ГБОУ ВПО Нижегородский государственный технический университет им. Р. Е. Алексеева
Институт радиоэлектроники и информационных технологий, кафедра "Вычислительные системы и технологии"

СОГЛАСОВАНО

Доцент каф. ВСТ

_____ *Гай В. Е.*

“____” _____ _____

ТЕХНОЛОГИИ РАСПРЕДЕЛЁННОЙ ОБРАБОТКИ ДАННЫХ

Отчет к лабораторной работе №2

РАЗРАБОТКА РАСПРЕДЕЛЁННОЙ СИСТЕМЫ ОБРАБОТКИ ДАННЫХ

Студент гр. 13-В-2

_____ *Смирнова С. В.*

“____” _____ _____

2015

Подп. и дата
Инв. дубл.
Взам. инв.
Подп. и дата
Инв. подл.

СОДЕРЖАНИЕ

1 ТРЕБОВАНИЯ К РАБОТЕ

Разработанный программный комплекс должен состоять из Сервера и Клиента. Функции сервера: хранение удалённого объекта, предоставляющего доступ к заданиям для обработки и результату обработки. Предусмотреть на сервере возможность одновременного доступа к критической секции кода нескольких клиентов. Критическая секция кода - та, к которой гипотетически одновременно могут обратиться несколько клиентов.

Функции клиента (на сервере хранится список клиентов - эта функция уже предусмотрена исходным кодом библиотеки RemoteBase):

а) Управляющие функции (выполняет только один клиент из всего множества клиентов, выполнение данной функции должно выполняться через вызов методов удалённого объекта (удалённый объект хранится на сервере)):

- Формирование и ведение списка заданий (под ведением понимается удаление уже обработанных и предоставление клиенту задания по запросу);

- Получение, объединение и вывод результатов вычислений (результаты вычислений должны выводиться в каждом клиенте, для этого необходимо проверять окончание обработки всех данных по таймеру; объединение результатов вычисление также можно реализовать с использованием таймера);

- Устанавливает флаг того, что управляющий клиент назначен, на сервере сохраняется идентификатор клиента;

б) Вычислительные функции

- Запрос задания с сервера (клиент должен запросить задание только после того, как эти задания были сформированы);

- Обработка данных;

- Отправка результатов обработки на сервер.

2 ВЫПОЛНЕНИЕ ЛАБОРАТОРНОЙ РАБОТЫ

2.1 Вариант задания

Вариант 13: Разработать алгоритм сортировки массива чисел методом пузырька

2.2 Листинг программы

2.2.1 Сервер

```
  using    System;
using SortLibrary;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

namespace SortServer
{
    class Server
    {
        TcpChannel channel;
        public void Start()
        {
            channel = new TcpChannel(8080);
            ChannelServices.RegisterChannel(channel, false);
            RemotingConfiguration.RegisterWellKnownServiceType(typeof(
               SharedObject), "Lab", WellKnownObjectMode.Singleton);
            ourServer.Print("Сервер запущен!");
        }

        public void Stop()
```

```
        {
            ChannelServices.UnregisterChannel(channel);
            ourServer.Print("Сервер остановлен!");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Server server = new Server();
            server.Start();
            Console.In.ReadLine();
            server.Stop();
        }
    }
}
```

2.2.2 Клиент

```
  using    System;
using SortLibrary;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;

namespace SortClient
{
    class Shell
    {
        TcpChannel chan;
        SharedObject obj;
        int[] arr;
        Task task;

        public Shell()
        {
            chan = new TcpChannel();
            ChannelServices.RegisterChannel(chan, false);
            obj = (SharedObject)Activator.GetObject(typeof(SortLibrary
               .SharedObject), "tcp://localhost:8080/Lab");
        }
```

```
        public int sort()
        {
                task = obj.GetTask();
                if (task == null)
                    return 0;

                arr = obj.GetData(task);

                Console.Out.WriteLine("Полученные данные:");
                display();

            for (int i = 0; i <= arr.Length - 1; i++)
            {
                for (int j = i + 1; j < arr.Length; j++)
                {
                    if (arr[j] < arr[i])
                    {
                        var spam = arr[i];
                        arr[i] = arr[j];
                        arr[j] = spam;
                    }
                }
            }


            Console.Out.WriteLine("Обработанные данные:");
                display();
                obj.Finish(task, arr);
            return 1;
        }

        void display()
        {
            for (int i = 0; i < SharedObject.clientPortion; i++)
            {
                Console.Out.Write("{0}\t", arr[i].ToString());
            }
            Console.Out.WriteLine();
        }
    }
```

```
class Program
{


    static void Main(string[] args)
    {
        Shell shellObj = new Shell();
        Console.Out.WriteLine("Клиент запущен!");

        while (shellObj.sort() != 0)
            Console.In.ReadLine();

        Console.Out.WriteLine("Все задачи решены!");
        Console.ReadLine();


    }
}
}
```

