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

Сравнение сортировок.

Выполнил:

студент института ИТММ

гр. 3821Б1ПМ3

Афанасьев К.О

Проверил:

заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений

Лебедев И.Г

Нижний Новгород

2021г.

# Содержание

[Содержание 1](#_Toc88648576)

[Введение 2](#_Toc88648577)

[Постановка задачи 3](#_Toc88648578)

[Руководство пользователя 4](#_Toc88648579)

[Первая программа 4](#_Toc88648580)

[Вторая программа 5](#_Toc88648581)

[Руководство программиста 8](#_Toc88648582)

[*Описание структуры кода программ* 8](#_Toc88648583)

[Первая программа 8](#_Toc88648584)

[Вторая программа 9](#_Toc88648585)

[Описание алгоритмов 11](#_Toc88648587)

[Сортировка «пузырьком» 12](#_Toc88648588)

[Сортировка «вставками» 13](#_Toc88648589)

[Быстрая сортировка 14](#_Toc88648590)

[Эксперименты 16](#_Toc88648591)

[Заключение 18](#_Toc88648592)

[Список литературы 19](#_Toc88648593)

[Приложение 20](#_Toc88648594)

[Приложение 1.1 21](#_Toc88648595)

[Приложение 1.2 22](#_Toc88648596)

# 

# Введение

Программирование можно рассматривать как искусство, науку, ремесло. Программирование - это искусство получения ответов от машины. Для этого в узком смысле нужно составить специальный код для технического устройства, а в широком - разработать программы на языках программирования, т.е. не просто составить код, а выполнить интеллектуальную работу по составлению высокоразумных программ для решения различных задач во всех сферах человеческой деятельности.

Программирование - процесс описания последовательности действий решения задачи средствами конкретного языка программирования и оформление результатов описания в виде программы. Эта работа требует точности, аккуратности и терпения. Команды машине должны формулироваться абсолютно четко и полно, не должны содержать никакой двусмысленности.

В нашей работе на языке программирования «С» будет реализовано три вида сортировок: пузырьковая, сортировка вставками и быстрая сортировка, а также сравним их быстродействие при одинаковых входных данных.

# Постановка задачи

Реализовать сортировки массивов данных (тип данных определяется преподавателем(int)) 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

## Первая программа

После запуска программы необходимо ввести три значения – количество элементов массива, минимальный и максимальный элементы массива (см. рис. 1)

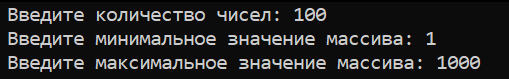


Рис.1 Запуск программы

Если никаких неполадок не возникнет, выйдет сообщение о том, что массив успешно сгенерирован (см. рис. 2).

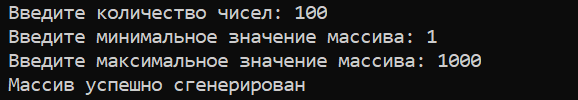


Рис. 2 Генерация массива

## 

## Вторая программа

При запуске второй программы перед пользователем должно быть выведено простое меню, в котором пользователю предлагается выбрать какое-нибудь действие, введя число (см. рис. 3).

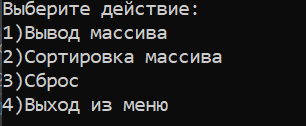


Рис.3 Меню программы

Если пользователь введёт число 1, то программа выведет массив (см. рис.4).

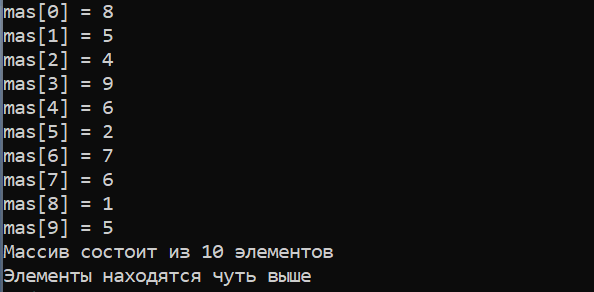


Рис. 4 Вывод массива

Если пользователь напишет в консоли число 2, то программа предложит выбрать следующие сортировки: пузырьком, вставками и быстрая. Пользователю необходимо выбрать нужную сортировку (см. рис. 8).

