**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**отчет**

**по лабораторной работе №1**

**по дисциплине «Построение и анализ алгоритмов»**

**Тема: «Программирование и исследование алгоритмов сортировки»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6382 |  | Черкасова Е.И. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2017

**Цель работы.**

Реализовать выбранный алгоритм сортировки и провести экспериментальное испытание (исследование) алгоритма (программной реализации).

**Постановка задачи.**

Вариант 21.

В наборе S имеется n вещественных чисел. Задано также вещественное число x. Содержатся ли в S два таких элемента, что их сумма равна x.

Указание. Если набор S отсортирован, то решить задачу можно за время O(n).

**Требования.**

Построить, обосновать и испытать алгоритм, решающий поставленную задачу. Провести экспериментальное испытание алгоритма сортировки Шелла. При этом оценить эффективность алгоритма: время выполнения (с использованием компьютерного таймера). По результатам испытаний необходимо проанализировать:

1) зависимость времени выполнения от количества элементов в заданной последовательности.

2) соотношение измеренных и теоретически предсказанных значений.

Рекомендуется провести сравнения (по тем показателям, по которым это возможно) с каким-либо «эталонным» алгоритмом (реализацией), например, с системной функцией qsort или с какой-либо ещё библиотечной функцией.

**Описание алгоритмов.**

1. Алгоритм, решаюющий поставленную задачу.

**Шаг 1.** После упорядочивания по возрастанию вещественных чисел из S, находится элемент, меньший или равный x.

**Шаг 2.** Теперь рассматривается только часть последовательности от начала до рассматрриваеого элемента (“усеченная” последовательность). Вычисляется разница между этим элементом и х. Если в последовательности присутствует 0 и вычисленная разница равна 0, то ответ на задачу положительный. Если она не 0, алгоритм переходит к шагу 3.

**Шаг 3.**Двигаясь от найденного элемента к началу последовательности, каждый следующий элемент сравнивается с данной разницей и если оказывается ей равным, то это означает положительный ответ на задачу. Если же разница больше проверяемого элемента, то учитывая, что последовательность упорядочена, дальнейшие проверки не имеют смысла. Если в “усеченной” последовательности еще есть элементы, то рассматриваемым становится следующий за элементом, найденном на предыдущем шаге, и алгоритм возвращается к шагу 2. Иначе ответ на задачу отрицательный, т.е. в последовательности S нет таких 2 элементов, что их сумма равна х.

2. Сортировка Шелла

Это [алгоритм сортировки](https://ru.wikipedia.org/wiki/Алгоритм_сортировки), являющийся усовершенствованным вариантом [сортировки вставками](https://ru.wikipedia.org/wiki/Сортировка_вставками). Идея метода Шелла состоит в сравнении элементов, стоящих не только рядом, но и на определённом расстоянии друг от друга.

При сортировке Шелла сначала сравниваются и сортируются между собой значения, стоящие один от другого на некотором расстоянии  (о выборе значения  [см. ниже](https://ru.wikipedia.org/wiki/Сортировка_Шелла" \l "Выбор_длины_промежутков)). После этого процедура повторяется для некоторых меньших значений , а завершается сортировка Шелла упорядочиванием элементов при  (то есть обычной [сортировкой вставками](https://ru.wikipedia.org/wiki/Сортировка_вставкой)). Эффективность сортировки Шелла в определённых случаях обеспечивается тем, что элементы «быстрее» встают на свои места (в простых методах сортировки, например, [пузырьковой](https://ru.wikipedia.org/wiki/Сортировка_методом_пузырька), каждая перестановка двух элементов уменьшает количество [инверсий](https://ru.wikipedia.org/wiki/Инверсия_(перестановка)) в списке максимум на 1, а при сортировке Шелла это число может быть больше).

Невзирая на то, что сортировка Шелла во многих случаях медленнее, чем [быстрая сортировка](https://ru.wikipedia.org/wiki/Быстрая_сортировка), она имеет ряд преимуществ:

- отсутствие потребности в памяти под стек;

- отсутствие деградации при неудачных наборах данных — быстрая сортировка легко деградирует до O(n²), что хуже, чем худшее гарантированное время для сортировки Шелла.

* Среднее время работы алгоритма зависит от длин промежутков - d, на которых будут находиться сортируемые элементы исходного массива ёмкостью  на каждом шаге алгоритма. Существует несколько подходов к выбору этих значений:

**ДОП вики**

**Спецификация программы.**

Ограничения на входные данные: входные данные - массив вещественных чисел (с количеством элементов от 2 до 1 000 000), больших 0, но меньших 1000000.

