Федеральное государственное образовательное бюджетное учреждение

высшего профессионального образования

«Нижегородский Государственный Университет им. Н.И. Лобачевского» (ННГУ)

Институт Информационных Технологий Математики и Механики

Отчёт по лабораторной работе

Генерирование случайных чисел и работа с ними

Выполнил:

студент группы 3821Б1ФИ3

Канаков Р.А.

Проверил:

заведующий лабораторией суперкомпьютерных технологий и

высокопроизводительных вычислений

Лебедев И.Г.

Нижний Новгород 2021г.

Содержание

[Введение 3](#_Toc87296705)

[Постановка задачи 3](#_Toc87296706)

[Руководство пользователя 4](#_Toc87296707)

[Руководство программиста 5](#_Toc87296708)

[Эксперимент 10](#_Toc87296709)

[Заключение 11](#_Toc87296710)

[Список литературы 11](#_Toc87296711)

## 

## Введение

Программирование — процесс и искусство создания компьютерных программ.

Си - компилируемый статически типизированный язык программирования общего назначения, разработанный в 1969—1973 годах сотрудником Bell Labs Деннисом Ритчи.

Сортировка массива — процесс изменения расположения отдельных элементов массива в определённом виде логической последовательности в соответствии с рядом правил, которые определенны пользователем. Первые прототипы современных методов сортировки появились уже в XIX веке.

## Постановка задачи

Сравнение сортировок.

Реализовать сортировки массивов данных float, задаваемых: обязательно случайно, дополнительно с клавиатуры или из файла.

Реализовать сортировки: пузырьком, вставкой, быстрая.

Сравнить время работы, сделать выводы.

Первая программа создает текстовый файл с записанными в него числами. Программа принимает количество чисел n, максимальное и минимальное значение.

Вторая программа читает текстовый файл с набором чисел, выводит консольный интерфейс (печать, сортировка, сброс, выход), выполняет выбранные действия.

**Руководство пользователя**

1 программа.

Перед использованием необходимо задать количество элементов. Это можно сделать с помощью аргумента командной строки.

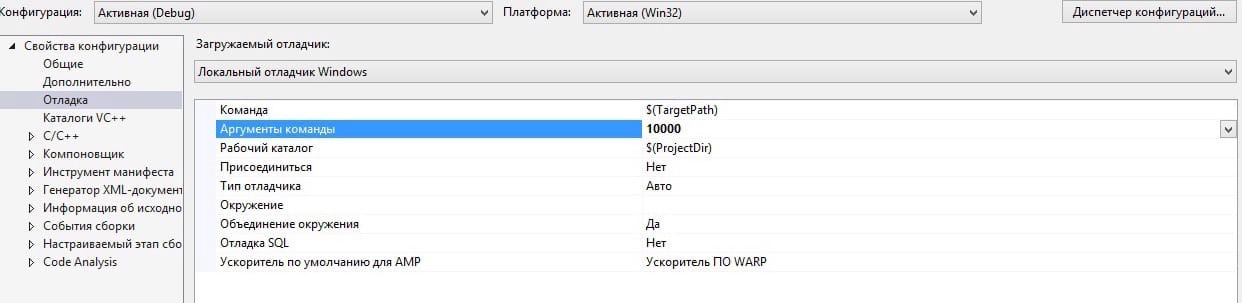


Рис. 1. Пример ввода количества элементов в Visual Studio.

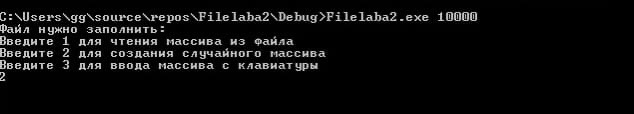


Рис. 2. Пример ввода количества элементов в консольном приложении.

При запуске нас встречает консольный интерфейс. Программа создает файл и дается 3 варианта чтобы его заполнить:

1. Считать из файла
2. Заполнить случайными числами из диапазона, который вводится пользователем
3. Ввести числа с клавиатуры

После заполнения программа завершается.

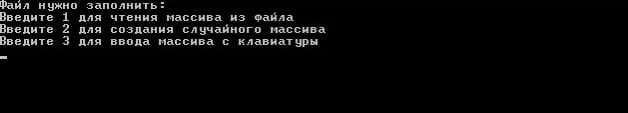


Рис.3. Выбор варианта заполнения файла.

2 программа.

Аналогично с первой программой, во второй необходимо задать количество элементов, а также ввести 0 или 1 для запрета или разрешения делать сортировки соответственно.