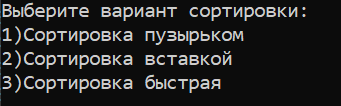


Рис. 5 Выбор сортировки

Далее пользователь должен выбрать вариант сортировки опять же введя число. Затем выйдет соответствующее сообщение о времени выбранной сортировки. (см. рис. 9)



Рис. 5 Время сортировки

Чтобы всё сбросить, пользователю необходимо ввести число 3 – система проведет сброс данных. (см. рис. 6).

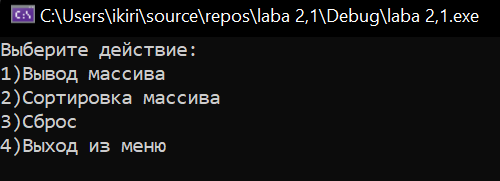


Рис.6 Сброс данных

Для завершения работы программы пользователю достаточно ввести число 4, оно осуществляет выход из программы. (см.рис.7)

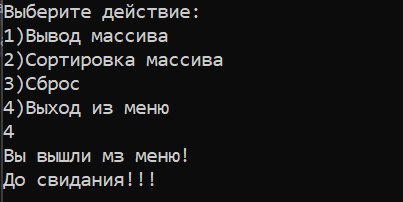


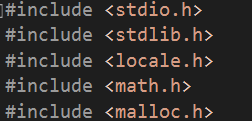
рис.7. Выход из меню

# Руководство программиста

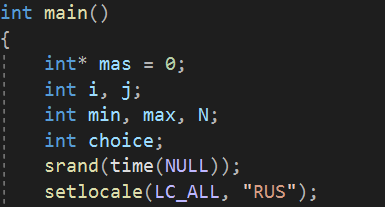
# *Описание структуры кода программ*

# *Первая программа*

1. Подключение библиотек.

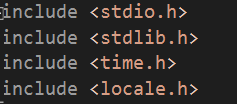


1. Объявление функции main() и всех переменных.

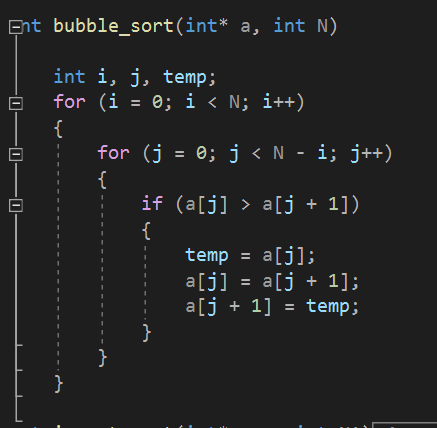


### Вторая программа

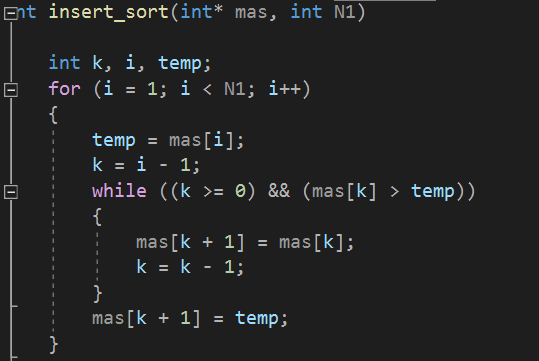
1. Подключение библиотек, с которыми предстоит работать.



