Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского"

Институт информационных технологий, математики и механики

Отчёт по лабораторной работе

Сортировка массивов

Выполнил:

студент ф-та ИТММ гр. 3821Б1ФИ3

Дурандин В.Е.

Проверил:

зав. лабораторией СТиВВ

Лебедев И.Г.

Нижний Новгород

2021 г.

Содержание

[Введение 3](#_Toc89985382)

[1. Постановка задачи 4](#_Toc89985383)

[2. Руководство пользователя 5](#_Toc89985384)

[3. Руководство программиста 7](#_Toc89985385)

[ Описание структуры программы 7](#_Toc89985386)

[ Описание структур данных 7](#_Toc89985387)

[ Описание алгоритмов 8](#_Toc89985388)

[4. Эксперименты 12](#_Toc89985389)

[5. Заключение 13](#_Toc89985390)

[6. Литература 14](#_Toc89985391)

[7. Приложения 15](#_Toc89985392)

[ Приложение 1 15](#_Toc89985393)

[ Приложение 2 16](#_Toc89985394)

[ Приложение 3 17](#_Toc89985395)

[ Приложение 4 18](#_Toc89985396)

# Введение

Программирование — процесс и искусство создания компьютерных программ с помощью языков программирования.

В узком смысле слова, программирование рассматривается как кодирование — реализация одного или нескольких взаимосвязанных алгоритмов на некотором языке программирования. В более широком смысле, программирование — процесс создания программ, то есть разработка программного обеспечения.

Язык программирования Cи был создан Деннисом Ритчи из AT&T Bell Laboratories как язык программирования высокого уровня для разработки операционной системы UNIX. В настоящее время язык широко используется в различных областях. C особенно популярен у системных программистов, потому что позволяет писать программы просто и кратко.

Сортировка массива — процесс принятия и расположение отдельных элементов массива в определённом виде логической последовательности в соответствии с рядом правил, которые определенны пользователем. Процесс включает в себя шаги через массив, один элемент за один раз, и тестирование элемента против окружающих элементов, чтобы определить, требуется ли его переместить в другой индекс в массиве.

# Постановка задачи

Создать две программы для сравнения сортировок массива типа unsigned char.

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.

Помимо двух программ нужно реализовать:

* + 1. Возможность запуска сортировки через параметры командной строки.
    2. Вычисление первой нормы вектора (массива).
    3. Вычисление второй нормы вектора (массива).
    4. Вычисление Гельдеровой нормы вектора (массива).
    5. Вычисление бесконечной нормы вектора (массива).
    6. Нормировку вектора (массива).

# Руководство пользователя

При запуске обеих программ нас встречает консольный интерфейс.

* **Программа 1**

От пользователя требуется ввести цифру от 1 до 3 для задания массива и дальнейшей записи его в файл.

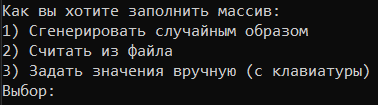


Рисунок 1. Выбор задания массива

Выбор 1 (Сгенерировать случайным образом): Пользователю нужно ввести количество элементов, максимальное и минимальное значения (диапазон) самому вручную, а дальше сгенерируются сами числа случайным образом.

Выбор 2 (Считать из файла): Пользователю требуется заранее создать текстовый файл с записанными в него данными, а именно: количество чисел, минимальное и максимальное значения диапазона, заданное количество чисел из диапазона.

Выбор 3 (Задать значения вручную (с клавиатуры)): Пользователь сам задаёт количество чисел, диапазон, и сами числа из диапазона.

* **Программа 2**

После задания массива из первой программы, мы можем дальше работать с числами, а именно с их сортировкой.

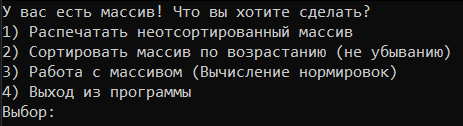


Рисунок 2. Выбор работы с массивом

Выбор 1 (Распечатать неотсортированный массив): Пользователь может распечатать ранее заданный массив, чтобы убедиться, что массив не отсортирован, а после уже отсортировать его.

