Федеральное государственное образовательное бюджетное учреждение
высшего образования
**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

Калужский филиал Финуниверситета

Факультет «Экономика и бизнес - технологии»
Кафедра «Бизнес – информатика и информационные технологии»

КОМПЬЮТЕРНЫЙ ПРАКТИКУМ

Методические указания по выполнению практических занятий
для студентов, обучающихся по направлению 38.03.05 Бизнес - информатика
профиль «ИТ – менеджмент в бизнесе»
очная и заочная форма обучения

Одобрено кафедрой «БИ и ИТ»
(протокол № 13 от 26 июня 2018 г.)

Калуга 2018

Методические указания разработал:
Кандидат физико – математических наук, доцент С.В. Пономарев

Методические указания по выполнению практических занятий по дисциплине «Компьютерный практикум» предназначены для студентов, обучающихся по направлению подготовки 38.03.05 Бизнес - информатика, профиль «ИТ – менеджмент в бизнесе» по очной и заочной форме обучения.

В методических указаниях излагаются требования к структуре и содержанию практических занятий, представлены рекомендации по написанию практической работы, варианты практических заданий.

Содержание

Введение

Практические работы предназначен для обучения студентов практическому программированию на языке C#.
Базовым уровнем работы студентов на практических занятиях является знание математики, черчения (рисования), информатики. Все работы объединены единым подходом, основанным на разработке программного обеспечения на языке C# типовых бизнес – процессов создания, редактирования и работы с документами.

Основной задачей является:

- получение навыков самостоятельной работы со средой MS VisualStudio на персональных компьютерах;
- обучение программированию на языке C#;
- знание и умение выполнять сборку программ.
-

1. Методические указания по подготовке и выполнению практических занятий

1.1 Структура практических занятий

Перегрузка методов

При определении методов какого-нибудь класса в программах необходимо указать тип возвращаемого методом значения, а также количество параметров и тип каждого из них.

Например, программистом была разработана функция с именем *sum(),* которая суммировала два целых значения. Если требуется использовать подобную функцию для сложения трех целых значений, следует создать функцию с другим именем. Аналогично если требуется использовать подобную функцию для сложения значений типа *float,* то необходимо еще одна функция с еще одним именем.

Чтобы избежать дублирования функций, язык C# позволяет определять несколько функций с одним и тем же именем. В процессе компиляции C# принимает во внимание количество аргументов, используемых каждой функцией, и затем вызывает именно требуемую функцию. Предоставление компилятору выбора среди нескольких функций называется **перегрузкой.** Перегрузка является одним из способов ***реализации полиморфизма*.**

Перегрузка методов позволяет использовать одно и то же имя для нескольких функций с разным количеством или разным типом параметров.

Перегрузка методов – это один из способов, которым достигается полиморфизм в языке C#. Две и более функции могут иметь одно и то же имя, а отличаться набором аргументов в интерфейсе (описании).

Полиморфизм – позволяет использовать один и тот же интерфейс при реализации целого круга различных действий.

Наиболее распространенным видом перегрузки методов является перегрузка конструкторов в классе.

Перегрузка операций

Как известно, в языке C# тип переменной определяет набор значений, которые она может хранить, а также набор операций, которые можно выполнять над этой переменной. Например, над значением переменной типа int программа может выполнять сложение, вычитание, умножение и деление. С другой стороны, использование оператора “плюс” для сложения двух экземпляров реализованного программистом класса лишено смысла.

Когда в программе определяется класс, то по существу определяется новый тип данных. Тогда язык C# позволяет определить операции, соответствующие этому новому типу данных.

Перегрузка операций состоит в изменении смысла операции при использовании его с определенным классом.

Например, пусть имеется:

myclass a,bc;…//a,b,c-экземпляры класса myclass

c=a+b; //*перегруженная операция* сложения для класса myclass

Перегрузка операций обычно применяется для классов, описывающих математические или физические понятия, то есть таких классов, для которых требуется выполнить соответствующие операции.