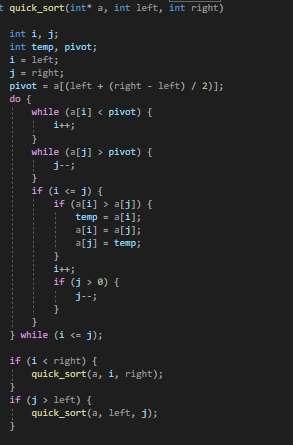
1. Сортировка пузырьком



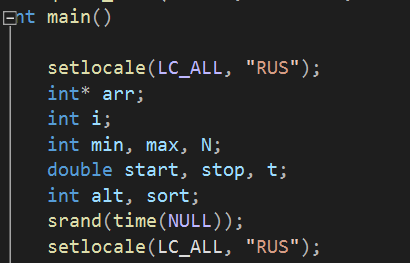
1. Сортировка вставками



1. Быстрая сортировка



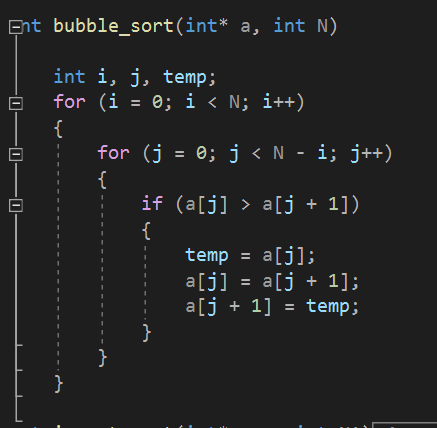
1. Объявление функции main(). Объявление всех переменных.



## Описание алгоритмов

### Сортировка «пузырьком»

Упорядоченный массив создается на том же участке памяти, где находится исходная последовательность. Идея метода состоит в том, чтобы попарно сравнивать соседние элементы. Каждый проход начинается с начала последовательности. Сравнивается первый элемент со вторым: если порядок между ними нарушен, то они меняются местами. Затем сравниваются второй с третьим, третий с четвертым и так далее до конца массива; элементы с неправильным порядком в паре меняются местами. В итоге, после первого прохода, максимальный (или минимальный, в зависимости от вида сортировки: по возрастанию/по убыванию) элемент будет находится на последнем месте в массиве, он как бы “всплывет” наверх

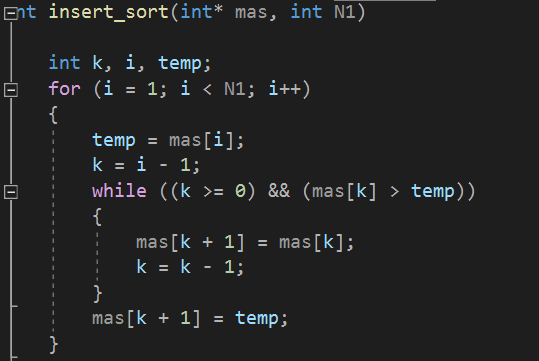


При **сортировке пузырьком**максимальное число сравнений составляет 0,5N², а среднее число сравнений приблизительно 0,25N².

### Сортировка «вставками»

  Первый элемент в массиве образует уже отсортированную последовательность. Сравниваем второй элемент с первым. Если порядок между ними нарушен, то первый элемент передвигается на одну позицию вправо. Теперь отсортированный массив состоит из двух элементов.

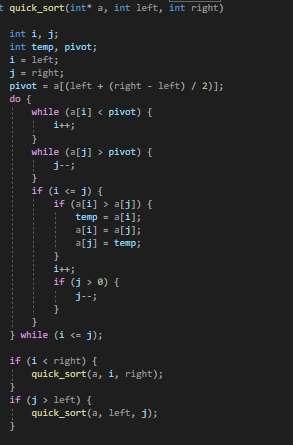
Далее, в течение каждой итерации, берем следующий элемент (третий, четвертый и т.д) и сравниваем его поочередно с другими элементами в уже отсортированном списке, **начиная с конца** этого списка. Если порядок между сравниваемыми элементами нарушен, то меняем их местами, если нет, то “вставка” нового элемента закончена, переходим к следующему.



**Сортировка вставками** имеет большую вычислительную сложность. Поэтому она эффективна на небольших наборах данных. Рекомендуется использовать этот метод на наборах размером до десятков элементов. Сортировка вставками эффективна на последовательностях с данными, которые уже частично отсортированы

***Быстрая сортировка***

Для достижения наибольшей эффективности желательно производить обмен элементов на больших расстояниях. В массиве выбирается некоторый элемент, называемый разрешающим. Затем он помещается в то место массива, где ему полагается быть после упорядочивания всех элементов. В процессе отыскания подходящего места для разрешающего элемента производятся перестановки элементов так, что слева от них находятся элементы, меньшие разрешающего, и справа — большие (предполагается, что массив сортируется по возрастанию).