Выбор 2 (Сортировать массив по возрастанию (неубыванию)): Пользователю предлагается 3 варианта сортировки и вариант вернуться назад, чтобы была возможность распечатать ещё неотсортированный массив или выйти из программы. При выборе одной из сортировок произойдёт сортировка и вывод массива, а также время работы сортировки.:

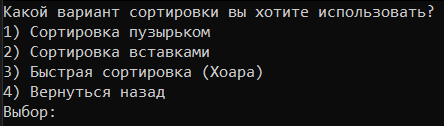


Рисунок 3. Выбор варианта сортировки

Выбор 3 (Работа с массивом (Вычисление нормировок): Пользователь может рассчитать некоторые нормы вектора, при нажатии одного из вариантов выведется результат работы:

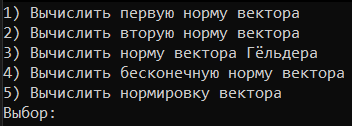


Рисунок 4. Выбор варианта вычисления нормы вектора

Выбор 4 (Выход из программы): Судя по названию, мы понимаем, что произойдёт завершение программы.

# Руководство программиста

## Описание структуры программы

Программа содержит два модуля: Project\_1 и Project\_2.

1. Функция позволяет выделить память под массив и заполнить его случайными числами из заданного диапазона. (Project\_1)

|  |
| --- |
| void create\_mas(int n, int min, int max, unsigned char\*\* mas) |

1. Функция помогает пользователю задать количество чисел и диапазон. (Project\_1)

|  |
| --- |
| void gen\_range(int \*n, int \*min, int \*max) |

1. Функция записывает числа в файл. (Project\_1)

|  |
| --- |
| void print\_mas(FILE\* f, int n, unsigned char\* mas) |

1. Функция сортировки массива пузырьком. (Project\_2)

|  |
| --- |
| void bubble\_sort(unsigned char\* mas, int size) |

1. Функция сортировки вставками. (Project\_2)

|  |
| --- |
| void insert\_sort(unsigned char\* mas, int size) |

1. Функция сортировки Хоара (быстрая). (Project\_2)

|  |
| --- |
| void quick\_sort(unsigned char\* mas, int left, int right) |

1. Функция вычисления первой нормы вектора. (Project\_2)

|  |
| --- |
| void firstNorma(unsigned char\* mas, int size) |

1. Функция вычисления второй нормы вектора. (Project\_2)

|  |
| --- |
| void secondNorma(unsigned char\* mas, int size) |

1. Функция вычисления Гёльдеровой нормы вектора. (Project\_2)

|  |
| --- |
| void pNorma(unsigned char\* mas, int size, double p) |

1. Функция вычисления бесконечной нормы вектора. (Project\_2)

|  |
| --- |
| void infNorma(unsigned char\* mas, int size) |

1. Функция вычисления нормировки вектора. (Project\_2)

|  |
| --- |
| void normirovka(unsigned char\* mas, int size) |

## Описание структур данных

Массив находится в файле с расширением txt. Формат файла: первая строка содержит число – количество чисел массива, вторая строка содержит 2 числа – диапазон чисел (минимальное и максимальное), с третьей до конечной строки находятся числа типа unsigned char в заданном диапазоне.

Во второй программе используются два массива. Первый массив содержит исходные (неотсортированные) числа, а второй, соответственно, содержит уже отсортированный исходный массив.

## Описание алгоритмов

Консольный интерфейс в обеих программах реализован с помощью цикла while и switch внутри него. Цикл while нужен для проверки ввода пользователем значения.

Алгоритм сортировки «Пузырьком» состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции как пузырёк в воде, отсюда и название алгоритма).

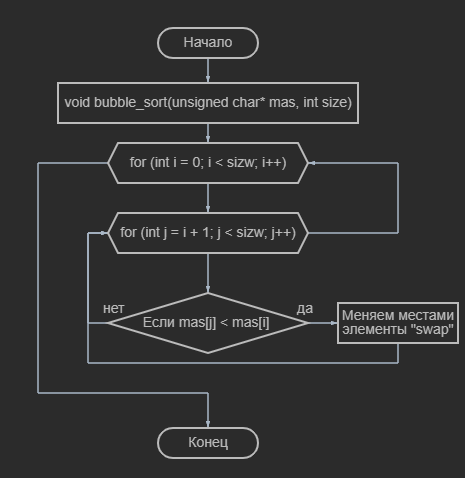


Рисунок 5. Сортировка пузырьком блок-схема

Сортировка вставками – простой алгоритм сортировки, преимущественно использующийся в учебном программировании. К положительной стороне метода относится простота реализации, а также его эффективность на частично упорядоченных последовательностях, и/или состоящих из небольшого числа элементов. Тем не менее, высокая вычислительная сложность не позволяет рекомендовать алгоритм в повсеместном использовании.