Общий синтаксис объявления перегруженной операции:

[атрибуты] спецификаторы operator тело операции,

где:

Спецификаторы – public,static,extern

operator – ключевое слово, определяющее перегруженную операцию

тело операции-действия, которые выполняются при использовании операции в выражении

Перегружать можно только стандартные операции.

Алгоритм перегрузки операции:

1. Определить класс, которому данная операция будет назначена.
2. Для перегрузки операций используется ключевое слово **operator.**
3. Переопределяя операцию, необходимо указать метод, который C# вызывает каждый раз, когда класс использует перегруженную операцию. Этот метод, в свою очередь, выполняет соответствующую операцию.

Правила перегрузки операции:

1. Операция должна быть объявлена как public static
2. Параметры в операцию должны передаваться по значению (не ref, не out)
3. Двух одинаковых перегруженных операций в классе не должно быть

Если программа перегружает операцию для определенного класса, то смысл этой операции изменяется *только для указанного класса*, оставшаяся часть программы будет продолжать использовать эту операцию для выполнения ее стандартных действий.

Перегрузка унарных операций

К унарным операциям, которые можно перегружать в языке C# относятся:

- унарные + и –
- логическое !,
- ++,--
- true, false – обычно перегружаются для типов SQL

Синтаксис объявления перегруженной унарной операции:

public static тип_возвр_знач operator унарная_операция (один параметр),

где параметр – это класс, для которого перегружается данная операция

Например,

```
public static myclass operator ++(myclass x)
public static int operator +(myclass x)
public static bool operator  true(myclass x)
```

Перегруженная операция возвращает:

- унарные + и –, ! величину любого типа
- ++,-- величину типа класса
- true, false – величину типа bool

Префиксные и постфиксные ++ и – не различаются при перегрузке.

Пример перегрузки унарных операций на примере класса “Одномерный массив”

```
class MyArray
{
public MyArray(int size)
{
length =size;
a=new int[length];
}
public MyArray(params int [] mas)
{
length =mas.length;
a=new int[length];
for (int i=0;i<length;i++)
a[i]=mas[i];
}
public static MyArray operator ++(MyArray x) //перегрузка унарного
оператора ++
{
MyArray temp=new MyArray(x.length);
for (int i=0;i<length;i++)
temp[i]=++x.a[i]; //попробуйте temp.a[i]=++x.a[i]
return temp;
}
//индексатор, в случае выхода за рамки массива – генерируется
исключение!
public int this [int i]
{
get {if (i>=0  && i<length) return a[i]; else throw new
IndexOutOfRangeException();}
set { if (i>=0  && i<length) a[i]=value; else throw new
IndexOutOfRangeException();}
}
public void Print(string name)
{
Console.WriteLine(name+”:”);
for (int i=0;i<length;i++)
```

```
Console.WriteLine(“\t”+a[i]);
Console.WriteLine();
}
public void Enter()
{
//в цикле - ввод элементов массива – реализуйте сами!
}
//данные класса – сам массив и его размерность
int [] a;
int length;
}
…Main()
{
    try
    {
    MyArray a1=new MyArray(5,2,-1,1,-2);
    a1.Print(“Первый массив ”);
    a1++; //теперь к экземпляру класса можно применить операцию
++
    a1.Print(“Использование операции ++ для всех элементов массива
”);
    MyArray a2=new MyArray(5);
    a2.Enter();
    a2.Print(“Второй массив ”);
    a2++;
    a2.Print(“Использование операции ++ для всех элементов
массива”);
    }
    catch (Exception e)
    {Console.WriteLine(e.Message);}
}…
```

.

1.2 Требования к содержанию и оформлению практических занятий

Делегат – особый вид класса, хранящий ссылки на методы. Делегат – класс (ссылочный тип), инкапсулирующий (содержащий в себе) метод с указанной сигнатурой и возвращаемым типом.

Все делегаты являются объектами типа System.Delegate или System.MulticastDelegate.

Общий синтаксис объявления делегата:

[спецификаторы] delegate *тип возвращаемого значения метода* имя делегата ([параметры]), где

delegate – ключевое слово.