Место и форма представления входных данных: входные данные расположены либо в файле, либо вводятся с клавиатуры, либо генерируются случайным образом самой программой (в указанном диапазоне и указанной длины), представляют собой последовательность вещественных чисел, где все элементы, кроме последнего, образуют последовательность - S, а последний есть х.

Выходные данные. Результат работы программы отображается в консоли. При необходимости могут выводиться в файл.

Также в программе присутствуют проверки на все входные данные, включая вспомогательные для диалога с пользователем. На любом этапе выполнения программы при некорректном вводе данных есть возможность их повторного ввода или же можно завершить работу.

Реализовано сравнение времени работы используемого метода сортировки и библиотечной функции qsort, проверка отсортированного массива.

**Алгоритм работы программы.**

1. Выбор способа ввода данных

2. Считывание исходных данных и их проверка

3. Копирование данных в 2 вектора (для сортировок)

4. Сортировка S последовательности 2 методами

5. Работа функции, решающей задачу

6. Проверка отсортированных массивов на идентичность

7. Рассчет затраченного времени на каждую сортировку и на решение поставленной задачи

**Описание основных функций.**

Спецификация функций приведена в Таблице 1.

***Таблица 1.*** *Описание функций*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Тип возвращаемого значения* | *Имя Функции* | *Входные данные* | *Выходные данные* | *Назначение* |
| void | cinclear | - | - | очистка буфера ввода |
| void | safe\_exit | - | - | закрытие файлов перед выходом из программы до ее завершения |
| bool | need\_exit | - | true - выход  false - повторный ввод | осуществление выбора дальнейшего действия при ошибке во входных данных |
| bool | incheck | string& buf - cсылка на введенные данные  int& n - ссылка для передачи числа в программу | true - число, удолетворяющее требованиям задачи  false - не число | проверка введенной строки: является ли числом в определенных границах |
| void | generate | ostream &file - файл для записи результата; size\_t n - количество элементов | - | генерация случайной последовательности вещественных чисел |
| void | see\_file | fstream &file - ссфлка на файловый поток | - | вывод содержимого файла на консоль, не извлекая данные |
| void | read\_vector | istream& my\_stream - ссылка на поток для чтения vector<float> &s - вектор, куда будут считаны данные | - | для считывания данных в вектор вещественных чисел |
| bool | read\_vector | vector<float> &s - вектор вещ. чисел | true - успешно  false - пустой вектор | вывести на экран вектор |
| void | template <typename T> shell\_sort | T arr[] -  size\_t distance - | - | сортировка Шелла |
| bool | if\_there\_is\_sum | vector<float>& s\_ptr - вектор данных  const float& x - сумма чисел | true - да  false - нет | ответ на поставленную задачу |

**Исследование.**

Зависимость времени работы функций сортивкии и функции, решающей поставленную задачу (каждая рассматривалась отдельно), от количества элементов в S приведена на Рис.1 (до 1000 элементов) и Рис.2 (от 50000 до 1000000 элементов), построенных на основе Таблицы 2.

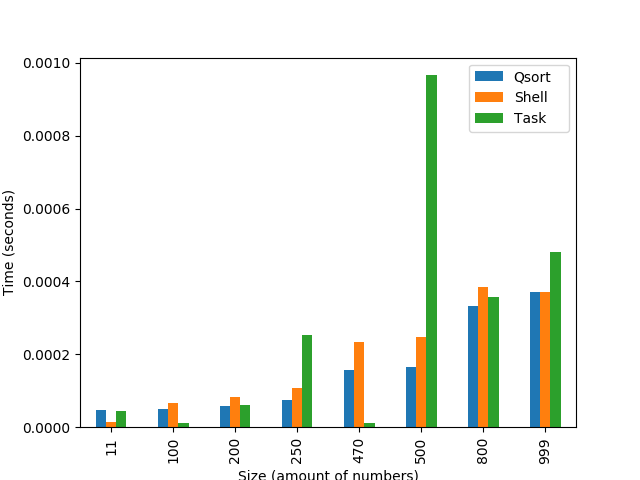
Исходя из анализа графиков, можно сделать вывод о том, что сортировка Шелла достаточно эффективна, а на больших данных время ее работы даже меньше, чем при сортировке того же массива с помощью qsort, которая выгодна в отношении времени и на малых массивах, и на больших..

