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

**Институт информационных технологий, математики и механики**

**ОТЧЕТ**

По лабораторной работе

На тему:

**«Генерация случайных чисел. Выполнение операций с**

**ними.»**

**Выполнил:**

Студент группы 3821Б1ФИ3

Черепанов А.М.

**Проверил:**

Заведующий лабораторией суперкомпьютерных технологий и высокопроизводительных вычислений

Лебедев И.Г.

Нижний Новгород

2021 г

Оглавление

[Введение. 3](#_Toc89791253)

[1. Постановка задачи. 4](#_Toc89791254)

[2. Руководство пользователя. 5](#_Toc89791255)

[3. Руководство программиста. 8](#_Toc89791256)

[Заключение. 13](#_Toc89791257)

[Литература. 14](#_Toc89791258)

[Приложение. 15](#_Toc89791259)

[Приложение 1 15](#_Toc89791262)

[Приложение 2 16](#_Toc89791263)

# Введение.

Программирование - это интересный, полезный и увлекательный процесс, благодаря которому мы, с помощью специальных команд, заставляем компьютер, выполнять для нас различные задачи, от выполнения операций с числами и навигации, до управления самолетами, спутниками и прочей техникой.

Существуют десятки алгоритмов сортировки, каждый из которых хорош в чём-то своём. На одних легко объяснять принципы сортировки, другие хороши при работе с большими массивами, третьи оптимизированы по скорости, четвёртые — по числу процессорных циклов, компактности кода и т. д.

# Постановка задачи.

Сравнение сортировок.

Реализовать сортировки массивов данных (тип данных “long 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.

Используется две программы. После запуска первой программы выводится сообщение на экран: «Введите N» (см. Рисунок 1)

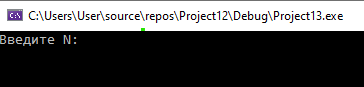


Рисунок 1. Консоль

Если выбранное вами число окажется меньше равно нуля, программа выведет на экран: «Error N<=0!» (см. Рисунок 2), а так же завершит работу.

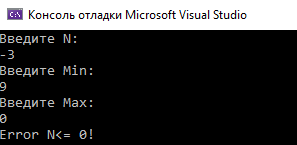


Рисунок 2. Ошибка 1

Введите, выбранное вами, наименьшее значение. Для примера выбрано число 1. (см. Рисунок 3)

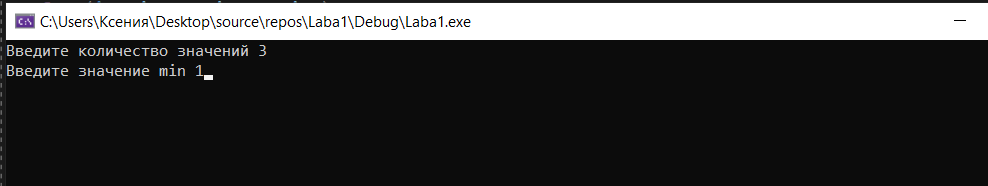


Рисунок 3. Введение значения

Введите, выбранное вами, наибольшее значение. Для примера выбрано число 9. (см. Рисунок 4)

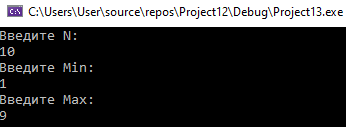


Рисунок 4. Введение значения

Если, введённое вами, наименьшее значение окажется больше наибольшего значения, программа выведет на экран: «Error! min>max», и завершит работу. (см. Рисунок 5)

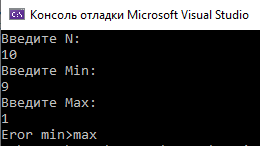


Рисунок 5. Ошибка 2

Если введённые данные оказались коректными, в файле появятся числа. (см. Рисунок 6)

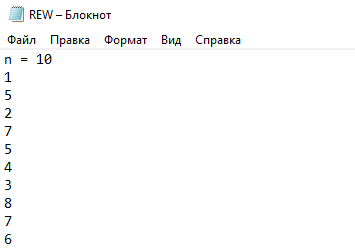


Рисунок 6. Числа в файле

Если в процессе работы программы вы введи некорректные данные,запустите программу заново, введите новые значения.