2.2.3 Библиотека

```
using System;
using System.Collections.Generic;

namespace SortLibrary
{
    public class SharedObject : MarshalByRefObject
    {
        const int dataCount = 10;  // всего задач
        const int tasksCount = 2;   // на сколько частей делятся задач
            и (иными словами, это – кол–во клиентов, необходмых для цел
            ьного выполнения работы)
        public static int clientPortion = dataCount / tasksCount;
        Queue<Task> QueTasks; // храним задачи, ожидающие обработку
        Object tasksLock;

        public List<int> maxes = new List<int>();
        public List<int> mins = new List<int>();

        int[] dataArray;
        Object dataLock;

        public SharedObject()
        {
            QueTasks = new Queue<Task>();
            GenerateData();
            GenerateTasks();

            tasksLock = new Object();
            dataLock = new Object();
        }

        void GenerateTasks()
        {
            ourServer.Print("Распределение задач...");
            Task temp;
```

```
            ourServer.Print("Всего элементов в матрице:{0}", dataCount
                );
            ourServer.Print("Клиентов:{0}", tasksCount);
            ourServer.Print("Кол–во элементов на клиента:{0}",
                clientPortion);

            for (int i = 0; i < tasksCount; i++)
            {
                temp = new Task();
                ourServer.Print("\nИнициализация счетчика элементов, в
                    ыделенных для клиента #{0}", i + 1);
                temp.start = i * clientPortion;
                ourServer.Print("Индекс начального элемента: {0}",
                    temp.start + 1);
                temp.stop = temp.start + clientPortion - 1;
                ourServer.Print("Индекс конечного элемента: {0}", temp
                    .stop + 1);
                QueTasks.Enqueue(temp);                    //добавле
                    ние задачи в конец очереди
            }
            ourServer.Print("\nЗадачи успешно распределены!");
        }

        void GenerateData()
        {
            ourServer.Print("Заполнение массива случайными элементами
                ...");
            Random r = new Random();
            dataArray = new int[dataCount];

            for (int i = 0; i < dataCount; i++)
            {
                dataArray[i] = r.Next(100);
                Console.Out.Write("{0}\t", dataArray[i].ToString());
            }
        }

        static int[] Sort(int[] temp)
        {
            for (int i = 0; i <= temp.Length - 1; i++)
            {
```

```
            for (int j = i + 1; j < temp.Length; j++)
            {
                if (temp[j] < temp[i])
                {
                    var spam = temp[i];
                    temp[i] = temp[j];
                    temp[j] = spam;
                }
            }
        }
        return temp;
    }

    public int[] GetData(Task task)
    {
        ourServer.Print("Клиент получил следующие данные для обраб
           отки:");
        int[] temp = new int[clientPortion];
        int j = 0;
        for (int i = task.start; i <= task.stop; i++)
        {
            temp[j] = dataArray[i];
            Console.Out.Write("{0}\t", dataArray[i].ToString());
            j++;
        }
        return temp;
    }

    public Task GetTask()
    {
        ourServer.Print("\nКлиент запросил задачу");
        lock (tasksLock)
        {
            if (QueTasks.Count == 0)
            {
                ourServer.Print("Задач больше нет!");
                return null;
            }
            else
                return QueTasks.Dequeue();
        }
```

```
}

public void Finish(Task task, int[] data)
{

    ourServer.Print("\nКлиент завершил обработку данных");
    lock (dataLock)
    {
        int j = 0;
        for (int i = task.start; i <= task.stop; i++)
        {

            dataArray[i] = data[j]; //соединяем сортированный
               части в целый массив
            j++;
            Console.Out.Write("{0}\t", dataArray[i].ToString()
               ); //отображаем только часть, обработанную дан
               ным клиентом

        }
        Console.Out.WriteLine();
    }

    if (QueTasks.Count == 0)
    {
        ourServer.Print("Все задачи решены");
        Console.Out.WriteLine("\n\n");
        Console.Out.WriteLine("Результаты работы клиентов, пом
           ещенные в один массив:");
        for (int i = 0; i < dataCount; i++)
        {
            Console.Out.Write("{0}\t", dataArray[i].ToString()
               ); //выводим массив, состоящий из двух сортир
               ованых половинок
        }
        Console.Out.WriteLine("\n\n");
        dataArray = Sort(dataArray);      //финальная сортировк
           а

        Console.Out.WriteLine("Дополнительна сортировка на сер
           вере:");
```

```
                for (int i = 0; i < dataCount; i++)
                {
                    Console.Out.Write("{0}\t", dataArray[i].ToString()
                        ); //отображение результирующего, полностью отс
                        ортированного массива
                }
                Console.Out.WriteLine();
            }
        }
    }



    [Serializable]
    public class Task
    {
        public int start = 0, stop = 0;
    }

    public class ourServer
    {
        public static void Print(String msg)
        {
            Console.WriteLine(msg);
        }
        public static void Print(String msg, int param1)
        {
            Console.WriteLine(msg, param1);
        }
    }
}
```