Фукция для решения задачи лабораторной, показывает неплохие результаты на небольших данных, однако на больший массивах эффективность снижается.

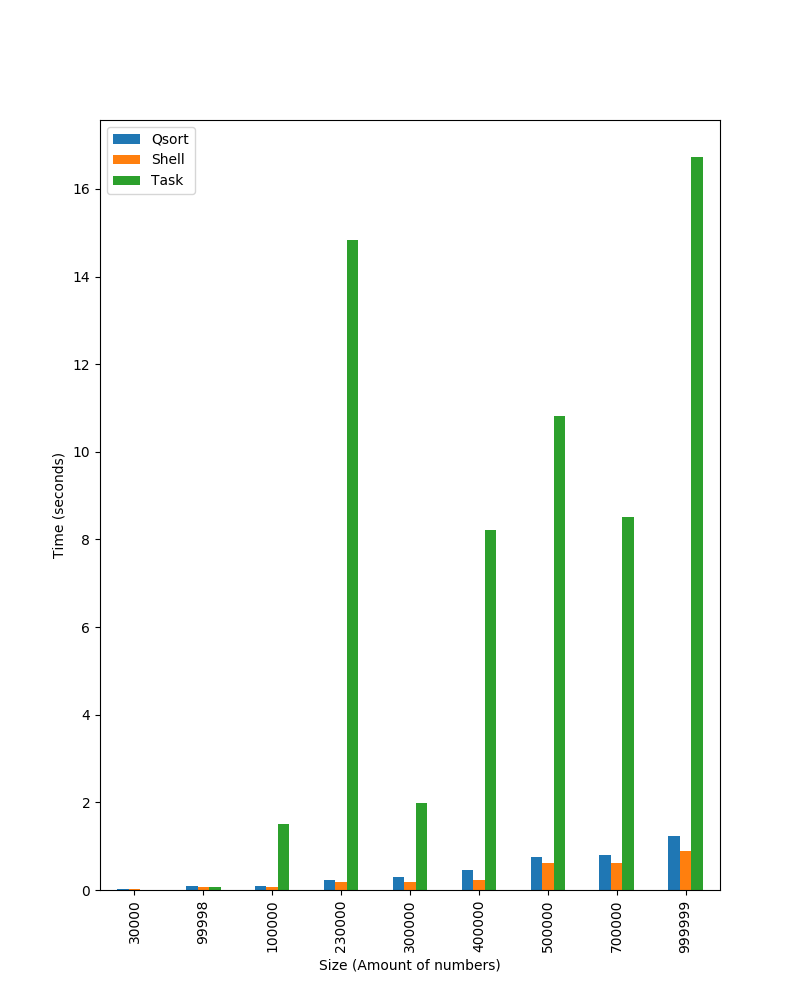
***Таблица 2.*** *Зависимость времени алгоритмов от размера массива*

|  |  |  |  |
| --- | --- | --- | --- |
| Размер массива | Время сортировки Шелла, с | Время быстрой сортировки qsort, с | Время решения задачи, с |
| 200  11  250  100  800  500  999  470 | 3.86e-04  2.48e-04  3.72e-04  2.24e-04  2.00e-06  1.39e-04  6.70e-05  1.50e-04 | 3.32e-04  1.64e-04  3.71e-04  1.53e-04  9.00e-06  1.13e-04  4.80e-05  1.43e-04 | 0.000358  0.000966  0.000482  0.000473  0.000007  0.000110  0.000014  0.000112 |
| 99998  300000  999999  30000  100000  500000  700000  400000  230000 | 0.074769  0.177278  0.898316  0.014634  0.075732  0.60579  0.617457  0.235878  0.184444 | 0.090564  0.291641  1.22846  0.02284  0.086071  0.75492  0.798376  0.453514  0.234696 | 5.79320e-02  1.98419e+00  1.67349e+01  6.83800e-03  1.51550e+00  1.08179e+01  8.50064e+00  8.21389e+00  1.48326e+01 |

***Рис.1.*** *Исследование на небольших значениях*

****

***Рис.1.*** *Исследование на небольших значениях*

**

**Пример работы программы.**

Press Enter to start

Select type of data input. 1 - file, 2 - keyboard, 3 - generate

1

File is:

23.23 32.32 10.1 19.1

29.2

The sequence S is:

23.23 32.32 10.1 19.1

The x is: 29.2

>After shell\_sort

The sequence S is:

10.1 19.1 23.23 32.32

>After qsort

The sequence S is:

10.1 19.1 23.23 32.32

>Checking sorts

Perfect! The sorts gave equal sequences!

