, которые вычисляются по формуле $y_p = p\,\mathrm{arctg}\,p - \ln\sqrt{1+p^2}\ \left(p=\overline{1,n}\right)$. Найти и вывести максимальный элемент массива

Задание 2. Основные функциональные возможности работы со строковым типом.

Методические рекомендации.

1. Повторите теоретический материал по работе со строковым типом.

2. Разработайте приложение, которое вводит многострочный текст из файла и из всех лишних пробелов оставляет один.

3. Предоставьте отчет о выполнении задания с описание объектно-ориентированных характеристик разработанного класса.

Написать программу поиска дубликатов в массиве, используя цикл foreach.

Создать описанный ниже класс, реализовав для него принципы инкапсуляции, определив свойства и индексаторы:

а) создать класс вектор, в котором есть массив вещественных чисел произвольной размерности, перегруженные операции сложения, вычитания, умножение двух векторов, скалярное произведение двух векторов, умножение вектора на скаляр;

б) создать класс complex, в котором комплексное число представляется в полярной системе координат, то есть имеется радиус r и угол fi, перегрузить операции сложения, вычитания, умножение и деления (с проверкой) двух

комплексных чисел, создать функции перевода из полярной системы координат в декартову и наоборот;

в) создать интерфейс объемные тела, содержащий функции площадь боковой поверхности, площадь основания, объем. Реализовать классы правильная четырехугольная пирамида, конус, правильная треугольная пирамида, наследовавшие описанный интерфейс. Данные-члены расположить в закрытой секции. Переопределить функции сравнения equal для сравнения по значению, преобразования в строку toString. Используя созданный класс определить три различных множества и проверить работу всех функций для каждого из них.

Задание 3. Основные функциональные возможности работы с классами.

Методические рекомендации.

1. Повторите теоретический материал.

2. Разработать класс, который содержит динамическую информацию о наличии автобусов в автобусном парке.

Сведения о каждом автобусе содержат:

- номер автобуса;
- фамилию и инициалы водителя;
- номер маршрута.

Программа должна обеспечивать:

- начальное формирование данных в виде списка о всех автобусах;
- при выезде каждого автобуса из парка вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;
- при въезде каждого автобуса в парк вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте, и записывает эти данные в список автобусов, находящихся в парке;

- по запросу выдаются сведения об автобусах, находящихся в парке, или об автобусах, находящихся на маршруте.

3. Предоставьте отчет о выполнении задания с описание объектно-ориентированных характеристик разработанного класса.

Тема 2. Интерфейс пользователя в технологии .Net

2.1. Пользовательский интерфейс в оконных приложениях

Для создания простейшего оконного приложения понадобится минимум два класса из пространства имен System.Windows.Forms [2]:

1. Класс **Application**, предназначенный для управления приложением;
2. Класс **Form**, описывающий форму.

Существует два обязательных требования для создания оконного приложения:

1. Определить класс, порожденный от класса System.Windows.Forms.**Form**, иерархия приведена на рис. 2.1;
2. В методе Main вызвать статический метод Run класса System.Windows.Forms.**Application**, передав в качестве параметра экземпляр класса, порожденного от класса System.Windows.Forms.Form.

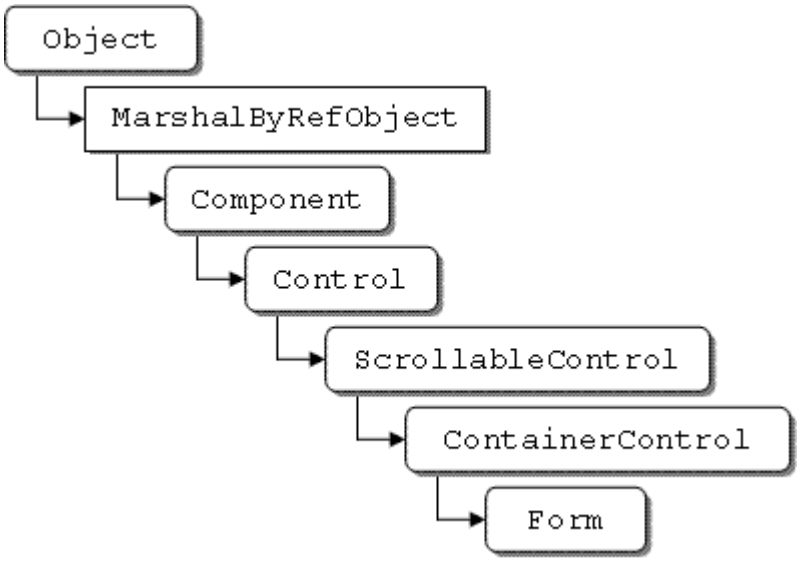

Рис. .2.1 - Иерархия классов System.Windows.Forms.Form

Событием (event) называется некоторое действие, вызванное либо действиями пользователя, например, нажатие клавиши мыши или клавиатуры, либо некоторой программой, например, завершение копирования файла. Работа с событиями в C# соответствует модели "издатель - подписчик", согласно

которой некоторый класс "публикует" (определяет) событие, которое он может инициировать, а другие классы могут "подписаться" на это событие. Класс-подписчик должен реализовать метод, который будет вызван при возникновении события. Такой метод называется обработчиком события (event handler).

Событие является свойством класса, опубликовавшего его [4]. При создании оконного приложения достаточно:

1. Подписаться на событие. Синтаксис:

имя-объекта-издателя.событие **+= new EventHandler(**имя-обработчика**);**

2. Реализовать обработчик события:

private **void** *имя-обработчика***(**object sender, EventArgs e**) { … }**

Обработчики события не должны возвращать значение и должны принимать два параметра:

1. Источник события - объект класса System.Object;
2. Объект, содержащий информацию о событии - экземпляр класса EventArgs или любого другого класса, порожденного данным.

Например, класс System.Windows.Forms.Form содержит событие Click, которое инициируется при нажатии кнопки мыши на форме. Чтобы форма реагировала на нажатие кнопки мыши на ней, нужно создать обработчик данного события, для этого необходимо в некотором методе класса формы (определенного пользователем) подписаться на событие и реализовать обработчик события.

Базовым в иерархии классов элементов управления является класс System.Windows.Forms.Control [6]. В классе Control определен вложенный класс ControlCollection. Этот класс представляет коллекцию для хранения списка элементов управления, ассоциированных с данным элементом управления.

Пользовательский интерфейс реализуется с помощью элементов, представленных на рис. 2.1.

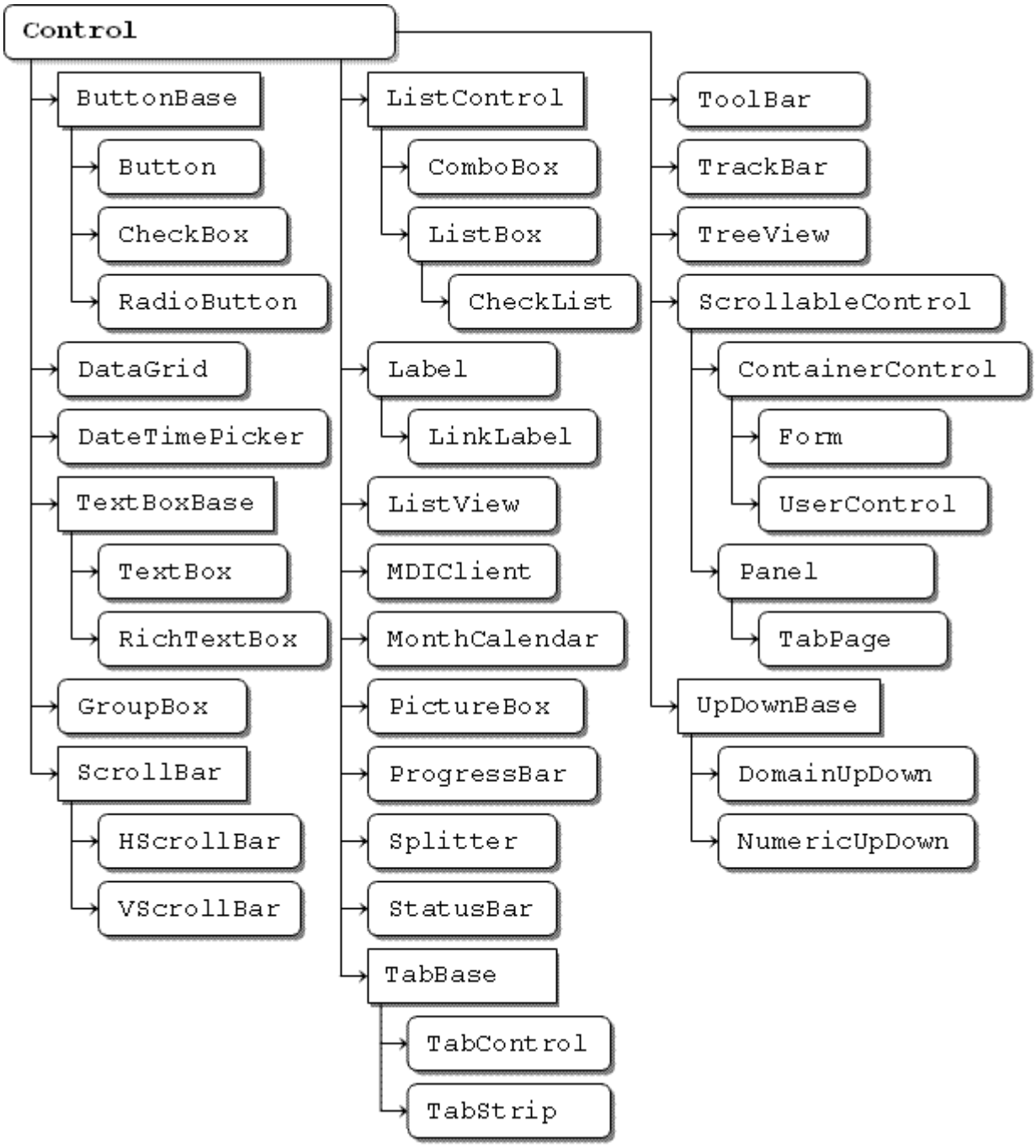

Рис. 2.1 – Диаграмма иерархии наследования элементов управления

Поскольку класс Form является потомком класса Control, то он содержит поле класса ControlCollection [21]. Итак, для того чтобы добавить элемент управления на форму, необходимо:

1. создать элемент управления;
2. добавить созданный элемент управления в коллекцию ControlCollection. Если этого не сделать, элемент управления на форме отображен не будет.

2.2. Пользовательский нтерфейс веб-приложения

Страницы ASP.NET называются веб-формами. являются важной частью приложения ASP.NET. Веб-формы позволяют создавать веб-приложения с интерфейсом, основанным на элементах управления. При запуске веб-формы ASP.NET считывает весь файл с расширением .aspx, генерирует соответствующие объекты и инициирует ряд событий, а разработчик реагирует на эти события с помощью полностью объектно-ориентированного кода.

Модель веб-форм ASP.NET [4]:

• визуализация согласно стандарту XHTML Strict. Страницы веб-форм теперь всегда будут на 100% совместимыми с XHTML (если только не нарушать правила XHTML самостоятельно);

• назначение предсказуемых клиентских идентификаторов. Чтобы каждый элемент управления получал уникальный идентификатор в визуализируемом HTML-коде, в ASP.NET используется система генерации длинных имен. Это приводит к усложнению при использовании таких идентификаторов в клиентском сценарии JavaScript;

• Постоянная переадресация. С целью улучшения поисковой оптимизации в ASP.NET теперь можно перенаправлять запросы с кодом состояния HTTP 301, который обозначает постоянную переадресацию. Когда роботы поисковых систем получают это сообщение, они обновляют свои каталоги.

Так как веб-приложения выполняются на сервере, то обработка страниц выполняется методом обратной отправки, которая посылает на сервер страницу (и всю предоставленную пользователем информацию) при выполнении определенных действий пользователем. Когда система ASP.NET получает страницу, она генерирует соответствующие серверные события для уведомления вашего кода.

Обработка событий в ASP.NET происходит следующим образом [7]:

1. при первом запуске страницы ASP.NET создает объекты этой страницы и ее элементов управления. Далее выполняется код инициализации, после чего

страница преобразуется в HTML и возвращается клиенту, а созданные объекты удаляются из памяти сервера;

2. на каком-то этапе пользователь выполняет действие, инициирующее обратную отправку данных, например, щелкает на кнопке. Тогда страница отправляется серверу вместе со всеми данными формы;

3. ASP.NET перехватывает эту возвращаемую страницу и снова воссоздает ее объекты, возвращая их в то состояние, в котором они находились тогда, когда эта страница в последний раз отправлялась клиенту;

4. далее ASP.NET проверяет, какая именно операция привела к обратной отправке данных, и генерирует соответствующие события (например, Button .Click), на которые разработчик может предусмотреть в своем коде определенную реакцию. На этом этапе чаще всего выполняются серверные операции (вроде обновления базы данных или чтения данных из файла), а затем изменять объекты элементов управления так, чтобы они отображали уже новую информацию;

5. измененная страница преобразуется в HTML и возвращается клиенту. Объекты страницы удаляются из памяти. В случае если происходит еще одна обратная отправка данных, ASP.NET повторяет действия, перечисленные в пунктах 2-4.

В Visual Studio поддерживаются два пути создания веб-приложений на базе ASP.NET [19, 20]:

• разработка на основе проекта. При создании веб-проекта в Visual Studio генерируется файл проекта с расширением .csproj (при условии, что код пишется на языке C#), в котором фиксируется информация обо всех включаемых в состав проекта файлах и сохраняются кое-какие отладочные параметры. При запуске веб-проекта перед открытием веб-браузера Visual Studio сначала компилирует весь написанный код в одну сборку;

• разработка без использования проекта. Это альтернативный подход, который подразумевает создание просто веб-сайта безо всякого файла проекта. При таком подходе Visual Studio предполагает, что каждый файл в каталоге веб-сайта (и всех его подкаталогах) является частью веб-приложения. В этом

случае Visual Studio не требуется предварительно компилировать код. Вместо этого ASP.NET компилирует уже сам веб-сайт при первом запросе какой-нибудь входящей в его состав страницы.

Место размещения отвечает за то, где будут храниться файлы веб-сайта. В левой части этого окна есть четыре кнопки, позволяющие выбирать различные варианты для размещения файлов:

• File System (Файловая система). Это самый простой вариант, поскольку он подразумевает просто просмотр дерева дисков и каталогов или общих ресурсов, отображаемых другими компьютерами в сети, и выбор походящего каталога. При желании создать новый каталог нужно всего лишь щелкнуть на значке Create New Folder (Создать новую папку), который отображается в правом верхнем углу дерева каталогов. (Заставить Visual Studio создать новый каталог также можно и путем добавления имени нового каталога в конце пути.);

• Local IIS (Локальный IIS). Этот вариант позволяет просматривать виртуальные каталоги, которые делает доступными предоставляющее веб-хостинг программное обеспечение IIS, при условии, конечно, что таковое установлено на данном компьютере;

• FTP Site (FTP-сайт). Этот вариант является не таким удобным, как поиск нужного каталога, поскольку предполагает ввод всей информации, которая необходима для установки соединения: имени FTP-сайта, номера порта, названия каталога, имени пользователя и пароля;

• Remote Site (Удаленный веб-сайт). Этот вариант позволяет получать доступ к определенному веб-сайту с определенным URL-адресом по протоколу HTTP. Чтобы он работал, на веб-сервере, к которому требуется получить доступ, должен быть установлен компонент FrontPage Extensions. Вдобавок, при подключении потребуется ввести имя пользователя и пароль.

Visual Studio предоставляет три режима для просмотра веб-страницы: Source (Исходный код), Design (Конструктор) и Split (Комбинированный). Выбрать желаемый режим можно, просто щелкнув на одной из трех имеющих соответствующие названия кнопок в нижней части окна с веб-страницей. В

режиме Source отображается разметка страницы (т.е. дескрипторы элементов управления HTML и ASP.NET), в режиме Design — отформатированное изображение того, как страница будет выглядеть в окне веб-браузера, а в режиме Split — комбинированное представление, позволяющее просматривать одновременно и разметку страницы, и ее окончательный внешний вид.

Добавить элемент управления ASRNET на страницу легче всего, перетащив его из находящейся слева панели Toolbox (Панель инструментов) [11].

Перетащить элемент управления можно как в область видимой структуры страницы (в режиме визуального конструктора), так и в определенную позицию в ее разметке (в режиме исходного кода) [16]. И в том и в другом случае результат будет одинаковым. Существует также и альтернативный вариант: вручную ввести дескриптор требуемого элемента управления в режиме Source. В таком случае представление, отображаемое в режиме Design, не обновляется до тех пор, пока не будет либо выполнен щелчок в области проектирования окна, либо нажата комбинация клавиш <Ctrl+S> для сохранения веб-страницы.

2.5. Вопросы для подготовки к контролю знаний

1. Какой тег html соответствует серверному элементу управления <asp:Label>?

а) <Span>

б) <Input Type="Text">

в) <img>

г) <Table>

2. Какой тег html соответствует серверному элементу управления <asp:ListBox>?

а) <Select>

б) <Input Type=”Text”>

в) <Span>

г) <img>

3. Какой тег html соответствует серверному элементу управления <asp:DropDownList>?

а) <Select>

б) <Input Type=”Text”>

в) <Span>

г) <img>

4. Какой тег html соответствует серверному элементу управления <asp:TextBox>?

а) <input Type=”Text”>

б) <Select>

в) <Input Type="Hidden">

г) <Input Type="Radio">

5. Какой тег html соответствует серверному элементу управления <asp:TextBox>?

а) <input Type="Password”>

б) <Select>

в) <Input Type="Hidden">

г) <Input Type="Radio">

6. Какой тег html соответствует серверному элементу управления <asp:TextBox>?

а) <Textarea>

б) <Select>

в) <Input Type="Hidden">

г) <Input Type="Radio">

7. Какой тег html соответствует серверному элементу управления <asp:HiddenField>?

а) <Input Type="Hidden">

б) <Input Type="Radio">

в) <input Type=”Text”>

г) <Span>

8. Какой тег html соответствует серверному элементу управления <asp:RadioButton>?

а) <Input Type="Radio">

б) <Input Type="Hidden">

в) <Span>

г) <Select>

9. Какой тег html соответствует серверному элементу управления <asp:RadioButtonList>?

а) <Input Type="Radio">

б) <Input Type="CheckBox">

в) <Input Type="button">

г) <Select>

10. Какой тег html соответствует серверному элементу управления <asp:CheckBox>?

а) <Input Type="CheckBox">

б) <Select>

в) <Input Type="Radio">

г) <Span>

11. Какой тег html соответствует серверному элементу управления <asp:CheckBoxList>?

а) <Input Type="CheckBox">

б) <Select>

в) <Input Type="Radio">

г) <Span>

12. Какой тег html соответствует серверному элементу управления <asp:Button>?

а) <Input Type="button">

б) <Input Type="Radio">

в) <Select>

г) <Input Type="CheckBox">

13. Какой тег html соответствует серверному элементу управления <asp:Image>?

а) <img>

б) <picture>

в) <Input Type="Radio">

г) <a>

14. Какой тег html соответствует серверному элементу управления <asp:ImageButton>?

а) <Input Type="image">

б) <img>

в) <Input Type="button">

г) <Input Type="Radio">

15. Какой тег html соответствует серверному элементу управления <asp:Table>?

а) <Table>

б) <img>

в) <Input Type="button">

г) <Select>

16. Какой тег html соответствует серверному элементу управления <asp:Panel>?