После завершения первой программы следует запустить вторую. После ее запуска на экран будет выведен консольный интерфейс. (см. Рисунок 7)

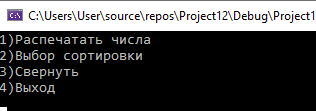


Рисунок 7. Консольный интерфейс

Консольный интерфейс представляет собой набор из четырех команд:

Команды:

1. Распечатать числа

2. Выбор сортировки

3. Свернуть

4. Выход

Команда 1 печатает массив на экран

Команда 2 вызывает вторую консоль с выбором вида сортировки. (см. Рисунок 8)

Команда 3 возвращает командную строку к исходному состоянию

Команда 4 завершает работу программы

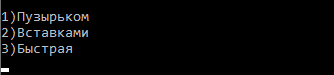


Рисунок 8 Консоль 2

# 

# Руководство программиста.

***Описание структуры программы***

Первая программа состоит из одного модуля int main() {…}, в котором находиться код создания чисел и записи их в файл.

Вторая программа также состоит из одного модуля int main() {…}, в котором находится код консоли.

Функции:

BubbleSort – Сортирует массив пузырьком

VstavkaSort - Сортирует массив вставкой

QuickSort - Сортирует массив по способу Тони Хоарома

***Описание алгоритмов.***

1. **Алгоритм записи случайных чисел в файл.**

|  |
| --- |
| fnumbers = fopen("D:\\REW.txt", "w");  for (int i = 0; i < N; i++)  {  mas[i] = (((double)rand()) / RAND\_MAX) \* (max - min) + min;    }  fprintf(fnumbers,"n = %d\n", N);  for (int i = 0; i < N; i++)  {  fprintf(fnumbers, "%ld\n", mas[i]);  }  fclose(fnumbers);  free(mas);  return 0; |

**2. Алгоритм консольного интерфейса**

Начало

alt = 4?

alt?

Забор “alt”

short=3

**alt=1**

alt=2

alt=3

Распечатать

числа

Забор “short”

Свернуть

short=3

short=3

Вставкой

Быстрая

Пузырьком

Конец

нет

да

**3.1 Сортировка пузырьком.**

|  |
| --- |
| int bubble\_sort(long 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;  }  }  }  } |

**3.2 Сортировка вставкой.**

|  |
| --- |
| int insert\_sort(long int\* mas, int n)  {  int tmp = 0;  for (int i = 1; i < n; i++)  {  tmp = mas[i];  for (int j = i - 1; j > 0; j--)  {  if (mas[j] > tmp)  {  mas[j + 1] = mas[j];  mas[j] = tmp;  }  }  }  } |

**3.3. Быстрая сортировка.**

|  |
| --- |
| int quick\_sort(long 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);  }  } |

**4. Эксперименты.**

Для упрощения понимания результатов эксперимента используем таблицу. Введём разное количество данных и сравним время работы.

|  |  |  |
| --- | --- | --- |
| **Тип сортировки** | **Размер входных данных** | **Время работы, мс** |
| Пузырьком | 10000 | 24200 |
| Вставками | 10000 |  |
| Быстрая | 10000 | 1 |
| Пузырьком | 100000 |  |
| Вставками | 100000 |  |
| Быстрая | 100000 |  |

Из таблицы видно, что быстрее всего работает быстрая сортировка, на 10 000 элементах она работает меньше 1 мс.

# Заключение.

В ходе лабораторной работы была написана программа на языке программирования «С», которая полностью выполняет поставленную задачу, а именно:

«Сравнение сортировок.

Реализовать сортировки массивов данных (тип данных “Long 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».

Интерфейс программы простой, понятный и удобный в использовании, все сообщения выводятся на русском языке и при своих небольших размерах содержат нужную информацию, которую необходимо донести до пользователя.

В ходе сравнения сортировок можно сделать вывод: «быстрая сортировка (quick sort)» выполняет сортировку чисел быстрее остальных типов сортировки и очень быстро работает с любыми типами данных; сортировка «пузырьком» занимает большее время чем «quick sort», но тоже довольно быстро сортирует данные; «сортировка вставками» работает медленнее всех на большом диапазоне чисел, ее следует использовать на коротком отрезке не больше, чем десятки, или уже на частично отсортированном массиве данных.