>Solving task : if in S there is 2 elements with sum of 29.2 ?

Yes! 19.1 and 10.1

The amount of elements in S is 4

Time of shell\_sort : 0.000013 second(s)

Time of qsort : 0.000005 second(s)

Time to solve the task with previous input sorting : 0.000004 second(s)

4999,0.002803,0.00298,0.025305

49999,0.03573,0.038862,0.36502

**Тестирование.**

Тесты на корректных входных данных представлены в Талице 3.

***Таблица 3.*** *Тестирование корректных данных*

|  |  |  |  |
| --- | --- | --- | --- |
| ***№*** | ***Неотсортированный массив и X*** | ***Отсортированный массив*** | ***Результат и дополнительные данные*** |
| 1 | 48.4878 88.7657 78.6368 21.9796 65.4434 41.411 48.9115 83.1428 20.2904 88.8446 17.71 82.3601 32.3739 28.4533 68.8848 64.4186 43.2786 54.2085 9.8195 27.2873 73.7463 96.3027 31.7863 52.3112 98.0886 51.5989 6.36985 63.0339 35.3109 88.9947  x = 65.3134 | 6.36985 9.8195 17.71 20.2904 21.9796 27.2873 28.4533 31.7863 32.3739 35.3109 41.411 43.2786 48.4878 48.9115 51.5989 52.3112 54.2085 63.0339 64.4186 65.4434 68.8848 73.7463 78.6368 82.3601 83.1428 88.7657 88.8446 88.9947 96.3027 98.0886 | >Solving task : if in S there is 2 elements with sum of 65.3134 ? - No!  Time of shell\_sort : 0.000022 second(s)  Time of qsort : 0.000011 second(s)  Time to solve the task with previous input sorting : 0.000010 second(s) |
| 2 | 48.9356 91.621 52.8845 31.8574 59.4503  x = 96.6936 | 31.8574 48.9356 52.8845 59.4503 91.621 | >Solving task : if in S there is 2 elements with sum of 96.6936 ? - No!  Time of shell\_sort : 0.000016 second(s)  Time of qsort : 0.000007 second(s)  Time to solve the task with previous input sorting : 0.000005 second(s) |
| 3 | 48.9968 31.3781 22.4737 54.9601 68.2228 27.1734 39.8345 84.5406 49.7665 98.0746 13.8356 8.7592 26.8529 99.1271 81.1899 65.7083 37.848 59.9217 99.3237 18.907 3.28649 18.0602 27.5941 81.6921 91.3217 70.6274 48.0422 25.8849 66.1571 57.5345  x = 73.3112 | 3.28649 8.7592 13.8356 18.0602 18.907 22.4737 25.8849 26.8529 27.1734 27.5941 31.3781 37.848 39.8345 48.0422 48.9968 49.7665 54.9601 57.5345 59.9217 65.7083 66.1571 68.2228 70.6274 81.1899 81.6921 84.5406 91.3217 98.0746 99.1271 99.3237 | >Solving task : if in S there is 2 elements with sum of 73.3112 ? - No!  Time of shell\_sort : 0.000023 second(s)  Time of qsort : 0.000018 second(s)  Time to solve the task with previous input sorting : 0.000012 second(s) |
| 4 | 23.23 32.32 10.1 19.1 2.3 22.11 2.2 3.1 0 23.23 32.32 10.1 19.1 2.3 22.11 2.2 3.1 0 23.23 32.32 10.1 19.1 2.3 22.11 2.2 3.1 0 23.23 32.32 10.1 19.1 2.3 22.11 2.2 3.1 0  x = 0 | 0 0 0 0 2.2 2.2 2.2 2.2 2.3 2.3 2.3 2.3 3.1 3.1 3.1 3.1 10.1 10.1 10.1 10.1 19.1 19.1 19.1 19.1 22.11 22.11 22.11 22.11 23.23 23.23 23.23 23.23 32.32 32.32 32.32 32.32 | >Solving task : if in S there is 2 elements with sum of 0 ?  - Yes! 0 and 0  Time of shell\_sort : 0.000026 second(s)  Time of qsort : 0.000013 second(s)  Time to solve the task with previous input sorting : 0.000004 second(s) |
| 5 | 23.23 32.32 10.1 19.1  х = 29.2 | 10.1 19.1 23.23 32.32 | >Solving task : if in S there is 2 elements with sum of 29.2 ?  - Yes! 19.1 and 10.1  Time of shell\_sort : 0.000015 second(s)  Time of qsort : 0.000006 second(s)  Time to solve the task with previous input sorting : 0.000006 second(s) |
| 6 | 0 0 0 0 0 0 0 0 0 0 0 0  х = 1 | 0 0 0 0 0 0 0 0 0 0 0 0 | >Solving task : if in S there is 2 elements with sum of 1 ? - No!  Time of shell\_sort : 0.000015 second(s)  Time of qsort : 0.000008 second(s)  Time to solve the task with previous input sorting : 0.000006 second(s) |
| 7 | 98.2333 107.342 139.83 93.1011 177.776 86.4819 6.53622 186.252 185.084 146.504 185.4 122.072 7.61668 129.477 115.703  x = 82.9272 | 6.53622 7.61668 86.4819 93.1011 98.2333 107.342 115.703 122.072 129.477 139.83 146.504 177.776 185.084 185.4 186.252 | >Solving task : if in S there is 2 elements with sum of 82.9272 ? - No!  Time of shell\_sort : 0.000017 second(s)  Time of qsort : 0.000008 second(s)  Time to solve the task with previous input sorting : 0.000003 second(s) |

