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-1

_____ *Мишкорудный А. И.*

“____” _____ _____

2016

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 9:

Разработать алгоритм умножения матрицы a ($m \times n$ элементов) на вектор b (n элементов) по следующей формуле: $c_i = \sum_{j=1}^{n} a_{ij} b_j, 1 \le i \le m$.

Рисунок 1

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), "Work", WellKnownObjectMode.Singleton);
```

```
            ServerConsole.Print("Сервер запущен!");
        }

        public void Stop()
        {
            ChannelServices.UnregisterChannel(channel);
            ServerConsole.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;

        Task task;

        public Shell()
        {
            chan = new TcpChannel();
```

```
        ChannelServices.RegisterChannel(chan, false);
        obj = (SharedObject)Activator.GetObject(typeof(SortLibrary
            .SharedObject), "tcp://localhost:8080/Work");
    }

    public int sort()
    {
            task = obj.GetTask();

            if (task == null)
                return 0;
            int[,] ATemp;
            int[] BTemp;
            int[] C;
             obj.GetData(task, out BTemp, out ATemp);
           C= new int[SharedObject.n];
            Console.Out.Write("Полученные данные:");

        Console.Out.Write("\nСтроки матрицы A:\n");
        for (int i = task.start; i <= task.stop; i++) // строк мат
           рицы взяли столько, сколько указано для данного клиента
            (от строки start до строки stop)
            {
            Console.Out.WriteLine();
            for (int j = 0; j < SharedObject.n; j++) // столбцов -
                столько, сколько есть всего
                {
                    Console.Out.Write(ATemp[i,j]+"\t");
                }
            }

        Console.Out.Write("\n\nВектор-столбец B:\n");
        Console.Out.WriteLine();
        for (int j = 0; j < SharedObject.n; j++)
        {
            Console.Out.Write(BTemp[j] + "\n");    //кол-во строк в
               ектор-столбца обязательно равно кол-ву столбцов умн
               ожаемой на него матрицы
        }

        Console.Out.WriteLine("\nПроверка вычисления C[i]:");
```

```
            for (int i = task.start; i <= task.stop; i++)
            {
                C[i] = 0;
                Console.Out.WriteLine();
                for (int j = 0; j < SharedObject.n; j++)
                {
                    C[i] += ATemp[i, j] * BTemp[j];
                    Console.Out.Write(" {0} * {1} ", ATemp[i, j],
                       BTemp[j]);
                    if (j + 1 != SharedObject.n)
                    {
                        Console.Out.Write("+");
                    }
                    else
                    {
                        Console.Out.Write("= {0}", C[i]);
                    }
                }
                Console.Out.WriteLine();
            }


                obj.Finish(C);

            return 1;
        }

    }

    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("Задач больше нет!");
                Console.ReadLine();


