МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп'ютерні інформаційні технології»

**Лабораторна робота №9**

**з дисципліни «Основи програмування»**

**на тему: «**Файлове введення-виведення. Бінарні файли. Пошук і сортування даних.**»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Нежуміра О. І.

Дніпро, 2020

**Тема.** Файлове введення-виведення. Бінарні файли. Пошук і сортування даних.

**Мета:**

1) сформувати навики роботи з бінарними файлами;

2) засвоїти реалізацію файлового введення-виведення у стилі мови С++;

3) ознайомитися з алгоритмами пошуку і сортування даних. 3) постановка завдання (завдання до лабораторної роботи, індивідуальне завдання);

# **Постановка завдання (завдання до лабораторної роботи, індивідуальне завдання).**

Розробити багатофайлову програму для роботи з бінарними файлами для заданого індивідуального завдання і обраного рівня складності.

Вимоги до програми:

− імена файлів ввести з клавіатури;

− в оперативної пам’яті дозволяється зберігати не більше двох блоків (структур) з вхідного файлу (файл в оперативну пам’ять цілком не зчитувати!);

− передбачити перевірку всіх операцій роботи з файлами (відкриття, читання, запису);

− управління виконанням програми здійснити на основі текстового меню користувача;

− передбачити послідовну обробку необмеженої кількості файлів; Вимоги до тексту програми:

− специфікації програми і її функцій;

− коментарі щодо призначення блоків програми, дій окремих операторів для пояснення алгоритму;

− самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № з/п | Предметна область | Дані | Рівень складності | | | | |
| Е  (достатньо) | D  (задовільно) | C  (добре) | B  (дуже добре) | A  (відмінно) |
| 13 | Програміст | • прізвище та ініціали  • назва компанії  • стаж роботи  • знання (структура з трьох полів – ОС, мова програмування, БД) | • Додавання програміста в список  • Виведення списку програмістів | Виведення списку програмістів з вказаним стажем роботи | Пошук програмістів з найбільшим стажем роботи | Пошук вказаного програмісту в списку | Сортування списку програмістів за стажем роботи |

# **Зовнішні специфікації програми (вхідні та вихідні дані, функціональні вимоги до програми).**

*Формат вхідних даних:*

Таблиця 2

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування даних | Умовне позначення | Вимоги до данних | Приклад |
| Вибір | choice | Ціле число з діапазону [1;7] | 3 |
| Ім'я(Прізвище та ініциали) програміста | SE.Name | Масив символів довжиною до 80 | George Smith |
| Назва компанії програміста | SE.Company | Масив символів довжиною до 80 | SpaceX |
| Стаж роботи програмиста(у днях) | SE.WE | Натуральне число | 521 |
| Операційна система програміста | SE.Software.OS | Масив символів довжиною до 80 | Kali |
| Мова програмування програміста | SE.Software.PLang | Масив символів довжиною до 80 | C++ |
| База даних програміста | SE.Software.DB | Масив символів довжиною до 80 | MySQL |

*Формат вихідних даних:*

Таблиця 3

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування даних | Умовне позначення | Вимоги до данних | Приклад |
| Ім'я(Прізвище та ініциали) програміста | SE.Name | Масив символів довжиною до 80 | George Smith |
| Назва компанії програміста | SE.Company | Масив символів довжиною до 80 | SpaceX |
| Стаж роботи програмиста(у днях) | SE.WE | Натуральне число | 521 |
| Операційна система програміста | SE.Software.OS | Масив символів довжиною до 80 | Kali |
| Мова програмування програміста | SE.Software.PLang | Масив символів довжиною до 80 | C++ |
| База даних програміста | SE.Software.DB | Масив символів довжиною до 80 | MySQL |