Тесты на некорректных входных данных при которых программа либо корректно завершает работу, либо предлагается возможность ввести их повторно (неограниченное количество раз), представлены в Талице 4.

***Таблица 4.*** *Тестирование некорректных данных*

|  |  |  |
| --- | --- | --- |
| **№** | **Диалог с пользователем и ввод данных** | **Работа программы** |
| 1 | >Generation of input  Enter quantity numbers to generate. It should be less than 1000000 :  **klk;l** | ERROR! with input! Would you like to continue and change the mistake (press 1) of exit (press 2):  **54rey0''**  Incorrect type of input. Try again. |
| 2 | ERROR! with input! Would you like to continue and change the mistake (press 1) of exit (press 2):  **2** | >Exit |
| 3 | Select type of data input. 1 - file, 2 - keyboard, 3 - generate  **6** | Incorrect type of input. Try again.  Select type of data input. 1 - file, 2 - keyboard, 3 - generate |
| 4 | >Generation of input  Enter quantity numbers to generate. It should be less than 1000000 : 20  Enter first range:  **dkl** | ERROR! with input! Would you like to continue and change the mistake (press 1) of exit (press 2):  **1**  Enter first range: |
| 5 | Enter last range:  **9i9i** | ERROR! with input! Would you like to continue and change the mistake (press 1) of exit (press 2):  **1**  Enter last range: |
| 6 | Если файл пуст.  Select type of data input. 1 - file, 2 - keyboard, 3 - generate  **1** | File is empty!  >Exit |
| 7 | Select type of data input. 1 - file, 2 - keyboard, 3 - generate  **9** | Incorrect type of input. Try again.  Select type of data input. 1 - file, 2 - keyboard, 3 - generate |

**Вывод.**

В результате выполнения этой лабораторной работы была исследована сортировка массива естественным слиянием, найдена зависимость количества базовых операций и времени от длины сортируемого массива и проведено сравнение с сортировкой простым слиянием и библиотечной функцией сортировки qsort.

**ПРИЛОЖЕНИЕ А**

**Исходный код программы**

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <fstream>

#include <vector>

#include <string>

#include <limits>

#include <random>

using namespace std;

#define MAX\_SIZE 1000000

fstream in("input.txt", ios::in);

fstream gen("generated.txt", ios::out);

int flag\_null = 0;

float first\_el = 0.0;

float second\_el = 0.0;

void cinclear()

{

cin.clear(); // if the latest input was wrong - flushing bufer

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

bool need\_exit()

{

\_start:

cinclear();

cout <<"ERROR! with input! Would you like to continue and change the mistake (press 1) of exit (press 2):\n";

int switch\_on;

cin >> switch\_on;

while(true){

switch(switch\_on){

case 1:

cinclear();

return false; // continue

case 2:

return true; // exit

default:

cout << "Incorrect type of input. Try again.\n";

goto \_start;

