МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«**Дальневосточный федеральный университет**»

**ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**

**Кафедра прикладной информатики, механики, управления и программного обеспечения**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

**ПО ДИСЦИПЛИНЕ**

**ФУНДАМЕНТАЛЬНЫЕ СТРУКТУРЫ ДАННЫХ И АЛГОРИТМЫ**

Направление 09.03.04 «Программная инженерия»

Выполнил студент гр. Б9120-09.03.04прогин  
Матюнин Е. Я.  
Проверил: доцент кафедры ПММУПО, к.т.н.  
Остроухова С. Н.

Владивосток

2021

**Неформальная постановка задачи**

Входные данные: текстовый файл, в первой строке – целое число n - количество строк файла (10 ≤ n ≤ 1000000). Каждая из последующих строк содержит данные о студентах ДВФУ, разделенных 1 пробелом. Все данные во входном файле корректные.

Выходные данные: текстовый файл, в первой строке - время, затраченное на сортировку входного файла, каждая из последующих строк содержит отсортированные данные (с помощью сортировки по Шеллу или сортировки выбором по убыванию).

1. Отсортировать данные входного файла в соответствии с заданным порядком сортировки ключей: 1 ключ – дата рождения, 2 ключ – номер группы, 3 ключ – ФИО студента.

2. Данные в выходном файле должны быть представлены в табличном виде. 1-3 столбцы должны содержат значение 1-3 ключей, отсортированные по убыванию. Оставшиеся поля записей выводятся в удобном для вас порядке.

3. Проверить на устойчивость (привести примеры, доказывающие, что сортировка неустойчивая).

4. Сравнить по времени (без учета чтения из файла и записи в файл).

5. Определить последовательности, на которых достигается наихудшее и наилучшее значение времени.

**Описание алгоритма сортировки**

*Сортировка по Шеллу:*сравниваются элементы, стоящие на определённом расстоянии друг от друга, и в случае необходимости меняются местами. С каждым шагом расстояние между элементами уменьшается, когда расстояние станет равно 1, алгоритм совершит последний проход.

*Сортировка простым выбором:*  
Шаги алгоритма:

1. находим номер минимального значения в текущем списке
2. производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции)
3. теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы

**Текст программы**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <intrin.h>

#pragma intrinsic(\_\_rdtsc)

using namespace std;

void generate\_input(size\_t n, string out\_fyle) {

setlocale(LC\_ALL, "rus");

string lane;

string direct[] = { "Физика", "Психология", "Экономика", "Химия", "Геология",

"География", "Биология", "Архитектура", "Машиностроение", "Инноватика" };

string grp\_num[] = { "Б9120-09.03.04прогин","Б9119-09.03.04прогин", "Б9120-09.03.03прогин",

"М3121-26.04.02кто", "М3121-26.04.02ксо", "М4120-37.04.01пм", "С3120-08.05.01сгспо",

"С3216-20.05.01птпп", "С3216-21.05.04мрд", "А8121-04.06.01фх", "А8121-04.06.01бох" };

string FIO[] = { "Агеев Андрей Владиславович","Аксенова Алёна Егоровна","Анохин Марк Николаевич",

"Безруков Иван Никитич","Ильина Варвара Львовна","Ильина Дарья Константиновна","Казанцев Григорий Михайлович",

"Калугина Виктория Дмитриевна","Кириллова Алиса Демьяновна",

"Киселева Арина Александровна","Ковалев Роман Александрович",

"Ковалев Артём Егорович","Козина Василиса Марковна","Козырев Кирилл Иванович",

"Лебедева Анна Львовна","Орлов Илья Русланович","Сорокина Валерия Тимуровна","Шестакова Таисия Павловна" };

srand(time(0));

ofstream output(out\_fyle, 'w');

output << n;

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

output << "\n";

lane = "";

int phone1 = rand() % 900 + 100;

int phone2 = rand() % 9000 + 1000;

int month = rand() % 12 + 1;

int day;

if (month == 2) {

day = rand() % 27 + 1;

}

else if (month % 2 == 1 or month == 8) {

day = rand() % 31 + 1;

}

else day = rand() % 30 + 1;

int year = rand() % 4 + 2000;

lane += FIO[rand() % 18] + ' ' + direct[rand() % 10] + ' ' + grp\_num[rand() % 11];

lane += ' ' + (day < 10 ? '0' + to\_string(day) : to\_string(day)) + '.'