После запуска появляется консольный интерфейс, а также массив заполняется числами из файла.

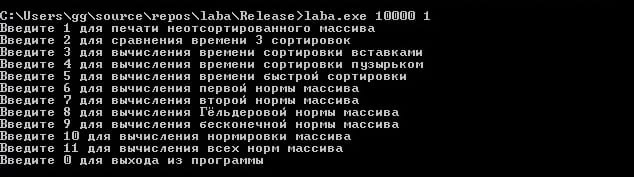


Рис. 4. Выбор работы над массивом.

1. Распечатывается неотсортированный массив
2. На экран выводится время трех сортировок (вставками, пузырьком и быстрая)

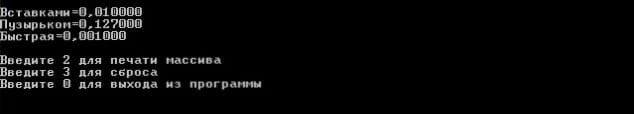


Рис. 5. Вывод времени трёх сортировок.

После вывода, мы можем

* Напечатать отсортированный массив
* Сбросить все до момента начала программы
* Завершить программу

1. Третий четвертый и пятый варианты практически совпадают со вторым, однако, в 3 мы выведем только время сортировки вставками, в 4 – пузырьком, в 5 – быстрой.

В 6, 7, 8, 9, 10 вариантах мы будем считать и выводить нормы массива.

В 11 мы выведем сразу все.

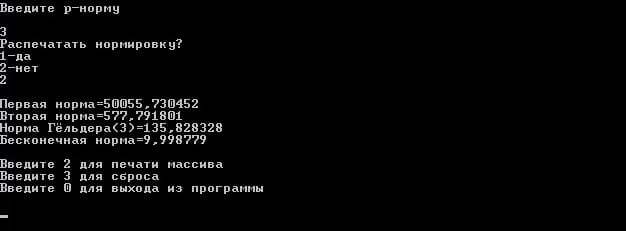


Рис. 6. Вывод всех норм массива(не включая нормировку).

# Руководство программиста

***Описание структуры кода программы***

Программа содержит два модуля: laba и Filelaba2.

1. Данная функция печатает массив

|  |
| --- |
| void PrintMas(float\* a, int n); |

1. Функция для очищения командной строки

|  |
| --- |
| void clrscr(); |

1. Сортировка Хоара (быстрая)

|  |
| --- |
| void Bistro(float\* a, int n); |

1. Сортировка Пузырьком

|  |
| --- |
| void Puzirek(float\* a, int n); |

1. Сортировка вставками

|  |
| --- |
| void Vstavki(float\* a, int n); |

**Описание структуры данных**

Массив находится в файле с расширением .txt. В нем находятся числа типа float.

Во второй программе содержатся 4 массива типа float, 1 для хранения неотсортированного массива, можно было обойтись еще 1 массивом, но нам необходимы еще 3, чтобы за 1 раз отсортировать массив тремя разными сортировками.

**Описание алгоритмов**

1. Алгоритм сортировки «пузырьком».

Во всех случаях осуществляется n2 проходов по массиву (где n – количество элементов массива, асимптотическая сложность – O(n2)). За первую итерацию максимальный элемент в массиве оказывается на последнем месте в массиве. За n итераций все элементы будут отсортированы.

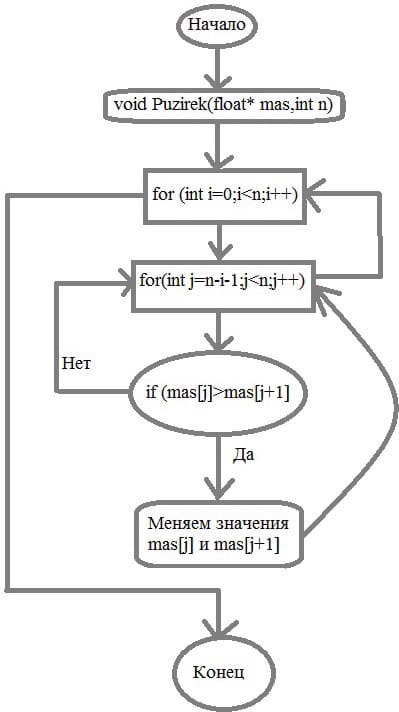


Рис. 7. Блок – схема сортировки «пузырьком».

1. Алгоритм сортировки «вставками».

