Федеральное Государственное Автономное Образовательное Учреждение Высшего Образования «Национальный Исследовательский Нижегородский Государственный Университет им. Н. И. Лобачевского».

Институт информационных технологий математики и механики.

Фундаментальная информатика и информационные технологии.

Отчёт по лабораторной работе

Модульное программирование. Виды сортировок.

Выполнила:

студентка факультета ИИТММ ФИИТ 2 группа

Тюлькина О. В.

Проверил:

ассистент кафедры математического обеспечения и суперкомпьютерных технологий ИИТММ

Лебедев И. Г.

Нижний Новгород

2019 г.

Содержание

[1. Введение 3](#_Toc26876141)

[2. Постановка задачи 4](#_Toc26876142)

[3. Руководство пользователя 5](#_Toc26876143)

[4. Руководство программиста 7](#_Toc26876144)

[4.1. Описание структур данных 7](#_Toc26876145)

[4.2. Описание алгоритмов 7](#_Toc26876146)

[4.3. Описание структуры программы 7](#_Toc26876147)

[5. Эксперименты 8](#_Toc26876148)

[6. Заключение 12](#_Toc26876149)

[7. Литература 13](#_Toc26876150)

[8. Приложение 14](#_Toc26876151)

## 1. Введение

Сейчас невозможно представить современный мир без программирования, так как именно благодаря ему стал возможен дальнейший прогресс в науке и технологиях. Программирование присутствует практически во всех сферах жизнедеятельности.

Люди научились писать большие по объему и сложности программы. Чтобы облегчить себе задачу, уменьшить объем и ускорить работу программы, программисты научились методу разбиения 1 сложной программы на несколько частей. Такое метод и используется в модульном программировании. Его основной принцип - «разделяй и властвуй». Причем каждый модуль выполняет свою задачу и действует согласно определенным правилам. Такой метод помимо удобства позволяет упростить тестирование программы и обнаружение ошибок.

С помощью данного метода напишем программу, которая позволит нам сортировать (упорядочивать) элементы массива.

## 2. Постановка задачи

Выполнить задание:

«Реализовать сортировки массивов данных (тип данных определяется преподавателем) 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.»

Вывести все данные и результат на экран.

## 3. Руководство пользователя

Программа №1.

1) Ввести количество элементов массива (т. е. его размер);

2)Ввести минимальное и максимальное значения;

3) Открыть папку программы;

4) Открыть созданный там файл и посмотреть, что вывела программа.

1 и 2 пункты – Рис.6

4 пункт – Рис.7

Программа №2.

1) Ввести размер массива (целое число).

2) Ввести цифру «1» - да или «0»- нет при вопросе «Хотите ли вы чтобы массив был показан?».

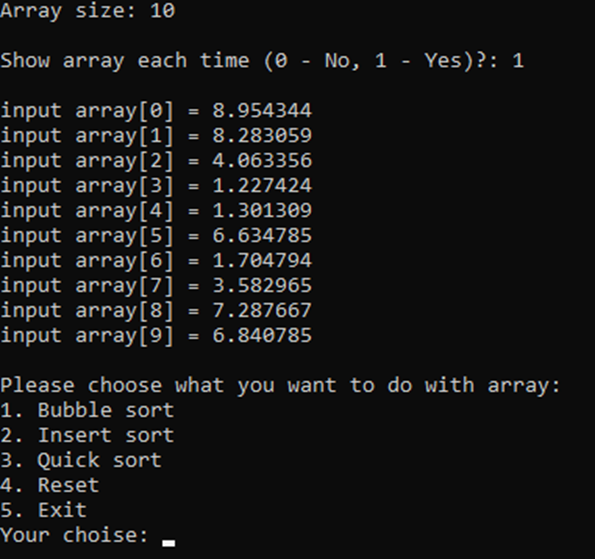


Рисунок 1. Ввод данных.

2) Выбрать 1 из 5 действий.

3) В случае выбора действия Quick sort нужно ввести первый и последний элемент массива.

4) После каждой сортировки будет показано время её выполнения в миллисекундах.