            }
        }
    }
```

2.2.3 Библиотека

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

namespace SortLibrary
{
    public class SharedObject : MarshalByRefObject
    {

        static int i;
        public static int m = 4; //кол–во строк
        public static int n = 4; //кол–во столбцов

        const int dataCount = 4; //всего строк в матрице
        const int tasksCount = 2;  //максимальное число задач

        Queue<Task> QueTasks; // очередь задач ожидающих обработки
        Object tasksLock;

        public int[,] A = new int[m,n];     //матрица для умножения
        public int[] B = new int[n];    //вектор–столбец для умножения
        public int[] C = new int[m];    //хранение результата
        Object dataLock;

        public SharedObject()
        {
            QueTasks = new Queue<Task>();
            CreateData();
            CreateTasks();

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

        void CreateTasks()
        {
            ServerConsole.Print("\n\nСоздание задач...\n");
            Task temp;
```

```
        //распределение массива поровну на каждого клиента
        int clientPortion = dataCount / tasksCount;
        ServerConsole.Print("Всего строк в матрице:{0}",dataCount)
            ;
        ServerConsole.Print("Клиентов:{0}",tasksCount);
        ServerConsole.Print("Кол-во строк на клиента:{0}",
            clientPortion);

        for (int i = 0; i < tasksCount; i++)
        {
            temp = new Task();
            ServerConsole.Print("\nИнициализация счетчика умножаем
                ых строк для клиента #{0}",i+1);
            temp.start = i * clientPortion;
            ServerConsole.Print("Начальная строка: {0}",temp.start
                +1);
            temp.stop = temp.start + clientPortion - 1;
            ServerConsole.Print("Конечная строка: {0}", temp.stop
                +1);
            QueTasks.Enqueue(temp);                    //добавле
                ние задачи в конец очереди
        }
        ServerConsole.Print("\nЗадачи успешно созданы и распределе
            ны!");
    }

    void CreateData()
    {
        ServerConsole.Print("\n\n\nИсходные данные:");