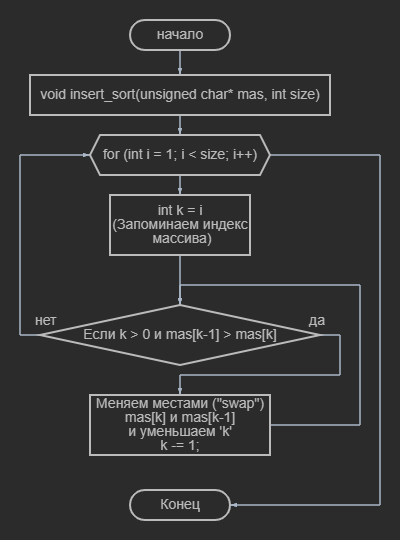


Рисунок 6. Сортировка вставками блок-схема

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.

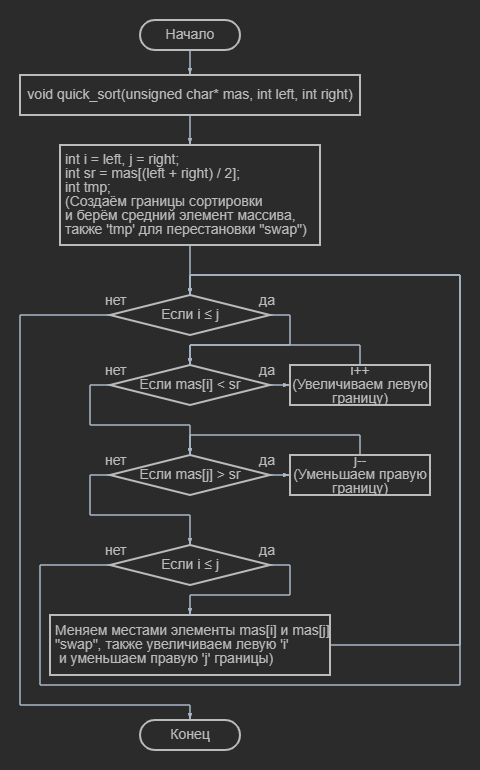
Отличительной особенностью быстрой сортировки является операция разбиения массива на две части относительно опорного элемента. Например, если последовательность требуется упорядочить по возрастанию, то в левую часть будут помещены все элементы, значения которых меньше значения опорного элемента, а в 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.

Рисунок 7. Быстрая сортировка Хоара блок-схема

Формула р-нормы (норма Гёльдера) вектора:

(1)

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

(2)

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

(3)

(4)

# Эксперименты

Таблица 1. Результаты работы программы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кол-во элементов | 100 000 | 250 000 | 500 000 | 1 000 000 |
| min : max | 0:255 | 0:255 | 0:255 | 0:255 |
| Сорт. Пузырьком, сек | 10.232 | 63.414 | 253.931 | N/A |
| Сорт. Вставками, сек | 8.395 | 52.66 | 219.639 | N/A |
| Быстрая сорт., сек | 0.008 | 0.021 | 0.041 | 0.084 |

Чем больше элементов, тем сортировка будет медленнее работать. Заметим, что самой быстрой сортировкой оказалась сортировка Хоара. Уже начиная с 250 000 элементов, первые две сортировки заняли много времени несмотря на то, что диапазон чисел не очень большой. Если количество элементов небольшое, то вполне спокойно можно использовать любую из данных сортировок. Но начиная с некоторого момента, сортировки пузырьком и вставками уже являются неэффективными, поэтому следует использовать сортировку Хоара, так как даже при большом значении элементов она очень быстро справляется.

# Заключение

Удалось создать две программы для сравнения сортировок массива типа unsigned char.