+ (month < 10 ? '0' + to\_string(month) : to\_string(month)) + '.' + to\_string(year) + " 8914" + to\_string(phone1) + to\_string(phone2);

output << lane;

}

output.close();

}

void split(string s, char delim, vector<string>& data)

{

while (!s.empty())

{

auto word\_end = s.find(delim);

if (word\_end == -1)

{

data.push\_back(s);

s.clear();

}

else

{

data.push\_back(s.substr(0, word\_end));

s.erase(0, word\_end + 1);

}

}

}

struct FIO

{

string f;

string i;

string o;

};

struct BirthDate

{

int dd;

int mm;

int yy;

};

struct Student

{

FIO fio;

string direct;

string group\_num;

BirthDate bdate;

string phone\_num;

};

bool operator <(Student& l, Student& r)

{

string tempL = l.fio.f + " " + l.fio.i + " " + l.fio.o;

string tempR = r.fio.f + " " + r.fio.i + " " + r.fio.o;

if (l.bdate.yy < r.bdate.yy)

return true;

else if (l.bdate.yy > r.bdate.yy)

return false;

else if (l.bdate.yy == r.bdate.yy)

{

if (l.bdate.mm < r.bdate.mm)

return true;

else if (l.bdate.mm > r.bdate.mm)

return false;

else if (l.bdate.mm == r.bdate.mm)

{

if (l.bdate.dd < r.bdate.dd)

return true;

else if (l.bdate.dd > r.bdate.dd)

return false;

else if (l.bdate.dd == r.bdate.dd)

{

if (l.group\_num < r.group\_num)

return true;

else if (l.group\_num > r.group\_num)

return false;

else if (l.group\_num == r.group\_num)

{

if (tempL < tempR)

return true;

else if (tempL >= tempR)

return false;

}

}

}

}

}

//вычисляет коэффициент седжвика с номером s

long calcSegvik(int s)

{

if (s % 2)

return 8 \* pow(2.0, s) - 6 \* pow(2.0, (s + 1) / 2) + 1;

else

return 9 \* pow(2.0, s) - 9 \* pow(2.0, s / 2) + 1;

}

//вычисляет последовательность длин подсписков

int increment(long inc[], long size)

{

int i = 0;

do {

inc[i] = calcSegvik(i);

} while (3 \* inc[i++] < size);

return i - 1;

}

//сортировка вставками начиная с индекса start с шагом inc

//сортирует только элементы, которые входят в множество {a | a попадает в start + k \* inc}

void insertSort(Student\*& stud, int size, int start, int inc)

{

int location;

Student newelement;

for (int i = start + inc; i < size; i += inc) {

newelement = stud[i];

location = i - inc;

while (location >= start && stud[location] < newelement) {

stud[location + inc] = stud[location];

location -= inc;

}

stud[location + inc] = newelement;

}

}

//сортировка Шелла

void ShellSort(int n, Student\*& stud)

{

long seq[40];

//в s хранится последний элемент (его индекс) из массива коэф. седжвика seq

for (int s = increment(seq, n); s >= 0; s--)

for (int i = 0; i < seq[s]; i++)

insertSort(stud, n, i, seq[s]);

}

//сортировка выбором

void ChoiceSort(int n, Student\*& stud)

{

// Перебираем каждый элемент массива (кроме последнего, он уже будет отсортирован к тому времени, когда мы до него доберемся)

for (int startIndex = 0; startIndex < n - 1; ++startIndex)

{

// В переменной smallestIndex хранится индекс наименьшего значения, которое мы нашли в этой итерации.

// Начинаем с того, что наименьший элемент в этой итерации - это первый элемент (индекс 0)

int smallestIndex = startIndex;

// Затем ищем элемент поменьше в остальной части массива

for (int currentIndex = startIndex + 1; currentIndex < n; ++currentIndex)

{

// Если мы нашли элемент, который меньше нашего наименьшего элемента,

if (stud[smallestIndex] < stud[currentIndex])

// то запоминаем его

smallestIndex = currentIndex;

}

// smallestIndex теперь наименьший элемент.