        //***ввод тестовых значений для проверки правильности вычи
            слений
        Console.Out.WriteLine("Матрица A:");
        int k = 1;
        for (int i = 0; i < m; i++)
        {
            Console.Out.WriteLine();
            for (int j = 0; j < n; j++)
            {
                A[i, j] = k;
                k++;
```

```
                Console.Out.Write("{0}\t", A[i, j].ToString());
            }
        }

        Console.Out.WriteLine("\n\nВектор–столбец B:");
        k = 1;
        for (int j = 0; j < n; j++)
        {
            B[j] = k;
            k++;
            Console.Out.Write("{0}\n", B[j].ToString());
        }
        //***


        // заполнение случайными значениями
        //ServerConsole.Print("Заполнение случайными значениями
            ...\n");
        //Random r = new Random();
        //for (int i = 0; i < m; i++)
        //{
        //    for (int j = 0; j < n; j++)
        //        A[i, j] = r.Next(100);
        //}
        //for (int j = 0; j < n; j++)
        //    B[j] = r.Next(100);

        //ServerConsole.Print("Исходная матрица и вектор–столбец у
            спешно заполнены случайными значениями!\n");

    }



    public void GetData(Task task, out int[] Btemp, out int[,]
        Atemp)
    {
        Atemp = new int[m, n];
        Btemp = new int[n];
        ServerConsole.Print("\nКлиент начал получение данных для о
            работки!");
```

```
            Console.Out.WriteLine("A[m,n] передаваемое клиенту:");
            for (int i = task.start; i <= task.stop; i++)
            {
                Console.Out.WriteLine();
                for (int j = 0; j < n; j++)
                {
                    Atemp[i, j] = A[i, j];
                    Console.Out.Write("{0}\t", Atemp[i, j].ToString())
                        ;
                }

            }
            Console.Out.WriteLine("\n\nB[n] передаваемое клиенту:");
            for (int j = 0; j < n; j++)
            {
                Btemp[j] = B[j];
                Console.Out.Write("{0}\n", Btemp[j].ToString());
            }
            ServerConsole.Print("Клиент получил данные для обработки!\
                n\n");
        }

        public Task GetTask()
        {