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 через параметры командной строки.
    2. Вычисление первой нормы вектора (массива).
    3. Вычисление второй нормы вектора (массива).
    4. Вычисление Гельдеровой нормы вектора (массива).
    5. Вычисление бесконечной нормы вектора (массива).
    6. Нормировку вектора (массива).

# Литература

1. Лекции Сысоева Александра Владимировича, 2021.
2. Практики и лабораторные Лебедева Ильи Геннадьевича, 2021.
3. С (язык программирования) – Режим доступа: <https://ru.wikipedia.org/wiki/Си_(язык_программирования)> – Загл. с экрана
4. Сортировка пузырьком – Режим доступа: <https://ru.wikipedia.org/wiki/Сортировка_пузырьком> - Загл. с экрана
5. Сортировка вставками – Режим доступа: <https://ru.wikipedia.org/wiki/Сортировка_вставками> - Загл. с экрана
6. Быстрая сортировка Хоара – Режим доступа: <https://ru.wikipedia.org/wiki/Быстрая_сортировка> - Загл. с экрана
7. Норма (математика) – Режим доступа:

<https://ru.wikipedia.org/wiki/Норма_(математика)> – Загл. с экрана

# Приложения

## Приложение 1

|  |
| --- |
| #include <stdio.h>  #include <locale.h>  #include <stdlib.h>  #include <time.h>  void create\_mas(int n, int min, int max, unsigned char\*\* mas) {  int i = 0;  \*mas = (unsigned char\*\*)malloc(n \* sizeof(unsigned char\*));  for (i = 0; i < n; i++)  (\*mas)[i] = rand() \* 1.0 / RAND\_MAX \* (max - min) + min;  }  void gen\_range(int \*n, int \*min, int \*max){  do {  printf("Введите количество чисел, которые вы хотите сгенерировать:\n");  scanf\_s("%d", n);  printf("Введите минимальное число диапазона:\n");  scanf\_s("%d", min);  printf("Введите максимальное число диапазона:\n");  scanf\_s("%d", max);  if (\*min > \*max || \*n <= 0 || \*min < 0 || \*max < 0) {  system("CLS");  printf("Вы неправильно задали значения, повторите снова!\n\n");  }  } while ((\*min > \*max) || (\*n <= 0) || (\*min < 0) || (\*max < 0));  }  void print\_mas(FILE\* f, int n, unsigned char\* mas) {  int i = 0;  for (i = 0; i < n; i++) fprintf(f, "%d\n", mas[i]);  }  int main()  {  int i, n, min, max, input;  unsigned char\* mas = 0;  setlocale(LC\_ALL, "Rus");  srand(time(NULL));  FILE\* file1 = 0;  FILE\* file2 = 0;  fopen\_s(&file1, "..\\text\_file.txt", "w");    do {  printf("Как вы хотите заполнить массив:\n1) Сгенерировать случайным образом\n2) Считать из файла\n3) Задать значения вручную (с клавиатуры)\nВыбор: ");  input = getch();  printf("%d\n", input - '0');    switch (input) {  case '1':  gen\_range(&n, &min, &max);  fprintf(file1, "%d\n%d %d\n", n, min, max);  create\_mas(n, min, max, &mas);  print\_mas(file1, n, mas);  fclose(file1);  break;  case '2':  fopen\_s(&file2, "..\\scanf.txt", "r");  fscanf\_s(file2, "%d\n%d", &n, &min);  fscanf\_s(file2, "%d\n", &max);  fprintf(file1, "%d\n%d %d\n", n, min, max);  mas = (unsigned char\*)malloc(n \* sizeof(unsigned char));  for (i = 0; i < n; i++) fscanf\_s(file2, "%d\n", &mas[i]);  print\_mas(file1, n, mas);  fclose(file1);  fclose(file2);  break;  case '3':  gen\_range(&n, &min, &max);  mas = (unsigned char\*)malloc(n \* sizeof(unsigned char));  printf("Введите значения:\n");  for (i = 0; i < n; i++) {  scanf\_s("%d", &mas[i]);  if (mas[i] < min || mas[i] > max) {  printf("Вы ввели значение не из нужного диапазона, перепишите число\n");  i -= 1;  }  }  fprintf(file1, "%d\n%d %d\n", n, min, max);  print\_mas(file1, n, mas);  break;  default:  system("CLS");  printf("Вы неправильно задали значения, повторите снова!\n\n");  }  } while (input > '3' && input < '1');  return 0;  } |