5) Затем можно снова выбрать 1 из 5 действий.

6) Перед тем как снова воспользоваться сортировкой нужно сначала выбрать Reset.

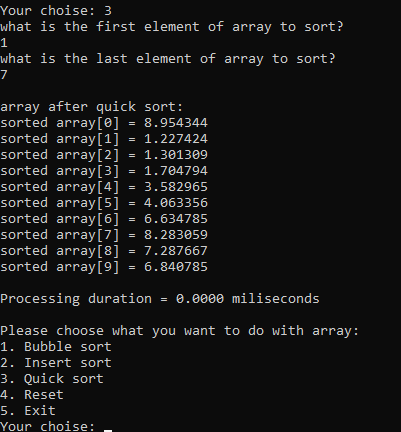


Рисунок 2. Пример работы быстрой сортировки.

## 4. Руководство программиста

## 4.1. Описание структур данных

*Целочисленный тип данных int* **-** служит для представления целых чисел.

*Вещественный тип данных long double* — тип данных, который используется для переменных, содержащих числа с дробной частью.

## 4.2. Описание алгоритмов

Суммирование – сложение переменных.

*Умножение* - Умножить одно целое число на другое значит повторить одно число столько раз, сколько в другом содержится единиц.

*Логическое* И - возвращает истину тогда и только тогда, когда оба аргумента являются истиной.

*Логическое* ИЛИ - истинен тогда, когда истиной является хотя бы один его аргумент.

*Запись в файл* – заполнение текстового файла нужной информацией. Форматированный вывод.

*Чтение из файла*- осуществляет форматированный ввод из файла, просматривает информацию в файле.

Условие - позволяет выполнять или не выполнять определенные участки кода, в зависимости от того является ли истинным или ложным условие этого оператора.

## 4.3. Описание структуры программы

Функция main - точка входа в программу.

Функция Rand– заполняет массив случайными числами.

Функция PrintInFile (или Print)– осуществляет запись информации в файл.

Функция BubbleSort – сортирует элементы массива методом пузырьковой сортировки.

Функция InsertSort - сортирует элементы массива, используя сортировку вставками.

Функция QuickSort - сортирует элементы массива методом быстрой сортировки.

Функция Copy – копирует элементы массива в другой массив.

Алгоритм работы: ввод данных, создание массивов, генерация случайных чисел, запись в файл, чтение файла, выбор действий (сортировка элементов массива, сброс, выход).

## 5. Эксперименты

Эксперимент №1 (Программа №2).

Введем размер массива: 10. Выбираем, чтобы нам показали элементы массива. После этого выберем 1 из 5 действий. В нашем случае – это пузырьковая сортировка (1). Тогда после нажатия на клавишу Enter, мы увидим 10 отсортированных в нужном порядке элементов.

Причем 10 элементов массива сортируются за очень маленькое время.

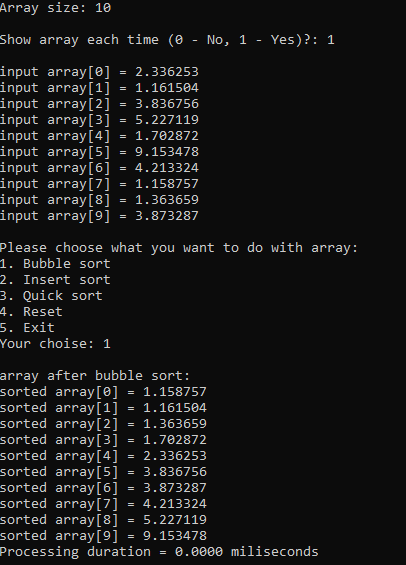


Рисунок 3. Эксперимент №1.

Эксперимент №2 (Программа №2)

Введем размер массива: 1000. Выбираем, чтобы нам не показывали элементы массива. После этого выберем 1 из 5 действий. В нашем случае – это пузырьковая сортировка (1). Тогда после нажатия на клавишу Enter, мы получим 1000 отсортированных в нужном порядке элементов. Причем 1000 элементов массива сортируются за время: 2 миллисекунды.