Быстрая сортировка представляет собой усовершенствованный метод сортировки, основанный на принципе обмена. Пузырьковая сортировка является самой неэффективной из всех алгоритмов прямой сортировки. Однако усовершенствованный алгоритм является лучшим из известных методом сортировки массивов. Он обладает столь блестящими характеристиками.

# 

# Эксперименты

В качестве эксперимента узнаем, какая из сортировок работает быстрее со случайными числами.

Для этого создадим массив, состоящий из 100.000 элементов (см. рис. 10).

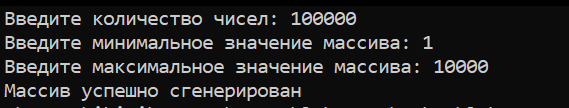


Рис. 12 Генерация массива

Затем запустим вторую программу и с её помощью отсортируем элементы с помощью сортировки пузырьком (см. рис. 11)



Рис. 11 Время сортировки пузырьком

Сортировка пузырьком продлилась примерно 20 секунд. Далее проводим данную процедуру для быстрой сортировки и сортировки вставкой.

Полученные в экспериментах данные приведены в таблице 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип сортировки** | **Сложность** | **Размер входных данных** | **Время работы, с** |
| Пузырьком |  | 100000 | ≈20 |
| Вставками |  | 100000 | ≈3,85 |
| Быстрая |  | 100000 | ≈0,004 |
| Пузырьком |  | 5000 | ≈0,034 |
| Вставками |  | 5000 | ≈0,011 |
| Быстрая |  | 5000 |  |

Таблица 1. Сравнение времени работы сортировок

Все эти данные были получены на одной и той же машине, при одном и том же диапазоне генерации случайных чисел.

# Заключение

В ходе данной работы, я написал две программы. Одна создает файл со случайно сгенерированными числами и записывает их туда. Вторая производит работу с этим файлом. Она сортировала массив благодаря трем сортировкам. Пузырьком, вставками и быстрая. Провели исследования и выясняли какая сортировка наиболее эффективная

По результатам экспериментов можно сделать вывод о том, что самая быстрая сортировка - это быстрая. Второе место занимает сортировка «вставками», которая выполнилась быстрее пузырьковой быстрее при одинаковом количестве данных. Ну и самой медленной оказалась сортировка «пузырьком», которая выполнялась дольше всех сортировок.

# Список литературы

1. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Алгоритмы: построение и анализ — 2-е. — М.: Вильямс, 2005. — 1296 с.
2. Алгоритмы / С. Дасгупта, Х. Пападимитриу, У. Вазирани; Пер. с англ. под ред. А. Шеня. –– М.: МЦНМО, 2014. –– 320 с.
3. Язык С в ХХI веке. Бен Клеменс
4. Изучаем программирование на C. Дэвид Гриффитс, Дон Гриффитс
5. C. Полное руководство. Герберт Шилдт Брайан Керниган, Деннис Ритчи. Язык программирования C. — Москва: Вильямс, 2015. — 304 с.
6. Бен Клеменс "Язык С в XXI веке
7. Дэвид Гриффитс, Дон Гриффитс "Изучаем программирование на C"

# Приложение 1

## Приложение 1.1

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <locale.h>  #include <math.h>  #include <malloc.h>  int main()  {  int\* mas = 0;  int i, j;  int min, max, N;  int choice;  srand(time(NULL));  setlocale(LC\_ALL, "RUS");  FILE\* file = 0;  fopen\_s(&file, "C:\\Users\\ikiri\\source\\repos\\laba 2,2\\laba 2,2\\numbers.txt", "w");  printf(" Введите количество чисел: ");  scanf\_s("%d", &N);  printf(" Введите минимальное значение массива: ");  scanf\_s("%d", &min);  printf(" Введите максимальное значение массива: ");  scanf\_s("%d", &max);  if ((N <= 0) || (max <= min))  {  printf(" Ошибка! Введите корректные данные");  return 0;  }  mas = malloc(N \* sizeof(int));  for (i = 0; i < N; i++)  {  mas[i] = (min + (max - min) \* rand() / RAND\_MAX);  }  printf(" Массив успешно сгенерирован");  fprintf(file, "n = %d\n", N);  for (i = 0; i < N; i++)  {  fprintf(file, "%d\n", mas[i]);  }  fclose(file);  return 0;  } |