Исходный код программы Project\_1: input.c

## Приложение 2

|  |
| --- |
| #ifndef \_\_FUNCTIONS\_H\_\_  #define \_\_FUNCTIONS\_H\_\_  //Сортировки  void bubble\_sort(unsigned char\* mas, int size);  void insert\_sort(unsigned char\* mas, int size);  void quick\_sort(unsigned char\* mas, int left, int right);  void firstNorma(unsigned char\* mas, int size); // Первая норма вектора  void secondNorma(unsigned char\* mas, int size);  void pNorma(unsigned char\* mas, int size, double p); // Норма вектора Гёльдера  void infNorma(unsigned char\* mas, int size); // Бесконечная норма вектора  void normirovka(unsigned char\* mas, int size); // Нормировка массива  #endif |

Исходный код программы Project\_2: Functions.h

## Приложение 3

|  |
| --- |
| #include "Functions.h"  #include <stdio.h>  #include <stdlib.h>  #include <math.h>  #include <string.h>  void bubble\_sort(unsigned char\* mas, int size) { // Сортировка пузырьком  for (int i = 0; i < size; i++)  for (int j = i + 1; j < size; j++)  if (mas[j] < mas[i]) {  int tmp = mas[j];  mas[j] = mas[i];  mas[i] = tmp;  }  }  void insert\_sort(unsigned char\* mas, int size) { // Сортировка вставками  for (int i = 1; i < size; i++) {  int k = i;  while (k > 0 && mas[k - 1] > mas[k]) {  int tmp = mas[k - 1];  mas[k - 1] = mas[k];  mas[k] = tmp;  k -= 1;  }  }  }  void quick\_sort(unsigned char\* mas, int left, int right) { // Быстрая сортировка(Хоара)  int i = left, j = right;  int sr = mas[(left + right) / 2];  int tmp;  while (i <= j) {  while (mas[i] < sr) i++;  while (mas[j] > sr) j--;  if (i <= j) {  tmp = mas[i];  mas[i] = mas[j];  mas[j] = tmp;  i++;  j--;  }  }  if (left < j) quick\_sort(mas, left, j);  if (right > i) quick\_sort(mas, i, right);  }  void firstNorma(unsigned char\* mas, int size)  {  int i = 0, x = 0;  for (i = 0; i < size; i++) x += mas[i];  printf("%d\n", x);  }  void secondNorma(unsigned char\* mas, int size)  {  int i = 0, sum = 0;  for (i = 0; i < size; i++) sum += pow(mas[i], 2);  printf("Результат вычисления второй нормы вектора: %g\n", sqrt(sum));  }  void pNorma(unsigned char\* mas, int size, double p)  {  int i = 0, x = 0;  for (i = 0; i < size; i++) x += pow(fabs(mas[i]), p);  printf("Результат вычисления первой нормы вектора: %g\n", pow(x, 1.0 / p));  }  void infNorma(unsigned char\* mas, int size)  {  int i = 0;  unsigned char max = 0;  for (i = 0; i < size; i++) if (mas[i] > max) max = mas[i];  printf("Результат вычисления бесконечной нормы вектора: %d\n", max);  }  void normirovka(unsigned char\* mas, int size)  {  int i = 0;  double x = 0;  for (i = 0; i < size; i++) x += pow(mas[i], 2);  x = round(sqrt(x));  for (i = 0; i < size; i++) printf("%d / %g --> %g\n", mas[i], x, mas[i] \* 1.0 / x);  } |

