Министерство образования и науки РФ
Брянский государственный технический университет

Кафедра: «Компьютерные технологии и системы»

Дисциплина: «Алгоритмические языки и программирование»

ОТЧЕТ

по контрольным работам
Вариант № 14

Выполнил студент гр: З-21-ИСТ-итпк-Б
Журавлёв Е. А.
Преподаватель:
Леонов Ю.А.

Брянск 2021

Контрольная работа № 1
Тема работы: Позиционная система счислений
Цель работы: ознакомление с видами систем счисления (СС) и приобретение практических навыков перевода из одной СС в другую, а также овладение элементарными арифметическими операциями над числами.
Формулировка задания: 1. Перевести заданное число из системы счисления (СС) «А» в «В». Число, которое необходимо перевести из СС «А» в «В», студент берет из строки «1 число» заданного варианта.
2. Выполнить сложение и вычитание двух представленных чисел («1 число» и «2 число»). Числа представлены для каждого варианта в системе счисления «А».

Решение

1. "1 число" - 3210_4

Основание СС "А" - 4, основание СС "В" - 5.
$3210_4 = 3\times10^3+2\times10^2+1\times10^1+0\times10^0=192+32+4+0=228_{10}$
228/5=45 ост: 3, 45/5=9 ост 0, 9/5=1 ост 4. 228_{10}=**1403_5**

2. "1 число" - 3210_4. "2 число" - 2012_4

3210_4	3210_4
$+2012_4$	-2012_4
11222_4	1132_4

$3210_4+2012_4=11222_4$
$3210_4-2012_4=1132_4$

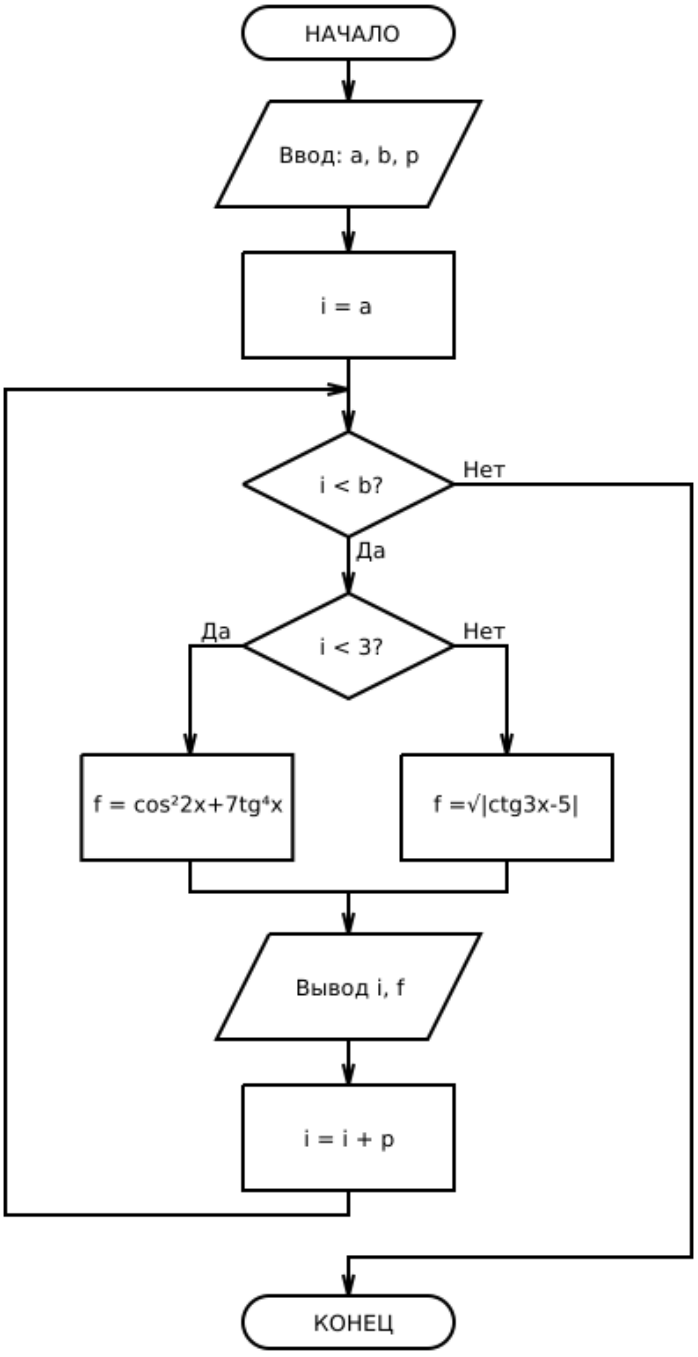
Заключение: навыки работы с системами счислений и выполнением арифметических операций над числами в различных системах счислений были “освежены” в памяти. Цель работы достигнута.

