МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

## Факультет информационных технологий и робототехники

Кафедра программного обеспечения вычислительной техники

и автоматизированных систем

**Отчет по лабораторной работе № 12**

по дисциплине: ”Основы алгоритмизации и программирования”

на тему: ***”*Алгоритмы поиска и сортировки*”***

Вариант 9

Выполнил**:** студенты группы *10701118*

Воробей И.А.

Приняла**:** ст.пр. Борисова И.М.

Минск 2019

## Цель работы: Изучить общие сортировки и алгоритмы поиска.

**Задание 1.** Считайте все записи из файла "9.dat" (Лаб.раб.9. Задание 2). Для чтения каждой отдельной записи осуществите динамический захват памяти. Предполагается, что к-во записей в файле заранее неизвестно.

Выполните сортировку записей по различным ключам:

|  |  |
| --- | --- |
| • | по фамилии (по возрастанию); |
| • | по окладу (по убыванию). |

Для каждого ключа используйте различные алгоритмы сортировки:

|  |  |
| --- | --- |
| • | сортировка выбором, |
| • | сортировка Шелла, |

Всего сортировка будет выполнена 4 раза, каждый раз сортируется массив в первоначальном состоянии.

Для каждого случая подсчитайте к-во сравнений и перестановок.

Оцените скорость, естественность и устойчивость каждого алгоритма.

## Оптимальную организацию функций продумайте самостоятельно.

### Код программы:

#include<iostream>

#include<fstream>

using namespace std;

struct NOTE {

char \*surname = new char;

char \*name = new char;

int \*d = new int[3];

int salary;

};

NOTE\* read(int\* num)

{

NOTE \*mass=new NOTE;

ifstream in;

in.open("9.txt",ios::binary|ios::in);

int n = 0;

for (;; n++){

NOTE \*ms1 = new NOTE;

if (!in.read((char\*)ms1, sizeof(NOTE)))

break;

mass[n] = \*ms1;

}

in.close();

\*num = n;

return mass;

}

void output(NOTE \*ar, int k){

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

cout << "Element number " << i << "\n";

cout << "Surname : " << ar[i].surname;

cout << "\nName : " << ar[i].name << "\nDate : ";

for (int k = 0; k < 3; k++)

cout << ar[i].d[k] << ".";

cout << "\nSalary : " << ar[i].salary << "\n\n";

}

}

void selection(NOTE \*ar,int k,bool fl){

int i1,comparison=0,permutation=0;

NOTE m;

for (int i = 0; i < k - 1; i++){

i1 = i;

for (int j = i + 1; j < k; j++){

if (fl){

comparison++;

if (strcmp(ar[i1].surname, ar[j].surname)>0)

i1 = j;

}

else{

comparison++;

if (ar[i1].salary< ar[j].salary)

i1 = j;

}

}

if (i != i1){

permutation++;

m = ar[i];

ar[i] = ar[i1];

ar[i1] = m;

}

}

output(ar, k);

cout << "Comparisons=" << comparison << "\nPermutations=" << permutation;

}

void shel(NOTE \*ar, int k, bool fl){

int flg, comparison = 0, permutation = 0;

NOTE n;

for (int gap = k / 2; gap > 0; gap = gap / 2)

do{

flg = 0;

for (int i = 0, j = gap; j < k; i++, j++){

if (fl){

comparison++;

if (strcmp(ar[i].surname, ar[j].surname)>0){

permutation++;

n = ar[j];

ar[j] = ar[i];

ar[i] = n;

flg = 1;

}

}

else{

comparison++;

if (ar[i].salary < ar[j].salary){

permutation++;

n = ar[j];

ar[j] = ar[i];

ar[i] = n;

flg = 1;

}

}

}

} while (flg);

output(ar, k);

cout << "Comparisons=" << comparison << "\nPermutations=" << permutation;

}

int write() {

int number;

cout << "Enter number of employees : ";

cin >> number;

NOTE \*ms = new NOTE[number];

NOTE \*ptr = ms;

void \*end = ms + number;

do{

cout << "\nEnter surname : "; cin >> ptr->surname;

cout << "Enter name : "; cin >> ptr->name;

cout << "Enter dd,mm,yyyy :"; cin >> ptr->d[0]; cin >> ptr->d[1]; cin >> ptr->d[2];

cout << "Enter salary : "; cin >> ptr++->salary;

} while (ptr<end);

ofstream fout;

fout.open("9.txt", ofstream::app | ios::binary | ios::out);

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

fout.write((char\*)&ms[n], sizeof(NOTE));

fout.close();

cout << "Entering was complet, massiv has loaded into the file.\n";

delete ms;

return number;

}

void main(){

bool fl = true;

int size=1,\*num=&size;