Первым делом проверяем первые два элемента, если первый больше второго, то меняем их местами. После этого увеличиваем количество рассматриваемых элементов на 1, т. е. на втором шаге мы рассматриваем первые три элемента и так далее. На i-ом шаге алгоритма, если рассмотреть массив слева относительно i (где i – номер элемента), то мы увидим отсортированный массив. Далее, путем операций сравнения, вставляем i-ый элемент на свое место и так далее.

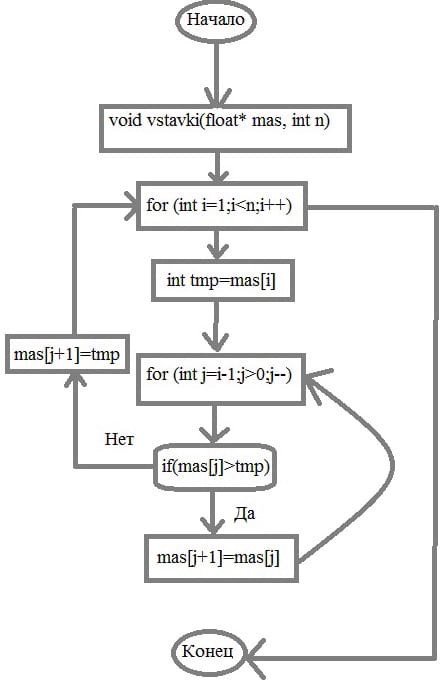


Рис. 8. Блок – схема сортировки вставками.

1. Алгоритм быстрой сортировки или сортировки Хоара.

Один из самых быстрых известных универсальных алгоритмов сортировки массивов: в среднем O(n\*log n) обменов при упорядочении n элементов; из-за наличия ряда недостатков на практике обычно используется с некоторыми доработками. Отличительной особенностью быстрой сортировки является операция разбиения массива на две части относительно опорного элемента. Например, если последовательность требуется упорядочить по возрастанию, то в левую часть будут помещены все элементы, значения которых меньше значения опорного элемента, а в правую элементы, чьи значения больше или равны опорному. Вне зависимости от того, какой элемент выбран в качестве опорного, массив будет отсортирован, но все же наиболее удачным считается ситуация, когда по обеим сторонам от опорного элемента оказывается примерно равное количество элементов. Если длина какой-то из получившихся в результате разбиения частей превышает один элемент, то для нее нужно рекурсивно выполнить упорядочивание, т. е. повторно запустить алгоритм на каждом из отрезков.

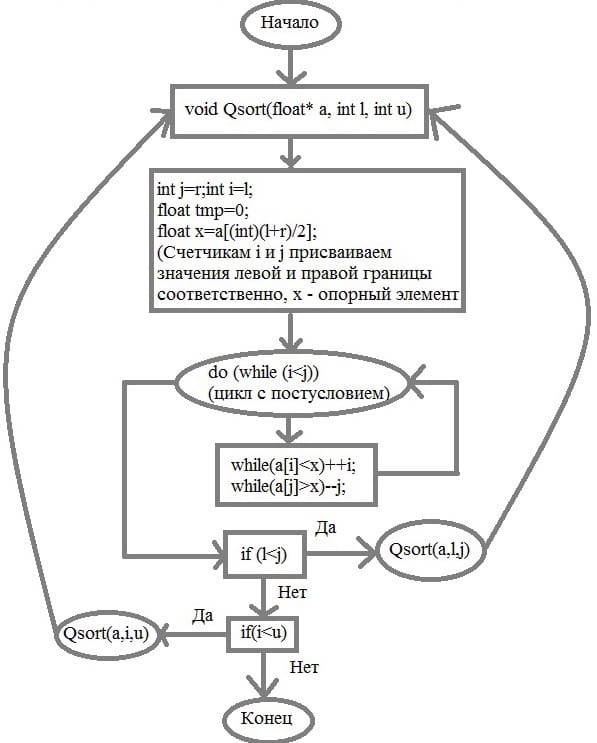


Рис. 9. Блок-схема быстрой сортировки.

Формула р-нормы (норма Гёльдера) вектора:

где x – исходный массив, i – индекс вектора (номер элемента вектора), n – количество элементов вектора.