а) <Div>

б) <Table>

в) <img>

г) <penel>

17. Какой тег html соответствует серверному элементу управления <asp:BulletedList>?

а)

б) <list>

в) <Table>

г) <img>

18. Какой тег html соответствует серверному элементу управления <asp:BulletedList>?

а)

б) <list>

в) <Table>

г) <img>

19. Какой тег html соответствует серверному элементу управления <asp:HyperLink>?

а) <A Href>

б)

в) <list>

г) <Table>

20. Какой серверный элемент управления соответствует html тегам <Input Type="Text"> <Input Type="Password"> <Textarea>?

а) <asp:TextBox>

б) <asp:ListBox>

в) <asp:Label>

г) <asp:Table>

21. Какой серверный элемент управления соответствует html тегам <Input Type="button"> <Input Type="submit">?

а) <asp:Button>

б) <asp:TextBox>

в) <asp:ListBox>

г) <asp:Label>

22. Какой серверный элемент управления соответствует html тегу <Span>?

а) <asp:Label>

б) <asp:Button>

в) <asp:TextBox>

г) <asp:ListBox>

23. Какой серверный элемент управления соответствует html тегу <img>?

а) <asp:Image>

б) <asp:Table>

в) <asp:Panel>

г) <asp:Picture>

24. Какой серверный элемент управления соответствует html тегу <Table>?

а) <asp:Table>

б) <asp:Panel>

в) <asp:Picture>

г) <asp:Button>

25. Какой серверный элемент управления соответствует тегу <Div>?

а) <asp:Panel>

б) <asp:Table>

в) <asp:Picture>

г) <asp:Button>

26. Определите тип атаки, подходящий под данное описание: атакующий подставляет в поле формы или URL сценарий, используемый приложением при формировании новой страницы:

а) межсайтовый скриптинг

б) атака на отказ

в) подслушивание

г) атака одним щелчком

27. Определите тип атаки, подходящий под данное описание: сервер атакуется ложными запросами, в результате чего система перегружается и настоящий трафик блокируется:

а) а) атака на отказ

б) межсайтовый скриптинг

в)подслушивание

г)атака одним щелчком

28. Определите тип атаки, подходящий под данное описание: атакующий использует специальное ПО для перехвата и чтения незашифрованных пакетов, передаваемых по сети:

а) подслушивание

б) атака на отказ

в) межсайтовый скриптинг

г) подмена значений скрытого поля

29. Определите тип атаки, подходящий под данное описание: атакующий подменяет значение скрытых полей, содержащих критичные для приложения данные:

а) подмена значений скрытого поля

б) подслушивание

в) атака на отказ

г) межсайтовый скриптинг

30. Определите тип атаки, подходящий под данное описание: серверу направляются опасные POST-запросы HTTP, выдаваемые за запросы от его собственных страниц:

а) атака одним щелчком

б) подслушивание

в) атака на отказ

г) межсайтовый скриптинг

31. Определите тип атаки, подходящий под данное описание: атакующий вычисляет или похищает идентификатор сеанса и подключается к серверу, используя сеанс другого пользователя

а) похищение сеанса

б) подслушивание

в) атака на отказ

г) межсайтовый скриптинг

32. Определите тип атаки, подходящий под данное описание: в поле формы, которое предназначено для ввода значений, включаемых в формируемые SQL- команды путем конкатенации, атакующий вводит

злонамеренный SQL код, в результате чего команды изменяется и выполняет нужные атакующему действия

а) внедрение SQL кода

б) подслушивание

в) атака на отказ

г) межсайтовый скриптинг

33. Назовите способ аутентификации, при котором запрос возвращается браузеру с определенным кодом состояния ATL:

а) Basic

б) Digest

в) Integrated Window

г) Anonymous

34. Какая из аутентификаций заключается в обмене информации между браузером и Web-сервером:

а) Integrated Window

б) Basic

в) Digest

г) Anonymous

35. Назовите уровень доверия при котором : приложения работают с полным доверием и могут выполнять любой код в контексте того процесса, в котором они работают:

а) Full

б) Hight

в) Medium

г) Low

36. . Назовите уровень доверия, при котором приложения наделяются большинством поддерживаемых разрешений. Такой режим подходит для приложений, которым необходимо достаточно широкие полномочия, в условиях, когда все же требуется снизить возможные риски

а) Hight

б) Full

в) Medium

г) Low

37. Назовите уровень доверия, при котором приложение может выполнять чтение и запись элементов своего каталога и взаимодействовать с базами данных:

а) Medium

б) Hight

в) Full

г) Low

38. Назовите уровень доверия, при котором приложение позволяет считывать его собственные ресурсы, но запрещено обращаться к ресурсам, расположенным вне его пространства:

а) Low

б) Medium

в) Hight

г) Full

39. Назовите уровень доверия, при котором приложение не может взаимодействовать с защищенными ресурсами. Данная установка подходит для непрофессиональных сайтов, которым достаточно поддержки обычного HTML и очень изолированной бизнес логики:

а) Minimal

б) Medium

в) Hight

г) Full

40. При помощи какого объекта можно обладать всей информацией о пользователе и можно программно изменять его пароль и сведения:

а) MembershipUser

б) ChangePasword

в) ResetPassword

г) Pasword

41. Какому из методов передается старый пароль:

а) ChangePasword

б) MembershipUser

в) ResetPassword

г) Pasword

42. Какой метод позволяет удалить старый и автоматически сгенерировать новый пароль:

а) ResetPassword

б) ChangePasword

в) MembershipUser

г) Pasword

43. События элемента управления Login, пользователь аутентифицирован:

а) Authencicate

б) LoggedIn

в) LogingIn

г) LogingOut

44. События элемента управления Login, пользователь успешно зашел на сайт после аутентификации:

а) LoggedIn

б) Authencicate

в) LogingIn

г) LogingOut

45. События элемента управления Login, после ввода пользователем учетных данных, но до аутентификации

а) LogingIn

б) LoggedIn

в) Authencicate

г) LogingOut

46. Какое из свойств класса Membership возвращает максимальное количество попыток ввода пароля перед блокированием пользователя

а) MaxinvalidName

б) ApplicationName

в) PasswordAttemptWindow

г) Provider

47. Какое из свойств класса Membership возвращает минимальную длину пароля:

а) MinRequiredpasswordLenght

б) ApplicationName

в) PasswordAttemptWindow

г) Provider

48. Какой из методов класса Membership генерирует случайный пароль, заданной длины:

а) GenerationPassword

б) GetAllUsers

в) DeletUsers

г) CreateUsers

49. Какое из свойств MambershipProvider указывает, поддерживает ли провайдер восстановление пароля:

а) EnablepasswordRettrieval

б) EnablePasswordReset

в) PasswordAttemptWindow

г) ApplicationName

50. Свойство Application:

а) содержит ссылку на объект типа HttpApplicationState, который ассоциируется с текущем приложением, в котором выполняется страница;

б) свойство содержит ссылку на объект типа HttpServerUtility;

в) это свойство содержит ссылку на объект типа HttpSessionState, который ассоциируется с текущим приложением, в котором выполняется страница;

г) позволяет получить объект типа Cache, принадлежащий приложению, в котором выполняется страница;

51. Используя какое свойство, можно получить доступ ко всей коллекции элементов управления, расположенных на странице?

а) Controls

б) Session

в) Server

г) Cache

52. Какое свойство возвращает ссылку на коллекцию элементов управления, проверяющих ввод пользователя?

а) Validators

б) Controls

в) Application

г) Cache

53. Какое свойство позволяет установить страницу, которая должна возникать в случае появления необрабатываемого исключения в вашем приложении.?

а) ErrorPage

б) IsValid

в) IsPostBack

г) ClientScript

54. Свойство Header:

а) содержит ссылку на объект HtmlHead, который ассоциируется с блоком <head> в HTML.

б) позволяет хранить ссылку на специальный объект ClientScriptManager. Этот объект позволяет объединять все методы, которые используются на клиенте;

в) содержит ссылку на экземпляр мастер-страницы, которая является шаблоном для данной страницы;

г) содержит ссылку на объект типа HttpSessionState, который ассоциируется с текущим приложением, в котором выполняется страница;

55. Свойство EnableTheming:

а) позволяет установить, следует ли использовать схемы при формировании данной страницы;

б) позволяет хранить ссылку на элемент управления Pager, с помощью которого можно разбить содержимое формы на несколько страниц;

в) содержит ссылку на объект типа SiteCounters, который позволяет учитывать посещения текущей страницы пользователями;

г) содержит ссылку на объект типа HttpSessionState, который ассоциируется с текущим приложением, в котором выполняется страница;

56. Какое свойство содержит имя мастер страницы, которая является шаблоном для текущей страницы?

а) Master PageFile

б) Pager

в) Master

г) ClientScript

57. Свойство PreviousPage:

а) возвращает объект, который ассоциируется со страницей при перекрестном переходе от одной странице к другой

б) позволяет определить, прошла ли успешно проверка страницы на корректность данных

в) используется при динамическом создании элементов управления

г) содержит ссылку на объект типа HttpSessionState, который ассоциируется с текущим приложением, в котором выполняется страница;

58. Что из ниже перечисленного не является свойством класса Page?

а) GetValidators

б) Title

в) IsPostBack

г) SetFocus

59. Метод MapPath:

а) этот метод в качестве параметров принимает относительный путь в виртуальной директории и возвращает полный абсолютный путь на диске сервера

б) используется для инициирования серверной проверки введенных данных на корректность. Используется совместно со свойством IsValid

в) этот метод позволяет установить фокус на элемент формы! В качестве параметров он может принимать как ссылку на элемент управления, так и его ID.

г) не существует такого метода

60. На каком этапе происходит загрузка персональных данных пользователя и схем?

а) предварительная инициализация

б) инициализация страницы

в) завершение инициализации

г) загрузка состояния элементов

61. На каком этапе страница инициирует событие Prelnit.?

а) предварительная инициализация

б) инициализация страницы

в) завершение инициализации

г) загрузка состояния элементов

62. На каком этапе страница инициирует сообщение Init?

а) инициализация страницы

б) предварительная инициализация

в) завершение инициализации

г) загрузка состояния элементов

63. На каком этапе происходит инициализация состояния страницы и создание всех элементов управления на странице?

а) предварительная инициализация

б) инициализация страницы

в) завершение инициализации

г) загрузка состояния элементов

64. На каком этапе происходит обработка данных, которые были переданы в запросе?

а) загрузка переданных данных

б) загрузка состояния элементов

в) предварительная загрузка

г) предварительная инициализация

65. Какому этапу соответствует событие PreLoad?

а) предварительная загрузка

б) загрузка переданных данных

в) загрузка состояния элементов

г) предварительная инициализация

66. Какому этапу соответствует событие InitCompIete?

а) завершение инициализации

б) загрузка переданных данных

в) загрузка состояния элементов

г) предварительная инициализация

67. На каком этапе происходит событие Load?

а) загрузка страницы

б) предварительная загрузка

в) загрузка переданных данных

г) предварительная инициализация

68. На каком этапе происходит вызов всех событий, соответствующих изменениям, которые проводились со страницей на клиенте?

а) вызов событий

б) загрузка страницы

в) завершение загрузки страницы

г) начало загрузки страницы

69. На каком этапе происходит событие LoadCompIete?

а) завершение загрузки

б) завершение инициализации

в) предварительная загрузка

г) загрузка переданных данных

70. Какому шагу соответствуют сразу 3 события?

а) предварительное представление

б) вызов событий

в) завершение загрузки

г) загрузка переданных данных

71. Какое из этих событий не соответствует этапу предварительного представления?

а) CompletePreLoad

б) PreRenderComplete

в) PreRender

г) Load

72. Какое из этих событий соответствует этапу предварительного представления?

а) SaveStateComplete

б) IsPostBack

в) PostBackUrl.

г) Load

73. На каком этапе происходит освобождение ресурсов, занятых страницей?

а) выгрузка страницы

б) представление

в) предварительное представление

г) инициализация страницы

74. Какому этапу соответствует событие Unload?

а) выгрузка страницы

б) представление

в) предварительное представление

г) загрузка переданных данных

75. Какой процесс в версии IIS 5.0 выполнял функции главного процесса Web-сервера:

а) Inetinfo.exe

б) Inet.exe

в) Webproc.exe

г) DLLHost.exe.

76. IIS 6.0 состоит из двух новых компонентов, выполняющих функции главного процесса Web-сервера:

а) HTTP.sys и WWW Service Administration and Monitoring

б) HTTP.sys и Inetinfo.exe

в) HTTP.sys и DLLHost.exe

г) WWW Service Administration and Monitoring и Inetinfo.exe.

77. Две новые концепции IIS 6.0 представляют собой:

а) пулы приложений (application pools) и рабочие процессы (worker processes)

б) application pools и HTTP.sys

в) HTTP.sys и worker processes

г) worker processes и Inetinfo.exe.

78. Каждый пул приложения (application pool) соответствует:

а) только одной очереди запросов в HTTP.sys и одному или более Windows-процессам, обрабатывающим эти запросы

б) одной или более очереди запросов в HTTP.sys и одному или более Windows-процессам, обрабатывающим эти запросы

в) одной или более очереди запросов в HTTP.sys и только одному Windows-процессу, обрабатывающему эти запросы

г) пул приложения вообще не соответствует ни очереди запросов в HTTP.sys, ни Windows-процессам.

79. IIS 6.0 поддерживает до:

а) 2 000 пулов приложений на сервер, и единовременно могут работать несколько таких пулов

б) 10 000 пулов приложений на сервер, и единовременно могут работать несколько таких пулов

в) 2 000 пулов приложений на сервер, и единовременно может работать только один подобный пул

г) один пул приложения на сервер, и единовременно могут работать до 2 000 таких пулов.

80. Вся обработка Web-приложений, в том числе загрузка ISAPI-фильтров и расширений, а также аутентификация и авторизация, выполняется новой DLL сервиса WWW, загружаемой в один или несколько рабочих хост-процессов, при этом исполняемый файл рабочего процесса называется:

а) W3wp.exe

б) Inet.exe

в) Inetinfo.exe

г) DLLHost.exe.

81. Компонент WWW Service Administration и Monitoring:

а) не исполняет код приложения

б) исполняет код приложения

в) напрямую зависит от кода и исполняет код приложения

г) напрямую зависит от кода и не исполняет код приложения.

82. Компонент HTTP.sys:

а) не исполняет код приложения

б) исполняет код приложения

в) напрямую зависит от кода и исполняет код приложения

г) напрямую зависит от кода и не исполняет код приложения.

83. Компонент WWW Service Administration и Monitoring выполняет две основные обязанности::

а) конфигурирование системы и управление рабочим процессом

б) конфигурирование системы и управление HTTP.sys

в) конфигурирование системы и управление worker processes

г) управление рабочим процессом и Inetinfo.exe.

84. Режим изоляции рабочего процесса в IIS 6.0 не имеет следующей особенности:

а) единственность пулов приложений

б) кэширование в режиме ядра

в) четкое разделение между пользовательским кодом и сервером

г) web-сады (Web gardens).

85. Основными преимуществами архитектуры обработки запросов в IIS 6.0 не являются:

а) большое количество перезапусков сервера

б) повышенная доступность приложения

в) улучшенная масштабируемость

г) мощная поддержка платформы приложений.

86. Назовите три основных усовершенствования в SSL (secure sockets layer) IIS:

а) Производительность, Объект сертификата, поддерживающий удаленное взаимодействие, Возможность выбора CSP

б) Производительность, Объект сертификата, поддерживающий удаленное взаимодействие, Улучшенная масштабируемость

в) Производительность, Улучшенная масштабируемость, Возможность выбора CSP

г) Улучшенная масштабируемость, Объект сертификата, поддерживающий удаленное взаимодействие, Возможность выбора CSP.

87. В Windows Server механизмом аутентификации для IIS служит интегрированый:

а) .NET Passport

б) .NET Password

в) .NET Manager

г) .NET Authorization.

88. Иерархическое хранилище параметров конфигурации (метабаза), используемых IIS, не поддерживает:

а) удаленное администрирование

б) типизацию данных

в) уведомление об изменениях и защиту

г) наследование.

89. Указать неверный вариант ответа. Новая XML-метабаза IIS 6.0 улучшает управляемость сервера, позволяя:

а) выполнять зависимое от сервера резервное копирование и восстановление

б) напрямую и безопасно редактировать конфигурационную метабазу

в) клонировать конфигурации Web-сайта и приложений

г) использовать эффективные утилиты командной строки, системы контроля версий и средства редактирования.

90. При записи метабазы на диск IIS:

а) добавляет к новому файлу MetaBase.xml номер версии и сохраняет его копию в архивной папке

б) перезаписывает файл MetaBase.xml, дописывая при этом дату изменения

в) перезаписывает файл MetaBase.xml, дописывая при этом дату изменения и сохраняет его копию в архивной папке

г) удаляет старый файл MetaBase.xml из архивной папки и помещает в нее новый файл

91. Указать неверный вариант ответа. В IIS предлагает два новых метода Admin Base Object (ABO): Export() и Import() позволяют:

а) при использовании любого из методов автоматически объединять текущую конфигурацию с импортируемой

б) экспортировать один узел или все дерево в XML с любого уровня метабазы

в) импортировать один узел или все дерево из XML-файла

г) защищать секретные данные паролем.

92. Какой из следующих сценариев выводит идентификаторы процессов и пулов приложений для выполняемых в данный момент рабочих процессов:

а) IISapp.vbs

б) IISconfg.vbs

в) IISback.vbs

г) IISext.vbs.

93. Какой из следующих сценариев экспортирует и импортирует конфигурацию IIS в XML-файл:

а) IISconfg.vbs

б) IISapp.vbs

в) IISback.vbs

г) IISext.vbs.

94. Web-сад - это пул приложений, содержащий несколько процессов, которые:

а) обслуживают запросы к этому пулу

б) обслуживают запросы к множеству пулов (в том числе и к этому пулу приложения)

в) не обслуживают запросы к этому пулу

г) обслуживают запросы к множеству пулов (исключая данный пул приложения).

95. При рабочих нагрузках, требующих операций с большим объемом кэшированных данных, IIS на системах с процессорами типа x86 можно настроить на поддержку кэша размером до:

а) 64 Гб

б) 64 Тб

в) 64 Мб

г) 64 Кб.

96. Серверная вспомогательная функция HSE_REQ_EXEC_URL позволяет ISAPI-расширениям:

а) перенаправлять запросы по другому URL

б) создавать запросы

в) принимать запросы от другого URL

г) выявлять реккурсивные запросы.

97. Указать неверный вариант ответа. В версиях IIS ниже 6.0 принимать все запросы к конкретному URL можно было только в ISAPI-фильтрах, но с ними связаны следующие проблемы:

а) ISAPI-фильтры локальны для используемого сервера

б) они не могут выполнять длительные операции

в) ISAPI-фильтры глобальны для всего Web-сайта

г) они не могут обращаться к телу запроса, если не являются фильтрами чтения необработанных данных.

98. Серверная вспомогательная функция HSE_REQ_REPORT_ UNHEALTHY позволяет ISAPI-расширению:

а) вызвать рабочий процесс IIS 6.0 и потребовать его повторного запуска

б) потребовать повторного запуска всех серверных рабочих процессов

в) прервать рабочий процесс (с возможностью продолжения работы процесса)

г) прервать рабочий процесс (без возможности продолжения работы процесса).

99. В IIS 5.0 за сжатие информации отвечал ISAPI-фильтр, и его можно было включить:

а) только для всего сервера

б) отдельно для каждого сайта

в) отдельно для документов с определенным расширением

г) только для определенной группы сайтов.