тип возвращаемого значения – это тот тип, который будет возвращать функция, на которую ссылается делегат.

параметры – это необязательные параметры, которые будут присутствовать у функции, на которую ссылается делегат.

Примеры объявления делегата:

```
delegate void MyDelegate(string s); //делегат, который может работать для всех методов, которые возвращают тип void и имеют один строковый параметр.

public delegate int mydel();

delegate double MyD (double x);

//”Почти универсальный делегат”

delegate void My (object o)
```

Алгоритм работы с делегатом:

1. Объявление делегата;

2. Определение метода, “подходящего” для работы с делегатом
3. В основной программе создание экземпляра делегата и связь его с определённой функцией;
4. Вызов метода через делегат.

Делегат может быть применён:

- Для обычной функции без класса;
- Для метода внутри класса.

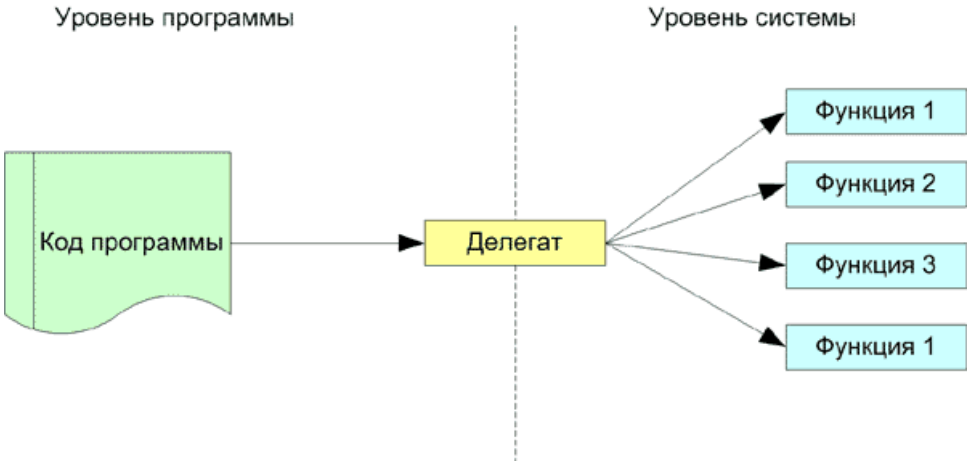
Пример делегата для обычной функции

```
class Program
  {
    //объявление делегата
    delegate void MyDelegate(string s);
    static void Myfunc(string s)
    {
      Console.WriteLine(s);
    }
    static void Main(string[] args)
    {
      //в конструкторе передается ссылка на функцию, которая делегируется (связыватся с делегатом)
      MyDelegate del = new MyDelegate(Myfunc);//создание экземпляра делегата
      del("Hello World");//вызов функции через делегат
      Console.ReadKey();
    }
  }
```

2. Определение варианта практического занятия

Делегаты используются для получения возможности определять вызываемый метод не при компиляции, а во время выполнения программы (динамически). То есть делегат может связываться не с одним методом, а с несколькими (экземпляры делегата могут содержать несколько ссылок на методы).

Благодаря этому, можно подсоединять к одному делегату несколько методов, каждый из которых при единственном обращении к делегату будет вызваться по цепочке. Таким образом, из программы будет виден лишь один делегат, за которым скрывается несколько методов (рисунок).



Delegate.Invoke или что там внутри? (для продвинутых)
Метод Invoke недокументирован. Он не является членом классов Delegate и MulticastDelegate. Соответственно, можно предположить, что это специальный метод, генерируемый компилятором. Его изучение показывает, что метод лишен какого-либо кода. Он попросту пустой и ничего не может делать. Так может показаться, если не принимать во внимание спецификатор runtime, используемый при определении метода. Данный спецификатор указывает на то, что метод будет реализован по ходу исполнения программы самой средой исполнения. Это означает, что код, обрабатывающий вызов метода, располагается в недрах самой среды исполнения. А она прекрасно осведомлена о внутреннем устройстве делегата и справится с вызовом всех его методов без дополнительной помощи со стороны программиста.
Код, реализующий делегат, также находится внутри среды исполнения. Оказывается, метод Invoke отвечает не только за вызов делегата, но

также за хранение информации о прототипе методов, на которые может ссылаться делегат. Прототип закодирован в самом методе Invoke, то есть его прототип полностью совпадает с прототипом делегата.