Норму при p = 1 называют *норма-сумма*, при р = 2 будет вторая норма (евклидова норма) вектора. А при p → ∞ будет формула: xi

Формула вычисления нормировки вектора (каждый элемент вектора разделим на результат данной формулы):

**Эксперимент**

Таблица 1. Результат работы программы (все числа в диапазоне от -300 до 300).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Количество элементов | 100’000 | 250’000 | 500’000 | 1’000’000 |
| Вставками, сек | 1,005 | 6,266 | 25,882 | 127,501 |
| Пузырьком, сек | 16,335 | 99,224 | 400,005 | 1655,011 |
| Быстрая, сек | 0,007 | 0,018 | 0,037 | 0,071 |

## Заключение

Удалось создать две программы для сравнения сортировок массива типа float.

Первая программа создаёт текстовый файл с записанными в него числами. Программа может создавать массив не только из случайных чисел (принимая количество чисел, максимальное и минимальное значения), но и получать их, задавая вручную, (с клавиатуры) и из файла (с уже готовым набором значений).

Вторая программа читает текстовый файл с набором чисел, созданный первой программой, выводит консольный интерфейс и реализует сортировки: пузырьком, вставкой, быстрая. Также нужно определить время работы сортировок.

Так же удалось реализовать:

* + 1. Возможность запуска сортировки и запись количества элементов массива через параметры командной строки.
    2. Вычисление первой нормы массива.
    3. Вычисление второй нормы массива.
    4. Вычисление Гёльдеровой нормы массива.
    5. Вычисление бесконечной нормы массива.
    6. Вычисление нормировки массива.

## Список литературы

1.C/C++. Программирование на языке высокого уровня / Т. А. Павловская. - СПб.: Питер, 2003. 461 с.: ил.

2.Лекции Лебедева Ильи Геннадьевича:

<https://cloud.unn.ru/s/CckzTzAAKNGqwNq>

<https://cloud.unn.ru/s/tZ8GY5TW8bGYaKb>

**Приложения**

***Приложение 1***

|  |
| --- |
| #include <stdlib.h>  #include <stdio.h>  #include <math.h>  #include <locale.h>  int main(int argc, char\*\* argv)  {  int n = 0, max = 0, min = 0, a = 0;  float c = 0, l = 0;  setlocale(LC\_ALL, "Rus");  sscanf\_s(argv[1], "%d", &n);  FILE\* fo;  FILE\* fo1;  errno\_t fa, fa1;  float\* mas = (float\*)malloc(n \* sizeof(float));  printf("Файл нужно заполнить:\n");  printf("Введите 1 для чтения массива из файла\n");  printf("Введите 2 для создания случайного массива\n");  printf("Введите 3 для ввода массива с клавиатуры\n");  scanf\_s("%d", &a);  if (a == 1)  {  fa = fopen\_s(&fo, "C:\\Users\\gg\\source\\repos\\Filelaba2\\output1.txt", "r");  for (int i = 0; i < n; i++)  fscanf\_s(fo, "%f", &(mas[i]));  fclose(fo);  fa1 = fopen\_s(&fo1, "C:\\Users\\gg\\source\\repos\\laba\\output.txt", "w");  for (int i = 0; i < n; i++)  fprintf(fo1, "%f\n", mas[i]);  fclose(fo1);  }  if (a == 2)  {  printf("Введите минимум ");  scanf\_s("%d", &min);  printf("Введите максимум ");  scanf\_s("%d", &max);  fa = fopen\_s(&fo, "C:\\Users\\gg\\source\\repos\\laba\\output.txt", "w");  for (int i = 0; i < n; i++)  {  float b = 0;  b = (float)rand();  l = (float)((b / RAND\_MAX) \* (max - min) + min);  fprintf(fo, "%f\n", l);  }  fclose(fo);  }  if (a == 3)  {  fa = fopen\_s(&fo, "C:\\Users\\gg\\source\\repos\\laba\\output.txt", "w");  for (int i = 0; i < n; i++)  {  scanf\_s("%f", &c);  fprintf(fo, "%f\n", c);  }  fclose(fo);  }  free(mas);  return 0;  } |

