**Лабораторная работа № 6**

**Методы. Делегаты**

**Цель работы:**

Приобретение навыков практического применения, закрепление знаний при создания простейших программ с использованием методов, делегатов.

**Задачи:**

1. Реализовать решение задания своего варианта с использованием делегатов;
2. В работе использовать разные типы параметров (параметры-значения, параметры-ссылки, выходные параметры, параметры-массивы, параметры по умолчанию)
3. Выполнить требования к лабораторным работам (см. файл «Требования к лабораторным»);
4. Ознакомиться с литературой по теме лабораторной работы;
5. Ответить на контрольные вопросы.

**Контрольные вопросы:**

1. Что такое делегаты, для чего применяются?

**Пояснения и примеры:**

[Делегат](http://msdn.microsoft.com/ru-ru/library/900fyy8e.aspx) — это тип, который определяет сигнатуру метода. При создании экземпляра делегата можно связать этот экземпляр с любым методом с совместимой сигнатурой. Метод можно запустить (или вызвать) с помощью экземпляра делегата.

Делегаты используются для передачи методов в качестве аргументов к другим методам. Обработчики событий — это ничто иное, как методы, вызываемые с помощью делегатов.Создание пользовательского метода и класса, например элемента управления Windows, может вызывать метод при возникновении определенного события. В следующем примере показано объявление делегата:

public delegate int PerformCalculation(int x, int y);

Делегату можно назначить любой метод из любого доступного класса или структуры, соответствующий сигнатуре делегата, которая состоит из типа возвращаемого значения и параметров. Этот метод должен быть статическим методом или методом экземпляра. Это позволяет программно изменять вызовы метода, а также включать новый код в существующие классы. Если сигнатура делегата известна, то можно назначить собственный метод.

 В следующем примере объявляется делегат с именем Del, который может инкапсулировать метод, использующий в качестве аргумента значение [string](http://msdn.microsoft.com/ru-ru/library/362314fe.aspx) и возвращающий значение [void](http://msdn.microsoft.com/ru-ru/library/yah0tteb.aspx):

public delegate void Del(string message);

Объект делегата обычно создается указанием имени метода, для которого делегат будет служить оболочкой, или с помощью [анонимного метода](http://msdn.microsoft.com/ru-ru/library/0yw3tz5k.aspx). После создания экземпляра делегата вызов метода, выполненный в делегате, передается делегатом в этот метод. Параметры, передаваемые делегату вызывающим объектом, передаются в метод, а возвращаемое методом значение (при его наличии) возвращается делегатом в вызывающий объект. Эта процедура называется вызовом делегата. Делегат, для которого создан экземпляр, можно вызвать, как если бы это был метод, для которого создается оболочка. Примеры.

// Create a method for a delegate.publicstaticvoid DelegateMethod(string message)

{

System.Console.WriteLine(message);

}

// Instantiate the delegate.

Del handler = DelegateMethod;

// Call the delegate.

handler("Hello World");

В следующем примере метод использует тип Del как параметр:

public void MethodWithCallback(int param1, int param2, Del callback)

{

callback("The number is: " + (param1 + param2).ToString());

}

Затем можно передать созданный ранее делегат в данный метод:

MethodWithCallback(1, 2, handler);

и получить следующие выходные данные в окне консоли:

The number is: 3

Следующий пример демонстрирует объявление, создание экземпляра и использование делегата. Класс BookDB инкапсулирует базу данных книжного магазина, в котором ведется база данных книг. Он предоставляет метод ProcessPaperbackBooks, который находит все книги в мягкой обложке по базе данных и вызывает делегат для каждой книги. Используемый тип delegate имеет имя ProcessBookDelegate. Класс Test использует этот класс для печати заголовков и средней цены книг в мягкой обложке.

Использование делегата способствует правильному разделению функций между базой данных книжного магазина и клиентским кодом. Клиентскому коду неизвестен порядок хранения книг и то, как код книжного магазина выполняет поиск книг в мягкой обложке. Коду книжного магазина неизвестно о выполняемой обработке книг с мягкой обложкой после их нахождения.

[[http://i.msdn.microsoft.com/Hash/030c41d9079671d09a62d8e2c1db6973.gif](javascript:void(0))**Пример**](javascript:void(0))

**C#**

// A set of classes for handling a bookstore:

namespace Bookstore

{

using System.Collections;

// Describes a book in the book list:

public struct Book

{

public string Title; // Title of the book.

public string Author; // Author of the book.

public decimal Price; // Price of the book.

public bool Paperback; // Is it paperback?

public Book(string title, string author, decimal price, bool paperBack)

{

Title = title;

Author = author;

Price = price;

Paperback = paperBack;

}

}

// Declare a delegate type for processing a book:

public delegate void ProcessBookDelegate(Book book);

// Maintains a book database.

public class BookDB

{

// List of all books in the database:

ArrayList list = new ArrayList();

// Add a book to the database:

public void AddBook(string title, string author, decimal price, bool paperBack)

{

list.Add(new Book(title, author, price, paperBack));

}

// Call a passed-in delegate on each paperback book to process it:

public void ProcessPaperbackBooks(ProcessBookDelegate processBook)

{

foreach (Book b in list)

{

if (b.Paperback)

// Calling the delegate:

processBook(b);

}

}

}

}

// Using the Bookstore classes:

namespace BookTestClient

{

using Bookstore;

// Class to total and average prices of books:

class PriceTotaller

{

int countBooks = 0;

decimal priceBooks = 0.0m;

internal void AddBookToTotal(Book book)

{

countBooks += 1;

priceBooks += book.Price;

}

internal decimal AveragePrice()

{

return priceBooks / countBooks;

}

}

// Class to test the book database:

class TestBookDB

{

// Print the title of the book.

static void PrintTitle(Book b)

{

System.Console.WriteLine(" {0}", b.Title);

}

// Execution starts here.

static void Main()

{

BookDB bookDB = new BookDB();

// Initialize the database with some books:

AddBooks(bookDB);

// Print all the titles of paperbacks:

System.Console.WriteLine("Paperback Book Titles:");

// Create a new delegate object associated with the static

// method Test.PrintTitle:

bookDB.ProcessPaperbackBooks(PrintTitle);

// Get the average price of a paperback by using

// a PriceTotaller object:

PriceTotaller totaller = new PriceTotaller();

// Create a new delegate object associated with the nonstatic

// method AddBookToTotal on the object totaller:

bookDB.ProcessPaperbackBooks(totaller.AddBookToTotal);

System.Console.WriteLine("Average Paperback Book Price: ${0:#.##}",

totaller.AveragePrice());

}

// Initialize the book database with some test books:

static void AddBooks(BookDB bookDB)

{

bookDB.AddBook("The C Programming Language", "Brian W. Kernighan and Dennis M. Ritchie", 19.95m, true);

bookDB.AddBook("The Unicode Standard 2.0", "The Unicode Consortium", 39.95m, true);

bookDB.AddBook("The MS-DOS Encyclopedia", "Ray Duncan", 129.95m, false);

bookDB.AddBook("Dogbert's Clues for the Clueless", "Scott Adams", 12.00m, true);

}

}

}

/\* Output:

Paperback Book Titles:

The C Programming Language

The Unicode Standard 2.0

Dogbert's Clues for the Clueless

Average Paperback Book Price: $23.97

\*/

**Литература:**

1. Павловская Т.А. С#. Делегаты 220 - 232 стр.,