MulticastDelegate.GetInvocationList

Возвращает список делегатов, находящихся в списке вызовов делегата.

public sealed override Delegate[] GetInvocationList();

Существует два недостатка механизма вызова функций, связанных с делегатом. И оба они связаны со случаем, когда в списке вызова делегата присутствует более одной функции.

- *Функции, на которые ссылаются делегаты, могут возвращать значения. Но мы их получить не сможем, поскольку штатная функция вызова Invoke возвращает значение, которая вернула последняя из опрошенных функций.*
- *При обращении к делегату, содержащему в списке вызова несколько функций, вовсе не гарантируется, что все функции из списка будут вызваны. Если одна из них сгенерирует исключение, то работа метода Invoke будет прервана и остальные функции вызваны не будут.*

Пример делегата для работы с несколькими методами

```
namespace ConsoleApplication1
{
    class Program
    {
        //объявление делегата
        delegate void MyDelegate(string s);
        // Первый метод, на который мы будем
ссылаться при помощи
        // делегата. Именно он будет вызывать
исключение,
        // не позволяющее обратиться ко второму
методу.
 static void f1(string s)
```

```
  {
    // Выведем значение переданного параметра,
    // а также уведомим пользователя о том, что данный
    // метод был вызван.
    Console.WriteLine("Функция 1 вызвана с параметром = {0}",s);
    // Преднамеренно выбросим исключение.
    throw new Exception();
  }
 // Второй метод, на который мы будем ссылаться из делегата.
  static void f2(string s)
  {
    // Сообщим пользователю о том, что метод
    // был вызван, а также выведем  значение переданного параметра.
    Console.WriteLine("Функция 2 вызывана с параметром = {0}",s);
  }
        static void Main(string[] args)
        {

                MyDelegate del = new MyDelegate(f1);
               // Присоединим к нему еще одну функцию.
               del += new MyDelegate(f2);
```

```
                // Последовательно пройдем по каждому
делегату, входящему
                // в список вызова ранее созданного
делегата.
                foreach (MyDelegate d in
del.GetInvocationList())
                {
                  // вызов функции  - в защищенный
блок!
                   try
                  {
                    d("Hello");
                  }
                  // Это блок обработки исключений,
произошедших
                  // в защищенном блоке.
                  catch(Exception ex)
                  {
                    // Сообщим пользователю о том,
что при попытке
                    // вызова одной из функций
произошло исключение.
                    Console.WriteLine("Oh mama, была
обнаружена исключительная ситуация!");
                  }
                }
            Console.ReadKey();         }     }}
```

3. Варианты тем практических занятий

Составить программу для ввода, вывода и обработки заданной структуры данных.

№1

В магазине формируется список лиц, записавшихся на покупку товара повышенного спроса. Каждая структура этого списка содержит: порядковый номер, Ф.И.О., домашний адрес покупателя и дату постановки на учет. Вывести список лиц в порядке очереди по датам постановки на учет.

№2

Список товаров, имеющихся на складе, включает в себя наименование товара, количество единиц товара, цену единицы и дату поступления товара на склад. Вывести список товаров, хранящихся больше месяца, стоимость которых превышает 1000000 руб.

№3

Для получения места в общежитии формируется список студентов, который включает Ф.И.О. студента, группу, средний балл, доход на члена семьи. Общежитие в первую очередь предоставляется тем, у кого доход на члена семьи меньше двух минимальных зарплат, затем остальным в порядке уменьшения среднего балла. Вывести список очередности предоставления мест в общежитии.

