Лабораторная работа № 8

Работа с обобщенными типами и алгоритмами сортировки

С алгоритмами сортировки ознакомиться посредством материалов из Интернета, а также воспользоваться файлом lect4c.pdf, расположенном в папке Lab8.

Задание 1.

В приведенной ниже программе демонстрируется использование двух алгоритмов сортировки.

```
// BubbleSortArray - Сортирует список планет по именам:
//    1. В алфавитном порядке.
//    2. По длине имен от коротких к длинным.
//    3. От длинных к коротким.
//    Использованы два алгоритма сортировки:
//
//    1. Алгоритм сортировки из класса Array.
//    2. Классический алгоритм пузырьковой сортировки.
using System;

namespace BubbleSortArray
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("5 ближайших планет располагаются вокруг Солнца в следующем
порядке: ");
      string[] planets =
        new string[] { "Меркурий", "Венера", "Земля", "Марс", "Юпитер" };
      foreach (string planet in planets)
      {
        // Символ \t вставляет табуляцию в строку вывода.
        Console.WriteLine("\t" +  planet);
      }

      Console.WriteLine("\nПланеты в алфавитном порядке: ");
      // Array.Sort() метод класса Array.
      // Array.Sort() работает в пределах массива planets, не
      // оставляя исходной копии. Необходимо скопировать старый
      // массив и работать с копией. Эта копия будет отсортирована.
      string[] sortedNames = planets;
      Array.Sort(sortedNames);
      // Показываем, что массив sortedNames содержит те же планеты,
      // но отсортированные.
      foreach (string planet in sortedNames)
      {
        Console.WriteLine("\t" + planet);
      }

      Console.WriteLine("\nСортировка по длине имени: ");
      // Этот алгоритм называется "Пузырковой сортировкой". Он простой, но
      // не очень эффективный. Метод Array.Sort() существенно эффективнее,
      // но здесь он не применим, так как сравниваются не строки, а их длины.
      int outer;  // Индекс внешнего цикла.
      int inner;  // Индекс внутреннего цикла.
      // Цикл от последнего индекса в первому: planets[3] к planets[0].
      for (outer = planets.Length - 1; outer >= 0; outer--)
      {
        // На каждом цикле проходим по всем элементам
        // ниже текущего элемента. Этот цикл проходит в восходящем
        // порядке, от planets[1] к planets[3]. Цикл for позволяет выполнять
        // обход массива в любом направлении.
        for (inner = 1; inner <= outer; inner++)
```

```
        {
          // Сравниваем соседние элементы. Если ранний более длинный обменяванием
          // их местами.
          if (planets[inner - 1].Length > planets[inner].Length)
          {
            // Временное хранение одной планеты
            string temp = planets[inner - 1];
            // Сейчас переписываем эту планету на место другой.
            planets[inner - 1] = planets[inner];
            // Наконец возвращаем планету, хранящеюся в temp,
            // помещая ее на место другой.
            // Этот процесс называет свопингом или обменом.
            planets[inner] = temp;
          }
        }
      }
      foreach (string planet in planets)
      {
        Console.WriteLine("\t" + planet);
      }

      Console.WriteLine("\nПомещаем планеты в обратном порядке: ");
      // Цикл в обратном порядке
      for(int i = planets.Length - 1; i >= 0; i--)
      {
        Console.WriteLine("\t" + planets[i]);
      }

      Console.WriteLine("\Нажать Enter для окончания программы...");
      Console.Read();
    }
  }
}
```

1. Ввести данную программу и выполнить ее.
2. Переписать данную программу, оформив алгоритм пузырьковой сортировки в виде класса. Класс должен быть описан в отдельном файле.

Задание 2. Разработать программу, демонстрирующую сортировку списка числовых значений. В качестве алгоритма использовать алгоритм быстрой сортировки. Предусмотреть возможность использовать клавиатуру при работе с кнопками.