Литература.  
1. Т.А. Павловская Учебник по программированию на языках высокого  
уровня(С/С++) – Режим доступа: http://cph.phys.spbu.ru/documents/First/books/7.pdf  
2. Бьерн Страуструп. Язык программирования С++ - Режим доступа:  
<http://8361.ru/6sem/books/Straustrup-Yazyk_programmirovaniya_c.pdf>

# Приложение.

## Приложение 1.

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include <locale.h>  #include <malloc.h>  #include <math.h>  int main() {  setlocale(LC\_ALL, "Rus");  int i = 0, N = 0;  long int min = 0, max = 0;  long int\* mas;  FILE\* fnumbers = 0;  printf\_s("Введите N: \n");  scanf\_s("%d", &N);  printf\_s("Введите Min: \n");  scanf\_s("%ld", &min);  printf\_s("Введите Max: \n");  scanf\_s("%ld", &max);  if (N <= 0)  {  printf\_s("Error N<= 0! \t");  return 0;  }  mas = (long int\*)malloc(N \* sizeof(long int));  if (mas == 0)  {  printf\_s("Минимальное 1! \t");  return 0;  }  if (min > max)  {  printf\_s("Error min>max \t");  return 0;  }  fnumbers = fopen("D:\\REW.txt", "w");  for (int i = 0; i < N; i++)  {  mas[i] = (((double)rand()) / RAND\_MAX) \* (max - min) + min;    }  fprintf(fnumbers,"n = %d\n", N);  for (int i = 0; i < N; i++)  {  fprintf(fnumbers, "%ld\n", mas[i]);  }  fclose(fnumbers);  free(mas);  return 0;  } |

## Приложение 2.

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include <locale.h>  int bubble\_sort(long 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(long int\* mas, int n)  {  int tmp = 0;  for (int i = 1; i < n; i++)  {  tmp = mas[i];  for (int j = i - 1; j > 0; j--)  {  if (mas[j] > tmp)  {  mas[j + 1] = mas[j];  mas[j] = tmp;  }  }  }  }  int quick\_sort(long 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");  long 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, "D:\\REW.txt", "r");  fscanf\_s(file, "n = %d\n", &N);  arr = malloc(N \* sizeof(long int));  for (i = 0; i < N; i++)  {  fscanf\_s(file, "%d\n", &(arr[i]));  }  fclose(file);  do  {  printf("1)Распечатать числа\n2)Выбор сортировки\n3)Свернуть\n4)Выход\n");  do  {  scanf\_s("%d", &alt);  if (alt > 4 || alt <= 0)  {  printf("!!!ERROR!!!");  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(N);    }  if (alt == 2)  {  printf("\n1)Пузырьком\n2)Вставками\n3)Быстрая\n");  do  {  scanf\_s("%d", &sort);  if (sort > 3 || sort <= 0)  {  printf("!!!EROR!!!");  return 0;  }  } while (sort > 3 || sort <= 0);  if (sort == 1)  {  printf("Соритровка завершена\n");  start = clock();  bubble\_sort(arr, N - 1);  stop = clock();  t = ((double)(stop - start)) / CLK\_TCK;  printf("Время выполнения %f\n\n", t);  }  else if (sort == 2)  {  printf("Соритровка завершена\n");  start = clock();  insert\_sort(arr, N);  stop = clock();  t = ((double)(stop - start)) / CLK\_TCK;  printf("Время выполнения %f\n\n", t);  }  else if (sort == 3)  {  printf("Соритровка завершена\n");  start = clock();  quick\_sort(arr, 0, N - 1);  stop = clock();  t = ((double)(stop - start)) / CLK\_TCK;  printf("Время выполнения %f\n\n", t);  }  }  if (alt == 3)  {  printf("Сортировка сброшена\n");  system("cls");  return main();  }  if (alt == 4)  {  printf("Програма завершена\n");  return 0;  }  } while (alt == 1 || alt == 2 || alt == 3 || alt == 4);  free(arr);  return 0;  } |