№4

В справочной автовокзала хранится расписание движения автобусов. Для каждого рейса указаны его номер, тип автобуса, пункт назначения, время отправления и прибытия. Вывести информацию о рейсах, которыми можно воспользоваться для прибытия в пункт назначения раньше заданного времени.

№5

На междугородной АТС информация о разговорах содержит дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента. Вывести по каждому городу общее время разговоров с ним и сумму.

№6

Информация о сотрудниках фирмы включает: Ф.И.О., табельный номер, количество проработанных часов за месяц, почасовой тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. Вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12% от суммы заработка.

№7

Информация об участниках спортивных соревнований содержит: наименование страны, название команды, Ф.И.О. игрока, игровой номер, возраст, рост, вес. Вывести информацию о самой молодой, рослой и легкой команде.

№8

Для книг, хранящихся в библиотеке, задаются: регистрационный номер книги, автор, название, год издания, издательство, количество страниц. Вывести список книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.

№9

Различные цехи завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают: наименование, количество, номер цеха. Для заданного цеха необходимо вывести количество выпущенных изделий по каждому наименованию в порядке убывания количества.

№10

Информация о сотрудниках предприятия содержит: Ф.И.О., номер отдела, должность, дату начала работы. Вывести списки сотрудников по отделам в порядке убывания стажа.

№11

Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: Ф.И.О., адрес, оценки. Определить количество абитуриентов, проживающих в г. Минске и сдавших экзамены со средним баллом не ниже 4.5, вывести их фамилии в алфавитном порядке.

№12

В справочной аэропорта хранится расписание вылета самолетов на следующие сутки. Для каждого рейса указаны: номер рейса, тип самолета, пункт назначения, время вылета. Вывести все номера рейсов, типы самолетов и времена вылета для заданного пункта назначения в порядке возрастания времени вылета.

№13

У администратора железнодорожных касс хранится информация о свободных местах в поездах дальнего следования на ближайшую неделю в следующем виде: дата выезда, пункт назначения, время отправления, число свободных мест. Оргкомитет международной конференции обращается к администратору с просьбой зарезервировать *m* мест до города *N* на *k-й* день недели с временем отправления поезда не позднее *t* часов вечера. Вывести время отправления или сообщение о невозможности выполнить заказ в полном объеме.

№14

Ведомость абитуриентов, сдавших вступительные экзамены в университет, содержит: Ф.И.О. абитуриента, оценки. Определить средний балл по университету и вывести список абитуриентов, средний балл которых выше среднего балла по университету. Первыми в списке должны идти студенты, сдавшие все экзамены на 5.

№15

В радиоателье хранятся квитанции о сданной в ремонт радиоаппаратуре. Каждая квитанция содержит следующую информацию: наименование группы изделий (телевизор, радиоприемник и т.п.), марку изделия, дату приемки в ремонт, состояние готовности заказа (выполнен, не выполнен). Вывести информацию о состоянии заказов на текущие сутки по группам изделий.

№16

Разработать программу формирования ведомости об успеваемости студентов. Каждая структура этой ведомости должна содержать: номер группы, Ф.И.О. студента, оценки за последнюю сессию. Вывести списки

студентов по группам. В каждой группе Ф.И.О. студентов должны быть расположены в порядке убывания среднего балла.

№17

В исполкоме формируется список учета нуждающихся в улучшении жилищных условий. Каждая структура этого списка содержит: порядковый номер, Ф.И.О., величину жилплощади на одного члена семьи и дату постановки на учет. По заданному количеству квартир, выделяемых по данному списку в течение года, вывести весь список с указанием ожидаемого года получения квартиры.

№18

Имеется список женихов и список невест. Каждая структура списка содержит пол, имя, возраст, рост, вес, а также требования к партнеру: наименьший и наибольший возраст, наименьший и наибольший вес, наименьший и наибольший рост. Объединить эти списки в список пар с учетом требований партнерам без повторений женихов и невест.

№19

В библиотеке имеется список книг. Каждая структура этою списка содержит: фамилии авторов, название книги, год издания. Вывести информацию о книгах, в названии которых встречается некоторое ключевое слово (ввести с клавиатуры).