Контрольная работа № 2
Тема работы: Изучение условных и циклических конструкций.
Цель работы: приобретение практических навыков при построении ветвящихся и повторяющихся процессов с использованием условных и циклических конструкций на примере табулирования функции на заданном числовом отрезке.
Формулировка задания: описать алгоритм поставленной задачи в виде блок-схемы и написать программу на языке C#, которая будет реализовывать табулирование функции для заданной системы уравнений на числовом промежутке [a, b] с шагом p. Данные должны выводиться в табличной форме, где каждому значению аргумента соответствует подсчитанное значение функции.

$$f(x) = \begin{cases} \cos^2 2x + 7tg^4 x, & x < 3 \\ \sqrt{|ctg3x - 5|}, & x \geq 3 \end{cases}$$

Решение:

Блок-схема алгоритма:



Листинг программы:

```
using System;
namespace lab
{
   class Program
   {
       static void Main(string[] args)
       {
           Console.WriteLine("\nПрограмма табулирвания значений системы:");
           Console.WriteLine("        ⎧  cos²2x+7tg⁴x, если |x|<3");
           Console.WriteLine("  f(x)= ⎨");
           Console.WriteLine("        ⎩  √|ctg3x-5|, если |x|>=3");
           Double a,b,p,f;//переменные условий табули. и хранения вывода самой
функции
           Console.Write("Нижняя граница отрезка, включительно: ");
               a = Convert.ToDouble(Console.ReadLine());
           Console.Write("Верхняя граница отрезка, включительно: ");
               b = Convert.ToDouble(Console.ReadLine());
           Console.Write("Шаг табулирования: ");
               p = Convert.ToDouble(Console.ReadLine());
           Console.WriteLine("\tx\t\tf(x)");
           for (Double i=a;i<=b;i+=p)
           {
               if (Math.Abs(i)<3)
f=Math.Pow(Math.Cos(2*i),2)+7*Math.Pow(Math.Tan(i),4);
               else f=Math.Sqrt(Math.Abs(1/Math.Tan(3*i)-5));
               Console.WriteLine("{0}\t\t{1}",i,f);
           }
       }
   }
}
```

Результат работы программы:

```
erambler@erambler-laptop:~/Projects/Labs/C#/lab2_14$ dotnet run

Табулирвание значений функции:
        ⎧  cos²2x+7tg⁴x, если |x|<3
  f(x)= ⎨
        ⎩  √|ctg3x-5|, если |x|>=3
Введите нижнюю границу отрезка. -1
Введите верхнюю границу отрезка. 4
Введите шаг табулирования. 0,75
        x               f(x)
-1              41,355169040454555
-0,25           0,7999079735535882
0,5             0,9154184009307424
1,25            574,911194940145
2               159,99146429998834
2,75            0,7057074337713692
3,5             2,1117363087161682
erambler@erambler-laptop:~/Projects/Labs/C#/lab2_14$
```

Заключение:

Поставленная задача была выполнена. Блок-схема составлена. Разработанный алгоритм реализован на языке C#. Программа выполняет табличный вывод табулирования заданной функции. Цель работы достигнута.