// Меняем местами наше начальное наименьшее число с тем, которое мы обнаружили

std::swap(stud[startIndex], stud[smallestIndex]);

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

string temp;

vector<string> record;

vector<string> sub\_record;

/\*generate\_input(10, "DVFUStudents1.txt");

generate\_input(100, "DVFUStudents2.txt");

generate\_input(1000, "DVFUStudents3.txt");

generate\_input(10000, "DVFUStudents4.txt");

generate\_input(50000, "DVFUStudents5.txt");

generate\_input(1000000, "DVFUStudents6.txt");\*/

ifstream fi("DVFUStudents1.txt");

ofstream fout1("DVFUStudentsOut1.txt");

ofstream fout2("DVFUStudentsOut2.txt");

getline(fi, temp);

const int n = stoi(temp);

Student\* students = new Student[n];

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

{

getline(fi, temp);

split(temp, ' ', record);

students[i].fio.f = record[0];

students[i].fio.i = record[1];

students[i].fio.o = record[2];

students[i].direct = record[3];

students[i].group\_num = record[4];

split(record[5], '.', sub\_record);

students[i].bdate.dd = stoi(sub\_record[0]);

students[i].bdate.mm = stoi(sub\_record[1]);

students[i].bdate.yy = stoi(sub\_record[2]);

students[i].phone\_num = record[6];

sub\_record.clear();

record.clear();

}

double begin = clock();

ShellSort(n, students);

double end = clock();

double t1 = (end - begin);

fout1 << t1;

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

{

fout1 << "\n";

//students[i].bdate.dd < 10 ? '0' + to\_string(day) : to\_string(day)

fout1 << (students[i].bdate.dd < 10 ? '0' + to\_string(students[i].bdate.dd) : to\_string(students[i].bdate.dd)) << "." <<

(students[i].bdate.mm < 10 ? '0' + to\_string(students[i].bdate.mm) : to\_string(students[i].bdate.mm)) << "." <<

students[i].bdate.yy << " ";

int l = 23 - students[i].group\_num.size();

fout1 << students[i].group\_num;

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

fout1 << " ";

string fio = students[i].fio.f + " " + students[i].fio.i + " " + students[i].fio.o;

l = 31 - fio.size();

fout1 << fio;

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

fout1 << " ";

l = 17 - students[i].direct.size();

fout1 << students[i].direct;

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

fout1 << " ";

fout1 << students[i].phone\_num;

}

//begin = clock();

//ChoiceSort(n, students);

//end = clock();

//double t2 = (end - begin);

//fout2 << t2;

//for (size\_t i = 0; i < n; i++)

//{

// fout2 << "\n";

// //students[i].bdate.dd < 10 ? '0' + to\_string(day) : to\_string(day)

// fout2 << (students[i].bdate.dd < 10 ? '0' + to\_string(students[i].bdate.dd) : to\_string(students[i].bdate.dd)) << "." <<

// (students[i].bdate.mm < 10 ? '0' + to\_string(students[i].bdate.mm) : to\_string(students[i].bdate.mm)) << "." <<

// students[i].bdate.yy << " ";

// int l = 23 - students[i].group\_num.size();

// fout2 << students[i].group\_num;

// for (size\_t i = 0; i < l; i++)

// fout2 << " ";

// string fio = students[i].fio.f + " " + students[i].fio.i + " " + students[i].fio.o;

// l = 31 - fio.size();

// fout2 << fio;

// for (size\_t i = 0; i < l; i++)

// fout2 << " ";

// l = 17 - students[i].direct.size();

// fout2 << students[i].direct;

// for (size\_t i = 0; i < l; i++)

// fout2 << " ";

// fout2 << students[i].phone\_num;

//}

fi.close();

fout1.close();

fout2.close();