2.3 Результат работы программы

Скриншот работы первого клиента представлен на Рис.1.

Скриншот работы второго клиента представлен на Рис.2.

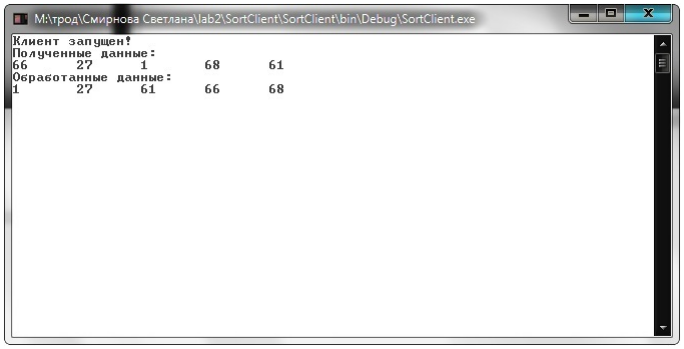

Рисунок 1

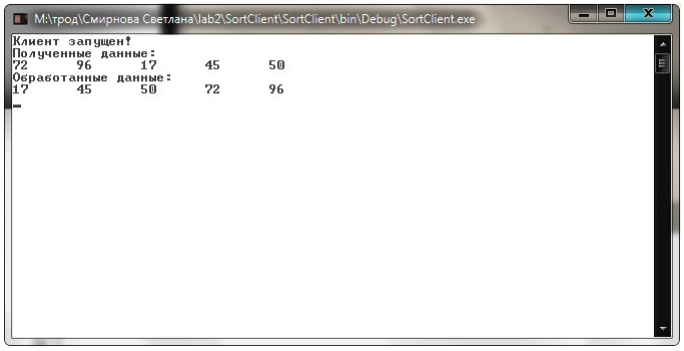

Рисунок 2

Скриншот работы сервера представлен на Рис.3.

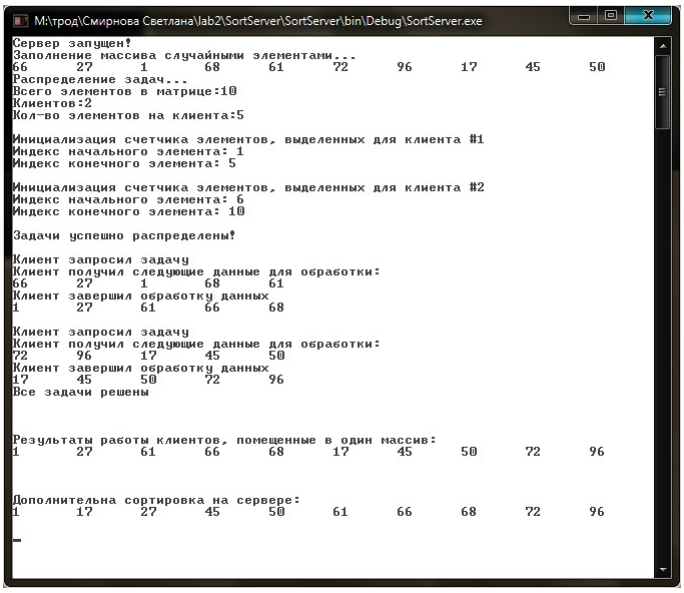

Рисунок 3

3 ВЫВОД

В результате выполнения лабораторной работы был получен программный комплекс, состоящий из сервера и клиента и реализующий алгоритм сортировки массива чисел методом пузырька.