## Приложение 1.2

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include <locale.h>  int bubble\_sort(int\* a, int N)  {  int i, j, temp;  for (i = 0; i < N; i++)  {  for (j = 0; j < N - i; j++)  {  if (a[j] > a[j + 1])  {  temp = a[j];  a[j] = a[j + 1];  a[j + 1] = temp;  }  }  }  }  int insert\_sort(int\* mas, int N1)  {  int k, i, temp;  for (i = 1; i < N1; i++)  {  temp = mas[i];  k = i - 1;  while ((k >= 0) && (mas[k] > temp))  {  mas[k + 1] = mas[k];  k = k - 1;  }  mas[k + 1] = temp;  }  }  int quick\_sort(int\* a, int left, int right)  {  int i, j;  int temp, pivot;  i = left;  j = right;  pivot = a[(left + (right - left) / 2)];  do {  while (a[i] < pivot) {  i++;  }  while (a[j] > pivot) {  j--;  }  if (i <= j) {  if (a[i] > a[j]) {  temp = a[i];  a[i] = a[j];  a[j] = temp;  }  i++;  if (j > 0) {  j--;  }  }  } while (i <= j);  if (i < right) {  quick\_sort(a, i, right);  }  if (j > left) {  quick\_sort(a, left, j);  }  }  int main()  {  setlocale(LC\_ALL, "RUS");  int\* arr;  int i;  int min, max, N;  double start, stop, t;  int alt, sort;  srand(time(NULL));  setlocale(LC\_ALL, "RUS");  FILE\* file = 0;  fopen\_s(&file, "C:\\Users\\ikiri\\source\\repos\\laba 2,2\\laba 2,2\\numbers.txt", "r");  fscanf\_s(file, "n = %d\n", &N);  arr = malloc(N\* sizeof(int));  for (i = 0; i < N; i++)  {  fscanf\_s(file, "%d\n", &(arr[i]));  }  fclose(file);  do  {  printf("Выберите действие:\n1)Вывод массива\n2)Сортировка массива\n3)Сброс\n4)Выход из меню\n");  do  {  scanf\_s("%d", &alt);  if (alt > 4 || alt <= 0)  {  printf("!!!ВВЕДИТЕ КОРРЕКТНЫЕ ДАННЫЕ!!!");  return 0;  }  } while (alt > 4 || alt <= 0);  if (alt == 1)  {  for (i = 0; i < N; i++)  {  printf("mas[%d] = %d\n", i, arr[i]);  }  printf("Массив состоит из %d элементов\n", N);  printf("Элементы находятся чуть выше\n");  }  if (alt == 2)  {  printf("Выберите вариант сортировки:\n1)Сортировка пузырьком\n2)Сортировка вставкой\n3)Сортировка быстрая\n");  do  {  scanf\_s("%d", &sort);  if (sort > 3 || sort <= 0)  {  printf("!!!ВВЕДИТЕ КОРРЕКТНЫЕ ДАННЫЕ!!!");  return 0;  }  } while (sort > 3 || sort <= 0);  if (sort == 1)  {  printf("Вы выбрали сортировку массива пузырьком:\n");  start = clock();  bubble\_sort(arr, N - 1);  stop = clock();  t = ((double)(stop - start)) / CLK\_TCK;  printf("Время сортировки равно % lf сек\n", t);  }  else if (sort == 2)  {  printf("Вы выбрали сортировку массива вставкой:\n");  start = clock();  insert\_sort(arr, N);  stop = clock();  t = ((double)(stop - start)) / CLK\_TCK;  printf("Время сортировки равно %lf сек\n", t);  }  else if (sort == 3)  {  printf("Вы выбрали быструю сортировку массива:\n");  start = clock();  quick\_sort(arr, 0, N - 1);  stop = clock();  t = ((double)(stop - start)) / CLK\_TCK;  printf("Время сортировки равно %lf сек\n", t);  }  }  if (alt == 3)  {  printf("Вы выполнили сброс\n");  system("cls");  return main();  }  if (alt == 4)  {  printf("Вы вышли мз меню!\n");  printf("До свидания!!!\n");  return 0;  }  } while (alt == 1 || alt == 2 || alt == 3 || alt == 4);  free(arr);  return 0;  } |