Начальный вид программы, изображен на рис.1

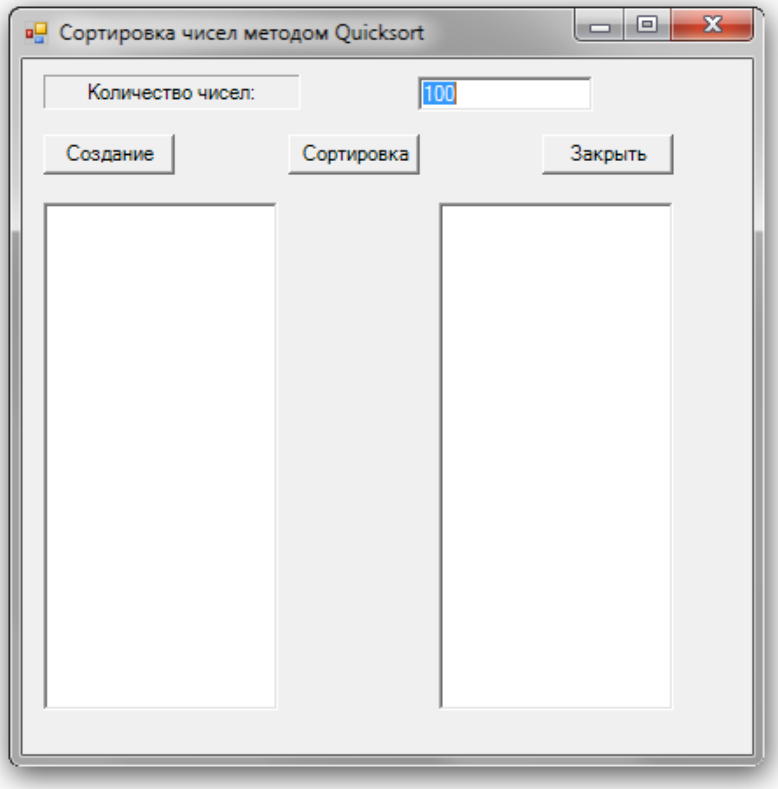

Рис.1. Начальное состояние программы

После щелчка по кнопке «Создание» или нажатия клавиш Alt+C в левом элементе ListBox должны появляться числа. (См. Рис.2).

Выбор кнопки «Сортировка» или нажатия клавиши Alt+O в правом элементе ListBox должны появиться отсортированные значения чисел (См. Рис.3).