NOTE \*mass = new NOTE, \*mass1 = new NOTE, \*mass2 = new NOTE,\*mass3 = new NOTE;

write();

mass=mass1=mass2=mass3=read(num);

size=\*num;

cout << "\nMassive in the ferst state :\n\n";

output(mass, size);

cout << "SELECTION sort by surname :\n\n";

selection(mass,size,true);

cout << "\nSELECTION sort by salary :\n\n";

selection(mass1, size,false);

cout << "\nSHEL sort by surname :\n\n";

shel(mass2, size, true);

cout << "\nSHEL sort by salary :\n\n";

shel(mass3, size, false);

system("pause");

}

**Скриншотырезультатов:**

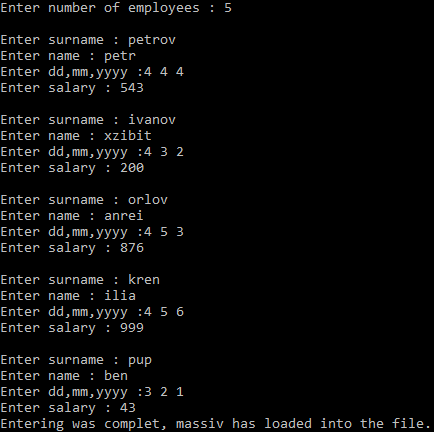
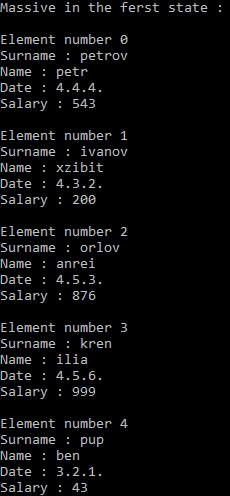
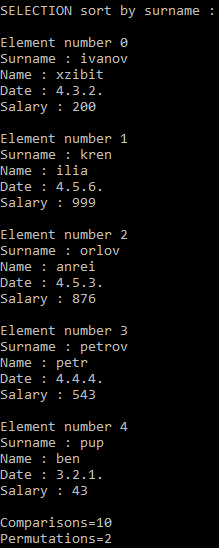
****

Рисунок 2

Рисунок 1

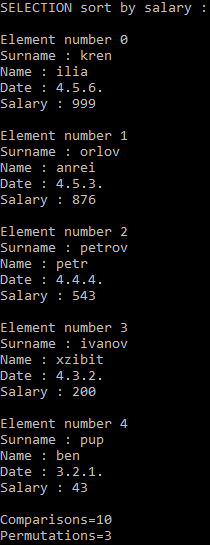
****

Рисунок 3

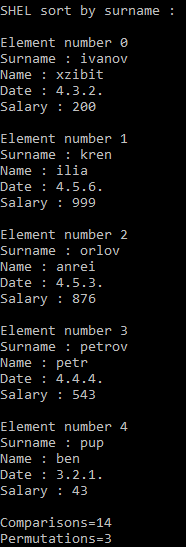
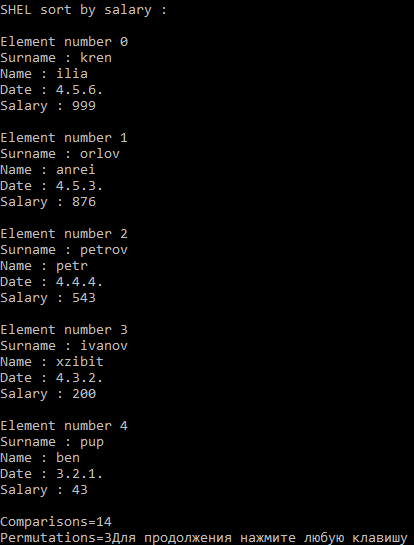
****

Рисунок 5

Рисунок 4

**Задание 2.** Напишите функцию search\_record для поиска записи в массиве из предыдущего задания по определенному ключу. Ключом может являться любое поле структуры (фамилия, имя, номер телефона, дата рождения, оклад).

Причем функция должна выполнять последовательный поиск, если массив не сортирован, и бинарный поиск, если массив сортирован.

Оптимальную организацию функции продумайте самостоятельно.