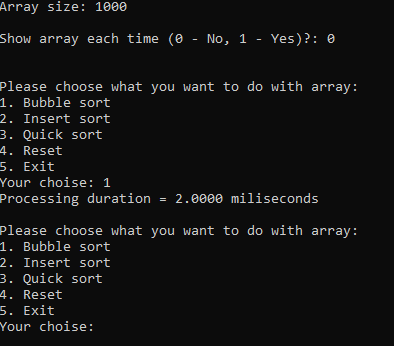


Рисунок 4. Эксперимент № 2. Пузырьковая сортировка.

Теперь сделаем сброс массива и выберем сортировку вставками (2). В итоге мы получим 1000 отсортированных элементов массива. Причем время выполнения данной сортировки составило уже 1 миллисекунду.

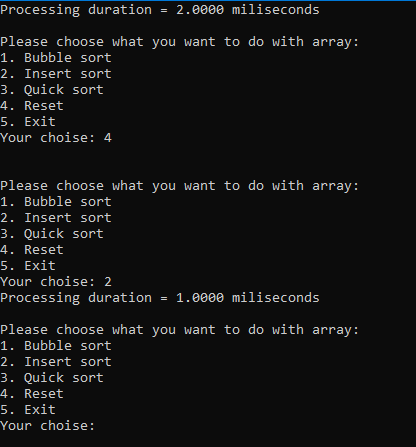


Рисунок 5. Эксперимент № 2. Сортировка вставками.

Теперь снова сделаем сброс массива и выберем быструю сортировку (3). Введем первый элемент – 1 и последний элемент – 999. Программа выполнила свою задачу за очень короткое время.

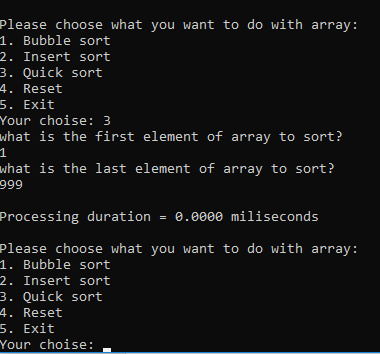


Рисунок 5. Эксперимент № 2. Быстрая сортировка.

В итоге, сравнив время выполнения каждой сортировки, мы можем сделать вывод, что быстрее всех справляется со своей задачей быстрая сортировка. Следующей по временя является сортировка вставками. Самое большое время по сравнению с другими было замечено у пузырьковой сортировки.

Таблица 1. Сравнение сортировок по времени их выполнения.

|  |  |  |
| --- | --- | --- |
| Название сортировки | Время выполнения | Вывод |
| Пузырьковая | 2 мс | Самая медленная из всех представленных в программе сортировок. |
| Вставками | 1 мс | Быстрее, чем пузырьковая, но медленнее чем быстрая сортировка. |
| Быстрая | 0.0 мс | Самая быстрая по времени выполнения сортировка. |

Эксперимент №3 (Программа №1).

Введем размер массива: n=5. Затем создаем диапазон от min до max элемента, для этого вводим максимальный и минимальный элементы (вещественные): min = 1, max = 2.5.



Рисунок 6. Эксперимент № 3. Ввод данных в программе №1.

После ввода данных программа закрывается. Зайдя в папку нашей программы, открываем лежащий там только что созданный файл. Он покажет нам 5 элементов массива в диапазоне от 1 до 2.5.

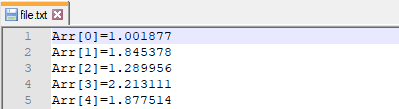


Рисунок 7. Эксперимент № 3. Файл с элементами массива.

## 6. Заключение

Я написала программу, позволяющую выполнить сортировку элементов массива, которые были заданы случайным образом. С помощью этой программы я показала какая сортировка занимает наименьшее количество времени. Данная программа была написана с помощью метода разбиения на модули, который помог мне сделать код менее объемным.

## 7. Литература

1. Брайан У. Керниган. Язык программирования Си / Деннис М. Ритчи – М.: Москва: Вильямс, 2015. — 304 с.