100. Для получения объекта DataSet, который заполнен результатами запроса к базе данных, необходимо выполнить запрос, обработать результаты, заполнить контейнер и закрыть подключение. Эти операции реализованы в объекте ADO.NET:

а) адаптере данных

б) провайдере данных

в) классе DataBinder

г) контейнере данных

101. В ADO.NET объект адаптера данных служит как:

а) двунаправленный мост между источником данных и объектом DataSet, предназначенный для заполнения этого объекта и передачи данных обратно в источник данных

б) однонаправленный канал, используемый только для чтения данных из их источника в таблицу

в) однонаправленный канал, используемый только для записи данных из таблицы в источник данных

г) обработчик информации, полученной как из источника данных, так и из объекта DataSet

102. Какой метод класса адаптера данных заполняет таблицу в памяти строками, прочитанными из источника данных?

а) Fill

б) FillSchema

в) FillLoadOption

г) InsertCommand

103. Какой метод класса адаптера данных возвращает параметры, заданные пользователем в строке запроса?

а) GetFillParameters

б) ReturnProviderSpecificTypes

в) TableMappings

г) FillSchema

104. При пакетном обновлении используются команды, представляющие операции:

а) вставки, обновления, удаления

б) вставки, обновления, удаления, переименования

в) вставки, удаления, выборки

г) только обновления

105. Метод Fill заполняет объекты DataSet или DataTable данными, прочитанными в результате выполнения команды:

а) SelectCommand

б) InsertCommand

в) UpdateCommand

г) FillLoadOption

106. Объект DataSet может содержать:

а) несколько таблиц, определения отношений между ними и определения ограничений

б) только таблицы DataTable

в) таблицы и отношения между ними

г) только ограничения для данных в DataTable

107. Имя каждого создаваемого объекта DataSet может

а) задаваться по умолчанию и его можно изменять

б) задаваться по умолчанию и его нельзя изменять

в) задаваться только программистом

г) задаваться только программистом без дальнейшего изменения

108. Какое действие будет выполнено, если созданная вами таблица уже имеется в составе объекта DataSet?

а) она обновляется

б) она игнорируется

в) она создается в другом объекте

г) на добавляется к исходной

109. Когда клиентское приложение добавляет в таблицу объекта DataSet новую строку, исходным значением всех ее ячеек является:

а) null

б) 0

в) 0.0

г) --

110. Что можно указать явно, задав значение свойства FillLoadOption?

а) как уже имеющиеся в этом объекте строки будут соединены со строками, загружаемыми из базы данных

б) должны ли фиксироваться операции вставки строк, осуществляемые при выполнении операции заполнения

в) должны ли в таблицу, при создании которой осуществляется операция заполнения, помещаться результирующие наборы данных

г) должны ли фиксироваться изменения, внесенные в ходе выполнения операции пакетного обновления

111. OverwriteChanges – значение свойства FillLoadOption – задается в том случае, если:

а) нужно инициализировать таблицы свежими данными

б) необходима синхронизация текущих данных объекта DataSet с данными из базы данных

в) текущие данные заменяются, а исходные остаются без изменения

г) исходные и текущие данные остаются без изменения

112. При каком значении свойства FillLoadOption исходная версия строки объекта Datatable обновляется значениями из загруженной строки данных?

а) PreserveChanges

б) Upsert

в) OverwriteChanges

г) FillSchema

113. Какое действие выполняет адаптер, если при сопоставлении таблиц соответствие найдено?

а) адаптер считывает заданное в соответствии имя объекта DataTable, а потом ищет этот объект среди таблиц объекта DataSet

б) адаптер возвращает коллекцию соответствий между таблицами источника данных и таблицами в памяти

в) адаптер присваивает отдельное имя каждому набору данных, полученному в результате выполнения запроса на сопоставление

г) адаптер определяет пару имен: исходного результирующего набора и таблицы в памяти

114. Кем должно определяться соответствие между таблицей, создаваемой для каждого набора значений, и результирующим набором данных?

а) программистом

б) пользователем

в) автоматически

г) адаптером данных

115. Если метод Fill не находит таблицу или столбец в списке соответствий, он может генерировать исключения, которые:

а) можно разрешать декларативно, выбрав одно из допустимых действий по их обработке

б) нельзя разрешать

в) можно разрешать декларативно, используя свойство ContinueFillOnError

г) разрешаются без дополнительно проводимых действий

116. Когда адаптер данных генерирует исключение – отсутствие элемента схемы, действие по его обработке задается в свойстве:

а) MissingSchemaAction

б) MissingMappingAction

в) TableMappingsAction

г) MissingColumnAction

117. Какое значение по умолчанию определено для свойства MissingMappingAction объекта адаптера?

а) Passthrough

б) Error

в) Ignore

г) Add

118. Какое из перечисленных значений не присуще свойству MissingSchemaAction объекта адаптера?

а) Upsert

б) Add

в) AddWithKey

г) Ignore

119. При каком значении свойства MissingSchemaAction объекта адаптера в объект DataSet добавляется недостающий элемент – DataTable или DataColumn?

а) Add

б) AddWithKey

в) Ignore

г) AddWithError

120. Какой параметр определяет, следует ли при формировании схемы объектов DataTable учитывать соответствия, заданные в коллекциях TableMappings и ColumnMappimgs адаптера данных?

а) MappingMode

б) Mapped

в) MappingTable

г) MappingAction

121. Какое свойство указывает, должны ли фиксироваться измененные в ходе выполнения операции пакетного обновления?

а) AcceptChangesDuringUpdate

б) AcceptChangesDuringFill

в) AcceptChangesDuringDelete

г) AcceptCommandDuringUpdate

122. Какое свойство можно установить во избежание остановки процесса пакетного обновления при генерации исключения во время ошибочного обновления строки?

а) ContinueUpdateOnError

б) IgnoreOnError

в) GetFillOnError

г) AcceptChangesDuringError

123. Какое свойство определяет размер блока записей, передаваемого за один раз при пакетном обновлении?

а) UpdateBatchSize

б) UpdateSpecificSize

в) returnBatchSize

г) returnActionSize

124. Как называются объекты, при помощи которых команды обновления данных в источнике могут генерироваться автоматически и передаваться непосредственно провайдеру данных?

а) построители команд

б) обработчик команд

в) генератор команд

г) управляемый объект

125. Специализированная зарезервированная папка Bin содержит:

а) все предкомпилированные сборки, необходимые для работы приложения;

б) файлы исходного кода классов, которые будут использоваться страницами;

в) файлы данных приложения;

г) глобальные для приложения файлы ресурсов.

126. Специализированная зарезервированная папка App_Browsers содержит:

а) файлы с информацией о возможностях браузера;

б) глобальные для приложения файлы ресурсов;

в) все предкомпилированные сборки, необходимые для корректной работы браузера;

г) файлы ресурсов для отдельных страниц.

127. Специализированная зарезервированная папка App_Data содержит:

а) файлы данных приложения;

б) файлы ресурсов для отдельных страниц;

в) файлы исходного кода классов, которые будут использоваться страницами;

г) файлы с информацией о возможностях браузера.

128. Специализированная зарезервированная папка App_GlobalResources содержит:

а) глобальные для приложения файлы ресурсов;

б) глобальные ресурсы для файлов данных;

в) глобальные файлы ресурсов для отдельных страниц;

г) файлы ресурсов для глобальных страниц.

129. Специализированная зарезервированная папка App_LocalResources содержит:

а) файлы ресурсов для отдельных страниц;

б) файлы ресурсов для глобальных страниц;

в) глобальные для приложения файлы ресурсов;

г) ресурсы, необходимые для работы всего приложения.

130. Специализированная зарезервированная папка App_Themes содержит:

а) определения поддерживаемых приложением тем;

б) список всех тем;

в) темы, созданные программистом и помещенные в данную папку для временного хранения;

г) стандартные определения тем.

131. Специализированная зарезервированная папка App_WebReferences содержит:

а) файлы, необходимые для связывания web-сервисов с приложением;

б) web-ссылки приложения;

в) необходимые настройки для web-файлов;

г) список стандартных web-сервисов.

132. Имена специализированных зарезервированных папок:

а) изменять нельзя

б) изменять можно;

в) присваиваются программистом один раз без дальнейшего их изменения;

г) изменять можно, имея доступ администратора.

133. Если web-страница является локализуемой, то во время формирования пользовательского интерфейса в ней используются:

а) строки, которые хранятся отдельно в виде ресурсов;

б) строки жестко закодированного текста;

в) троки, имеющие web-ссылки;

г) се строки, написанные в приложении.

134. Какого раздела не имеет файл страницы ASP.NET:

а) нет правильного ответа;

б) директива страницы;

в) программный код;

г) интерфейс страницы.

135. Страница контента – это:

а) страница, основанная на эталонной странице;

б) обычная страница .aspx;

в) файл, определяющий шаблон для группы страниц;

г) статическая страница.

136. Эталонная страница – это:

а) файл, определяющий шаблон для группы страниц;

б) обычная страница .aspx;

в) страница, основанная на странице контента;

г) статическая страница.

137. Статическая страница:

а) содержит код на языке гипертекстовой разметки HTML;

б) перед отправлением клиенту проходит цикл обработки на сервере;

в) это программа, модифицирующая запрашиваемые клиентом эталонные страницы;

г) это страница, информация которой отличается от просмотра к просмотру.

138. Динамическая страница:

а) это программа, модифицирующая запрашиваемые клиентом статические страницы;

б) это страница, информация которой не отличается от просмотра к просмотру;

в) это страница, содержание которой не зависит от того, кому она принадлежит;

г) перед отправлением клиенту на сервере не обрабатывается;

139. Страницы контента могут содержать:

а) только тэги вида <asp:Content>;

б) классические тэги HTML;

в) клиентские тэги <script>;

г) комментарии.

140. Страницы контента:

а) могут использоваться лишь совместно с эталонными страницами;

б) это независимые страницы, созданные программистом;

в) допускают литературную разметку;

г) допускают сценарии.

141. В специализированную зарезервированную папку App_Code можно помещать:

а) создаваемые для приложения вспомогательные и бизнес – классы;

б) файлы с информацией о возможностях браузера;

в) XML-файлы или базы данных Access;

г) глобальные для приложения файлы ресурсов.

142. Файлы классов из папки App_Code могут быть написаны:

а) только на языке VB.NET или C#;

б) на любом языке программирования;

в) только на языке VB.NET;

г) только на языке C#.

143. Развертыванием приложения называется:

а) рекурсивное копирование файлов в целевую папку;

б) копирование файлов и регистрация компонентов;

в) копирование файлов и конфигурирование компонентов;

г) рекурсивное копирование файлов с последующим редактированием реестра.

144. Развертывание приложения с использованием инсталляционного файла осуществляется в два этапа:

а) создание виртуального каталога, затем копирование необходимых файлов;

б) установка приложения, затем копирование необходимых файлов;

в) создание виртуального каталога, затем редактирование реестра;

г) установка приложения, затем редактирование реестра.

145. Раздел <location> в файлах machine.config и web.config:

а) позволяет ограничить действие заданных в нем установок определенной папкой;

б) позволяет распространить действие заданных в нем установок определенной папкой;

в) используется для комментариев;

г) данный раздел не допускается в вышеуказанных файлах.

146. Предкомпиляция на месте:

а) осуществляется после развертывания сайта, но до его открытия для общего доступа;

б) осуществляется до развертывания сайта;

в) осуществляется после развертывания сайта, но после его открытия для общего доступа;

г) осуществляется в случае внесения незначительных изменений в код приложения.

147. Как расшифровывается аббревиатура ADO?

а) ActiveX Data Object

б) Active Data Object

в) Active Data Objects

г) ActiveX Data Objects

148. ADO предназначена для работы:

а) С базами данных

б) С графикой

в) С музыкой

г) С текстом

149. ADO…

а) Является компонентом в рамках стратегии OLEDB

б) Является самостоятельным компонентом

в) Является компонентом DAO

г) Такой аббревиатуры не существует

150. ADO и DAO

а) Это различные механизмы реализации работы с базами данных

б) Это один и тот же механизм работы с базами данных

в) Это несравнимые понятия

г) Таких аббревиатур не существует

151. Объект класса Connection служит для создания:

а) соединения с базой данных

б) соединения с сервером

в) соединения с таблицей в базе данных

г) Такого класса не существует

152. Объект класса Command служит для:

а) Выполнения команд при работе с базой данных

б) Выполнения только запросов

в) Выполнения только хранимых процедур

г) Такого класса не существует

153. Какая из приведенных ниже команд выполнят запрос, возвращающий единственное значение?

а) ExecuteScalar

б) ExecuteReader

в) ExecuteNoReader

г) Такой команды не существует

154. Какая из приведенных команд выполнят запрос, не возвращающий значений?

а) ExecuteNoReader

б) ExecuteReader

в) ExecuteScalar

г) Такой команды не существует

155. Какая команда выполняет запрос, возвращающий выборку значений?

а) ExecuteReader

б) ExecuteNoReader

в) ExecuteScalar

г) Такой команды не существует

156. Какая команда выполняет запрос, создающий новую базу данных?

а) Такой команды не существует

б) ExecuteNoReader

в) ExecuteReader

г) ExecuteScalar

157. Для создания соединения с базой данных используются объекты класса:

а) Connection

б) Command

в) StoredProcedures

г) Соединение можно создавать с помощью объектов любого класса

158. Для выполнения запросов используются объекты класса:

а) Command

б) Connection

в) StoredProcedures

г) Для выполнения запроса нет необходимости создания объекта

159. Для хранения выборки данных можно использовать:

а) DataReader

б) DataObject

в) DataTable

г) Хранить выборку отдельно от таблицы не возможно

160. Для хранения выборки, содержащей одно значение можно использовать:

а) Любую переменную, тип которой совпадает с типом значения выборки

б) Невозможно создать выборку, содержащую единственное значение

в) Хранить выборку отдельно от таблицы не возможно

г) Такое не предусмотрено программно

161. Для создания параметров для запросов используются объекты класса

а) Parameter

б) Parametr

в) Param

г) Pam-Param

162. Объекта класса Parameter используются для:

а) Создания параметров для запросов

б) Создания любых параметров для функций

в) Не используются вообще

г) Такого класса не существует

163. Какое свойство CommandType объекта Command используется для выполнения хранимых процедур?

а) StoredProcedure

б) Text

в) Table

г) Такого свойства не существует

164. Какое свойство CommandType объекта Command для выполнения запросов?

а) Text

б) StoredProcedure

в) Table

г) Такого свойства не существует

165. Для выполнения хранимых процедур служит стандартная функция:

а) Такой функции не существует

б) ExecuteStoredProcedure

в) ExecStP

г) StoredProcedure

166. Хранимые процедуры названы так потому, что:

а) Хранятся на сервере

б) Хранятся на компьютере клиента

в) Во время выполнения хранятся в оперативной памяти сервера

г) Во время выполнения хранятся в оперативной памяти клиента

167. Время выполнения хранимых процедур:

а) Меньше времени выполнения аналогичного запроса

б) Больше времени выполнения аналогичного запроса

в) Меньше времени выполнения любого запроса

г) Равно времени выполнения аналогичного запроса

168. При работе с базой данных:

а) Существует возможность четкого ранжирования доступа к базе данных

б) Возможность ранжирования доступа носит декларативный характер и не имеет четкой формализации

в) Алгоритмы и механизмы доступа к базе данных необходимо создавать вручную

г) Все пользователи имеют права полного доступа к базе данных

169. При работе с БД в ASP базу данных можно создавать:

а) либо программно, либо с помощью какой-либо СУБД

б) Только программно

в) Только с помощью внешних приложений

г) Создавать нельзя

170. DataReader и DataSet :

а) Выполняют сходные задачи, но с помощью различных механизмов

б) Выполняют различные задачи

в) Не выполняют никаких задач

г) Играют индикативную роль при отображении результатов запросов

171. Использование какой библиотеки необходимо продекларировать для легитимной работы с БД?

а) System.Data.SQLClient

б) System.Data.DataBase

в) System.SQLLanguage

г) С БД можно работать без предварительной декларации каких-либо библиотек

172. Различные виды интерфейса пользователя.

173. Элементы управления для построения оконного пользовательского интерфейса, их компоновка, отображение данных.

174. Этапы создание приложения ASP.NET.

2.6. Практические задания

Методические рекомендации.

1. Изучите теоретический материал, изложенный в лекциях, в литературных источниках и информационных ресурсах по тематике задания.

2. Разработайте алгоритм выполнения задания.

3. Запишите алгоритм на языке программирования C#.

4. Подготовьте отчет о выполнении задания.

Задание 1. Создать приложение, выводящее на экран текущее время. При обновлении страницы должна происходить коррекция текущего времени.

Задание 2. Предыдущее задание, но время на часах клиента и сервера должно совпадать, если они находятся в разных точках земного шара.

Задание 3. Создать приложение, в котором используется элемент управления «RadioButtonList» (на примере голосования)

Задание 4. Создать приложение, в котором используется элемент управления «DropDown List» (на примере выбора товара в Интернет-магазине)

Задание 5. Создать приложение, в котором используется три рассмотренных элемента управления.

Задание 6. Изучите теоретический материал, изложенный в лекциях, в литературных источниках и информационных ресурсах по тематике задания. Создайте приложение на основе WinForm. Добавьте элементы управления для заполнения данных регистрации пользователя. Добавьте валидаторы для проверки корректности ввода данных пользователем. Протестируйте работу приложения.