}

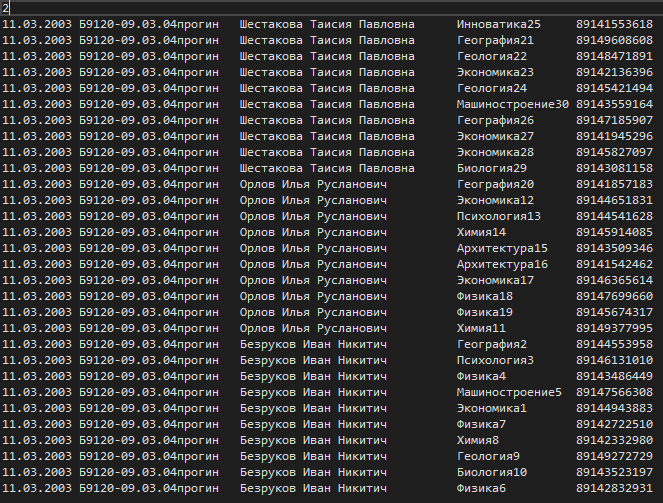
**Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовая ситуация | Входные и выходные данные | Время сортировки по Шеллу (шаг выбирается по Сэджвику) | Время сортировки простым выбором |
| Сортировка случайного массива (50 записей) | Вход: In\_DVFUStudents\_50.txt  Выход:  По Шеллу: Out\_DVFUStudents\_50.txt  Выбором: Out\_Selection\_DVFUStudents\_50.txt | 4 ms | 18 ms |
| Сортировка случайного массива (5000 записей) | Вход: In\_DVFUStudents\_5000.txt  Выход:  По Шеллу: Out\_DVFUStudents\_5000.txt  Выбором: Out\_Selection\_DVFUStudents\_5000.txt | 1441 ms | 159101 ms |
| Сортировка по второму ключу (одинаковые даты рождения) | Вход: In\_DVFUStudents\_key2.txt  Выход:  По Шеллу: Out\_DVFUStudents\_key2.txt  Выбором: Out\_Selection\_DVFUStudents\_key2.txt | 3 ms | 7 ms |
| Сортировка по третьему ключу (одинаковые даты рождения и номера групп) | Вход: In\_DVFUStudents\_key3.txt  Выход:  По Шеллу: Out\_DVFUStudents\_key3.txt  Выбором: Out\_Selection\_DVFUStudents\_key3.txt | 2 ms | 8 ms |

|  |  |  |  |
| --- | --- | --- | --- |
| Лучший случай для сортировки по Шеллу и выбором (массив отсортирован в правильном порядке, по убыванию) | Вход: In\_DVFUStudents\_best.txt  Выход:  По Шеллу: Out\_DVFUStudents\_best.txt  Выбором: Out\_Selection\_DVFUStudents\_best.txt | 102 ms | 6513 ms |
| Худший случай для сортировки по Шеллу (массив отсортирован в противоположном порядке, по возрастанию) | Вход: In\_DVFUStudents\_worstShell.txt  Выход:  По Шеллу: Out\_DVFUStudents\_worstShell.txt  Выбором: Out\_Selection\_DVFUStudents\_worstShell.txt | 193 ms | 6672 ms |
| Худший случай для сортировки выбором (массив отсортирован в правильном порядке, по убыванию, но последний элемент стоит на первом месте) | Вход: In\_DVFUStudents\_worstSelection.txt  Выход:  По Шеллу: Out\_DVFUStudents\_worstSelection.txt  Выбором: Out\_Selection\_DVFUStudents\_worstSelection.txt | 148 ms | 7605 ms |

**Проверка на устойчивость**

*Изображение выглядит как текст

Автоматически созданное описаниеСортировка по Шеллу*: входной файл – In\_DVFUStudents\_stable.txt, выходной файл – Out\_ DVFUStudents\_stable.txt.

Тест показал, что сортировка по Шеллу является неустойчивой, так как после сортировки порядок элементов поменялся (можно заметить на приведенных выше скриншотах по названию направления подготовки, что после сортировки порядок поменялся) (первый скриншот – до сортировки, второй – после сортировки)

*Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст, табличка

Автоматически созданное описаниеСортировка простыми вставками*: входной файл - In\_DVFUStudents\_stableSelection.txt, выходной файл - Out\_Selection\_DVFUStudents\_stable.txt.

Тест показал, что сортировка выбором является неустойчивой, так как после сортировки порядок элементов поменялся (можно заметить на приведенных выше скриншотах по названию направления подготовки, что после сортировки порядок поменялся (Орлов Илья)) (первый скриншот – до сортировки, второй – после сортировки)

**Вывод**

На небольших объемах данных скорость работы сортировок отличается незначительно, но при увеличении объема данных сортировка Шелла работает в разы быстрее сортировки простым выбором.

Обе сортировки показали лучший результат при уже отсортированном массиве.