            ServerConsole.Print("\nКлиент запросил задачу");
            lock (tasksLock)
            {
                if (QueTasks.Count == 0) //если задачи кончились
                {
                    ServerConsole.Print("Больше нет задач..."); //сооб
                        щим об этом
                    return null;
                }
                else
                    return QueTasks.Dequeue(); //если еще не кончились
                        – вернем следующую задачу, извлеченную из очер
                        еди
            }
        }

        public void Finish(int[] mas)
```

```
        {
            lock (dataLock)
            {
                for (int i = 0; i < n; i++)
                {
                    C[i] += mas[i];
                }
                ServerConsole.Print("\nКлиент успешно завершил задачу!
                    ");
            }

            if (QueTasks.Count == 0)
            {
                Console.Out.Write("\n\nПолученный результат:\n");
                for (i = 0; i < m; i++)
                    Console.Out.WriteLine("C[{0}]: {1}", i + 1, C[i].
                        ToString());
            }
        }
    }



    [Serializable]
    public class Task
    {
        public int start = 0, stop = 0;  //определение начала и конца
            диапазона
    }
    public class ServerConsole //вывод записи в консоли на сервере
    {
        //public static void Print(String msg, params int[] values)
            //используется ключевое слово params для передачи неопредел
            енного числа параметров в функцию
        //{
        //    Console.WriteLine(msg, values);
        //}

        //на случай проблем в работе params перегрузим метод Print
        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.2.

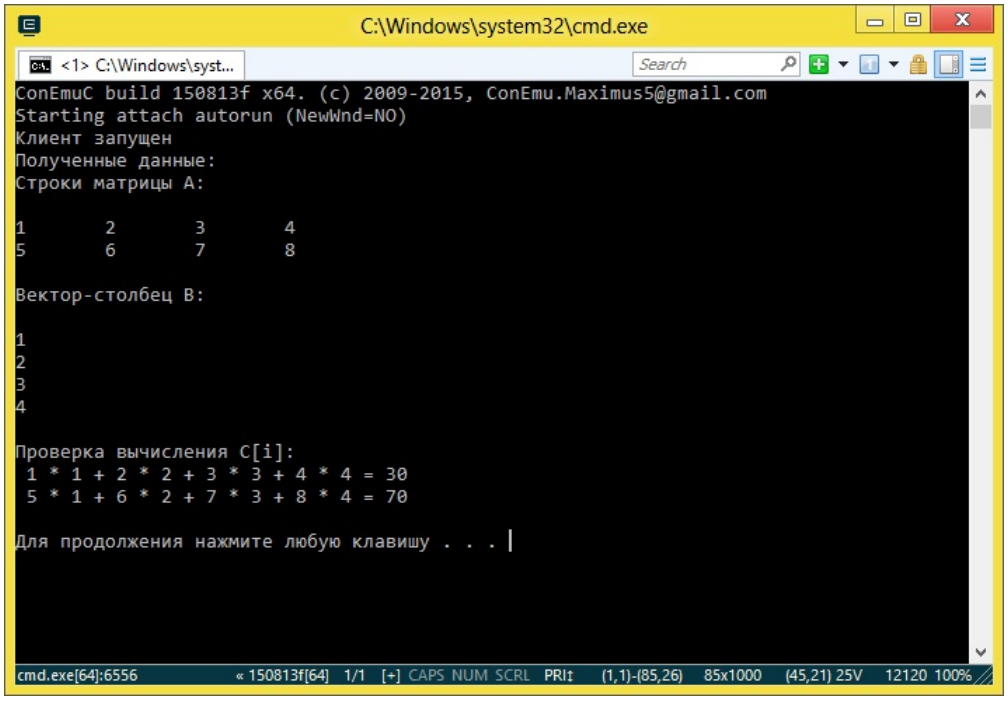

Рисунок 2

Скриншот работы второго клиента представлен на Рис.3.

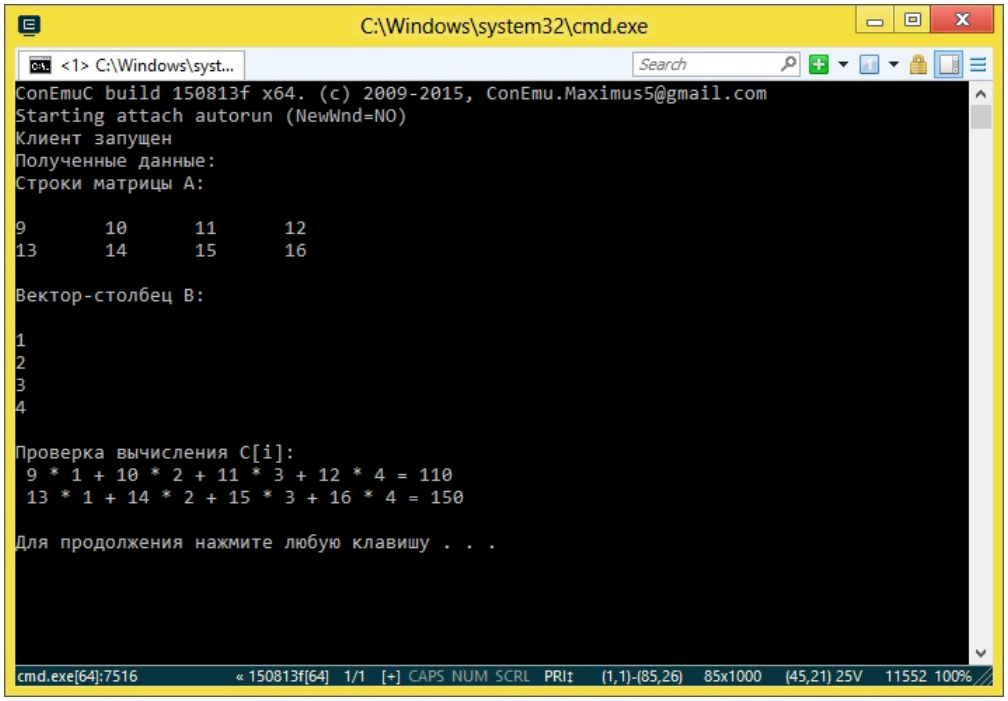

Рисунок 3

Скриншот работы сервера представлен на Рис.4.

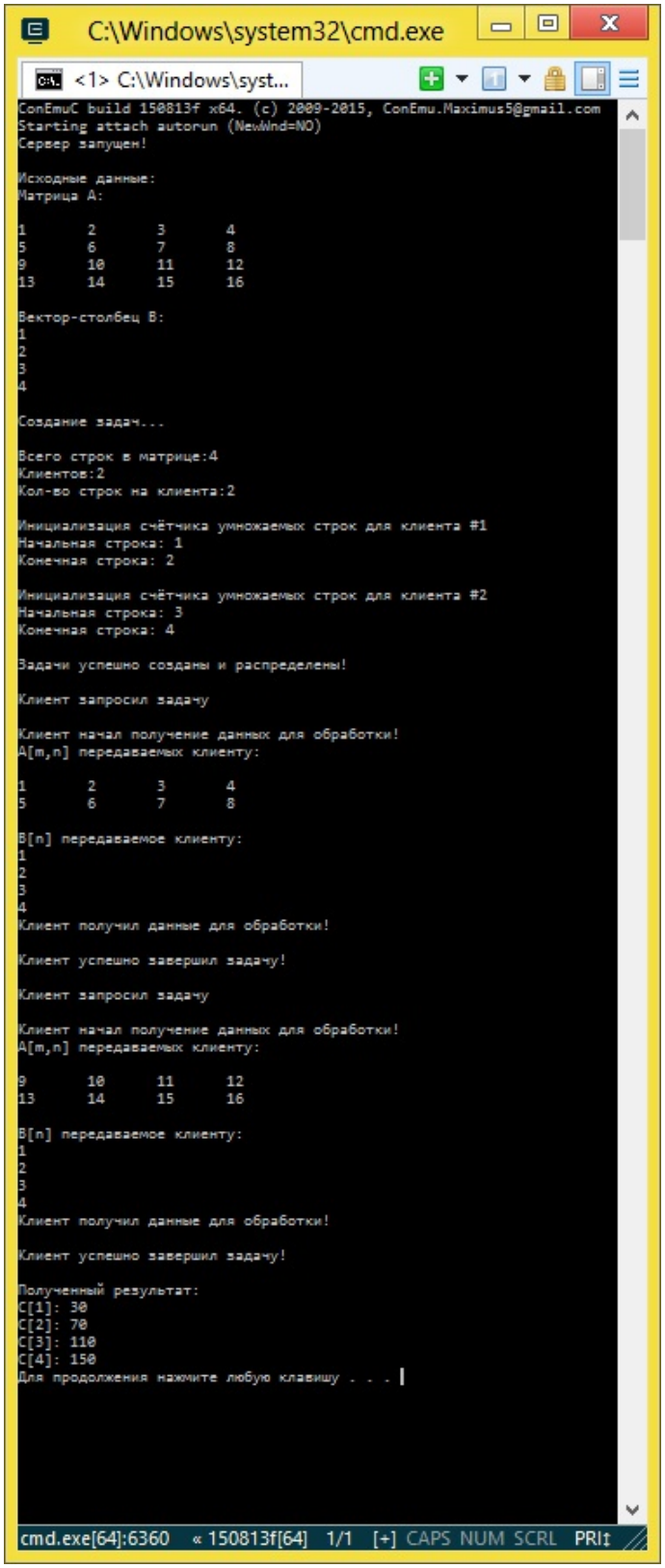

Рисунок 4

3 ВЫВОД

В результате выполнения лабораторной работы был получен программный комплекс, состоящий из сервера и клиента и реализующий алгоритм умножения матрицы на вектор-столбец.