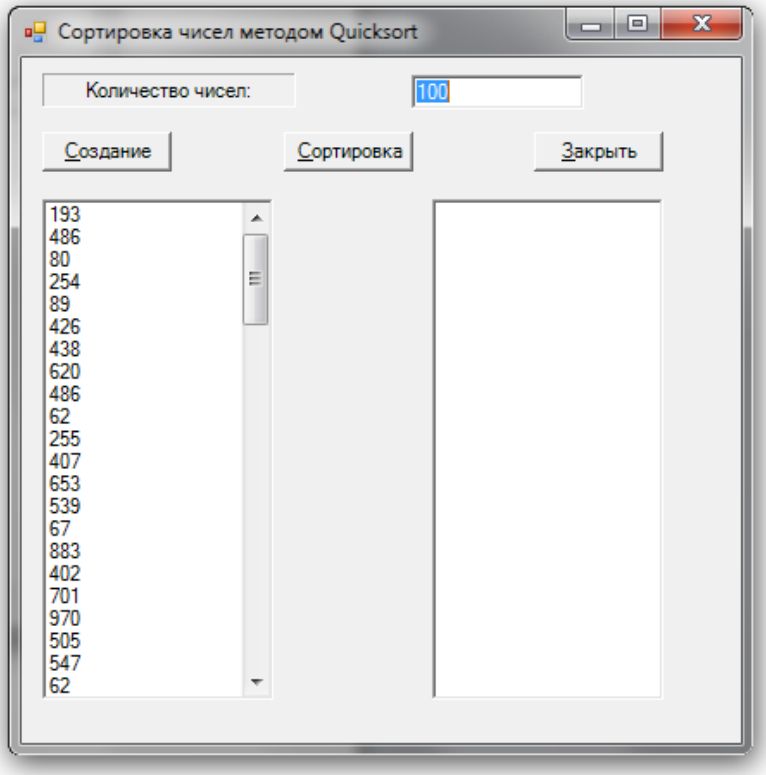

Рис.2. Демонстрация неотсортированного списка

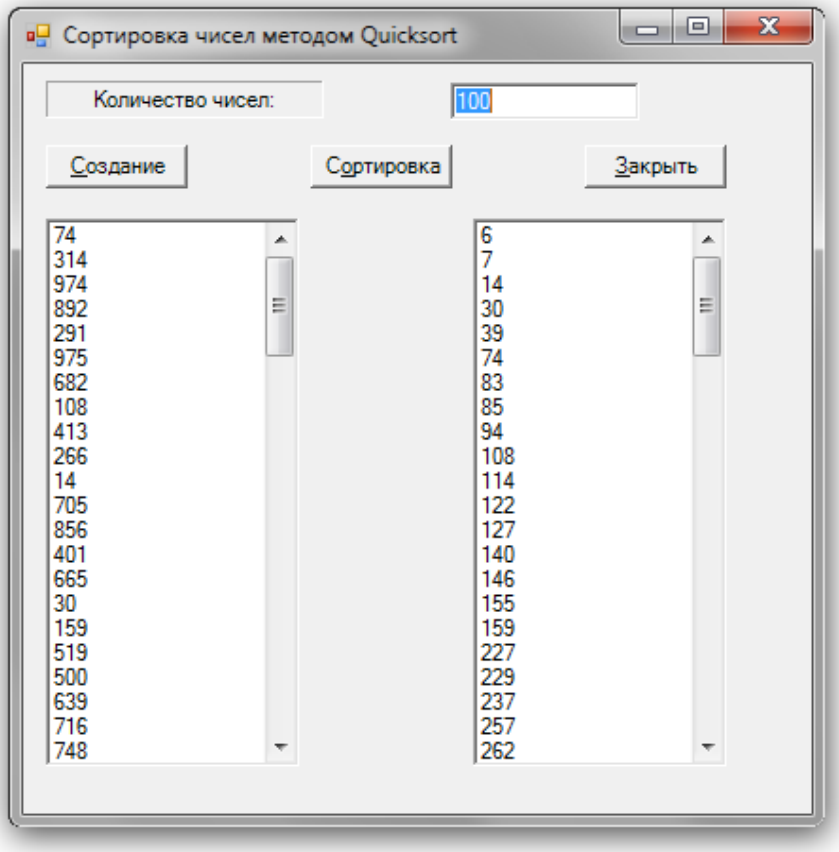

Рис.3. Демонстрация отсортированного списка.

Проверить работу программы для различного количества значений.

Задание 3. Используя обобщенные типы, переработать программу из задания 2, чтобы она могла обрабатывать числа типа int, long и double. Кроме того, программа должна выполнять сортировку строкового типа, выбирая значения из тестового поля. Предусмотреть возможность работы с клавиатурой используя кнопки.

Общий интерфейс программы изображен на Рис. 4.

Рис. 4. Вид программы при выполнении сортировки строкового типа.

Количество относится к числовым значениям, строковый тип сортируется на основе, количества значений, введенных в текстовое поле.

Задание 4. Для отсутствующих на занятии 08.05.2014.

1. Переписать текст программы из задания 1, используя алгоритм пирамидальной сортировки.

2. В переписанной программе оформить алгоритм пирамидальной сортировки в виде класса. Класс должен быть описан в отдельном файле.

Задание 5. Для отсутствующих на занятии 08.05.2014.

Переписать программу из задания 2. В качестве алгоритма использовать алгоритм сортировки Шелла. Предусмотреть возможность использовать клавиатуру при работе с кнопками.