Исходный код программы Project\_2: Functions.с

## Приложение 4

|  |
| --- |
| #include <stdio.h>  #include <locale.h>  #include <stdlib.h>  #include <math.h>  #include <string.h>  #include <time.h>  #include <stdbool.h>  #include "Functions.h"  int main(int argc, char\*\* argv)  {  setlocale(LC\_ALL, "Rus");  int n, min, max, input, p;  bool flag = 0;  unsigned char\* mas;  unsigned char\* arr;  FILE\* file;  fopen\_s(&file, "..\\text\_file.txt", "r");  fscanf\_s(file, "%d\n", &n);  fscanf\_s(file, "%d %d\n", &min, &max);  mas = (int\*)malloc(n \* sizeof(int));  arr = (int\*)malloc(n \* sizeof(int));  for (int i = 0; i < n; i++)  fscanf\_s(file, "%d\n", &mas[i]);  memcpy(arr, mas, n \* sizeof(unsigned char));  reset: printf("У вас есть массив! Что вы хотите сделать?\n1) Распечатать неотсортированный массив\n2) Сортировать массив по возрастанию (не убыванию)\n3) Работа с массивом (Вычисление нормировок)\n4) Выход из программы\nВыбор: ");  do {  if (flag == 0 && argc > 1 && argv[1][0] >= '1' && argv[1][0] <= '4') sscanf\_s(argv[1], "%d", &input);  else scanf\_s("%d", &input);  switch (input) {  case 1:  for (int i = 0; i < n; i++) printf("arr[%d] = %d\n", i, arr[i]);  system("pause");  case 2:  system("CLS");  printf("Какой вариант сортировки вы хотите использовать?\n1) Сортировка пузырьком\n2) Сортировка вставками\n3) Быстрая сортировка (Хоара)\n4) Вернуться назад\nВыбор: ");  do {  if (flag == 0 && argc > 2 && argv[2][0] >= '1' && argv[2][0] <= '4') sscanf\_s(argv[2], "%d", &input);  else scanf\_s("%d", &input);  flag = 1;  switch (input) {  case 1:  system("CLS");  clock\_t b\_time = clock();  bubble\_sort(mas, n);  clock\_t e\_time = clock() - b\_time;  printf("Сортировка массива пузырьком...\n");  for (int i = 0; i < n; i++) printf("mas[%d] = %d\n", i, mas[i]);  printf("Сортировка заняла %g сек.\n", (double)e\_time / CLOCKS\_PER\_SEC);  system("pause");  system("CLS");  goto reset;  case 2:  system("CLS");  b\_time = clock();  insert\_sort(mas, n);  e\_time = clock() - b\_time;  printf("Сортировка массива вставками...\n");  for (int i = 0; i < n; i++) printf("mas[%d] = %d\n", i, mas[i]);  printf("Сортировка заняла %g сек.\n", (double)e\_time / CLOCKS\_PER\_SEC);  system("pause");  system("CLS");  goto reset;  case 3:  system("CLS");  b\_time = clock();  quick\_sort(mas, 0, n - 1);  e\_time = clock() - b\_time;  printf("Сортировка массива методом Хоара(быстрая)...\n");  for (int i = 0; i < n; i++) printf("mas[%d] = %d\n", i, mas[i]);  printf("Сортировка заняла %g сек.\n", (double)e\_time / CLOCKS\_PER\_SEC);  system("pause");  system("CLS");  goto reset;  case 4:  system("CLS");  goto reset;  default:  printf("Вы ввели неверные значения, попробуйте заново!\n");  }  } while (input < 1 || input > 4);  case 3:  system("CLS");  printf("1) Вычислить первую норму вектора\n2) Вычислить вторую норму вектора\n3) Вычислить норму вектора Гёльдера\n4) Вычислить бесконечную норму вектора\n5) Вычислить нормировку вектора\nВыбор: ");  do {  scanf\_s("%d", &input);  switch (input) {  case 1:  system("CLS");  firstNorma(mas, n);  system("pause");  system("CLS");  goto reset;  case 2:  system("CLS");  secondNorma(mas, n);  system("pause");  system("CLS");  goto reset;  case 3:  system("CLS");  printf("Введите р-норму:\n");  scanf\_s("%d", &p);  pNorma(arr, n, p);  system("pause");  system("CLS");  goto reset;  case 4:  system("CLS");  infNorma(mas, n);  system("pause");  system("CLS");  goto reset;  case 5:  system("CLS");  normirovka(arr, n);  system("pause");  system("CLS");  goto reset;  default:  printf("Вы ввели неверные значения, попробуйте заново!\n");  }  } while (input < 1 || input > 4);  case 4:  break;  default:  printf("Вы ввели неверные значения, попробуйте заново!\n");  }  } while (input < 1 || input > 4);  free(mas);  free(arr);  return 0;  } |

Исходный код программы Project\_2: main.с