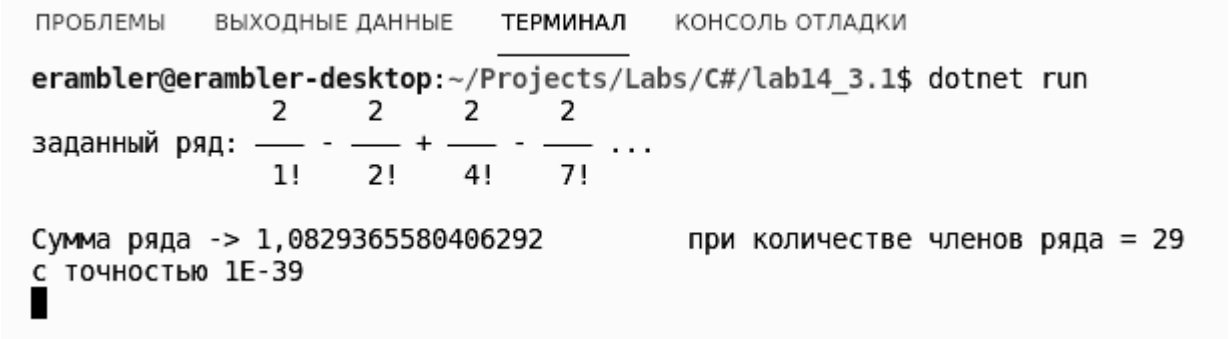
Контрольная работа № 3.1
Тема работы: Итерационные методы решения задач.
Цель работы: ознакомление с итерационными методами решения задач на примере вычисления суммы сходящегося ряда и нахождение корней нелинейных уравнений.
Формулировка задания: Требуется вычислить сумму S сходящегося ряда с заданной точностью e.

заданный ряд: $\frac{2}{1!} - \frac{2}{2!} + \frac{2}{4!} - \frac{2}{7!}$

Решение:

```
using System;
namespace lab3_1
{
    class Program
    {
        static void Main()
        {
            Console.WriteLine("               2     2     2     2");
            Console.WriteLine("заданный ряд: ——— - ——— + ——— - ——— ...");
            Console.WriteLine("               1!    2!    4!    7!");
            const float e = 0.000000001f;
            int i=0, l;
            double  nFact=1,//факториал
                    n=1,    //факториала
                    t,      //член ряда
                    sum=0,  //сумма полученная
                    diff;   //сумма предыдущая
            do {
                nFact=1;
                for (int k=1;k<=n;k++) nFact*=k; //Расчет факториала от n
                if (i%2!=0) l=-1; else l=1;      //Определяем знак
                t = l * (2/nFact);
                            diff=sum;
                                sum += t;
                                i++;
                                n+=i;
                }
            while (Math.Abs(sum-diff) > e);
            Console.WriteLine("\nСумма ряда -> {0}\t при количестве членов ряда =
{1}\nс точностью {2}",sum,n,e);
           Console.Read();
        }
    }
}
```

Результат выполнения программы:



Заключение: разработанная программа выполняет вычисление с заданной точностью. Цель работы была достигнута.

Контрольная работа № 3.2
Тема работы: Итерационные методы решения задач.
Цель работы: ознакомление с итерационными методами решения задач на примере вычисления суммы сходящегося ряда и нахождение корней нелинейных уравнений.
Формулировка задания: нахождение корней уравнения методом секущих на заданном отрезке. $e^x - 2(x-1)^2$

Решение

```
using System;
namespace Secant
{
    class Program
    {
    static float fEquation  (float x)
         return Convert.ToSingle((Math.Exp (x) - 2*(x-1)*(x-1)));
    static void Main()
        {   Console.WriteLine("уравнение: e ˣ -2(x-1)²");
            const float e = 0.0001f;
            float x0, x1, x2, fx0, fx1, fx2;
            Console.WriteLine("Введите начало и конец числового отрезка");
            x0 = Convert.ToSingle(Console.ReadLine());
            x1 = Convert.ToSingle(Console.ReadLine());
            do {fx0 = fEquation (x0);
                fx1 = fEquation (x1);
                x2  = x1 - fx1 * (x1 - x0) / (fx1 - fx0);
                fx2 = fEquation (x2);
                x0  = x1; x1 = x2;
            } while (Math.Abs(fx2) > e);
            Console.WriteLine("Значение корня уравнения на заданном числовом отрезке
= {0}", x2);
            Console.Read();
        }
    }
}
```

Результат выполнения программы:

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ  КОНСОЛЬ ОТЛАДКИ

erambler@erambler-desktop:~/Projects/Labs/C#/lab14_3.2$ dotnet run
уравнение: e ˣ -2(x-1)²
Введите начало и конец числового отрезка
0
1
Значение корня уравнения на заданном числовом отрезке = 0,21330993

erambler@erambler-desktop:~/Projects/Labs/C#/lab14_3.2$
```

Дополнительный комментарий: Согласно рекомендациям данным в методическом пособии, был построен график, для понимания поведения функции и отрезка с решением:

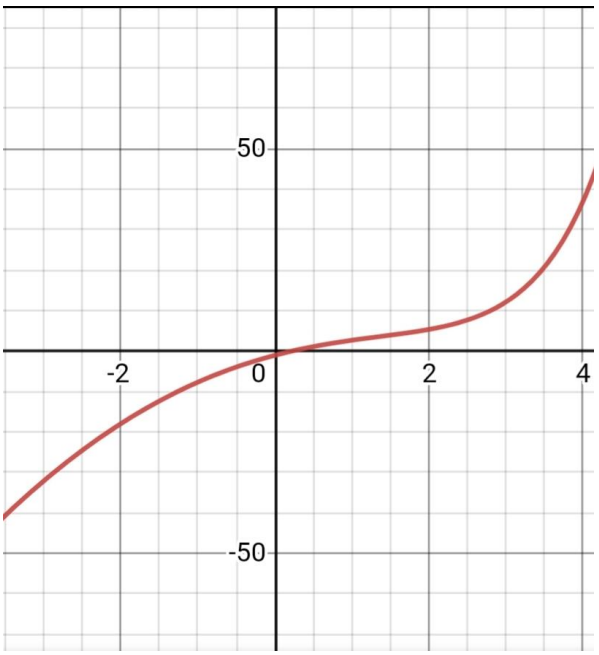


рис: график функции $f(x) = e^x - 2(x-1)^2$

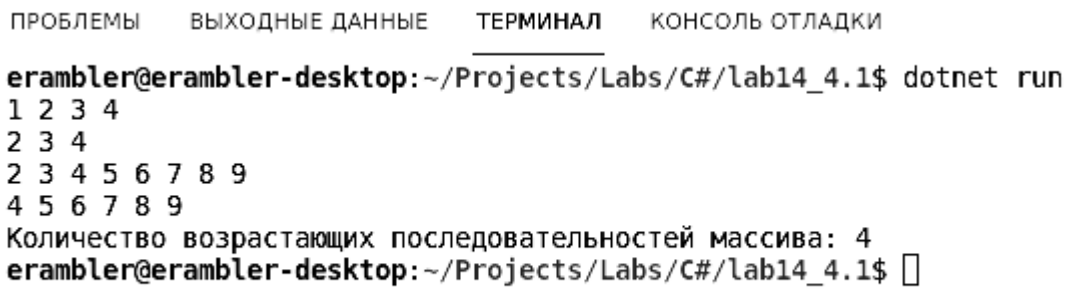
Заключение: в результаты выполнения работ по освоению итерационных методов были получены ценные навыки. Работа по вычислению корня уравнения выполнена. Результаты выполнения программы соответствуют теоретическим ожиданиям. Цель работы была достигнута.

Контрольная работа № 4.1
Тема работы: Работа с массивами
Цель работы: Изучение теоретических основ и овладение практическими навыками работы с массивами.
Формулировка задания: Дан массив размера n. Найти количество участков, на которых его элементы возрастают (участком считать последовательность от 3-х элементов).

Решение