№20

В магазине имеется список поступивших в продажу автомобилей. Каждая структура этого списка содержит: марку автомобиля, стоимость, расход топлива на 100 км, надежность (число лет безотказной работы), комфортность (отличная, хорошая, удовлетворительная). Вывести перечень автомобилей, удовлетворяющих требованиям покупателя, которые вводятся с клавиатуры в виде некоторого интервала допустимых значений.

№21

Каждая структура списка вакантных рабочих мест содержит: наименование организации, должность, квалификацию (разряд или образование), стаж работы по специальности, заработную плату, наличие социального страхования (да/нет), продолжительность ежегодного

оплачиваемого отпуска. Вывести список рабочих мест в соответствии с требованиями клиента.

№22

В технической службе аэропорта имеется справочник, содержащий записи следующей структуры: тип самолета, год выпуска, расход горючего 1000 км. Для определения потребности в горючем техническая служба запрашивает расписание полетов. Каждая структура расписания содержит следующую информацию: номер рейса, пункт назначения, дальность полета. Вывести суммарное количество горючего, необходимое для обеспечения полета на следующие сутки.

№23

Для участия в конкурсе на замещение вакантной должности сотрудника фирмы желающие подают следующую информацию: Ф.И.О., год рождения, образование(среднее, специальное, высшее), знание иностранных языков (английский, немецкий, французский, владею свободно, читаю и перевожу со словарем), владение компьютером (MSDOS,Windows), стаж работы, наличие рекомендаций. Вывести список претендентов в соответствии с требованиями руководства фирмы.

№24

При постановке на учет в ГАИ автолюбители указывают следующие данные: марка автомобиля, год выпуска, номер двигателя, номер кузова, цвет, номерной знак, Ф.И.О и адрес владельца. Вывести список автомобилей, проходящих техосмотр в текущем году, сгруппированных по маркам автомобилей. Учесть, что если текущий год четный, техосмотр проходят автомобили с четными номерами двигателей, иначе - с нечетными номерами.

№25

Для участия в конкурсе исполнителей необходимо заполнить следующую анкету: Ф.И.О., год рождения, название страны, класс музыкального инструмента (гитара, фортепиано, скрипка, виолончель). Вывести список самых молодых лауреатов конкурса по классам инструментов в порядке занятых мест.

№26

Список группы студентов содержит следующую информацию: Ф.И.О., рост и вес. Вывести Ф.И.О. студентов, рост и вес которых чаще всего встречаются в списке.

№27

Список группы студентов содержит следующую информацию: Ф.И.О., рост и вес. Вывести Ф.И.О. студентов, рост и вес которых являются в списке уникальными.

4. Критерии оценки

Оценка качества выполненной практического занятия проверяется по следующим критериям, оцененным в оценке «Плюс» или «Минус»:

Если практическое занятие выполнена полностью в соответствии с заданием – то в преподавательскую ведомость (Excel файл) ставится «Плюс», в противном случае ставится «Минус».

5. Список рекомендуемой литературы, интернет – ресурсов и справочно-правовых систем

5.1. Нормативные акты

Законодательство РФ

5.2 Основная литература

1. Информационные системы и технологии в экономике и управлении : учебник для бакалавров / под ред. В. В. Трофимова. — 4-е изд., перераб. и доп. — М. : Издательство Юрайт, 2014.

2. Гаврилов, М.В. Информатика и информационные технологии: Учебник для бакалавров / М.В. Гаврилов, В.А. Климов; Рецензент Л.В. Кальянов, Н.М. Рыскин. - М.: Юрайт, 2013. - 378 с.

5.3 Дополнительная литература

1. Гаврилов, Л. П. Инновационные технологии в коммерции и бизнесе : учебник для бакалавров / Л. П. Гаврилов. — М. : Издательство Юрайт, 2013. — 372 с. — Серия : Бакалавр.

2. Информационные системы и технологии в экономике и управлении : учебник для бакалавров / под ред. В. В. Трофимова. — 4-е изд., перераб. и доп. — М. : Издательство Юрайт, 2013.