***Приложение 2***

|  |
| --- |
| #include "Header.h"  #include <stdlib.h>  #include <stdio.h>  #include <math.h>  #include <locale.h>  #include <time.h>  void Qsort(float\* mas, int l, int u)  {  int i = l;  int j = u;  float tmp = 0;  float x = mas[(int)((l + u) / 2)];  do  {  while (mas[i] < x)++i;  while (mas[j] > x)--j;  if (i <= j)  {  tmp = mas[i];  mas[i] = mas[j];  mas[j] = tmp;  i++;  j--;  }  } while (i < j);  if (l < j)  Qsort(mas, l, j);  if (i < u)  Qsort(mas, i, u);  }  void Bistro(float\* a, int n)  {  Qsort(a, 0, n - 1);  }  void Puzirek(float\* a, int n)  {  int i = 0, j = 0;  float tmp = 0;  for (; i < n; i++)  {  for (j = 0; j < n - i - 1; j++)  {  if (a[j] > a[j + 1])  {  tmp = a[j];  a[j] = a[j + 1];  a[j + 1] = tmp;  }  }  }  }  void Vstavki(float\* a, int n)  {  int i = 1, j = 0;  float tmp = 0;  for (; i < n; i++)  {  tmp = a[i];  for (j = i - 1; j > 0; j--)  {  if (a[j] > tmp)  a[j + 1] = a[j];  else  break;  }  a[j + 1] = tmp;  }  }  void PrintMas(float\* a, int n)  {  for (int i = 0; i < n; i++)  printf("%f\n", a[i]);  printf("\n");  }  void clrscr()  {  system("@cls || clear");  } |

***Приложение 3***

|  |
| --- |
| #ifndef \_\_HEADER\_H\_\_  #define \_\_HEADER\_H\_\_  void Bistro(float\* a, int n);  void Puzirek(float\* a, int n);  void Vstavki(float\* a, int n);  void PrintMas(float\* a, int n);  void clrscr();  #endif |