```
using System;
namespace Subsequences
{
    class Program
    {
    static void Main()
        {
        const int n = 30;           //  Размер массива
        int counter=0,              //  счётчик возрастающих элементов
            quantity=0;             //  счётчик последовательностей,
                                    //удовлетворяющих условию задачи
        int [] A = new int[n]  {1,1,1,1,1,1,2,3,4,2,
                                3,4,2,3,4,5,6,7,8,9,
                                9,9,9,4,5,6,7,8,9,9};
        for (int i = 1; i < A.Length; i++) // Перебор массива со второго элемента
        {
            if (A[i]>A[i-1])
            {
              if (counter==0) Console.Write("{0} ",A[i-1]); // Выводим первый
элемент
                counter++;                  //  Засчитываем элемент
                Console.Write("{0} ",A[i]); //  Вывод элемента
            } else
            {
                if (counter >=2)        //  Условие длины последовательности
                {
                    counter=0;          //  Сброс счётчика
                    quantity++;         //  Засчитываем последовательность
                    Console.WriteLine();//  Возврат строки
                }
            }
        }
        if (counter >=2) quantity++; // Засчитываем последовательность, которая
может
                                     //оказаться в конце массива
        Console.WriteLine("Количество возрастающих последовательностей массива:
{0}",quantity);
        Console.Read();
    }
  }
}
```

Результат выполнения программы:



Заключение: Получены навыки работы с массивами. Решена задача с нахождением возрастающих последовательностей. Цель работы была достигнута.

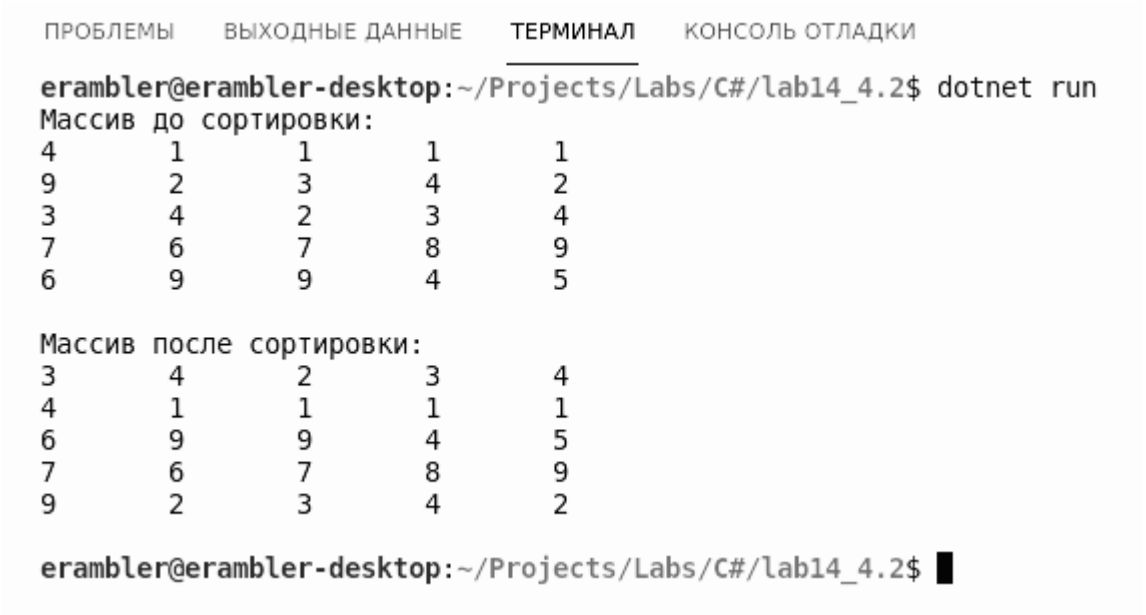
Контрольная работа № 4.2
Тема работы: Работа с массивами
Цель работы: Изучение теоретических основ и овладение практическими навыками работы с массивами.
Формулировка задания: Дан массив размера n × n. Необходимо упорядочить (переставить) строки массива по возрастанию значений первых элементов строк.

Решение

```
using System;
namespace ArrayBubble
{   class Program
    {
    const int n = 5;        //  Дан массив размера n × n
    static int [,] A = new int[n,n] {{4,1,1,1,1},
                                     {9,2,3,4,2},
                                     {3,4,2,3,4},
                                     {7,6,7,8,9},
                                     {6,9,9,4,5}};

