**1. База даних співробітників фірми містить наступні дані: паспортні дані, освіта,**

**спеціальність, посада, оклад. Описати структуру запису, який би містив вказані дані та**

**організувати збереження записів у файлі. Передбачити процедури додавання, вилучення**

**записів та пошуку записів за довільним запитом.**

#include "stdafx.h"

#include <iostream>

#include<string>

#include<fstream>

using namespace std;

struct Worker {

string ID;

string education;

string speciality;

string position;

double salary;

};

void inputOneWorker(Worker &worker) {

cout << "--------------------------------\n";

cout << "Passport: "; cin >> worker.ID;

cout << "Education: "; cin >> worker.education;

cout << "Speciality: "; cin >> worker.speciality;

cout << "Position: "; cin >> worker.position;

cout << "Salary: "; cin >> worker.salary;

cout << "--------------------------------\n";

}

void inputWorkers(Worker\* &workers, int& workersCount) {

cout << "Amount of workers: ";

cin >> workersCount;

workers = new Worker[workersCount];

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

inputOneWorker(workers[i]);

}

void copyWorkers(Worker &newWorker, Worker &oldWorker) {

newWorker.ID = oldWorker.ID;

newWorker.education = oldWorker.education;

newWorker.speciality = oldWorker.speciality;

newWorker.position = oldWorker.position;

newWorker.salary = oldWorker.salary;

}

void coutOneWorker(Worker& oneWorker) {

cout << "Passport: " << oneWorker.ID << endl;

cout << "Education: " << oneWorker.education << endl;

cout << "Speciality: " << oneWorker.speciality << endl;

cout << "Position: " << oneWorker.position << endl;

cout << "Salary: " << oneWorker.salary << endl;

cout << "--------------------------------\n";

}

void coutWorkers(Worker\* workers, int workersAmount) {

cout << "\n\_\_\_\_\_\_\_\_List of workers\_\_\_\_\_\_\_\_\_\n";

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

coutOneWorker(workers[i]);

}

void addWorker(Worker\* &workers, int& workersCount, Worker& newWorker) {

Worker\* workersNew = new Worker[workersCount+1];

for (int i = 0; i < workersCount; i++) //coping previous workers data

copyWorkers(workersNew[i], workers[i]);

workersNew[workersCount] = newWorker;

workersCount++;

delete[] workers; //delete previous array of workers data

workers = workersNew; //now workers has address of workersNew array

}

bool equal(Worker worker\_1, Worker worker\_2) {

return (worker\_2.ID == "-" || worker\_1.ID == worker\_2.ID) &&

(worker\_2.education == "-" || worker\_1.education == worker\_2.education) &&

(worker\_2.speciality == "-" || worker\_1.speciality == worker\_2.speciality) &&

(worker\_2.position == "-" || worker\_1.position == worker\_2.position) &&

(worker\_2.salary == -1 || worker\_1.salary >= worker\_2.salary);

}

Worker inputCriterionForSearch() {

Worker worker;

cout << "Enter passport (or -) : "; cin >> worker.ID;

cout << "Enter education (or -): "; cin >> worker.education;

cout << "Enter speciality (or -): "; cin >> worker.speciality;

cout << "Enter position (or -): "; cin >> worker.position;

cout << "Enter salary (or -1): "; cin >> worker.salary;

return worker;

}

void deleteWorkerByIndex(Worker\* &workers, int& workersCount, int workerIndex) {

for (int j = workerIndex + 1; j < workersCount; j++)

{

workers[j - 1] = workers[j];

}

workersCount--;

}

void deleteWorkersByCriterion(Worker\* &workers, int& workersCount, Worker criterion) {

for (int i = 0; i < workersCount; )

{

if (equal(workers[i], criterion))

{

deleteWorkerByIndex(workers, workersCount, i);

}

else

i++;

}

}

bool searchWorkers(Worker\* workers, int workersCount, Worker criterion) {

bool found = false;

cout << "---------wanted worker(s)-------\n";

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

if (equal(workers[i], criterion)) {

coutOneWorker(workers[i]);

found = true;

}

}

return found;

}

void saveToFileWorker(ofstream& file, Worker worker)

{

file << worker.ID << endl;

file << worker.education << endl;

file << worker.speciality << endl;

file << worker.position << endl;

file << worker.salary << endl;

}

void saveToFileWorkers(Worker\* workers, int workersCount, string fileName)

{

ofstream file(fileName);

file << workersCount << endl;

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

{

saveToFileWorker(file, workers[i]);

}

file.close();

}

void readFromFileWorker(ifstream& file, Worker& worker)

{

file >> worker.ID;

file >> worker.education;

file >> worker.speciality;

file >> worker.position;

file >> worker.salary;

}

void readFromFileWorkers(Worker\* &workers, int & workersCount, string fileName)

{

ifstream file(fileName);

file >> workersCount;

workers = new Worker[workersCount];

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

{

readFromFileWorker(file, workers[i]);

}

file.close();

}

void main()

{

Worker\* workersList=0;

int workersCount = 0;

Worker criterion, temp;

string fileName;

int c;

do {

cout << "1. Add"<< endl;

cout << "2. Remove by index" << endl;

cout << "3. Remove by criterion" << endl;

cout << "4. Search." << endl;

cout << "5. Save to file" << endl;

cout << "6. Read from file" << endl;

cout << "7. Print" << endl;

cout << "8. Exit" << endl;

cout << "Your choise: ";

cin >> c;

switch(c)

{

case 1:

cout << "------- Enter one worker's info -------------" << endl;

inputOneWorker(temp);

addWorker(workersList, workersCount, temp);

cout << "--- Done ----- " << endl;

break;

case 2:

cout << "Your index:";

int t;

cin >> t;

deleteWorkerByIndex(workersList, workersCount, t);

cout << "--- Done ----- " << endl;

break;

case 3:

//criterion = inputCriterionForSearch();

cout << "------- Enter criterion -------------" << endl;

deleteWorkersByCriterion(workersList, workersCount, inputCriterionForSearch() );

cout << "--- Done ----- " << endl;

break;

case 4:

cout << "------- Enter criterion -------------" << endl;

if (!searchWorkers(workersList, workersCount, inputCriterionForSearch()))

cout << "No such workers." << endl;

cout << "--- Done ----- " << endl;

break;

case 5:

cout << "File name :";

cin >> fileName;

saveToFileWorkers(workersList, workersCount, fileName);

cout << "--- Done ----- " << endl;

break;

case 6:

cout << "File name :";

cin >> fileName;

if (workersList!=0)

{

delete[] workersList;

}

readFromFileWorkers(workersList, workersCount, fileName);

cout << "--- Done ----- " << endl;

break;

case 7:

coutWorkers(workersList, workersCount);

cout << "--- Done ----- " << endl;

break;

}

system("pause");

} while (c != 8);

delete[] workersList;

workersList = 0;

}