***Приложение 4***

|  |
| --- |
| #include "Header.h"  #include <stdlib.h>  #include <stdio.h>  #include <math.h>  #include <locale.h>  #include <time.h>  int main(int argc, char\*\* argv)  {  FILE\* fo;  errno\_t fa;    setlocale(LC\_ALL, "Russian");  int a = 1, b = 0, min = 0, max = 0, n = 0, k = 0, f = 0, l = 0;    sscanf\_s(argv[1], "%d", &n);  sscanf\_s(argv[2], "%d", &k);  if (k != 1)  k = 0;  float\* mas = (float\*)malloc(n \* sizeof(float));  float\* mas1 = (float\*)malloc(n \* sizeof(float));  float\* mas2 = (float\*)malloc(n \* sizeof(float));  float\* mas0 = (float\*)malloc(n \* sizeof(float));  for (int i = 0; i < n; i++)  {  mas[i] = 0;  mas1[i] = 0;  mas2[i] = 0;  mas0[i] = 0;  }    fa = fopen\_s(&fo, "C:\\Users\\gg\\source\\repos\\laba\\output.txt", "r");  for (int i = 0; i < n; i++)  fscanf\_s(fo, "%f", &(mas[i]));  fclose(fo);  for (int i = 0; i < n; i++)  {  mas1[i] = mas[i];  mas2[i] = mas[i];  mas0[i] = mas[i];  }  while (a != 0)  {  printf("Введите 1 для печати неотсортированного массива\n");  printf("Введите 2 для сравнения времени 3 сортировок\n");  printf("Введите 3 для вычисления времени сортировки вставками\n");  printf("Введите 4 для вычисления времени сортировки пузырьком\n");  printf("Введите 5 для вычисления времени быстрой сортировки\n");  printf("Введите 6 для вычисления первой нормы массива\n");  printf("Введите 7 для вычисления второй нормы массива\n");  printf("Введите 8 для вычисления Гёльдеровой нормы массива\n");  printf("Введите 9 для вычисления бесконечной нормы массива\n");  printf("Введите 10 для вычисления нормировки массива\n");  printf("Введите 11 для вычисления всех норм массива\n");  printf("Введите 0 для выхода из программы\n\n");  scanf\_s("%d", &a);  if (a != 1 && a != 2 && a != 3 && a != 4 && a != 5 && a != 6 && a != 7 && a != 8 && a != 9 && a != 10 && a != 11 && a != 0)  {  clrscr();  printf("Не то вводишь\n\n");  }  if ((k == 0) && ((a == 5) || (a == 2) || (a == 3) || (a == 4)))  {  clrscr();  printf("Сортировки делать нельзя(\n\n");  a = 12;  }  if (a == 1)  {  clrscr();  PrintMas(mas, n);  }  if (a == 2)  {  clrscr();  clock\_t start1 = clock();  Vstavki(mas, n);  clock\_t end1 = clock();  printf("Вставками=%lf\n", (double)(end1 - start1) / CLOCKS\_PER\_SEC);  clock\_t start2 = clock();  Puzirek(mas1, n);  clock\_t end2 = clock();  printf("Пузырьком=%lf\n", (double)(end2 - start2) / CLOCKS\_PER\_SEC);  clock\_t start3 = clock();  Bistro(mas2, n);  clock\_t end3 = clock();  printf("Быстрая=%lf\n\n", (double)(end3 - start3) / CLOCKS\_PER\_SEC);  }  if (a == 3)  {  clrscr();  clock\_t start1 = clock();  Vstavki(mas, n);  clock\_t end1 = clock();  printf("Вставками=%lf\n\n", (double)(end1 - start1) / CLOCKS\_PER\_SEC);  }  if (a == 4)  {  clrscr();  clock\_t start2 = clock();  Puzirek(mas, n);  clock\_t end2 = clock();  printf("Пузырьком=%lf\n\n", (double)(end2 - start2) / CLOCKS\_PER\_SEC);  }  if (a == 5)  {  clrscr();  clock\_t start3 = clock();  Bistro(mas, n);  clock\_t end3 = clock();  printf("Быстрая=%lf\n\n", (double)(end3 - start3) / CLOCKS\_PER\_SEC);  }  if (a == 6)  {  clrscr();  long double x = 0;  for (int i = 0; i < n; i++)  x += fabs(mas[i]);  printf("Первая норма=%lf\n\n", x);  }  if (a == 7)  {  clrscr();  long double x = 0, w = 0;  for (int i = 0; i < n; i++)  w += pow(mas[i], 2);  x = sqrt(w);  printf("Вторая норма=%lf\n\n", x);  }  if (a == 8)  {  clrscr();  long double x = 0, w = 0;  double p = 0;  printf("Введите p-норму\n\n");  scanf\_s("%lf", &p);  if (p < 1)  break;  for (int i = 0; i < n; i++)  w += pow(fabs(mas[i]), p);  x = pow(w, (long double)(1 / p));  printf("\nНорма Гёльдера(%.0lf)=%lf\n\n", p, x);  }  if (a == 9)  {  clrscr();  Bistro(mas, n);  printf("Бесконечная норма=%f\n\n", mas[n - 1]);  }  if (a == 10)  {  clrscr();  long double x = 0, w = 0;  for (int i = 0; i < n; i++)  w += pow(mas[i], 2);  x = sqrt(w);  printf("Нормировка:\n");  for (int i = 0; i < n; i++)  printf("%f\n", mas[i] \* (1.0 / x));  printf("\n");  }  if (a == 11)  {  clrscr();  double x1 = 0, x2 = 0, x3 = 0, x4 = 0, w2 = 0, w3 = 0, w4 = 0, p = 0;  printf("Введите p-норму\n\n");  scanf\_s("%lf", &p);  if (p < 1)  break;  //Первая норма  for (int i = 0; i < n; i++)  x1 += fabs(mas[i]);  //Вторая норма  for (int i = 0; i < n; i++)  w2 += pow(mas[i], 2);  x2 = sqrt(w2);  //Норма Гёльдера  for (int i = 0; i < n; i++)  w3 += pow(fabs(mas[i]), p);  x3 = pow(w3, 1.0 / p);  //Нормировка  for (int i = 0; i < n; i++)  w4 += pow(mas[i], 2);  x4 = sqrt(w4);  printf("\nРаспечатать нормировку?\n1-да\n2-нет\n\n");  scanf\_s("%d", &a);  if (a == 1)  {  printf("Нормировка:\n");  for (int i = 0; i < n; i++)  printf("%f\n", mas[i] \* (1.0 / x4));  }  if (a == 2)  a = 12;  //Бесконечная норма  Bistro(mas, n);  printf("\nПервая норма=%lf\n", x1);  printf("Вторая норма=%lf\n", x2);  printf("Норма Гёльдера(%.0lf)=%lf\n", p, x3);  printf("Бесконечная норма=%f\n\n", mas[n - 1]);  }  if (a == 5 || a == 2 || a == 3 || a == 4)  {  printf("Введите 2 для печати массива\n");  printf("Введите 3 для сброса\n");  printf("Введите 0 для выхода из программы\n\n");  scanf\_s("%d", &a);  clrscr();  }  if (a == 2)  {  clrscr();  PrintMas(mas, n);  printf("\nВведите 3 для сброса\n");  printf("Введите 0 для выхода из программы\n\n");  scanf\_s("%d", &a);  clrscr();  }  for (int i = 0; i < n; i++)  {  mas[i] = mas0[i];  mas1[i] = mas0[i];  mas2[i] = mas0[i];  }  }    free(mas);  free(mas1);  free(mas2);  free(mas0);  return 0;  } |