    static void Main()
        {   Console.WriteLine("Массив до сортировки:");
            ShowMeArray();        //  Показываем массив на экран
            Bubble();             //  Запускаем сортировку
            Console.WriteLine("Массив после сортировки:");
            ShowMeArray();        //  Показываем массив на экран
       }
    static void ShowMeArray ()    //  Показываем массив на экран
    {   for (int i=0;i<n;i++)
        {
            for (int j=0;j<n;j++) Console.Write("{0}\t",A[i,j]);
            Console.WriteLine();
        }
        Console.WriteLine();
    }
    static void Bubble ()           // Начинаем итерацию сортировки элементов
    {                               // Простым, но не очень эффективным способом
        bool iscorrect = true;      // Будем считать, что массив отсортирован,
        for (int i = 1; i < n; i++) // Однако убедимся в этом, перебрав его
        {   if (A[i,0]<A[i-1,0])     // Если нашли больший элемент, то
            {   iscorrect = false;   // запоминаем, что нужна ещё одна итерация
                int swap;            // и, для перемены строк, создадим переменную,
                for (int j=0;j<n;j++)// и, наконец, поменяем местами строки.
                {   swap=A[i,j];
                    A[i,j]=A[i-1,j];
                    A[i-1,j]=swap;
                }
            }
        }
        if (!iscorrect) Bubble();    // Запускаем рекурсию, до победного, если это
надо.
    }
  }
}
```

Результат выполнения программы:



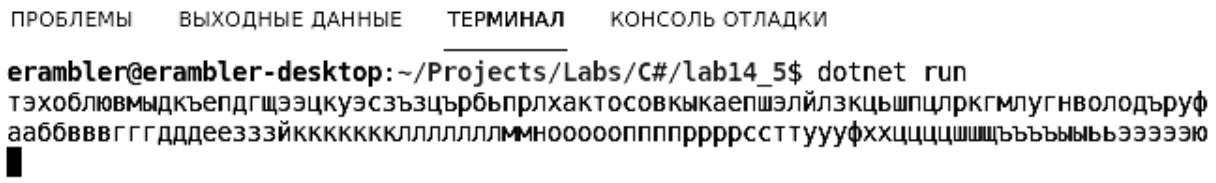
Заключение: В результаты выполнения работы был освоен метод “пузырьковой” сортировки, освоено понимание реализации пространства имён на языке C#, а также, работа с двумерными массивами. Цель работы была достигнута.

Контрольная работа № 5
Тема работы: методы сортировки данных
Цель работы: изучение методов сортировки и приобретение практических навыков в программировании данных методов.
Формулировка задания: написать программу на языке C#, выполняющую задачу сортировки массива строчных русских букв методом слияния.

Решение

```
using System;
namespace ArrayBubble
{
    class Program
    {
        static void Main()
        {
            const int n = 80;           //  длина массива
            Random rnd = new Random();
            char[] arr = new char[n];   // 1) выделение памяти под массив;
            for ( int i = 0; i < n; i++)// 2) заполнение неупорядоченными данными;
                arr[i] = Convert.ToChar(rnd.Next(1072,1103));
            Console.WriteLine(arr);        // 3) вывод неупорядоченных данных;
            MergeSort (arr,0,arr.Length-1);// 4) сортировка массива указанным
методом;
            Console.WriteLine(arr);        // 5) вывод упорядоченных данных.
            Console.ReadKey();
        }
        static void MergeSort(char[] A, int low, int high)
                            // Взято из решения, описанного в методическом пособии.
        {                   // Изменен тип получаемого массива с int на char
            if (low < high)
            {
                int center = (low + high) / 2;
                MergeSort(A, low, center);
                MergeSort(A, center + 1, high);
                Merge(A, low, center, high);
            }
        }
        static void Merge(char[] A, int low, int center, int high)
        {
            int i = low, j = center + 1, tmpPos = 0;
            char[] tmp = new char[high - low + 1];
            while (i <= center && j <= high)
            {
                if (A[i] < A[j]) tmp[tmpPos++] = A[i++];
                else tmp[tmpPos++] = A[j++];
            }
            while (j <= high) tmp[tmpPos++] = A[j++];
            while (i <= center) tmp[tmpPos++] = A[i++];
            for (tmpPos = 0; tmpPos < tmp.Length; tmpPos++) A[low + tmpPos] =
tmp[tmpPos];
        }
    }
}
```

Результат выполнения программы:



Дополнительный комментарий: Работа выполнена в системе с кодовой страницей unicode, поэтому использованы числа, кодирующие символы от 1072 до 1103. В случае с таблицей ansi-1251 нужно использовать 224 - 255

Заключение: В результате проведенной работы получены навыки работы с системой документации компании Microsoft, а именно - описание использованных классов. Изучены методы класса Random. Использован алгоритм сортировки. Основная цель работы достигнута.