Задание 7. Рассмотрим приложение для интерполяции функциии различными методами. Приложение представлено в виде оконного приложения, которое является отдельным потоком в операционной системе. Использована интегрированная среда Visual Studio.Net. Приведите разбор текста приложения, приведенного в листинге.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Interpolation
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new MainForm());
      }
   }
}
namespace Interpolation
{
   partial class MainForm
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;
      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }
      #region Windows Form Designer generated code
      /// <summary>
      /// Required method for Designer support - do not modify
      /// the contents of this method with the code editor.
```

```
        /// </summary>
        private void InitializeComponent()
        {
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea1 =
new System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend2 = new
System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Title title2 = new
System.Windows.Forms.DataVisualization.Charting.Title();
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea2 =
new System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend1 = new
System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Title title1 = new
System.Windows.Forms.DataVisualization.Charting.Title();
            this.comboBox1 = new System.Windows.Forms.ComboBox();
            this.chart1                                    =                                    new
System.Windows.Forms.DataVisualization.Charting.Chart();
            this.buttonBuild = new System.Windows.Forms.Button();
            this.toNumericUpDown = new System.Windows.Forms.NumericUpDown();
            this.fromNumericUpDown                         =                                    new
System.Windows.Forms.NumericUpDown();
            this.nodesNumericUpDown                        =                                    new
System.Windows.Forms.NumericUpDown();
            this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();
            this.label5 = new System.Windows.Forms.Label();
            this.buttonClean = new System.Windows.Forms.Button();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.UserRadioButton = new System.Windows.Forms.RadioButton();
            this.OptimRadioButton = new System.Windows.Forms.RadioButton();
            this.EquidRadioButton = new System.Windows.Forms.RadioButton();
            this.button = new System.Windows.Forms.Button();
            this.chart3                                    =                                    new
System.Windows.Forms.DataVisualization.Charting.Chart();
            this.groupBox3 = new System.Windows.Forms.GroupBox();
            this.FreeNodeTextBox = new System.Windows.Forms.TextBox();
            this.NewNodeBbutton = new System.Windows.Forms.Button();
            ((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.toNumericUpDown)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.fromNumericUpDown)).BeginIni
t();
```

```
((System.ComponentModel.ISupportInitialize)(this.nodesNumericUpDown)).BeginI
nit();
            this.tableLayoutPanel1.SuspendLayout();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.chart3)).BeginInit();
            this.groupBox3.SuspendLayout();
            this.SuspendLayout();
            //
            // comboBox1
            //
            this.comboBox1.Dock = System.Windows.Forms.DockStyle.Fill;
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Items.AddRange(new object[] {
            "sin(x)",
            "cos(x)",
            "log(x)"});
            this.comboBox1.Location = new System.Drawing.Point(146, 3);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(216, 21);
            this.comboBox1.TabIndex = 0;
            //
            // chart1
            //
            chartArea1.AxisX.TextOrientation                                    =
System.Windows.Forms.DataVisualization.Charting.TextOrientation.Horizontal;
            chartArea1.Name = "ChartArea1";
            this.chart1.ChartAreas.Add(chartArea1);
            this.tableLayoutPanel1.SetColumnSpan(this.chart1, 5);
            this.chart1.Dock = System.Windows.Forms.DockStyle.Fill;
            legend2.Enabled = false;
            legend2.Name = "Legend1";
            this.chart1.Legends.Add(legend2);
            this.chart1.Location = new System.Drawing.Point(3, 32);
            this.chart1.Name = "chart1";
            this.tableLayoutPanel1.SetRowSpan(this.chart1, 4);
            this.chart1.Size = new System.Drawing.Size(706, 305);
            this.chart1.TabIndex = 1;
            this.chart1.Text = "chart1";
            title2.BorderWidth = 3;
            title2.Name = "Title1";
            title2.Text = "Функции";
            title2.TextStyle                                                    =
System.Windows.Forms.DataVisualization.Charting.TextStyle.Frame;
            this.chart1.Titles.Add(title2);
```

```
        this.chart1.DoubleClick += new
System.EventHandler(this.chart1_DoubleClick);
        this.chart1.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.chart1_MouseDown);
        //
        // buttonBuild
        //
        this.buttonBuild.Dock = System.Windows.Forms.DockStyle.Fill;
        this.buttonBuild.Location = new System.Drawing.Point(368, 3);
        this.buttonBuild.Name = "buttonBuild";
        this.buttonBuild.Size = new System.Drawing.Size(141, 23);
        this.buttonBuild.TabIndex = 2;
        this.buttonBuild.Text = "Построить";
        this.buttonBuild.UseVisualStyleBackColor = true;
        this.buttonBuild.Click += new System.EventHandler(this.buttonBuild_Click);
        //
        // toNumericUpDown
        //
        this.toNumericUpDown.Location = new System.Drawing.Point(33, 44);
        this.toNumericUpDown.Maximum = new decimal(new int[] {
        1000,
        0,
        0,
        0});
        this.toNumericUpDown.Minimum = new decimal(new int[] {
        100,
        0,
        0,
        -2147483648});
        this.toNumericUpDown.Name = "toNumericUpDown";
        this.toNumericUpDown.Size = new System.Drawing.Size(102, 20);
        this.toNumericUpDown.TabIndex = 4;
        this.toNumericUpDown.ValueChanged += new
System.EventHandler(this.toNumericUpDown_ValueChanged);
        //
        // fromNumericUpDown
        //
        this.fromNumericUpDown.Location = new System.Drawing.Point(33, 13);
        this.fromNumericUpDown.Maximum = new decimal(new int[] {
        1000,
        0,
        0,
        0});
        this.fromNumericUpDown.Minimum = new decimal(new int[] {
        100,
        0,
```

```
        0,
        -2147483648});
        this.fromNumericUpDown.Name = "fromNumericUpDown";
        this.fromNumericUpDown.Size = new System.Drawing.Size(102, 20);
        this.fromNumericUpDown.TabIndex = 5;
        this.fromNumericUpDown.ValueChanged += new
System.EventHandler(this.fromNumericUpDown_ValueChanged);
        //
        // nodesNumericUpDown
        //
        this.nodesNumericUpDown.Location = new System.Drawing.Point(6, 19);
        this.nodesNumericUpDown.Minimum = new decimal(new int[] {
        2,
        0,
        0,
        0});
        this.nodesNumericUpDown.Name = "nodesNumericUpDown";
        this.nodesNumericUpDown.Size = new System.Drawing.Size(120, 20);
        this.nodesNumericUpDown.TabIndex = 6;
        this.nodesNumericUpDown.Value = new decimal(new int[] {
        2,
        0,
        0,
        0});
        this.nodesNumericUpDown.ValueChanged += new
System.EventHandler(this.nodesNumericUpDown_ValueChanged);
        //
        // tableLayoutPanel1
        //
        this.tableLayoutPanel1.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.T
op | System.Windows.Forms.AnchorStyles.Bottom)
        | System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
        this.tableLayoutPanel1.ColumnCount = 6;
        this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
16.66667F));
        this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
25.85784F));
        this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
17.15686F));
```

```
        this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
19.85294F));
        this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
3.553921F));
        this.tableLayoutPanel1.ColumnStyles.Add(new
System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Percent,
16.66667F));
        this.tableLayoutPanel1.Controls.Add(this.label5, 0, 0);
        this.tableLayoutPanel1.Controls.Add(this.comboBox1, 1, 0);
        this.tableLayoutPanel1.Controls.Add(this.buttonBuild, 2, 0);
        this.tableLayoutPanel1.Controls.Add(this.chart1, 0, 1);
        this.tableLayoutPanel1.Controls.Add(this.buttonClean, 3, 0);
        this.tableLayoutPanel1.Controls.Add(this.groupBox1, 5, 1);
        this.tableLayoutPanel1.Controls.Add(this.groupBox2, 5, 2);
        this.tableLayoutPanel1.Controls.Add(this.button, 5, 3);
        this.tableLayoutPanel1.Controls.Add(this.chart3, 0, 5);
        this.tableLayoutPanel1.Controls.Add(this.groupBox3, 5, 5);
        this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);
        this.tableLayoutPanel1.Name = "tableLayoutPanel1";
        this.tableLayoutPanel1.RowCount = 7;
        this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle());
        this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute,
76F));
        this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute,
121F));
        this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute,
42F));
        this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute,
72F));
        this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute,
103F));
        this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute,
50F));
        this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute,
20F));
```

```
        this.tableLayoutPanel1.RowStyles.Add(new
System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Absolute,
20F));
        this.tableLayoutPanel1.Size = new System.Drawing.Size(859, 513);
        this.tableLayoutPanel1.TabIndex = 7;
        //
        // label5
        //
        this.label5.AutoSize = true;
        this.label5.Dock = System.Windows.Forms.DockStyle.Fill;
        this.label5.Font = new System.Drawing.Font("Monotype Corsiva", 11.25F,
System.Drawing.FontStyle.Italic, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
        this.label5.Location = new System.Drawing.Point(3, 0);
        this.label5.Name = "label5";
        this.label5.Size = new System.Drawing.Size(137, 29);
        this.label5.TabIndex = 10;
        this.label5.Text = "f(x) = ";
        this.label5.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
        this.label5.Click += new System.EventHandler(this.label5_Click);
        //
        // buttonClean
        //
        this.buttonClean.Dock = System.Windows.Forms.DockStyle.Fill;
        this.buttonClean.Location = new System.Drawing.Point(515, 3);
        this.buttonClean.Name = "buttonClean";
        this.buttonClean.Size = new System.Drawing.Size(164, 23);
        this.buttonClean.TabIndex = 11;
        this.buttonClean.Text = "Сбросить";
        this.buttonClean.UseVisualStyleBackColor = true;
        this.buttonClean.Click += new
System.EventHandler(this.buttonClean_Click);
        //
        // groupBox1
        //
        this.groupBox1.Controls.Add(this.label2);
        this.groupBox1.Controls.Add(this.label1);
        this.groupBox1.Controls.Add(this.toNumericUpDown);
        this.groupBox1.Controls.Add(this.fromNumericUpDown);
        this.groupBox1.Dock = System.Windows.Forms.DockStyle.Fill;
        this.groupBox1.Location = new System.Drawing.Point(715, 32);
        this.groupBox1.Name = "groupBox1";
        this.groupBox1.Size = new System.Drawing.Size(141, 70);
        this.groupBox1.TabIndex = 12;
        this.groupBox1.TabStop = false;
        this.groupBox1.Text = "Интервал";
```

```
        //
        // label2
        //
        this.label2.AutoSize = true;
        this.label2.Location = new System.Drawing.Point(7, 49);
        this.label2.Name = "label2";
        this.label2.Size = new System.Drawing.Size(22, 13);
        this.label2.TabIndex = 1;
        this.label2.Text = "До";
        //
        // label1
        //
        this.label1.AutoSize = true;
        this.label1.Location = new System.Drawing.Point(7, 20);
        this.label1.Name = "label1";
        this.label1.Size = new System.Drawing.Size(20, 13);
        this.label1.TabIndex = 0;
        this.label1.Text = "От";
        //
        // groupBox2
        //
        this.groupBox2.Controls.Add(this.UserRadioButton);
        this.groupBox2.Controls.Add(this.OptimRadioButton);
        this.groupBox2.Controls.Add(this.EquidRadioButton);
        this.groupBox2.Controls.Add(this.nodesNumericUpDown);
        this.groupBox2.Location = new System.Drawing.Point(715, 108);
        this.groupBox2.Name = "groupBox2";
        this.groupBox2.Size = new System.Drawing.Size(132, 112);
        this.groupBox2.TabIndex = 13;
        this.groupBox2.TabStop = false;
        this.groupBox2.Text = "Узлы";
        //
        // UserRadioButton
        //
        this.UserRadioButton.AutoSize = true;
        this.UserRadioButton.Location = new System.Drawing.Point(6, 92);
        this.UserRadioButton.Name = "UserRadioButton";
        this.UserRadioButton.Size = new System.Drawing.Size(122, 17);
        this.UserRadioButton.TabIndex = 9;
        this.UserRadioButton.TabStop = true;
        this.UserRadioButton.Text = "Пользовательские";
        this.UserRadioButton.UseVisualStyleBackColor = true;
        this.UserRadioButton.CheckedChanged += new
System.EventHandler(this.UserRadioButton_CheckedChanged);
        //
        // OptimRadioButton
```

```
            //
            this.OptimRadioButton.AutoSize = true;
            this.OptimRadioButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.OptimRadioButton.ForeColor = System.Drawing.Color.SeaGreen;
            this.OptimRadioButton.Location = new System.Drawing.Point(6, 69);
            this.OptimRadioButton.Name = "OptimRadioButton";
            this.OptimRadioButton.Size = new System.Drawing.Size(108, 17);
            this.OptimRadioButton.TabIndex = 8;
            this.OptimRadioButton.TabStop = true;
            this.OptimRadioButton.Text = "Чебышевские";
            this.OptimRadioButton.UseVisualStyleBackColor = true;
            //
            // EquidRadioButton
            //
            this.EquidRadioButton.AutoSize = true;
            this.EquidRadioButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.EquidRadioButton.ForeColor = System.Drawing.Color.Maroon;
            this.EquidRadioButton.Location = new System.Drawing.Point(6, 45);
            this.EquidRadioButton.Name = "EquidRadioButton";
            this.EquidRadioButton.Size = new System.Drawing.Size(112, 17);
            this.EquidRadioButton.TabIndex = 7;
            this.EquidRadioButton.TabStop = true;
            this.EquidRadioButton.Text = "Равностоящие";
            this.EquidRadioButton.UseVisualStyleBackColor = true;
            //
            // button
            //
            this.button.Location = new System.Drawing.Point(715, 229);
            this.button.Name = "button";
            this.button.Size = new System.Drawing.Size(141, 34);
            this.button.TabIndex = 14;
            this.button.Text = "Вычислить";
            this.button.UseVisualStyleBackColor = true;
            this.button.Click += new System.EventHandler(this.button_Click);
            //
            // chart3
            //
            chartArea2.AxisX.Maximum = 10D;
            chartArea2.Name = "ChartArea1";
            this.chart3.ChartAreas.Add(chartArea2);
            this.tableLayoutPanel1.SetColumnSpan(this.chart3, 5);
            this.chart3.Dock = System.Windows.Forms.DockStyle.Fill;
```

```
        legend1.Enabled = false;
        legend1.Name = "Legend1";
        this.chart3.Legends.Add(legend1);
        this.chart3.Location = new System.Drawing.Point(3, 343);
        this.chart3.Name = "chart3";
        this.tableLayoutPanel1.SetRowSpan(this.chart3, 2);
        this.chart3.Size = new System.Drawing.Size(706, 167);
        this.chart3.TabIndex = 17;
        this.chart3.Text = "chart3";
        title1.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(204)));
        title1.Name = "Title1";
        title1.Text = "Погрешность";
        this.chart3.Titles.Add(title1);
        //
        // groupBox3
        //
        this.groupBox3.Controls.Add(this.FreeNodeTextBox);
        this.groupBox3.Controls.Add(this.NewNodeBbutton);
        this.groupBox3.Location = new System.Drawing.Point(715, 343);
        this.groupBox3.Name = "groupBox3";
        this.groupBox3.Size = new System.Drawing.Size(132, 97);
        this.groupBox3.TabIndex = 15;
        this.groupBox3.TabStop = false;
        this.groupBox3.Text = "Произвольный узел";
        //
        // FreeNodeTextBox
        //
        this.FreeNodeTextBox.Location = new System.Drawing.Point(6, 13);
        this.FreeNodeTextBox.Name = "FreeNodeTextBox";
        this.FreeNodeTextBox.Size = new System.Drawing.Size(112, 20);
        this.FreeNodeTextBox.TabIndex = 1;
        //
        // NewNodeBbutton
        //
        this.NewNodeBbutton.Enabled = false;
        this.NewNodeBbutton.Location = new System.Drawing.Point(6, 39);
        this.NewNodeBbutton.Name = "NewNodeBbutton";
        this.NewNodeBbutton.Size = new System.Drawing.Size(112, 23);
        this.NewNodeBbutton.TabIndex = 0;
        this.NewNodeBbutton.Text = "Добавить";
        this.NewNodeBbutton.UseVisualStyleBackColor = true;
        this.NewNodeBbutton.Click += new
System.EventHandler(this.NewNodeBbutton_Click);
        //
```

```
            // MainForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(859, 513);
            this.Controls.Add(this.tableLayoutPanel1);
            this.Name = "MainForm";
            this.Text = "Интерполяция";
            this.Load += new System.EventHandler(this.Form1_Load);
            ((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.toNumericUpDown)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.fromNumericUpDown)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.nodesNumericUpDown)).EndInit();
            this.tableLayoutPanel1.ResumeLayout(false);
            this.tableLayoutPanel1.PerformLayout();
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.groupBox2.ResumeLayout(false);
            this.groupBox2.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)(this.chart3)).EndInit();
            this.groupBox3.ResumeLayout(false);
            this.groupBox3.PerformLayout();
            this.ResumeLayout(false);
        }
        #endregion
        private System.Windows.Forms.ComboBox comboBox1;
        private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
        private System.Windows.Forms.Button buttonBuild;
        private System.Windows.Forms.NumericUpDown toNumericUpDown;
        private System.Windows.Forms.NumericUpDown fromNumericUpDown;
        private System.Windows.Forms.NumericUpDown nodesNumericUpDown;
        private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Button buttonClean;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.RadioButton UserRadioButton;
        private System.Windows.Forms.RadioButton OptimRadioButton;
        private System.Windows.Forms.RadioButton EquidRadioButton;
        private System.Windows.Forms.Button button;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.TextBox FreeNodeTextBox;
        private System.Windows.Forms.Button NewNodeBbutton;
        private System.Windows.Forms.DataVisualization.Charting.Chart chart3;
```

```
    }
}
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
namespace Interpolation
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }
        Mathos.Parser.MathParser parser;
        int nodes, NODES;
        Decimal from;
        Decimal to;
        Decimal step;
        double[] X, Y;
        double[] X_l, Y_l;
        double[] X_Tn, Y_Optimal;
        double[] Tn, Lagr;
        double[] Y_Free;
        double[] Y_Error2;
        double[] Y_Error;
        double[] Y_Error3;
        double[] xList;
       List<double> list = new List<double>();
       List<double> Ylist = new List<double>();
         Series function = new Series("func");
        Series lagrange = new Series("Lagrange");
        Series optimal = new Series("Chebyshev");
        Series free = new Series("free");
       Series LagrangeError = new Series ("LaggangeError");
       Series ChebyshevError = new Series("ChebyshevError");
       Series UserError = new Series("UserError");
        private void Form1_Load(object sender, EventArgs e)
        {
```

```
            parser = new Mathos.Parser.MathParser(true, true, true);
            //chart1.ChartAreas[0].AxisX.ScaleView.Zoom(0, 100);
            chart1.ChartAreas[0].CursorX.IsUserEnabled = true;
            chart1.ChartAreas[0].CursorX.IsUserSelectionEnabled = true;
            chart1.ChartAreas[0].AxisX.ScaleView.Zoomable = true;
           // chart1.ChartAreas[0].AxisX.ScrollBar.IsPositionedInside = true;
           // chart1.ChartAreas[0].AxisY.ScaleView.Zoom(0, 100);
            chart1.ChartAreas[0].CursorY.IsUserEnabled = true;
            chart1.ChartAreas[0].CursorY.IsUserSelectionEnabled = true;
            chart1.ChartAreas[0].AxisY.ScaleView.Zoomable = true;
           // chart1.ChartAreas[0].AxisY.ScrollBar.IsPositionedInside = true;
            chart1.ChartAreas[0].AxisX.MajorGrid.LineDashStyle                    =
ChartDashStyle.Dash;
            chart1.ChartAreas[0].AxisY.MajorGrid.LineDashStyle                    =
ChartDashStyle.Dash;
            chart1.ChartAreas[0].AxisX.LabelStyle.Format = "0.00";
            chart1.ChartAreas[0].AxisY.LabelStyle.Format = "0.00";
            chart3.ChartAreas[0].AxisX.ScaleView.Zoom(0, 100);
            chart3.ChartAreas[0].CursorX.IsUserEnabled = true;
            chart3.ChartAreas[0].CursorX.IsUserSelectionEnabled = true;
            chart3.ChartAreas[0].AxisX.ScaleView.Zoomable = true;
            chart3.ChartAreas[0].AxisX.ScrollBar.IsPositionedInside = true;
 //           chart3.ChartAreas[0].AxisY.ScaleView.Zoom(0, 100);
            chart3.ChartAreas[0].CursorY.IsUserEnabled = true;
            chart3.ChartAreas[0].CursorY.IsUserSelectionEnabled = true;
            chart3.ChartAreas[0].AxisY.ScaleView.Zoomable = true;
            //chart3.ChartAreas[0].AxisY.ScrollBar.IsPositionedInside = true;
            chart3.ChartAreas[0].AxisX.MajorGrid.LineDashStyle                    =
ChartDashStyle.Dash;
            chart3.ChartAreas[0].AxisY.MajorGrid.LineDashStyle                    =
ChartDashStyle.Dash;
          chart3.ChartAreas[0].AxisX.LabelStyle.Format = "0.00";
             Bitmap  LagrColor = new Bitmap(17, 17);
             using (Graphics gr = Graphics.FromImage(LagrColor))
             {
             }
        }
        private void buttonBuild_Click(object sender, EventArgs e)
        {
            list.Clear();
            Ylist.Clear();
            chart1.Series.Clear();
            chart3.Series.Clear();
            chart1.ChartAreas[0].AxisX.Minimum = Double.NaN;
            chart1.ChartAreas[0].AxisY.Minimum = Double.NaN;
            chart1.ChartAreas[0].AxisX.Maximum = Double.NaN;
```