}

}

}

void safe\_exit()

{

cout << "\n>Exit" <<endl;

in.close(); gen.close(); exit(1);

}

bool incheck(string& buf, int& n) // not for real numbers

{

int i = 0;

while(i < buf.size()){

if(!isdigit(buf[i++])) return false; // not a number

}

n = atoi(buf.c\_str());

if(n < 0 || n > 1000000) return false; // error value

return true;

}

void generate(ostream &file, size\_t n) //sequence generation

{

int range\_first;

int range\_last;

string str;

\_gen1:

cout << "Enter first range: \n";

cin >> str;

if ( !incheck(str, range\_first)) {

if(need\_exit()) safe\_exit();

else goto \_gen1;

}

\_gen2:

cout << "Enter last range: \n";

str.clear(); cin >> str;

if ( !incheck(str, range\_last)) {

if(need\_exit()) safe\_exit();

else goto \_gen2;

}

if (range\_first >= range\_last ){

if (need\_exit()) safe\_exit();

else goto \_gen1;

}

static default\_random\_engine rnd(time(0));

static uniform\_real\_distribution<float> d(range\_first, range\_last);

while (n > 0) { //write to file

float l = d(rnd);

file << l;

file << " ";

n--;

}

}

void see\_file(fstream& file)

{

if ( !file )

{ cout << "ERROR! with opening file" << endl; safe\_exit(); }

file.seekg(0, ios::end);

int len = file.tellg();

if(len==0){

cout << "\nFile is empty!\n";

safe\_exit();

}

file.seekg(0, ios::beg);

cout << "\nFile is: \n";

int pos = 0;

while(pos != len){

char e;

e = file.peek();

file.seekg(++pos, ios::beg);

cout << e;

}

file.seekg(0, ios::beg);

cout <<endl;

}

void read\_vector(istream& my\_stream, vector<float> &s)

{

float f;

int flag\_met\_null = 0; // while 0 is not met

string fs;

while ( my\_stream >> fs && fs != "l"){

f = stof(fs, 0);

s.push\_back(f);

if( !flag\_met\_null && !f) { // for catching 0 in S

flag\_null = 1;

flag\_met\_null = 1;

}

}

}

bool print\_vector(vector<float> &s)

{

if(s.empty()) {

cout << "S is empty" <<endl;

return false;

}

cout <<endl<< "The sequence S is:\n";

for(int i = 0; i < s.size(); i++){

cout << s[i] << " ";

}

cout << endl<<endl;

return true;

}

template <typename T>

void shell\_sort(T arr[], size\_t distance)//, size\_t &quantit\_of\_equals, size\_t &quantit\_of\_assign)

{

cout <<">Shell\_sorting"<<endl;

for (int d = distance / 2; d > 0; d /= 2) {

cout << "\nThe distance is " << d <<endl <<endl;

for (size\_t i = 0; i < distance; i++) {

T temp = arr[i];

int j;

cout << " tmp = " << temp << endl;

for (j = i; (j >= d) && (arr[j - d] > temp) ; j -= d){ //,quantit\_of\_equals++) {

// cout << arr[j] << " = " << arr[j - d] << endl; //

arr[j] = arr[j - d]; //quantit\_of\_assign++;

}

// cout << arr[j] << " = " << temp << endl;

arr[j] = temp; //quantit\_of\_assign++; //

}

}

}

int compare(const void\* a, const void\* b)

{

float fa = \*(const float\*) a;

float fb = \*(const float\*) b;

return (fa > fb) - (fa < fb);

}

bool if\_there\_is\_sum( vector<float>& s\_ptr , const float& x)

{

float tmp = 0.0;

for (int i = s\_ptr.size()-1; i > 0; i--){

if ( x < s\_ptr[i] ) continue;

cout << "The possition when x >= s[i] was found. i = " << i << ": " << s\_ptr[i] << endl;

first\_el = s\_ptr[i];

if (flag\_null && x == s\_ptr[i]) return true;

tmp = x - s\_ptr[i];

second\_el = tmp;

cout << " tmp is " << tmp <<endl;

for (int j = i+1; j >= 0; j--){ // 1 candidate is found, finding the 2d

if( tmp == s\_ptr[j] ){ // 2 candidates for sum of x were found

return true;