2. М. Бен-Ари. Языки программирования. Практический сравнительный анализ. / М. Бен-Ари. — М.: Москва: Мир, 2000. — 366 с.

3. Подбельский, В. В. Курс программирования на языке Си: учебник/ С. С. Фомин, В. В. Подбельский, - М.: ДМК Пресс, 2012. — 318 с.

4. Прата С. Язык программирования С: Лекции и упражнения / С. Прата — М.: Вильямс, 2006. — С. 960.

## 8. Приложение

Программа №1

**Файл prog1.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "prog.h"

int main(int argc, char\*argv[])

{

int n;

long double max, min;

FILE\* f;

printf("Array size: ");

scanf\_s("%d", &n);

printf("\n");

printf("Min element: \n");

scanf\_s("%lf", &min);

printf("Max element: \n");

scanf\_s("%lf", &max);

long double \*Arr = (long double\*)malloc(n \* sizeof(long double));

Gen(n, Arr, max, min);

f = fopen("file.txt", "w+");

Print(f, n, Arr);

fclose(f);

free(Arr);

getchar();

getchar();

return 0;

}

**Файл prog2.cpp**

#include "prog.h"

#include <stdlib.h>

void Gen(int n, long double\*Arr, long double max, long double min)

{

int i = 0;

for (i = 0; i < n; i++)

Arr[i] = long double(rand()) / long double(RAND\_MAX) \* (max - min) + min;

}

void Print(FILE\*f, int n, long double \*Arr)

{

int i = 0;

for (i = 0; i < n; i++)

{

fprintf(f, "Arr[%d]=%lf\n", i, Arr[i]);

}

}

**Файл prog.h**

#ifndef \_PROG\_H\_

#define \_PROG\_H\_

#include <stdio.h>

void Gen(int n, long double\* Arr, long double max, long double min);

void Print(FILE\*f, int n, long double\*Arr);

#endif

Программа №2.

**Файл main.cpp**

#include "prog.h"

int main(int argc, char\*argv[])

{

int n;

printf("Array size: ");

scanf("%d", &n);

printf("\n");

int showArray;

printf("Show array each time (0 - No, 1 - Yes)?: ");

scanf("%d", &showArray);

printf("\n");

int first, last, i;

double startTime, endTime, duration;

long double minVal = 1., maxVal = 10.;

long double \*randArr = (long double\*)malloc(n \* sizeof(long double));

long double \*inArr = (long double\*)malloc(n \* sizeof(long double));

long double \*sortedArr = (long double\*)malloc(n \* sizeof(long double));

srand(unsigned int(time(NULL)));

Rand(n, randArr, maxVal, minVal);

FILE\* f;

f = fopen("file.txt", "w+");

PrintInFile(f, n, randArr);

fclose(f);

f = fopen("file.txt", "r");

for (i = 0; i < n; i++)

{

fscanf(f, "arr[%d]=%lf\n", &i, &inArr[i]);

sortedArr[i] = inArr[i];

if (showArray == 1)

printf("input array[%d] = %lf\n", i, sortedArr[i]);

}

fclose(f);

printf("\n");

int choise = -1;

while (choise != 5)

{

printf("Please choose what you want to do with array:\n");

printf("1. Bubble sort\n");

printf("2. Insert sort\n");

printf("3. Quick sort\n");

printf("4. Reset\n");

printf("5. Exit\n");

printf("Your choise: ");

scanf("%d", &choise);

while (!(choise == 1 || choise == 2 || choise == 3 || choise == 4 || choise == 5))

{

printf("Wrong number.\nYour choise: ");

scanf("%d", &choise);

}

switch (choise)

{

case 1:

startTime = double(clock()) / double(CLOCKS\_PER\_SEC);

BubbleSort(n, sortedArr);

endTime = double(clock()) / double(CLOCKS\_PER\_SEC);

if (showArray == 1)

{

printf("\narray after bubble sort:\n");

for (int i = 0; i < n; i++)

{

printf("sorted array[%d] = %lf\n", i, sortedArr[i]);