```
chart1.ChartAreas[0].AxisY.Maximum = Double.NaN;
chart3.ChartAreas[0].AxisX.Minimum = Double.NaN;
chart3.ChartAreas[0].AxisY.Minimum = Double.NaN;
chart3.ChartAreas[0].AxisX.Maximum = Double.NaN;
chart3.ChartAreas[0].AxisY.Maximum = Double.NaN;
try
{
    from = fromNumericUpDown.Value;
  to = toNumericUpDown.Value;
   nodes = Convert.ToInt32(nodesNumericUpDown.Value);
    step = (to - from) / (nodes - 1);
   //------------------------//
   double Dots = chart1.Size.Width*0.5;
   double STEP = (double)(to - from) / Dots;
    NODES = Convert.ToInt32(Math.Ceiling(Dots));
   xList = new double[NODES];
   Y = new double[NODES];
   xList[0] =(double) from;
   for (int i = 1; i < xList.Length; i++)
   {
      xList[i] = xList[i - 1] + STEP;
   }

   for (int i = 0; i < xList.Length; i++)
   {
      parser.ProgrammaticallyParse("x := " +
xList[i].ToString("F10").Replace(",", "."));
      Decimal y = parser.Parse(comboBox1.Text);
      Y[i] = Convert.ToDouble(y);
   }
   //---------------------//
   function.ChartType = SeriesChartType.Spline;
   function.BorderWidth = 4;
   function.Color = Color.DarkOrange;
   function.Points.DataBindXY(xList, Y);
   chart1.Series.Add(function);
   X_l = new double[nodes];
   Y_l = new double[nodes];
   for (int i = 0; i < nodes; i++)
   {
  Decimal Xi = from + i * step;
 parser.ProgrammaticallyParse("x := " + Xi.ToString("F10").Replace(",", "."));
      Decimal y = parser.Parse(comboBox1.Text);
      Y_l[i] = Convert.ToDouble(y);
      X_l[i] = (double)Xi;
      // listBox1.Items.Add(y + " " + Xi);
```

```
            }
            Lagr = new double[NODES];
            for (int i = 0; i < NODES; i++)
            {
              // Decimal Xi = from + i * step;
          Lagr[i] = LagrangePolynom((double)from, (double)to, xList[i], Y_l, nodes-1);
            //    listBox1.Items.Add(Lagr[i]);
            }
            lagrange.ChartType = SeriesChartType.Spline;
            lagrange.Color = Color.Transparent;
            lagrange.Points.DataBindXY(xList, Lagr);
            lagrange.BorderWidth = 2;
            chart1.Series.Add(lagrange);
            Y_Optimal = new double[nodes];
            X_Tn = Chebyshev((double)from, (double)to, nodes);
            for (int i = 0; i < nodes; i++)
            {
parser.ProgrammaticallyParse("x := " + X_Tn[i].ToString("F10").Replace(",", "."));
               Decimal y = parser.Parse(comboBox1.Text);
               Y_Optimal[i] = (double)y;
            }
            Tn = new double[NODES];
            for (int i = 0; i < NODES; i++)
            {
               Tn[i] = Lagrange_2(xList[i], X_Tn, Y_Optimal, nodes);
            }
            optimal.ChartType = SeriesChartType.Spline;
            optimal.Color = Color.Transparent;
            optimal.Points.DataBindXY(xList, Tn);
            optimal.BorderWidth = 2;
            chart1.Series.Add(optimal);
            free.ChartType = SeriesChartType.Spline;
            free.Color = Color.Transparent;
            chart1.Series.Add(free);
            //Для пользовательских узлов
            //click = new double[nodes];
           Y_Error = new double[NODES];
            Y_Error2 = new double[NODES];
           for (int i = 0; i < NODES; i++)
            {
              Y_Error[i] = Math.Abs(Y[i] - Lagr[i]);
             Y_Error2[i] = Math.Abs(Y[i] - Tn[i]);
            }
           chart3.ChartAreas[0].AxisX.Maximum = 10.0;
           LagrangeError.BorderWidth = 2;
            LagrangeError.ChartType = SeriesChartType.Spline;
```

```
            LagrangeError.Color = Color.Transparent;
            LagrangeError.Points.DataBindXY(xList, Y_Error);
           chart3.Series.Add(LagrangeError);
           //
            ChebyshevError.BorderWidth = 2;
            ChebyshevError.ChartType = SeriesChartType.Spline;
            ChebyshevError.Color = Color.Transparent;
            ChebyshevError.Points.DataBindXY(xList, Y_Error2);
            chart3.Series.Add(ChebyshevError);
            UserError.BorderWidth = 2;
            UserError.ChartType = SeriesChartType.Spline;
            UserError.Color = Color.Transparent;
            chart3.Series.Add(UserError);
            //
            if (comboBox1.Text == "log(x)" && fromNumericUpDown.Value < 1)
               fromNumericUpDown.Value = 1;
            if (nodesNumericUpDown.Value == 0)
               throw new Exception("Выберите количество узлов");
            //Сгенерируем исключение, если границы интервала равны
            if (from == to)
            { throw new Exception("Верхняя и нижня границы интервала не могут
совпадать"); }
            else if (from <= 0 && comboBox1.Text.Contains("log") &&
comboBox1.Text.Contains("x"))
            {
               throw new Exception("Неверно задан интервал для логарифма");
            }
            if (comboBox1.Text == "")
            {
               throw new Exception("Введите функцию");
            }

         }

         catch (Exception ex)
         {
            MessageBox.Show("Ошибка", ex.Message + "\n");
         }
      }
      private void fromNumericUpDown_ValueChanged(object sender, EventArgs e)
      {
         toNumericUpDown.Minimum = fromNumericUpDown.Value;
         fromNumericUpDown.Maximum = toNumericUpDown.Value;
      }
      private void toNumericUpDown_ValueChanged(object sender, EventArgs e)
      {
```

```
            toNumericUpDown.Minimum = fromNumericUpDown.Value;
            fromNumericUpDown.Maximum = toNumericUpDown.Value;
        }
        double LagrangePolynom(double a, double b, double x, double[] y, int n)
        {
            double h = (b - a) / n; //шаг
            double t = (x - a) / h; //величина, которая зависит от x и от шага
            double sk = (n % 2 == 0) ? 1 : -1;
            for (int k = 1; k <= n; k++)
                sk = sk / k;
            double S = 0;
            for (int k = 0; k <= n; k++)
            {
                double P = 1;
                for (int i = 0; i <= n; i++)
                    if (i != k) P = P * (t - i);
                S = S + y[k] * sk * P;
                sk = sk * (-1) * (n - k) / (k + 1);
            }
            return S;
        }
        double Lagrange_2(double x, double[] X, double[] Y, int nodes)
        {
            double result = 0;
            for (int i = 0; i < nodes; i++)
            {
                double P = 1.0;
                for (int j = 0; j < nodes; j++)
                {
                    if (j != i)
                        P *= (x - X[j]) / (X[i] - X[j]);
                }
                result += Y[i] * P;
            }
            return result;
        }
        double[] Chebyshev(double a, double b, int nodes)
        {
            double[] X1 = new double[nodes];
            for (int i = 0; i < nodes; i++)
            {
X1[i] = (a + b) * 0.5 + (b - a) * Math.Cos((Math.PI * (2 * i)) / (2 * (nodes - 1))) * 0.5;
//   X1[i] = 0.5 * ((b - a) * Math.Cos((2 * i - 1) * Math.PI / (2 * nodes + 2)) + (b + a));
            }
            Array.Sort(X1);
            return X1;
```

```
        }
        private void button_Click(object sender, EventArgs e)
        {
            if (EquidRadioButton.Checked)
            {
                lagrange.Color = Color.Maroon;
                LagrangeError.Color = Color.Maroon;
            }
            if (OptimRadioButton.Checked)
            {
                optimal.Color = Color.SeaGreen;
                ChebyshevError.Color = Color.SeaGreen;
            }
            if (UserRadioButton.Checked)
            {
                UserError.Color = Color.Black;
                free.Color = Color.Red;

            }
        }
        private void chart1_MouseDown(object sender, MouseEventArgs e)
        {
            chart3.ChartAreas[0].AxisX.Maximum = 10.0;
            UserError.Color = Color.Red;
            // chart3.Series.Clear();
            chart1.ChartAreas[0].AxisX.ScaleView.Zoomable = Enabled;
            chart1.ChartAreas[0].AxisY.ScaleView.Zoomable = Enabled;
            try
            {
double node = Math.Round(chart1.ChartAreas[0].AxisX.PixelPositionToValue(e.X), 4);
              // FreeNodeTextBox.Text = list.Count.ToString();

              FreeNodeTextBox.Text = node.ToString();
               if (UserRadioButton.Checked)
               {
                  if (list.Count == 0)
                  {
                     list.Add((double)toNumericUpDown.Value);
                     list.Add((double)fromNumericUpDown.Value);
                  }
if (node >= (double)fromNumericUpDown.Value&& node <=
(double)toNumericUpDown.Value)
                { list.Add(node);
                  list.Sort();
                  free.ChartType = SeriesChartType.Spline;
                  free.Color = Color.DarkBlue;
```

```
            double[] xxx = list.ToArray();////
            nodesNumericUpDown.Value=xxx.Length;
            Y_Free = new double[xxx.Length];
            using (System.IO.StreamWriter file = new
System.IO.StreamWriter(@"1.txt"))
            {
           if (xxx.Length >= 3)
            {
               for (int i = 0; i < xxx.Length; i++)
               {
                  parser.ProgrammaticallyParse("x := " +
xxx[i].ToString("F10").Replace(",", "."));
                     Decimal y = parser.Parse(comboBox1.Text);
                     Y_Free[i] = Convert.ToDouble(y);
                     file.WriteLine("xxx=" + xxx[i] + "yyyy=" + Y_Free[i]);
                  }
            }
              double []UserY = new double[NODES];
              Y_Error3 = new double[NODES];
               for (int i = 0; i < NODES; i++)
               {
                  UserY[i] = Lagrange_2(xList[i], xxx, Y_Free, xxx.Length);
                  Y_Error3[i] = Math.Abs(Y[i] - UserY[i]);
               }
              free.Points.DataBindXY(xList, UserY);
              chart1.Refresh();
              UserError.Points.DataBindXY(xList, Y_Error3);
              chart3.Refresh();
            }
         }
         }
         else MessageBox.Show("Узел выходит за интервал");
      }
      catch (Exception ex)
      {
        MessageBox.Show("Ошибка в узлах  " + ex.Message + "\n");
      }
   }
   private void NewNodeBbutton_Click(object sender, EventArgs e)
   {
      try
      {
      }

      catch (Exception ex)
      {
```

```
            MessageBox.Show("Достаточное количество узлов");
        }

    }
    private void UserRadioButton_CheckedChanged(object sender, EventArgs e)
    {
        NewNodeBbutton.Enabled = true;
    }
    private void buttonClean_Click(object sender, EventArgs e)
    {
        chart1.Series.Clear();
        chart1.ChartAreas[0].AxisX.Minimum = Double.NaN;
        chart1.ChartAreas[0].AxisY.Minimum = Double.NaN;
        chart1.ChartAreas[0].AxisX.Maximum = Double.NaN;
        chart1.ChartAreas[0].AxisY.Maximum = Double.NaN;
        chart3.Series.Clear();
        chart3.ChartAreas[0].AxisX.Minimum = Double.NaN;
        chart3.ChartAreas[0].AxisY.Minimum = Double.NaN;
        chart3.ChartAreas[0].AxisX.Maximum = Double.NaN;
        chart3.ChartAreas[0].AxisY.Maximum = Double.NaN;
    }
   private void nodesNumericUpDown_ValueChanged(object sender, EventArgs e)
    {
    }

    private void chart1_DoubleClick(object sender, EventArgs e)
    {
    }
    private void label5_Click(object sender, EventArgs e)
    {
    }

  }
}
```

Тема 3. Использование технологии ADO.Net для работы с базами данных

3.1. Основы ADO.NET

Платформа .NET Framework включает собственную технологию доступа к данным – ADO.NET. (ActiveX Data Objects.NET). Эта технология состоит из управляемых классов, позволяющих приложениям .NET подключаться к источникам данных (обычно реляционным базам данных), выполнять команды и управлять автономными данными. Использование этой технологии не почти не зависит от вида приложения: веб-приложение, клиент-серверные настольные приложения, однопользовательские приложения, подключаемые к локальной базе данных.

В ASP.NET можно получить информацию из базы данных без использования классов ADO.NET. Для этого используются такие варианты:

• элемент управления SqlDataSource. Элемент SqlDataSource позволяет определять запросы декларативно. Можно подключать SqlDataSource к элементам управления (например, GridView), предоставляя страницам возможность редактирования и обновления данных без необходимости в коде ADO.NET. При этом ADO.NET используется неявно. Недостатком этого способа является размещение логики базы данных в ту часть страницы, которая касается разметки;

• LINQ (Language Integrated Query) to Entities. С помощью LINQ to Entites можно сгенерировать модель данных с поддержкой на этапе проектирования в Visual Studio. Надлежащая логика для доступа к базе данных в таком случае генерируется автоматически. LINQ to Entites поддерживает внесение обновлений, генерирует безопасные и правильно оформленные операторы SQL и предоставляет широкие возможности в плане настройки. Кроме того, LINQ to Entities теперь заменяет собой более простую модель LINQ to SQL, которой разработчики приложений ASP.NET пользовались ранее. Вдобавок LINQ to

Entities служит основой для новой системы поставки данных под названием ASP.NET Dynamic Data.

В ADO.NET используется многоуровневая архитектура, которая обращается вокруг небольшого числа ключевых концепций, таких как объекты Connection, Command и DataSet.

Однако архитектура ADO.NET серьезно отличается от классической архитектуры ADO методом работы с разными поставщиками БД. В ADO программисты всегда используют обобщенный набор объектов, независимо от лежащих в их основе источников данных. Например, для извлечения записи из базы данных Oracle используется тот же класс Connection, что применяется для решения той же задачи в SQL Server. Это не касается технологии ADO.NET, которая использует модель поставщиков данных.

Поставщик данных (data provider) — это набор классов ADO.NET, которые позволяют получать доступ к определенной базе данных, выполнять команды SQL и извлекать данные. По сути, поставщик данных – это мост между вашим приложением и источником данных.

Ниже перечислены классы, которые входят в состав любого поставщика данных:

• Connection. Объект этого класса используется для установки соединения с источником данных;

• Command. Объект этого класса используется для выполнения команд SQL и хранимых процедур;

• DataReader. Объект этого класса предоставляет быстрый однонаправленный доступ только для чтения к данным, извлеченным из запроса;

• Data Adapter. Объект этого класса решает две задачи. Первая — наполнение DataSet (автономная коллекция таблиц и отношений) информацией, извлеченной из источника данных. Вторая — применение изменений к источнику данных в соответствии с модификациями, произведенными в DataSet.

ADO.NET не содержит объектов обобщенных поставщиков данных. Вместо этого имеется набор специализированных поставщиков для различных источников данных.

Каждый поставщик данных имеет специфическую реализацию классов Connection, Command, DataReader и DataAdapter, оптимизированных для конкретных реляционных систем управления базами данных (СУБД). Например, для создания подключения к базе данных SQL Server используется класс соединения по имени SqlConnection.

Одной из ключевых идей, лежащих в основе модели поставщиков ADO.NET, является расширяемость. Другими словами, разработчики могут создавать собственные поставщики для патентованных источников данных. В действительности доступно множество подтверждающих это примеров, которые демонстрируют, как создавать настраиваемые поставщики ADO.NET, служащие оболочками для нереляционных хранилищ данных, таких как файловая система или служба каталогов. Некоторые независимые производители также продают собственные поставщики данных для .NET.

ADO.NET также имеет другой уровень стандартизации — DataSet. Класс DataSet – это контейнер данных общего назначения, которые извлекаются из одной или более таблиц источника данных. DataSet полностью обобщен; другими словами, специализированные поставщики не определяют собственных специализированных версий класса DataSet.

3.2. Фундаментальные классы ADO.NET

ADO.NET имеет два типа объектов: основанные на соединении и основанные на содержимом [3]:

• объекты, основанные на соединении. Существуют объекты поставщика данных, такие как Connection, Command и DataReader. Они позволяют подключаться к базе данных, выполнять операторы SQL, перемещаться по результирующему набору, доступному только для чтения, и наполнять DataSet. Объекты, основанные на соединении, специфичны для типа источника данных

и находятся в специфичных для поставщика пространствах имен (таких как System. Data. SqlClient для поставщика SQL Server);

• объекты, основанные на содержимом Эти объекты в действительности лишь "упаковывают" данные. Они включают DataSet, DataColumn, DataRow, DataRelation и др. Они полностью независимы от типа источника данных и определены в пространстве имен System. Data.

Классы ADO.NET группируются в несколько пространств имен. Каждый поставщик имеет свое собственное пространство имен, а обобщенные классы вроде DataSet находятся в пространстве имен System.Data.

Поставщик ADO.NET — это просто набор классов ADO.NET (с реализацией Connection, Command, DataAdapter и DataReader), которые поставляются в сборке типа библиотеки классов. Обычно все эти классы поставщика имеют один и тот же префикс. Например, префикс OleDb применяется для поставщика OLE DB в ADO.NET, который предусматривает реализацию объекта Connection по имени OleDbConnection.

Класс Connection. Класс Connection позволяет устанавливать соединения с источником данных, с которым нужно взаимодействовать. Перед тем, как можно будет делать что-то еще (в том числе извлечение, удаление, вставка или обновление данных), понадобится установить соединение.

Хотя строки соединений варьируются в зависимости от реляционной СУБД и используемого поставщика данных, несколько фрагментов информации необходимы почти всегда:

• сервер, на котором размещается база данных. Если сервер базы данных всегда расположен на том же компьютере, что и приложение ASP.NET, поэтому вместо имени компьютера применяется псевдоним localhost;

• база данных, которую следует использовать;

• как база данных производит аутентификацию. Поставщики данных Oracle и SQL Server предоставляют возможность выбора — применить определенные учетные данные аутентификации либо подключиться как текущий пользователь системы. Последний вариант обычно более

предпочтителен, поскольку в этом случае не понадобится помещать информацию о пароле в код или конфигурационные файлы.

Для присоединения пользовательского экземпляра базы данных необходимо установить параметр User Instances в True (в строке соединения) и в параметре AttachDBFilename предоставить имя нужной базы данных. Значение Intital Catalog указывать не понадобится.

Вот пример строки соединения в случае применения такого подхода:

```
myConnection.ConnectionString =
 @"Data Source=localhost\SQLEXPRESS;" + "Integrated Security=SSPT;" +
@"AttachDBFilename=|DataDirectory|\имяБД.mdf;User Instance=True";
```

Здесь присутствует еще одна хитрость. Имя файла начинается с | DataDi rectory |. Это автоматически указывает на папку AppData внутри каталога веб-приложения. В таком случае не нужно задавать полный путь к файлу, который может перестать быть корректным после перемещения веб-приложения на веб-сервер. ADO.NET всегда будет искать файл по имени *имяБД*.mdf в каталоге AppData.

Пользовательские экземпляры — удобное средство, когда имеется веб-сервер, обслуживающий много веб-приложений с базами данных, которые часто добавляются и удаляются. Это средство также хорошо работает в сочетании с другими высокоуровневыми компонентами ASP.NET, такими как профили и система членства. По умолчанию эти компоненты создают файловые базы данных для SQL Server Express, которые сокращают работы по конфигурированию:

```
SqlClient     Integrated Security=true;  -- or -- Integrated Security=SSPI;
OleDb         Integrated Security=SSPI;
Odbc  Trusted_Connection=yes;
OracleClient          Integrated Security=yes;
```

Значение Integrated Security=true вызывает исключение при работе с поставщиком OleDb.

После выбора строки соединения управлять подключением очень легко — нужно просто использовать методы Open () и Close ().

Классы Command и DataReader [5]. Класс Command позволяет выполнить SQL-оператор любого типа. Хотя класс Command можно использовать для решения задач определения данных (таких как создание и изменение баз данных, таблиц и индексов), все же более вероятно его применение для выполнения задач манипулирования данными (вроде извлечения и обновления записей в таблице).

Специфичные для поставщика классы Command реализуют стандартную функциональность, в точности как классы Connection. В данном случае небольшой набор ключевых свойств и методов, используемых для выполнения команд через открытое соединение, определяется интерфейсом IDbConnection.

Прежде чем использовать команду, необходимо выбрать ее тип, установить ее текст и привязать к соединению. Всю эту работу можно выполнить, установив значения соответствующих свойств (CommandType, CommandText и Connection), либо передать необходимую информацию в аргументах конструктора.

Значения CommandType:

CommandType.Text – команда будет выполнять прямой оператор SQL. Оператор SQL указывается в свойстве CommandText. Это — значение по умолчанию;

CommandType.StoredProcedure – команда будет выполнять хранимую процедуру в источнике данных. Свойство CommandText представляет имя хранимой процедуры ;

CommandType.TableDirect – команда будет опрашивать все записи таблицы. CommandText — имя таблицы, из которой команда извлечет все записи. Эта опция предназначена только для обратной совместимости с

некоторыми драйверами OLE DB. Она не поддерживается поставщиком данных SQL Server и не работает так хорошо, как тщательно направленный запрос.

Фрагмент кода на C# имеет вид:

SqlCommand cmd = new SqlCommand(" *текст_запроса* ", *имя_соединения*) ;

Методы Command реализуются с помощью функций:

ExecuteNonQuery() – выполняет команды, отличные от select, такие как SQL-операторы вставки, удаления или обновления записей. Возвращаемое значение указывает количество строк, обработанных командой;

ExecuteNonQuery () –можно использовать для выполнения команд определения данных, которые создают, изменяют и уничтожают объекты базы данных (наподобие таблиц, индексов, ограничений и т.п.);

ExecuteScalar() – Выполняет запрос select и возвращает значение первого поля первой строки из набора строк, сгенерированного командой. Этот метод обычно применяется при выполнении агрегатной команды select, использующей функции вроде COUNT () или SUM () для вычисления единственного значения;

ExecuteReader() – Выполняет запрос SELECT и возвращает объект DataReader, который является оболочкой однонаправленного курсора, доступного только для чтения.

Класс DataReader. Класс DataReader позволяет читать данные, возвращенные командой SELECT, по одной строке за раз, в однонаправленном, доступном только для чтения потоке. Иногда это называют пожарным курсором. Использование DataReader — простейший путь получения данных, но ему недостает возможностей сортировки и связывания автономного объекта Data Set. Однако DataReader представляет наиболее быстрый способ доступа к данным.

Пример использования

protected void Page_Load(object sender, System.EventArgs e)

```
{
// Создать объекты Command и Connection.
string connectionString =
WebConfigurationManager.ConnectionStrings["Northwind"].ConnectionString
;
SqlConnection con = new SqlConnection(connectionString);
string sql = "SELECT * FROM Employees";
SqlCommand cmd = new SqlCommand(sql, con) ;
```

Соединение открывается, и команда выполняется методом ExecuteReader (), который возвращает Sql DataReader, как показано ниже:

```
// Открыть объект Connection и получить объект DataReader.
con.Open ();
SqlDataReader reader = cmd.ExecuteReader();
```

Получив DataReader, можно организовать цикл для прохождения по его записям, вызывая метод Read () в теле цикла. Этот метод перемещает курсор строки на следующую запись (при первом вызове — на первую строку). Метод Read () также возвращает булевское значение, означающее наличие последующих строк для чтения. В следующем примере цикл продолжается до тех пор, пока Read () не вернет false, после чего элегантно завершается.

Информация из каждой записи затем объединяется в одну длинную строку. Чтобы обеспечить быстрое выполнение манипуляций со строками, вместо обычных объектов строк используется StringBuilder (из пространства имен System.Text).

```
// Пройти в цикле по записям и построить HTML-строку.
StringBuilder htmlStr = new StringBuilder("");
while (reader.Read())
{
htmlStr.Append("<li>");
htmlStr.Append(reader["TitleOfCourtesy"]);
htmlStr.Append(" <b>");
```

```
htmlStr.Append(reader.GetStringA) ) ;
htmlStr.Append("</b>, ") ;
htmlStr.Append(reader.GetStringB) ) ;
htmlStr.Append(" - employee from ") ;
htmlStr.Append(reader.GetDateTimeF) .ToString("d") ) ;
htmlStr.Append("</li>") ;
}
```

Метод ExecuteNonQuery () выполняет команды, которые не возвращают результирующих наборов, такие как INSERT, DELETE или UPDATE. Метод ExecuteNonQuery () возвращает одну порцию информации — количество обработанных записей (или -1, если команда отлична от INSERT, DELETE или UPDATE).

Ниже показан пример использования команды DELETE с динамическим построением SQL-строки.

```
SqlConnection con = new SqlConnection(connectionString);
string sql = "DELETE FROM Employees WHERE EmployeelD = " + empID.ToString();
SqlCommand cmd = new SqlCommand (sql, con);
try
{
con.Open();
int numAff = cmd.ExecuteNonQuery();
HtmlContent.Text += string.Format("<br />Deleted <b>{0}</b> record(s)<br />",
numAff);
}
catch (SqlException exc)
{
HtmlContent.Text += string.Format("<b>Error:</b> {0}<br /><br />", exc.Message);
```

```
}
finally
{ con.Close();}
```

3.3. Расширенные элементы управления-контейнеры для работы с данными

Познакомитесь поближе с тремя наиболее мощными элементами управления данными: GridView, DetailsView, FormView и ListView [18].

Grid View — исключительно гибкий табличный элемент управления, предназначенный для демонстрации данных в виде двумерной сетки (grid), или экранной таблицы, состоящей из строк pi столбцов. Он включает в себя широкий диапазон встроенных средств, включая выделение, разбиение на страницы и редактирование. К тому же он может быть расширен с помощью шаблонов. Огромным преимуществом GridView перед DataGrid является его поддержка сценариев без кода. Используя GridView, можно без написания кода решать множество распространенных задач, таких как перемещение по страницам и выделение. В DataGrid для получения тех же средств нужно было кодировать обработку событий.

Свойство GridView AutoGenerateColumns по умолчанию имеет значение true. В этом случае GridView использует рефлексию для исследования объекта данных и нахождения полей (для записи) или свойств (для пользовательского объекта). Затем он создает столбцы для каждого из них в том порядке, в котором их обнаруживает.

Эта автоматическая генерация столбцов хороша для быстрого создания тестовых страниц, но не предоставляет необходимой гибкости, которая обычно требуется.

Если нужно скрыть столбцы, изменить порядок их следования или настроить некоторые аспекты их отображения, такие как форматирование и текст заголовков, то понадобится установить AutoGenerateColumns в false и

определить столбцы самостоятельно в разделе <Columns> дескриптора элемента управления GridView.

Типы столбцов, явно определяемых в GridView, имеют названия:

BoundField - Основной тип, представляющий отображаемые данные поля;

CheckBoxField - Столбец, представленный флажками для отображения логических данных типа bit;

HyperLinkField - Отображает содержимое полей в форме гиперссылки;

ImageField - Отображает графические данные из двоичного поля;

ButtonField - Отображает содержимое полей в форме кнопок;

CommandField - Представляет управляющий столбец с кнопками редактирования;

TemplateField- Специфицирует отображение множественных полей, настраиваемых элементов управления и произвольного HTML, используя шаблон. Обладает наибольшей гибкостью, но требует больших усилий в настройке.

Сортировка. Средства сортировки GridView позволяют переупорядочить результирующий набор строк GridView, щелкая на заголовке столбца. Это удобно, и это легко реализовать. Чтобы разрешить сортировку, нужно установить свойство GridView.AllowSorting в true. Далее понадобится определлть SortExpression для каждого столбца, который может быть отсортирован. Теоретически выражение сортировки может использовать любой синтаксис, который понимает элемент управления источником данных. На практике выражение сортировки почти всегда принимает форму, используемую в конструкции ORDER BY запроса SQL. Это значит, что выражение сортировки может включать единственное поле или список полей, разделенный запятыми, и с необязательным словом ASC или DESC, добавленным после имени столбца, которое позволяет сортировать в восходящем или нисходящем порядке.

Вот как определить столбец FirstName, чтобы строки сортировались в алфавитном порядке по имени:

```
<asp:BoundField DataField="FirstName11
```

HeaderText="First Name" SortExpression="FirstName"/>

Если два раза щелкнуть на заголовке столбца FirstName в строке, то первый щелчок отсортирует его по алфавиту в прямом порядке, а второй — в обратном.

Не все источники данных поддерживают сортировку.

Шаблоны. До сих пор в примерах элемент управления GridView использовался для отображения данных с помощью отдельно привязанных столбцов для каждого поля. Если требуется поместить множественные значения в одну ячейку или иметь неограниченные возможности настройки содержимого ячейки, добавляя HTML-дескрипторы и серверные элементы управления, то для этого нужно применять TemplateField. TemplateField позволяет определять полностью настраиваемый шаблон для столбца. Внутрь шаблона можно добавлять дескрипторы элементов управления, произвольные элементы HTML и выражения привязки данных.

Элементы TemplateField:

HeaderTemplate - определяет внешний вид и содержимое ячейки заголовка;

FooterTemplate - определяет внешний вид и содержимое ячейки нижнего колонтитула;

ItemTemplate - определяет внешний вид и содержимое каждой ячейки данных (если не используется AlternatingltemTemplate) или каждой нечетной ячейки (если используется);

AlternatingltemTemplate - используется в сочетании с ItemTemplate для различного форматирования четных и нечетных строк;

EditItemTemplate - определяет внешний вид и элементы управления, используемые в режиме редактирования;

InsertItemTemplate - определяет внешний вид и элементы управления, используемые при вставке новой записи.

DetailsView. Элемент, который выводит только одну запись в отличие от GridView, поддерживает разбиение на страницы.

В отличие от GridView, DetailsView позволяет вставлять записи. Для этого нужно установить значение свойства AutoGenerateInsert Button= "True". При отображении появится кнопка New. Ее нажатие переводит элемент в режим вставки, по умолчанию для каждого поля генерируются TextBox-ы.

Если источник данных для DetailsView — SqlDataSource, то у него должны быть определены свойство InsertCommand и набор параметров.

У DetailsView имеются пары событий, которые происходят при связывании с данными, при переходе из режима просмотра в режим вставки, при перелистывании страницы.

FormView. Еще один новый элемент FormView похож на DetailsView, но отличается от него тем, что нуждается в шаблоне для своего представления:

3.3. Пример оконного приложения с базой данных

Создать приложение Win32 "Абитуриент" для центра математического просвещения на факультете математики и информационных технологий.

Цель приложения – предоставить возможность хранения информации об одарённых детях региона, абитуриентах, учащихся, которые принимали участие в различных конкурсах, обработку данной информации, поиск записей по критериям, обеспечение связи средствами рассылки сообщений непосредственно через приложение.

Основными пользователями системы являются сотрудники центра математического просвещения.

Для построения проектируемой системы была выбрана технология ADO.NET – это набор классов (фреймворк) для работы с базами данных, а также XML файлами. Аббревиатура ADO расшифровывается как ActiveX Data Objects. Данная технология имеет методы и классы для извлечения и обработки данных.

Можно определить два типа архитектуры подключения:

1. Архитектура, подключенная к базе: Подсоединена к БД на протяжении всего рабочего времени.

2. Архитектура, не подсоединённая к БД: приложение, автоматически подключается/отключается в процессе работы. Приложения на такой архитектуре используют временные данные хранящиеся на стороне клиента (DataSet).

В проектируемой системе реализована двухуровневая модель архитектуры клиент-серверного приложения. В приложении можно выделить два программных уровня (слоя):

– уровень интерфейса;

– уровень работы с БД.

Приложение взаимодействует с базой данных Access через интерфейс пользователя. Выбранная база данных хороша тем, что для связи с компонентами базы данных Access не требует наличия локального сервера. Для полного функционирования приложения достаточно лишь наличие установленной оболочки Access на персональном компьютере, а она входит в стандарт MS Office.

Уровень интерфейса представляет собой набор компонентов для визуализации данных, возможности редактирования этих данных, и блок для рассылки сообщений (рис. 3.1).

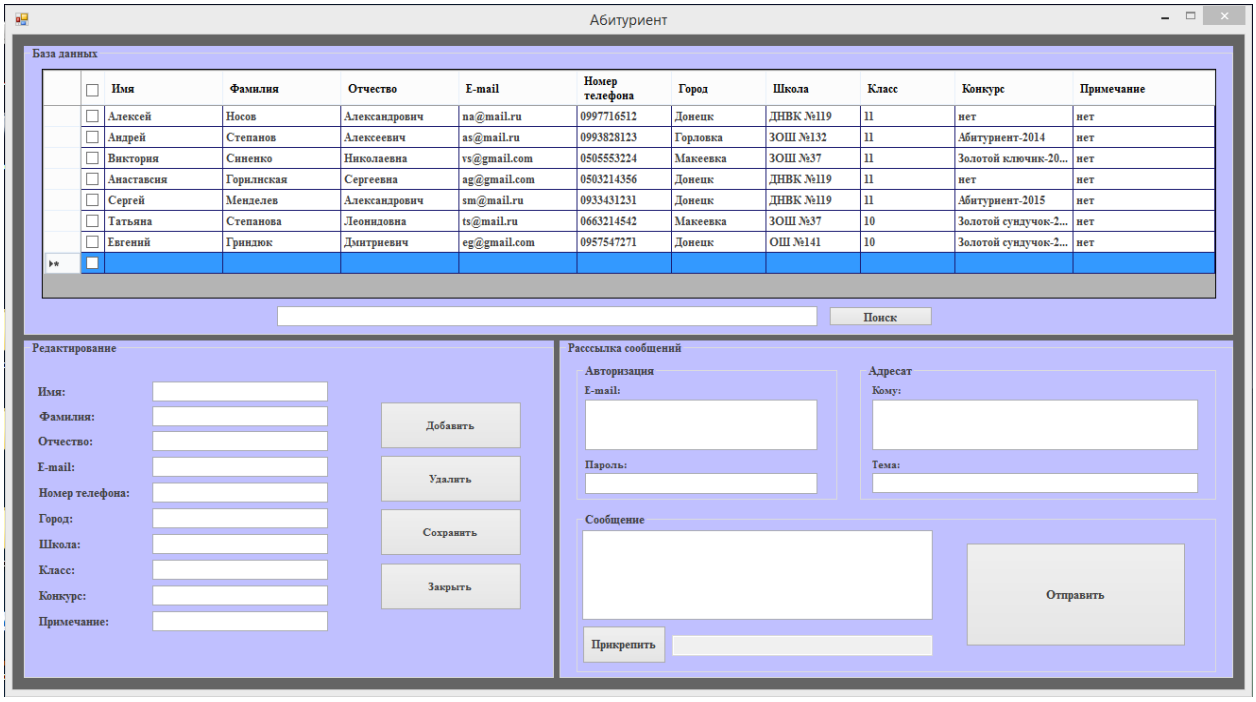

Рис. 3.1 - Интрефейс приложения «Абитуриент»

Для функционирования приложения "Абитуриент" была разработана и создана база данных (БД) Access, состоящая из одной таблицы - AbitDB, которая предусматривает информацию обо всех одарённых детях региона (табл. 3.1)

Таблица 3.1 - Таблица AbitDB

№ п/п	Наименование столбца	Тип данных	Назначение столбца
1.	Имя	Текстовый	содержит имя
2.	Фамилия	Текстовый	содержит фамилию
3.	Отчество	Текстовый	содержит отчество
4.	E-mail	Текстовый	содержит электронный адресс
5.	Номер телефона	Текстовый	содержит номер телефона
6.	Конкурс	Текстовый	содержит список конкурсов, в которых абитуриент, или учащийся принимал участие
7.	Школа	Текстовый	содержит информацию об учебном заведении
8.	Город	Текстовый	содержит город, в котором проживает абитуриент
9.	Класс	Текстовый	содержит класс, в котором учится одарённый ребёнок
10.	Примечание	Текстовый	содержит краткую информацию об абитуриенте

Рассмотрим фрагмент кода, в котором приложение подключается и взаимодействует с базой данных:

```
private void Form1_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу "abiturientDataSet1.AbitDB". При необходимости она может быть перемещена или удалена.
        this.abitDBTableAdapter1.Fill(this.abiturientDataSet1.AbitDB);
    }
    private void textBox1_TextChanged(object sender, EventArgs e){  }//Имя
```

```
    private void button2_Click(object sender, EventArgs e)
//Добавить новую запись
    {
       abitDBBindingSource1.AddNew();
    }
    private void button4_Click(object sender, EventArgs e) //Выйти
    {
       if (MessageBox.Show("Вы действительно хотите выйти?", "Выход",
MessageBoxButtons.YesNo, MessageBoxIcon.Question,
          MessageBoxDefaultButton.Button1) == DialogResult.Yes)
                { this.Close(); }
    }
    private void button5_Click(object sender, EventArgs e)//Удалить
    {
       if (MessageBox.Show("Вы действительно хотите удалить запись?",
"Удалить", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
          MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            { abitDBBindingSource1.RemoveCurrent(); }
    }
    private void button6_Click(object sender, EventArgs e)//Сохранить
    {
       abitDBBindingSource1.EndEdit();
       abitDBTableAdapter1.Update(abiturientDataSet1.AbitDB);      }
```

Данный фрагмент создаёт подключение к базе данных и осуществляет работу кнопок “Добавить”, “Удалить”, “Сохранить” и “Выйти”. Кнопки “Удалить” и “Выйти” имеют в себе обработчики событий для подтверждения действий удаления и выхода соответственно.

Далее рассмотрим, как осуществляется рассылка сообщений и прикрепление файлов:

```
private void button8_Click(object sender, EventArgs e)//Рассылка сообщений
    {
```

```
        try
        {
            SmtpClient client = new SmtpClient("smtp.mail.ru", 25);
            MailMessage message = new MailMessage();
            message.From =  new MailAddress(textBox9.Text); // от кого
            foreach (string recepient in textBox12.Text.Split(';')) // кому
            {
               message.To.Add(new MailAddress(textBox12.Text));
            }
            message.Body = textBox13.Text; //  текст сообщения
            message.Subject = textBox11.Text; // тема сообщения
            client.UseDefaultCredentials = false;
            client.EnableSsl = true;
            client.DeliveryMethod = SmtpDeliveryMethod.Network;
            if (textBox16.Text != "")
            {
               foreach (string Attachments in textBox16.Text.Split(';'))
// прикрепление
               {
                  message.Attachments.Add(new Attachment(textBox16.Text));
               }