}

if( s\_ptr[j] < tmp ) { // break if the difference between the 1st and the 2d candidates is bigger that the next candidate

cout << "As s\_ptr[j] < tmp go to find another candidate for sum of x (" << s\_ptr[j] << " < " << tmp << " )" <<endl;

break;

}

}

}

return false;

}

template <typename T>

void copy\_vector(vector <T>& orig, vector <T>& dest)

{

for(int i = 0; i < orig.size(); i++){

dest.push\_back(orig[i]);

}

}

template <typename T>

bool is\_vectos\_equal(vector <T>& v1, vector <T>& v2)

{

if(v1.size() != v2.size()) return false;

for(int i = 0; i < v1.size(); i++){

if (v1[i] != v2[i]) return false;

}

return true;

}

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

{

vector <float> s;

float x = 0;

if(!in || !gen) safe\_exit();

cout << "Press Enter to start";

int i = 0;

while(true){

cinclear();

cout << "\nSelect type of data input. 1 - file, 2 - keyboard, 3 - generate" <<endl;

int type = 0;

cin >> type;

switch(type){

case 1:

see\_file(in);

read\_vector(in, s); // read s

x = s[s.size()-1]; // get x

s.pop\_back();

if(!print\_vector(s)) continue; // print s with checking the input

cout << "The x is: " << x <<endl<<endl;

break;

case 2:

cout << "\nStart entering float sequence S with space delim, MAX\_SIZE = " << MAX\_SIZE << ", press 'l + Enter' to finish" <<endl;

read\_vector(cin, s);

cout << "Enter x: ";

cin >> x;

if(!print\_vector(s)) continue;

cout << "The x is: " << x <<endl<<endl;

break;

case 3:{

cout << "\n>Generation of input";

\_case3:

cout <<"\nEnter quantity numbers to generate. It should be less than 1000000 : ";

string n\_str;

cin >> n\_str; // read a number as string for check

int n = 0;

if ( !incheck(n\_str, n)) {

if(need\_exit()) safe\_exit();

else goto \_case3;

}

if (n<3) {

if(need\_exit()) safe\_exit();

else goto \_case3;

}

n += 1;

gen.close();

fstream gen("generated.txt", ios::in|ios::out);

generate(gen, n);

see\_file(gen);

read\_vector(gen, s); // read s, the last elem will be x

x = s[s.size()-1]; // get x

s.pop\_back();

if(!print\_vector(s)) continue; // print s with checking the input

cout << "The x is: " << x <<endl<<endl;

break;}

default:

cout << "Incorrect type of input. Try again.\n";

continue;

}

break;

}

clock\_t start, end;

vector <float> s1;

vector <float> s2;

copy\_vector<float>(s, s1);

copy\_vector<float>(s, s2);

// time check of shell\_sort

start = clock();

shell\_sort(s1.data(), s1.size());

end = clock();

double shell\_time = (double) (end - start);

cout <<endl<< ">After shell\_sort\n";

print\_vector(s1);

// time check of qsort

start = clock();

qsort(s2.data(), s2.size(), sizeof(float), compare);

end = clock();

double quick\_time = (double) (end - start);

cout << ">After qsort\n";

print\_vector(s2);

// check if the sorts gave qual sequences

cout <<">Checking sorts" <<endl;

if(is\_vectos\_equal(s1, s2)) cout <<"Perfect! The sorts gave qual sequences!" <<endl<<endl;

else {

cout <<"ERROR!! The sorts gave not qual sequences" <<endl;

safe\_exit();

}

// solving the task

cout <<">Solving task : if in S there is 2 elements with sum of " << x << " ?" <<endl;

start = clock();

bool res = if\_there\_is\_sum(s1, x);

end = clock();

double task\_time = (double) (end - start);

if (res) cout << "Yes! " << first\_el << " and " <<second\_el <<endl;

else cout << "No!" << endl;

// before getting an executable task time check that console dialog dors not exist

cout << "\nThe amount of elements in S is " << s.size() <<endl;

cout << std::fixed <<"\nTime of shell\_sort : "<< quick\_time/((double) CLOCKS\_PER\_SEC) << " second(s)" <<endl;

cout << std::fixed <<"\nTime of qsort : "<< shell\_time/((double) CLOCKS\_PER\_SEC) << " second(s)" <<endl;

cout << std::fixed << "\nTime to solve the task with previous input sorting : "<< task\_time / ((double) CLOCKS\_PER\_SEC) << " second(s)"<<endl;

in.close();

gen.close();

return 0;

}