Работа с классом List<T>

В данной работе с помощью класса **List<T>** реализуются основные операции со списком книг, в том числе сортировка списка по разным критериям.

Класс **List<T>**, который (как и класс **ArrayList)** в языке C# называется **коллекцией**, способен хранить объекты произвольного типа (или любого класса). Класс **List<T>** объявлен в пространстве имен **System.Collections.Generic**.

В отличие от фиксированных статических массивов, в коллекциях List<T> и ArrayList размер увеличивается по мере необходимости, то есть классы List<T> и ArrayList поддерживают динамические массивы, расширяющиеся и сокращающиеся по мере необходимости.

Массив на основе коллекции List<T> или ArrayList создается с первоначальным размером. Если этот размер превышается, то массив автоматически расширяется. При удалении объектов из такого массива он автоматически сокращается.

Для примера используется список:

1. Visual Studio .NET 2003, Гарнаев, 2003.
2. Разработка приложений на C++ и C#, Секунов, 2003.
3. Программирование на языке C#, Фаронов, 2007.
4. C# и платформа .NET, Троелсен, 2010.
5. Библия C#, Фленов, 2011.

Приложение необходимо построить из одной формы, в которой предусмотреть ввод наименования книги, автора и года издания, кнопки и другие компоненты. Примерный внешний вид приложения показан на рисунке 1.

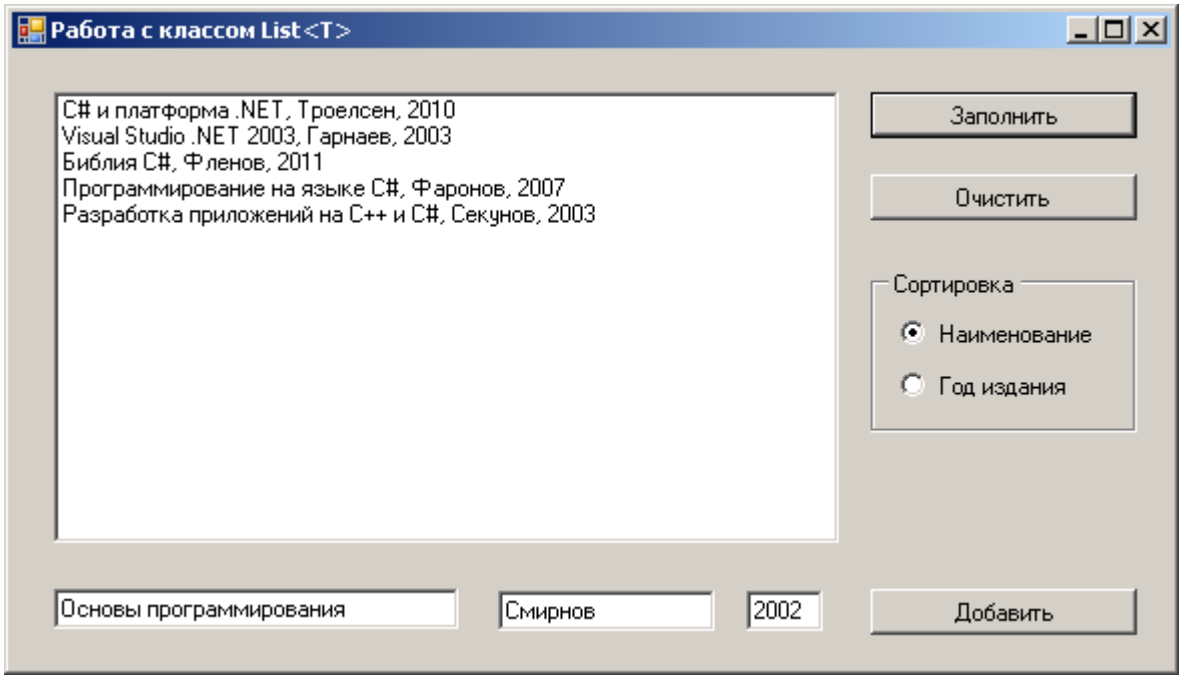

Рисунок 1 – Главная форма приложения

В проекте приложения необходимо использовать следующие компоненты:

- **TextBox** для ввода наименования и автора книги;
- **MaskedTextBox** для ввода года издания;
- **GroupBox, RadioButton** для типа сортировки;
- **ListBox** для вывода списка книг.

Далее рассматриваются элементы разработки. При завершении очередного шага рекомендуется сохранить проект, выполнить компиляцию и отладку.

1. Объявление нового класса TBook, отписывающего одну книгу

В программе создается новый класс **TBook**, содержащий параметры одной книги. Далее класс **TBook** будет использоваться в динамическом массиве, предназначенном для хранения списка книг.

Внутри класса Form1 главной формы приложения необходимо объявить новый класс **TBook**

```
public class TBook  // Содержит описание одной книги
{
   // поля класса
   internal string Title;
   internal int Year;
   // Конструкторы класса
   public TBook() { }   // Умалчиваемый конструктор
   public TBook(string kTitle, int kYear) // Конструктор с параметрами
   {
      Title = kTitle;
      Year = kYear;
    }
}
```

В приведенном классе **TBook** необходимо добавить поле для хранения автора книги.

2. Объявление массива (хранилища) книг и компаратора по наименованию

Динамический массив книг создается с помощью коллекции **List<T>** и класса **TBook.** Также требуется создать специальный класс-компаратор для выполнения сортировки списка книг по наименованию.