3.Абросимова, М.А. Информационные технологии в государственном и муниципальном управлении: Учебное пособие / М.А. Абросимова. - М.: КноРус, 2013. - 248 с

5.4 Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

http://www.intuit.ru/ - национальный открытый университет

6. Приложение:

— Титульный лист

Ссылка на образцы титульных листов различных видов текущего контроля: http://www.old.fa.ru/fil/kaluga/student/Pages/default.aspx

— Пример выполнения варианта практического занятия

Сборка – совокупность взаимосвязанных типов (классов, интерфейсов, структур, перечислений, делегатов и т.д.).

Сборка включает в себя:

- Манифест;
- Метаданные;
- Код на языке IL;

- Ресурсы.

Манифест – это набор метаданных о сборке, включающий информацию о:

- Всех файлах, входящих в состав сборки;
- Сведения о всех внешних сборках.

Манифест создаётся компилятором автоматически.

Метаданные рассматриваются, как *метаданные типов* – это сведения о типах, используемых в сборке. Эти данные содержат информацию о каждом типе, существующем в программе, а также о каждом его элементе. То есть, если под типом понимать класс, то для каждого класса описываются все его поля, методы, свойства, события.

Код на языке IL – код, который поддерживает выполнение приложения на любом типе компьютеров, для которых существует среда CLR.

Ресурсы – это, например, файлы изображений, помещаемых на форму, текстовые данные, иконка приложения и т.д. Хранение ресурсов внутри сборки обеспечивает их защиту и упрощает развёртывание приложения. Среда Visual Studio NET предоставляет возможность автоматического внедрения ресурсов в сборку.

Создание собственной библиотеки

Для того, чтобы создать собственную библиотеку необходимо выбрать файл проекта - шаблон ClassLibrary. Библиотека хранится в виде файла с расширением *.DLL. Для использования такой библиотеки необходимо создать собственный проект и включить в него с помощью команды Project->Add Reference файл собственной библиотеки с расширением dll.

Все классы, включенные в библиотеку, желательно объявлять с режимом доступа public.

Пример создания и использования собственной библиотеки

```
//Файл библиотеки с именем ClassLibrary1.dll
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ClassLibrary1
{
   public class Monster
   {
      public Monster(int сила, int умение, string имя)
      {   this.сила = сила;
         this.умение = умение;
         this.имя = имя;
      }
 public virtual void Passport()
```

```
{Console.WriteLine("Монстр {0} \t сила = {1} умение= {2}", имя, сила, умение);}
        public int Сила
        { get    { return сила;   }
          set    {if (value > 0) сила = value;
                  else сила = 0;    }
        }
        public int Умение
        { get    {return умение;  }
          set   {if (value > 0) умение = value;
                  else умение = 0;  }
        }
        public string Имя
        {get {return имя; }
        }
        string имя;
        int сила, умение;
    }
    //производный класс от монстра – демон, умеющий “думать”
    public class Daemon : Monster
    {
        public Daemon(int сила, int умение, string имя, int ум)
            : base(сила, умение, имя)
        { this.ум = ум; }
        public override void Passport()
        {
            Console.WriteLine("Демон {0} сила = {1} умение {2} ум = {3}", Имя,
Сила, Умение, ум);
        }
        public void Think()
        {
            Console.Write(Имя + "это ");
            for (int i = 0; i < ум; i++) Console.Write(" думает ");
            Console.WriteLine("...");
        }
        int ум;
    }
}
//Файл проекта, использующий созданную библиотеку
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ClassLibrary1; //ПОДКДЮЧЕНИЕ собственной библиотеки
```

```
namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         Monster a = new Monster(1, 1, "вася");
         a.Passport();
         Monster [] mas = new Monster[2];
         mas[0] = new Monster(2, 2, "Петя");
         mas[1] = new Daemon(0, 0, "Демон Вася", 1);
                  foreach (Monster x in mas)
            x.Passport();
                  Console.ReadKey();
      }
   }
}
```