            }
            client.UseDefaultCredentials = false;
            client.Credentials = new System.Net.NetworkCredential(textBox9.Text,
textBox10.Text); // авторизация пользователя
            client.Send(message);
            MessageBox.Show("Сообщение отправленно", "Отправленно",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            message = null;
        }
```

```
catch (Exception s)
    {
        MessageBox.Show(s.Message, "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
     }
```

Данный фрагмент кода реализовывает подключение к SMTP серверу, в качестве такого сервера был взят сервер “mail.ru”. Для рассылки сообщение, требуется в соответствующие поля ввести логин и пароль отправителя для авторизации, а в поле “Кому” – получателей. Стоит заметить, что получателей можно отбирать с помощью поиска по базе данных и поставив галочку на чекбоксе, который соответствует нужной записи, в поле “Кому” будет добавляться электронный адрес выбранного абитуриента (рис. 3.2).

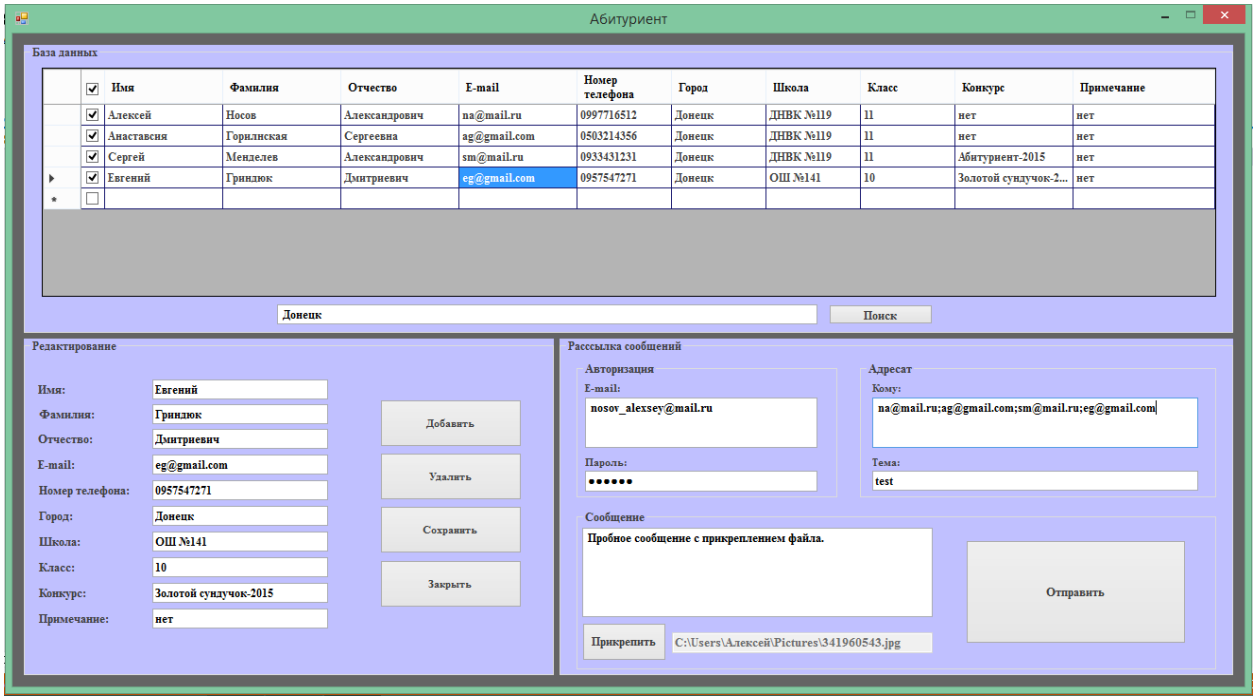

Рис. 3.2 - Отображение данных приложения «Абитуриент»

Для реализации прикрепления файлов был использован компонент для работы с диалоговыми окнами:

```
private void button7_Click_1(object sender, EventArgs e)
    {
```

```
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                textBox16.Text = openFileDialog1.FileName.ToString();
            }
        }
```

Данный код вызывает диалоговое окно (Проводник - рис.3.3)), для выбора файлов. Реализован множественный выбор файлов:

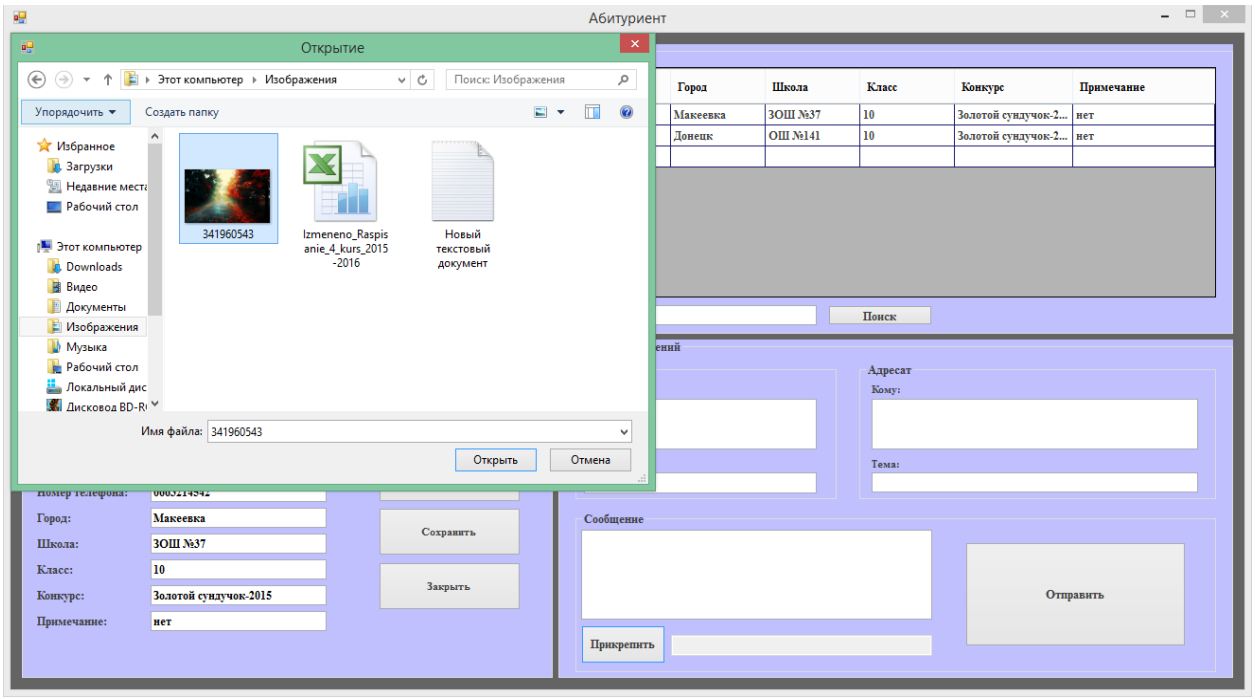

Рис. 3.3 - Интрефейс приложения «Абитуриент» с окном проводника

Далее рассмотрим реализацию поиска по базе данных:

```
private void button7_Click(object sender, EventArgs e)// Поиск
    {
      try
       {
         if (textSearch.Text == "")
             {
                MessageBox.Show("Введите критерий поиска");
             }
             else { string Search = textSearch.Text; }
```

```
                abitDBBindingSource1.Filter = "([Имя] LIKE '" + textSearch.Text
+ "') OR ([Фамилия] LIKE '" + textSearch.Text + "')" +
                "OR ([Отчество] LIKE '" + textSearch.Text + "') OR ([E-mail]
LIKE '" + textSearch.Text + "')" +
                "OR ([Номер телефона] LIKE '" + textSearch.Text + "') OR
    ([Город] LIKE '" + textSearch.Text + "')" +
                "OR ([Школа] LIKE '" + textSearch.Text + "') OR ([Класс] LIKE
    '" + textSearch.Text + "')" +
                "OR ([Конкурс] LIKE '" + textSearch.Text + "') OR
    ([Примечание] LIKE '" + textSearch.Text + "')";
            if (abitDBBindingSource1.Count != 0)
                {
                    dataGridView1.DataSource = abitDBBindingSource1;
                }
            else
                {
                    MessageBox.Show("Совпадений не найдено");
                    abitDBBindingSource1.Filter = null;
                    dataGridView1.ClearSelection();
                    dataGridView1.ReadOnly = true;
                    dataGridView1.MultiSelect = false;
                    dataGridView1.DataSource = abitDBBindingSource1;
                }
        }
        catch (Exception ex1)
        {
            MessageBox.Show(ex1.Message, "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
```

Данный фрагмент кода реализовывает поиск записей с помощью фильтра. С помощью такой конструкции возможен поиск по любому из доступных критериев, что очень удобно в использовании (рис. 3.4).

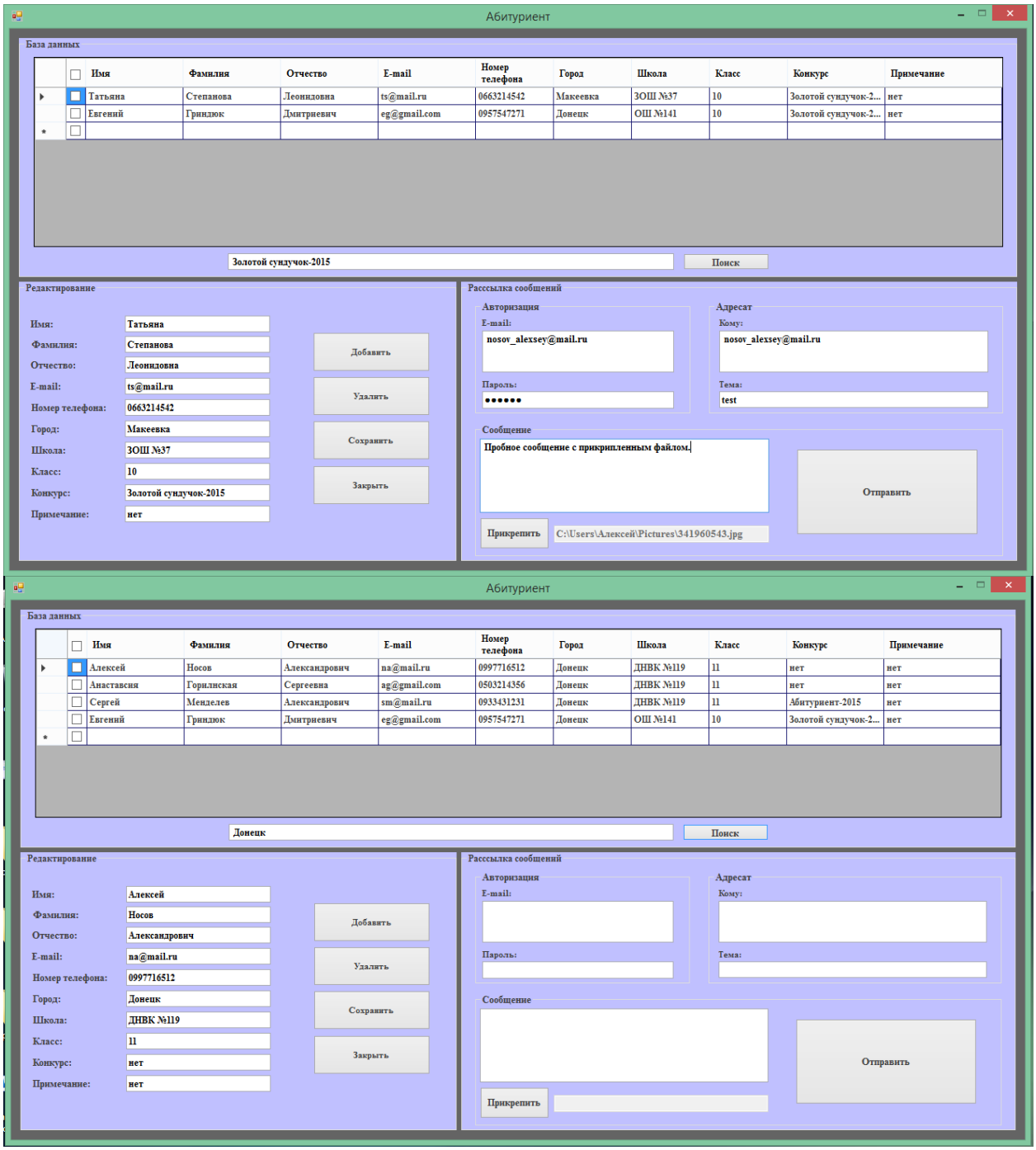

Рис. 3.4 - Использование критериев выбора приложения «Абитуриент»

Протестируем приложение и отправим пробное сообщение на почту (рис.3.5):

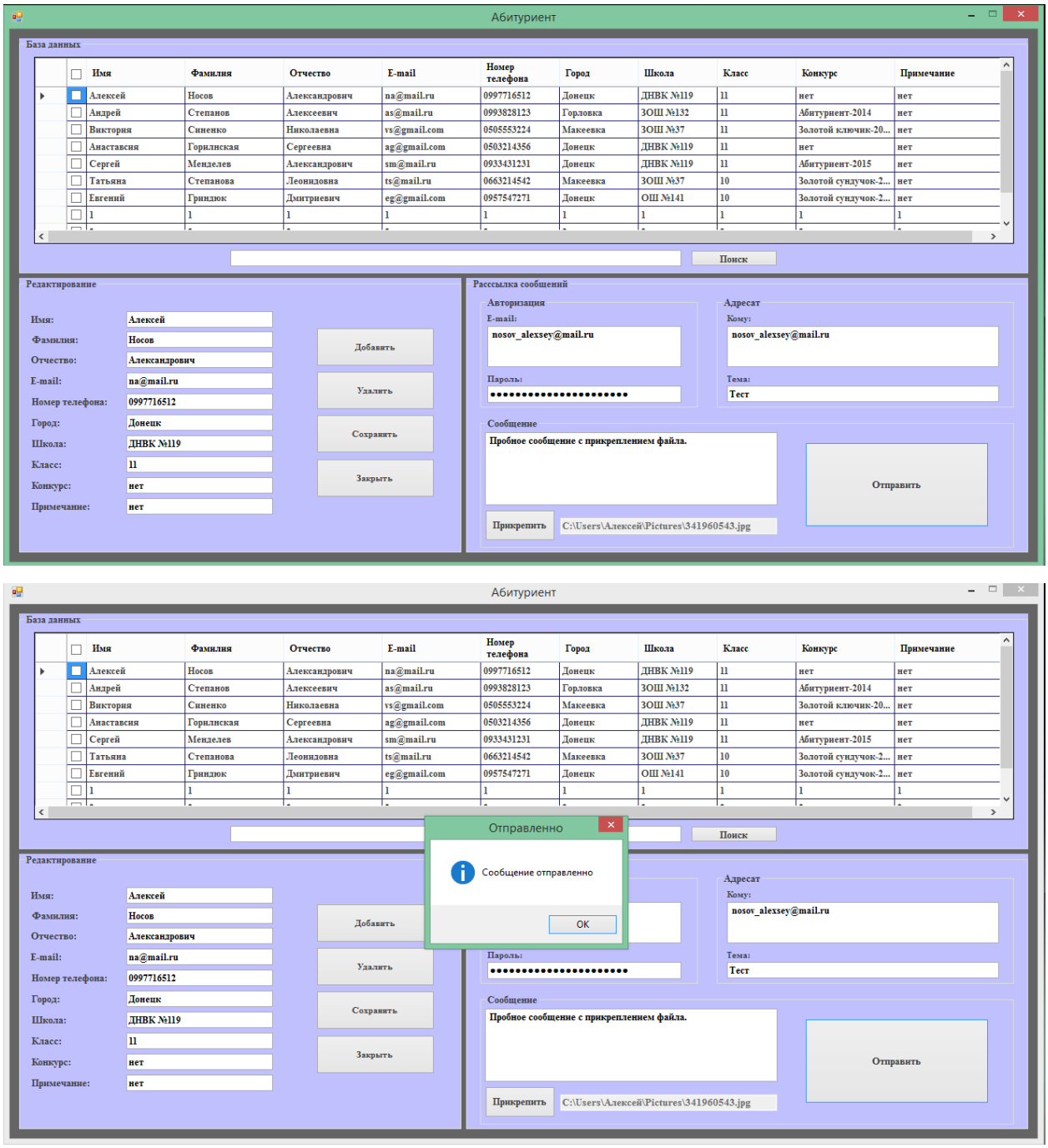

Рис. 3.5 - Отправка почты приложения «Абитуриент»

Сообщение отправлено. Приложение работает корректно.

3.4. Вопросы для подготовки к контролю знаний

1. Что чаще всего используется для хранения данных

а) СУБД

б) текстовый документ

в) CompareValidator

г) RangeValidator

2. В виде чего хранятся данные в СУБД

а) в виде таблиц

б) в виде страниц

в) в виде архивов

г) в виде строковых записей

3. Что представляет собой запись

а) строка таблицы

б) столбец таблицы

в) ячейка таблицы

г) поле в таблице

4. Что представляет собой поле

а) столбец таблицы

б) строка таблицы

в) ячейка таблицы

г) запись в таблице

5. Множество *таблиц* данных, связанных отношениями, составляют

а) базу данных

б) схему данных

в) представление

г) схему

6. Что из ниже перечисленных программ не является СУБД

а) Microsoft Power Point

б) Microsoft Access

в) Microsoft SQL Server 2000

г) FoxPro

7. Что представляет ADO.Net

а) набор классов для работы с внешними данными

б) набор классов для работы с внутренними данными

в) набор классов для работы с графикой

г) набор классов для обработки прерываний

8. Какой объект отвечает за соединение с базой данных

а) объект Connection

б) объект Command

в) объект DataAdapter

г) объект DataBase

9. Что является обязательным атрибутом объекта Connection

а) строка соединения

б) объект Command

в) объект DataAdapter

г) объект DataBase

10. Какое свойство объекта Connection хранит строку соединения с СУБД

а) ConnectionString

б) DataSource

в) Database

г) State

11. Что такое транзакция

а) последовательность команд, которая выполняется как одно целое

б) объект DataSource

в) объект Database

г) объект State

12. Что представляет собой класс DataSet

а) *представление* в памяти информации, считанной через ADO из баз данных или XML

б) последовательность команд, которая выполняется как одно целое

в) объект класса Database

г) объект класса State

13. Что обеспечивает класс DataAdapter

а) двусторонний обмен информацией между СУБД и приложением

б) последовательность команд, которая выполняется как одно целое

в) *представление* в памяти информации, считанной через ADO из баз данных или XML

г) объект класса State

14. Что делает объект Command

а) исполняет запрос *SQL*

б) обеспечивает двусторонний обмен информацией между СУБД и приложением

в) обеспечивает *представление* в памяти информации, считанной через ADO из баз данных или XML

г) создаёт транзакции - последовательность команд, которая выполняется как одно целое

15. Какая вкладка Visual Studio 2005 позволяет работать с соединениями баз данных

а) вкладка Server Explorer

б) вкладка Solution Explorer

в) вкладка Class Explorer

г) вкладка Project Explorer

16. В каком файле описываются строки соединения с СУБД

а) web.config

б) default.aspx

в) <Имя проекта>.mdf

г) < Имя проекта >.sln

17. Посредством какого свойства элементы управления связываются с элементом-источником

а) DataSourceID

б) DataTimeID

в) DataID

г) DataBindID

18. Для чего предназначены элементы-источники данных

а) для двустороннего обмена даными

б) для одностороннего обмена данными с сервера на клиент

в) для обмена данными между частями приложения

г) для обмена данными с ОС

19. Для чего нужно кэширование

а) увеличения эффективности работы с данными

б) для сжатия данных и передачи на сервер

в) для обмена данными между частями приложения

г) для обмена данными с ОС

20. В каком свойстве можно задать длительность кэширования:

а) CacheDuration

б) CacheExpiration Policy

в) DataSourceID

г) DataBindID

3.4. Практические задания

Методические рекомендации.

1. Изучите теоретический материал, изложенный в лекциях, в литературных источниках и информационных ресурсах по тематике задания.

2. Разработайте алгоритм выполнения задания.

3. Запишите алгоритм на языке программирования C#.

4. Подготовьте отчет о выполнении задания.

Задание 1. Разработать приложение содержащее 3 уровня: интерфейс пользователя, уровень данных и бизнес-логики. Бизнес-логику реализовать в виде класса, который содержит динамическую информацию о наличии автобусов в автобусном парке.

Сведения о каждом автобусе содержат:

- номер автобуса;
- фамилию и инициалы водителя;
- номер маршрута.

Задание 2. Необходимо обеспечить:

- начальное формирование данных в виде списка о всех автобусах;
- при выезде каждого автобуса из парка вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;
- при въезде каждого автобуса в парк вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте, и записывает эти данные в список автобусов, находящихся в парке;
- по запросу выдаются сведения об автобусах, находящихся в парке, или об автобусах, находящихся на маршруте.

Уровень данных реализовать с использованием ADO.Net.

Задание 3. Интерфейсы приложения реализовать в виде оконного приложения и веб-приложения (ASP.Net).

Задание 4. Сравнить интерфейсы приложения. Сформулировать отличия.

Тема 4. Технология LINQ для работы с данными

4.1. Введение в Linq

LINQ относится к одним из самых интересных средств языка C#. Аббревиатура LINQ означает Language-Integrated Query, т.е. язык интегрированных запросов. Это понятие охватывает ряд средств, позволяющих извлекать информацию из источника данных. Как вам должно быть известно, извлечение данных составляет важную часть многих программ. Например, программа может получать информацию из списка заказчиков, искать информацию в каталоге продукции или получать доступ к учетному документу, заведенному на работника. Как правило, такая информация хранится в базе данных, существующей отдельно от приложения. Так, каталог продукции может храниться в реляционной базе данных. В прошлом для взаимодействия с такой базой данных приходилось формировать запросы на языке структурированных запросов (SQL). А для доступа к другим источникам данных, например в формате XML, требовался отдельный подход.

LINQ дополняет C# средствами, позволяющими формировать запросы для любого LINQ-совместимого источника данных [10]. При этом синтаксис, используемый для формирования запросов, остается неизменным, независимо от типа источника данных.

Формальная структура запроса на LINQ имеет вид:

<переменная запроса> = from <переменная диапазона> in <источник данных> [where <условие, дающее логическое значение>]

[orderby <правило сортировки> [<направление сортировки>]] {select <возвращаемые данные> |

[group <возвращаемые данные> by <ключ группировки>]};

где <переменная запроса> – переменная, в которой хранится ссылка на правила запроса и с использованием которой можно будет выполнить запрос и получить

доступ к возвращаемым результатам. Должна быть одной из форм интерфейса IEnumerable<T>. Может быть объявлена как var; <переменная диапазона> – идентификатор, который используется внутри запроса для обращения к требуемым данным; <источник данных> – должен поддерживать интерфейсIEnumerable<T>. По <источнику данных> автоматически определяется тип <переменной диапазона>; <условие, дающее логическое значение> – условие, позволяющее из общего набора данных отобрать требуемые. Часто в условии используется <переменная диапазона>; <правило сортировки> – данные, по значениям которых будет осуществляться сортировка. Могут получаться на основе построения выражений. Часто использу ется <переменная диапазона>; <направление сортировки> – слова «ascending» или «descending», определяющие сортировку по возрастанию и убыванию, соответственно. По умолчанию используется сортировка по возрастанию, поэтому слово «ascending» можно не использовать; <возвращаемые данные> – данные, которые будут записаны в <переменную запроса>. Часто используется <переменная диапазона>; <ключ группировки> – данные, по которым будут осуществлена группировка. Часто задается с использованием<переменной диапазона>.

4.2. Использование средств языка XML в SQL

Производители баз данных формата XML высказывают в пользу своих продуктов те же аргументы, которые в свое время приводили производители объектно-ориентированных баз данных.

1. Поскольку огромное количество внешних данных представлено в формате XML, в базах данных удобнее всего использовать этот же формат и соответствующую модель данных.

2. Так как все большее количество пользователей осваивает HTML и XML, базы данных XML-формата также доступны для пользователей, как и реляционные базы данных SQL-типа.

Сравнение библиотек SYSTEM.XML и SYSTEM.XML. LINQ. Для начала, как и в случае с System.Xml, подключим необходимое пространство имен: using System.Xml.Linq.

Далее из множества классов выберем класс XDocument, создаем новый экземпляр этого класса и воспользуемся первым из двух подходов библиотеки Linq, заключающимся в использовании вложенных конструкторов.