Внутри класса Form1 главной формы приложения необходимо объявить массив книг **Books**, класс компаратора **myIComparer** и объект **compar.**

```
// массив (хранилище) книг
public static List<TBook> Books = new List<TBook>();

// класс компаратора для сравнения по наименованию
public class myIComparer : IComparer<TBook>
{
   public int Compare(TBook x, TBook y)
   {
      return string.Compare(x.Title, y.Title);
   }
}
// создать объект-компаратор по наименованию
myIComparer compar = new myIComparer();
```

3. Метод вывода списка книг

Для вывода списка книг (массива **Books**) в **listBox1** создается новый метод **OutNames** с одним формальным параметром, с помощью которого в метод будет передаваться массив **Books**.

В методе используется циклический оператор **foreach**. Оператор **foreach** в языке C# используется для получения необходимой информации из определённой коллекции.

Например, в данном методе в операторе **foreach** создается объект типа **TBook** в коллекции **list**.

```
public void OutNames(List<TBook> list)
{
   listBox1.Items.Clear();      // очистка listBox1
   foreach (TBook obj in list)  // вывод списка книг
      listBox1.Items.Add(obj.Title + ", " + obj.Author + ", " + obj.Year);
}
```

4. Обработчик события для кнопки «Заполнить»

Массив **Books** заполняется списком книг и выводится в объект типа **ListBox**.

```
// заполнить список книг
Books.Add(new TBook("Visual Studio .NET 2003", "Гарнаев", 2003));
Books.Add(new TBook("Разработка приложений на C++ и C#", "Секунов", 2003));
Books.Add(new TBook("Программирование на языке C#", "Фаронов", 2007));
Books.Add(new TBook("C# и платформа .NET", "Троелсен", 2010));
Books.Add(new TBook("Библия C#", "Фленов", 2011));

Books.Sort(compar); // сортировка по наименованию с помощью компаратора
OutNames(Books);    // вывод списка книг
```

5. Обработчик кнопки «Очистить»

Требуется очистить массив **Books** и объект для вывода **listBox1**.

```
Books.Clear();
listBox1.Items.Clear();
```

6. Обработчик кнопки «Добавить»

```
string stitle = textBox1.Text;
string sauthor = textBox2.Text;
if (stitle == "")   // если пустая строка
   {
      MessageBox.Show("Ошибка ввода!");
      return;
   }
Books.Add(new TBook(stitle, sauthor, int.Parse(maskedTextBox1.Text)));
Books.Sort(compar); // сортировка по наименованию с помощью компаратора
OutNames(Books);    // вывод списка книг
```

7. Обработка ошибок ввода

- Необходимо проверить ввод пустой строки в полях «Автор» и «Год издания».
- Обработать, исключить повторный ввод наименования книги. Для поиска существующего наименования рекомендуется использовать встроенный в коллекции **List<T>** метод **BinarySearch**. Перед вызовом этого метода список необходимо отсортировать.

В приложении необходимо предусмотреть вывод соответствующих сообщений, приведенных на рисунке 2.

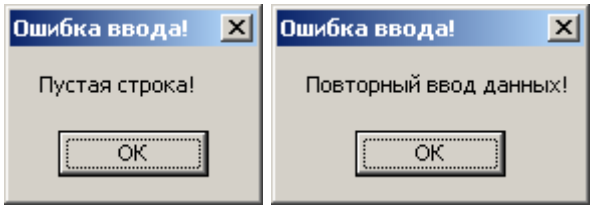

Рисунок 2 – Сообщения об ошибках ввода

8. Сортировка списка по разным критериям

Для сортировки списка книг по наименованию, автору и году издания требуется создать в классе Form1 новый метод. В этом методе необходимо использовать значение типа **radioButton1.Checked**, а также встроенный метод **Books.OrderBy(),** который выполняет упорядочивание по разным полям (критериям) объектов указанной коллекции. Примерный код:

```
if (radioButton1.Checked)
   OutNames(Books.OrderBy(критерий)); // упорядочить по наименованию
```

9. Удаление книги из списка

Для удаления строки из списка используется встроенный метод **Remove(строка)** или **RemoveAt(индекс).**

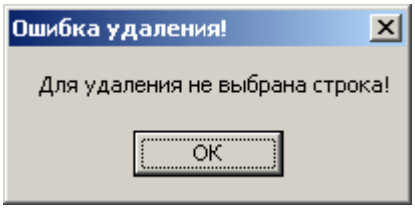

Рисунок 3 – Сообщение об ошибке удаления

Перед удалением должно появляться предупреждение, показанное на рисунке 4.

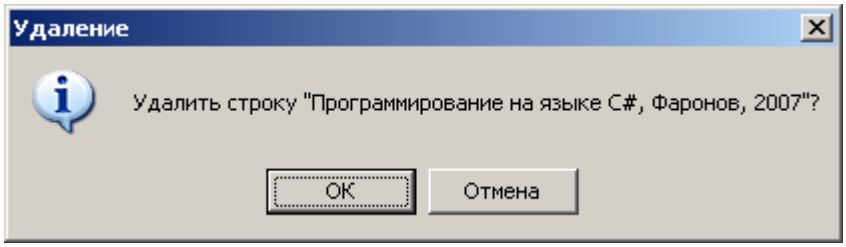

Рисунок 4 – Сообщение перед удалением