## Продемонстрируйте работу функции для всех вариантов поиска.

**Код программы:**

#include<iostream>

#include<fstream>

using namespace std;

struct NOTE {

char \*surname = new char;

char \*name = new char;

int \*d = new int[3];

int salary;

};

void output(NOTE \*ar, int k){

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

cout << "Element number " << i << "\n";

cout << "Surname : " << ar[i].surname;

cout << "\nName : " << ar[i].name << "\nDate : ";

for (int k = 0; k < 3; k++)

cout << ar[i].d[k] << ".";

cout << "\nSalary : " << ar[i].salary << "\n\n";

}

}

NOTE\* write(int\* size) {

int number;

cout << "Enter number of employees : ";

cin >> number;

\*size = number;

NOTE \*ms = new NOTE[number];

NOTE \*ptr = ms;

void \*end = ms + number;

do{

cout << "\nEnter surname : "; cin >> ptr->surname;

cout << "Enter name : "; cin >> ptr->name;

cout << "Enter dd,mm,yyyy :"; cin >> ptr->d[0]; cin >> ptr->d[1]; cin >> ptr->d[2];

cout << "Enter salary : "; cin >> ptr++->salary;

} while (ptr<end);

system("cls");

return ms;

}

void selection(NOTE \*ar, int k, bool fl){

int i1, comparison = 0, permutation = 0;

NOTE m;

for (int i = 0; i < k - 1; i++){

i1 = i;

for (int j = i + 1; j < k; j++){

if (fl){

comparison++;

if (strcmp(ar[i1].surname, ar[j].surname)>0)

i1 = j;

}

else{

comparison++;

if (ar[i1].salary< ar[j].salary)

i1 = j;

}

}

if (i != i1){

permutation++;

m = ar[i];

ar[i] = ar[i1];

ar[i1] = m;

}

}

output(ar, k);

cout << "Comparisons=" << comparison << "\nPermutations=" << permutation;

}

void search\_record(NOTE \* arr, int key,int size){

char \*xnm = new char;

switch(key){

case 1:

cout << "Enter searched name : ";

cin >> xnm;

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

if (!strcmp(xnm, arr[i].name)){

cout << "Surname : " << arr[i].surname;

cout << "\nName : " << arr[i].name << "\nDate : ";

for (int k = 0; k < 3; k++)

cout << arr[i].d[k] << ".";

cout << "\nSalary : " << arr[i].salary << "\n\n";

}

}

break;

case 2:

int x, left = 0, right = size;

cout << "Enter searched salary : ";

cin >> x;

int midd = 0;

while (1)

{

midd = (left + right) / 2;

if (x > arr[midd].salary) // если искомое больше значения в ячейке

right = midd - 1; // смещаем правую границу поиска

else if (x < arr[midd].salary) // если искомое меньше значения в ячейке

left = midd + 1; // смещаем левую границу поиска

else{ // иначе (значения равны)

cout << "Surname : " << arr[midd].surname;

cout << "\nName : " << arr[midd].name << "\nDate : ";

for (int k = 0; k < 3; k++)

cout << arr[midd].d[k] << ".";

cout << "\nSalary : " << arr[midd].salary << "\n\n";

break;

}

if (left > right){ // если границы сомкнулись

cout << "Element not exist.\n";

break;

}

}

}

}

void main(){

bool fl = true;

int size=1, \*num=&size,key;

NOTE \*mass = new NOTE,xxx;

mass=write(num);

cout << "Entering was complet.\n";

size = \*num;

cout << "\nMassive in the ferst state :\n\n";

output(mass, size);

cout << "\nSELECTION sort by salary :\n\n";

selection(mass, size, false);

do{

cout << "\n1)Name(unsorted)\t2)Salary(sorted)\t3)STOPE\nEnter key to search : "; cin >> key;

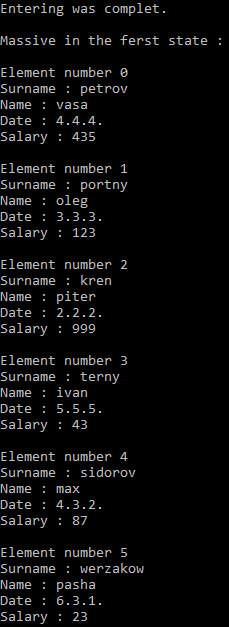
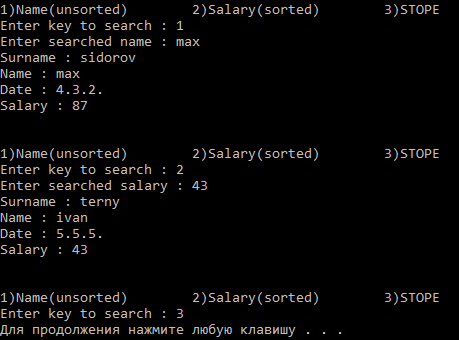
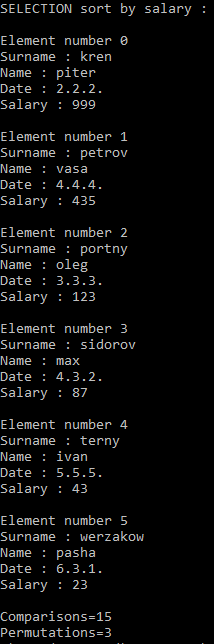
search\_record(mass, key, size);

} while (key != 3);

system("pause");

}

### Скриншотырезультатов:

**Вывод: я изучил общие сортировки и алгоритмы поиска**.