```
XDocument doc = new XDocument(
        new XElement("Items",
            new XElement("item",
                new XAttribute("id", trackId++),
                new XElement("name", "One"),
                new XElement("name2", "First")
                );
```

Воспользуемся вторым способом создания Xml-докумета.

```
XDocument docx = new XDocument();
        XElement Items = new XElement("Items");
        docx.Add(Items);
        XElement Item = new XElement("Item");
        Item.Add(new XAttribute("id", 1));
        XElement name = new XElement("name");
        name.Value = "One";
        Item.Add(name);
        XElement name2 = new XElement("name2");
        name2.Value = "First";
        Item.Add(name2);
```

В библиотеке Linq возможно создание документа и запись в него данных несколькими способами, в отличие от библиотеки Xml, что ограничивает

разработчика и осложняет ему работу с библиотекой Xml. Также необходимо выделить способы создания, которые очень осложнены и запутаны в библиотеке Xml, в отличие от Linq. Заметим, что при создании новых объектов Xml-документа, библиотека Linq, в отличие от Xml, предусматривает автоматическое открытие и закрытие создаваемого объекта.

В библиотеке Xml при создании документа, необходимо создавать экземпляр класса XmlTextWriter, который является, по своей сути, лишь способом создания. Нельзя создать документ с помощью конструктора класса, а Linq, в свою очередь, позволяет это сделать, как с помощью конструктора класса, так и с помощью специального метода для создания документов.

4.3. Вопросы для подготовки к контролю знаний

1) С какими СУБД работает API Linq To Sql ?
а) Microsoft SQL Server
б) Oracle
в) MySQL
г) все вышеперечисленные
2) Как Linq to Sql абстрагирует физическую структуру базы данных?
а) в виде бизнес-объектов
б) в виде таблиц
в) в виде иерархической структуры
г) в виде реляционных отношений
3) Классом, устанавливающий соединение с базой данных, является
а) DataContext
б) SQLConnect
в) DataEntities
г) DataMember
4) Обновление базы данных осуществляется вызовом метода
а) SubmitChanges( )
б) ExecuteQuery( )

в) ExecuteTransaction( )

г) Close( )

5) Сущностные классы LinqToSql создаются

а) непосредственно программистом, либо средствами инструмента командной строки SQLMetal или Linq To Sql Designer

б) только программистом

в) утилитой командной строки SQL Metal

г) инструментом Object Relational Designer, входящим в состав Visual Studio 2010

6) Приведен код:

Northwind db = new Northwind(@”DataSource=.\SQLEXPRESS;InitialCatalog=Northwind”);

Customer cust = (from c in db.Customers where c.CustomerID == “LONEP” select c).Single<Customer>();

Вызовет ли его запуск немедленное выполнение запроса?

а) да, вызов операции Single() вызовет немедленное выполнение запроса

б) да, запросы Linq To Sql всегда выполняются немедленно

в) нет, запросы Linq To Sql всегда выполняются отложенно

г) нет, запросы с условием всегда выполняются во время чтения данных

7) Запросы Linq to Sql возвращают объект типа

а) IQueryable<T>

б) IEnumerable<T>

в) опеределенный самим программистом

г) Table<T>

8) Модель классов Linq To Sql содержит

а) модели таблиц, и ассоциации между ними

б) таблицы и ограничения

в) только таблицы

г) таблицы, триггеры, и хранимые процедуры

9) Приведен код:

```
Customer customer = (from c in db.Customers
where c.CompanyName == “Alfreds Futterkiste”
select c).Single<Customer>();
//db.OrderDetails.DeleteAllOnSubmit(
//        customer.Orders.SelectMany(o => o.OrderDetails));
//db.Orders.DeleteAllOnSubmit(customer.Orders);
db.Customers.DeleteOnSubmit(customer);
db.SubmitChanges();
```

Таблица Customers связа с таблицей Orders, а та в свою очередь с таблицей OrderDetails, вызовет ли выполнение этого кода удаление объектов, зависимых от родительского ?

а) нет, сначала нужно удалить все связанные данные из таблицы OrderDetailes, затем из Orders, и только потом можно удалить запись из таблицы Customers

б) да, все ассоциации и связанные данные удаляются автоматически – за этим следит механизм Linq To Sql

в) нет, после удаления данных из таблицы Customers нужно удалить данные из таблицы Orders, а затем из OrderDetails

г) да, об удалении соответствующих связанных данных позаботится СУБД

10) Переопределение методов вставки, обновления и удаления производится

а) программистом с помощью реализации частичных методов, либо с помощью Object Relational Designer

б) только программистом и только при определении классов Linq To Sql

в) только программистом, использую механизм частичных методов

г) утилитой командной строки SQL Metal.

4.4. Практические задания

Методические рекомендации.

1. Изучите теоретический материал, изложенный в лекциях, в литературных источниках и информационных ресурсах по тематике задания.

2. Разработайте алгоритм выполнения задания.

3. Запишите алгоритм на языке программирования C#.

4. Подготовьте отчет о выполнении задания.

Задание 1. Используя LINQ выполнить задания по обработке одномерных массивов: нахождение максимума, поиск одинаковых элементов, количество совпадений элементов. Создайте консольное приложение.

Задание 2. Для разработанного в теме 3 приложения реализовать доступ к данным ADO.Net, используя LINQ.

Разработать приложение содержащее 3 уровня: интерфейс пользователя, уровень данных и бизнес-логики. Бизнес-логику реализовать в виде класса, который содержит динамическую информацию о наличии автобусов в автобусном парке.

Сведения о каждом автобусе содержат:

- номер автобуса;
- фамилию и инициалы водителя;
- номер маршрута.

Задание 2. Необходимо обеспечить:

- начальное формирование данных в виде списка о всех автобусах;
- при выезде каждого автобуса из парка вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;
- при въезде каждого автобуса в парк вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся

на маршруте, и записывает эти данные в список автобусов, находящихся в парке;

- по запросу выдаются сведения об автобусах, находящихся в парке, или об автобусах, находящихся на маршруте.

Уровень данных реализовать с использованием базы данных SqLite и LINQ.

Задание 4. Интерфейсы приложения реализовать в виде оконного приложения и веб-приложения (ASP.Net).

Задание 5. Сравнить интерфейсы приложения. Сформулировать отличия технологии ADO.Net и LINQ.

5. Темы творческих заданий

1. Технология LINQ и привнесенные ей добавления в язык: лямбда-функции, анонимные типы, методы-расширения (extension methods), инициализаторы классов (class initializers).
2. Технология LINQ и ее связь с реляционными базами данных.
3. Пользовательские элементы управления в ASP.NET.
4. Система построения приложений WPF (Windows Presentation Foundation).
5. Технология Parallel LINQ и использование на практике при параллельном программировании.
6. Новые возможности в C# 7 и их преимущества перед предыдущими версиями.
7. Реализация расширенной многопоточности на C#.
8. Использование интерфейсов IEnumerable и IEnumerator с обобщениями.
9. Контейнеры пользовательского интерфейса.
10. Объект DataSet: несколько таблиц, связи между таблицами.
11. Мастер-страницы в приложениях ASP.Net с элементами аутентификации, определения нескольких пользовательских тем, использованием карты сайта.
12. Сетевые возможности C#: сервер, клиент, URL адреса, сокеты. И их использование в приложениях.
13. Библиотеки для реализации параллельного программирования в C#.
14. Технология LINQ to Object с приведением различных типов объектов.
15. Реализация собственных классов обобщений.
16. Концепция сериализации.

СПИСОК ЛИТЕРАТУРЫ

Основная

1. Биллиг В. А. Основы программирования на C# : учеб. пособие / В.А. Биллиг. - М. : Интернет-ун-т информ. технологий : Бином. Лаб. знаний, 2006.- 483 с. АУЛ(6), АНЛ(1), Чз1(1)

2. Кариев Ч. А. Разработка Windows-приложений на основе Visual C# : учеб. пособие / Ч.А. Кариев. - М.: Интернет-Ун-т информ. технологий: БИНОМ. Лаб. знаний, 2007. - 767 с. + электрон. опт. диск (CD-ROM). АНЛ(1), Чз1(1)

3. Кариев Ч.А. Технология Microsoft ADO. NET: учеб. пособие / Ч.А. Кариев. - М. : Интернет-Ун-т информ. технологий : БИНОМ. Лаб. знаний, 2007.- 543 с. АУЛ(1), АНЛ(1), Чз1(1)

4. Клаус М. Язык программирования C#: Лекции и упражнения / М.Клаус.- СПб: ООО "ДиаСофтЮП", 2002.- 636 с.

5. Кузнецов С. Д. Базы данных: модели и языки : учеб. пособ. для студентов вузов, обучающихся по специальности " Прикладная математика и информатика" и "Информационные технологии" / С. Д. Кузнецов. - М. : Бином, 2008. - 720 с. Всего 50: АУЛ(49), АНЛ+Чз1(1)

6. Кулямин В. В. Технологии программирования. Компонентный подход: учеб. пособие / В. В. Кулямин. - М. : Интернет-ун-т информ. технологий : Бином. Лаб. знаний, 2007. - 463 с. АУЛ (3), АНЛ (1), Чз1 (1)

7. Мак-Дональд М. Microsoft ASP.NET 3.5 с примерами на C# 2008 для профессионалов / Мэтью Мак-Дональд, Марио Шпушта ; [пер. с англ. Я.П. Волковой и др.]. - 2-е изд. - Москва [и др.] : Вильямс, 2008. - 1420 с. + 1 электрон. опт. диск (CD-ROM). АНЛ (1), Чз1(1)

8. Марченко А. Л. Основы программирования на C# 2.0 : учеб. пособие / А.Л. Марченко.- Москва: Интернет-Ун-т информ. технологий: БИНОМ. Лаб. знаний, 2007.- 551 с. АУЛ(1), АНЛ(1), Чз1(1)

9. Прайс Д. Visual C# .NET: полное руководство / Д. Прайс, М. Гандэрлой. - Киев: Век+, 2011. - 957 с. Чз1(1)

10. Раттц-мл. Дж. С. LINQ: язык интегрированых запросов C#2008 для профессионалов / Джозеф Раттц-мл.- Москва: Вильямс, 2008. - 549 с. Чз1 (1).

11. Троелсен Э. C# и платформа .NET: Пер. с англ. / Э. Троелсен.- М. и др. : Питер, 2004. - 796 с. Чз1(1)

12. Фленов М.Е. Библия C#. - 3- е изд., перераб. и доп. / М.Е. Фленов.- СПб.: БХВ-Петербург, 2016.- 544 с.

13. Шарп Дж. Microsoft Visual C#. Подробное руководство. 8-е изд. / Джон Шарп.- СПб.: Питер, 2017. — 848 с.

14. Шилдт Г. C# 2.0 : полное руководство : классическое справочное руководство для версии языка C# 2.0, обновлен. и доп.: [пер. с англ.] / Г. Шилдт. - М. : ЭКОМ, 2007. - 961 с. Чз1(1)

15. Шилдт Г C# 4.0: полное руководство. – М.: ООО "И.Д. Вильяме", 2011. – 1056 с.

16. Эвери Дж. Microsoft ASP.NET : конфигурирование и настройка : [пер. с англ.] / Джеймс Эвери. - М. : БИНОМ. Лаб. знаний : СП ЭКОМ, 2005. - 269 с. АНЛ(1), Чз1(2)

Дополнительная

17. Евсеева О.Н., Шамшев А.Б. Основы языка C# 2005: Учебное пособие. - Ульяновск: УлГТУ, 2008. - 132 с. http://window.edu.ru/resource/863/58863

18. Евсеева О.Н., Шамшев А.Б. Работа с базами данных на языке C#. Технология ADO .NET: Учебное пособие. - Ульяновск: УлГТУ, 2009. - 170 с. http://window.edu.ru/resource/225/65225

19. Мак-Дональд М. Microsoft ASP.NET 4 с примерами на C# 2010 для профессионалов / Мак-Дональд М., Фримен Ад., Шпушта М. – М. : ООО "И.Д. Вильяме", 2011. – 1424 с.

20. Оньон Ф. Основы ASP.NET с примерами на C# / Оньон Ф.– М., 2003.– 304 с.

21. Петцольд Ч. Программирование для Microsoft Windows на C#: В 2-х т. / Ч.Петцольд.- М.: Издательско-торговый дом "Русская редакция", 2002.- 624 с.

22. Практикум по курсу "Объектно-ориентированное программирование" на языке C#: Учебное пособие / А.А. Андрианова, Л.Н. Исмагилов, Т.М. Мухтарова. - Казань: Казанский (Приволжский) федеральный университет, 2012. - 112 с. http://window.edu.ru/resource/949/79949

23. C# 4.0 и платформа .NET 4 для профессионалов / Кристиан Нейгел, Билл Ивьен, Джей Глинн, Карли Уотсон, Морган Скиннер.: Пер. с англ.- М. : ООО "И.Д. Вильяме", 2011. — 1440 с.

24. Скит Дж. C# для профессионалов. Тонкости программирования / Дж.Скит.- СПб.: Вильямс, 2014.- 608 с.

Информационные ресурсы

1. https://studfiles.net/preview/6211355/page:37/

2. http://nullpro.info/2013/samouchitel-po-c-dlya-nachinayushhix-01-osnovy-yazyka-peremennye-logika-cikly/

3. https://msdn.microsoft.com/ru-ru/library/bb399349(v=vs.110).aspx

4. http://codingcraft.ru/c_sharp_coding/auxiliary/linq.php

5. https://metanit.com/sharp/tutorial/15.1.php

УЧЕБНОЕ ИЗДАНИЕ

Рекомендовано Ученым советом
ГОУ ВПО «Донецкий национальный университет»
(протокол № 10 22.12.2016 г.)

ПРАКТИЧЕСКИЙ КУРС СОВРЕМЕННЫХ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Авдюшина Елена Владимировна

для студентов направления подготовки

01.02.04 Прикладная математика и информатика

Авторская верстка
Компьютерный дизайн: Е.В. Авдюшина

Адрес издательства:
ГОУ ВПО «Донецкий национальный университет»,
ул. Университетская, 24. г. Донецк, 283055

Подписано в печать 22.12.2016 г.
Формат 60×84/16. Бумага офисная.
Печать – цифровая. Усл.-печ. л. 10,5.
Тираж 100 экз. Заказ № 838 – декабрь 16.
Донецкий национальный университет
283001, г. Донецк, ул. Университетская, 24.
Свидетельство про внесение субъекта
издательской деятельности в Государственный реестр
серия ДК № 1854 от 24.06.2004г.