}

}

duration = endTime - startTime;

printf("Processing duration = %.4lf miliseconds\n\n",1000.\*duration);

break;

case 2:

startTime = double(clock()) / double(CLOCKS\_PER\_SEC);

InsertSort(n, sortedArr);

endTime = double(clock()) / double(CLOCKS\_PER\_SEC);

if (showArray == 1)

{

printf("\narray after insert sort:\n");

for (int i = 0; i < n; i++)

{

printf("sorted array[%d] = %lf\n", i, sortedArr[i]);

}

}

duration = endTime - startTime;

printf("Processing duration = %.4lf miliseconds\n\n",1000.\*duration);

break;

case 3:

printf("what is the first element of array to sort?\n");

scanf("%d", &first);

printf("what is the last element of array to sort?\n");

scanf("%d", &last);

startTime = double(clock()) / double(CLOCKS\_PER\_SEC);

QuickSort(sortedArr, first, last);

endTime = double(clock()) / double(CLOCKS\_PER\_SEC);

if (showArray == 1)

{

printf("\narray after quick sort:\n");

for (int i = 0; i < n; i++)

{

printf("sorted array[%d] = %lf\n", i, sortedArr[i]);

}

}

printf("\n");

duration = endTime - startTime;

printf("Processing duration = %.4lf miliseconds\n\n",1000.\*duration);

break;

case 4:

Copy(inArr, sortedArr, n);//reset

printf("\n");

if (showArray == 1)

for (int i = 0; i < n; i++)

{

printf("sorted array[%d] = %lf\n", i, sortedArr[i]);

}

printf("\n");

break;

}

}

fclose(f);

free(randArr);

free(sortedArr);

free(inArr);

getchar();

getchar();

return 0;

}

**Файл funct1.cpp**

#include "prog.h"

void Rand(int n, long double \*arr, long double maxVal, long double minVal)

{

for (int i = 0; i < n; i++)

{

arr[i] = long double(rand()) / long double(RAND\_MAX) \* (maxVal - minVal) + minVal;

}

}

void PrintInFile(FILE\*f, int n, long double \*arr)

{

for (int i = 0; i < n; i++)

{

fprintf(f, "arr[%d]=%lf\n", i, arr[i]);

}

}

void InsertSort(int n, long double \*arr)//сортировка вставками

{

int i, j;

long double t;

for (i = 0; i < n; i++)

{

j = i - 1;

t = arr[i];

while (j >= 0 && arr[j] > t)

{

arr[j + 1] = arr[j];

j--;

}

arr[j + 1] = t;

}

}

void BubbleSort(int n, long double \*arr)//сортировка пузырьковая

{

long double p = 0, s = 0;

int j = 0, i = 0;

for (i = 0; i < n; i++)

{

for (j = 0; j < i; j++)

{

s = arr[i];

p = arr[j];

if (arr[i] < arr[j])

{

arr[i] = p;

arr[j] = s;

}

}

}

}

void QuickSort(long double\* arr, int first, int last)

{

int i = first, j = last;

long double t;

long double x = arr[(first + last) / 2];//?

do {

while (arr[i] < x) i++;

while (arr[j] > x) j--;

if (i <= j)

{

t = arr[i];

arr[i] = arr[j];

arr[j] = t;

i++;

j--;

}

} while (i <= j);

if (i < last)

QuickSort(arr, i, last);

if (first < j)

QuickSort(arr, first, j);

}

void Copy(long double\*arr, long double\*sortedArrInsert, int n)

{

for (int i = 0; i < n; i++)

{

sortedArrInsert[i] = arr[i];

}

}

**Файл prog.h**

#ifndef \_PROG\_H

#define \_PROG\_H

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <time.h>

void Rand(int n, long double \*arr, long double maxVal, long double minVal);

void PrintInFile(FILE\*f, int n, long double \*arr);

//void Functon1(FILE\*f, int\*n, long double \*\*arr);

void BubbleSort(int n, long double\*arr);

void InsertSort(int n, long double \*arr);

void QuickSort(long double\* arr, int first, int last);

void Copy(long double\*arr, long double\*sortedArrInsert, int n);

#endif \_PROG\_H