# **Модульна структура програми.**

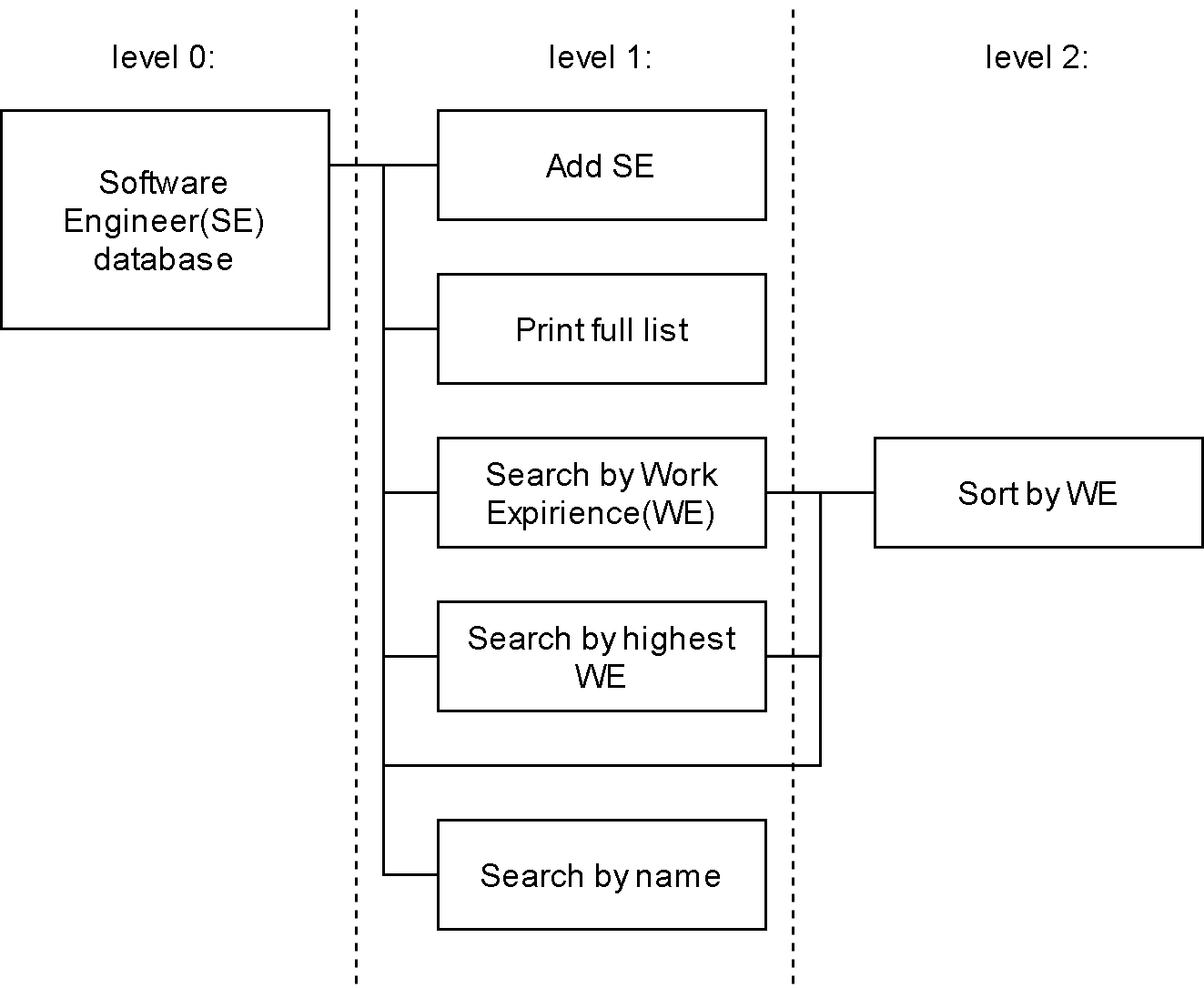


Рисунок 1

# **4. Алгоритми для розв’язання завдання всіх модулів програми.**

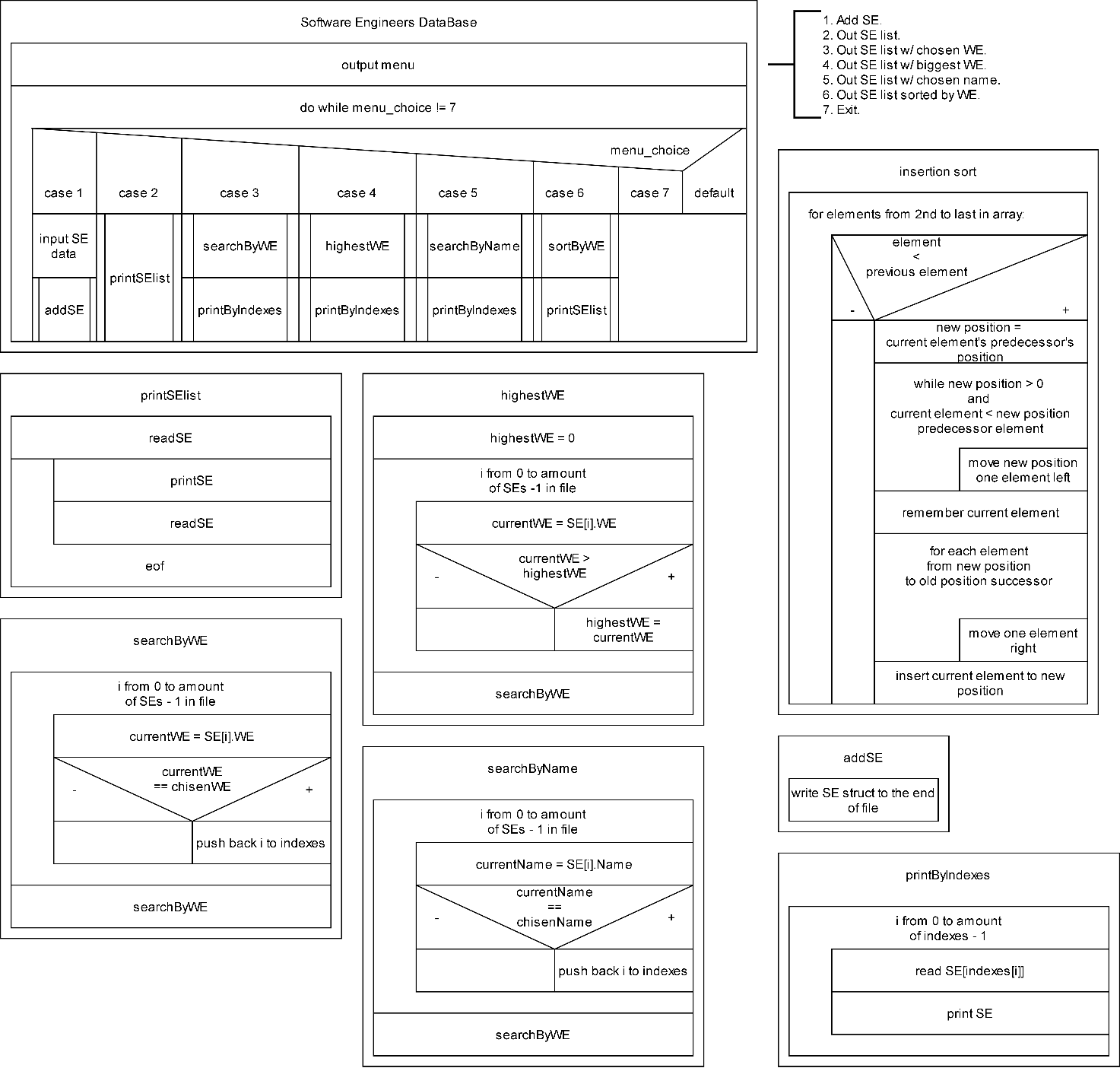


Рисунок 2

# **5. Набори тестів для перевірки правильності виконання всіх модулів програми.**

5.1 набір тестів для перевірки вводу даних (пункт меню користувача, поля структури).

Таблиця 4

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікуваний результат |
| 1 | Меню Пункт діапазон 1-6 | choice = 1 | Перехід до відповідного блоку програми |
| 7 | Меню Пункт 7 | choice = 7 | Вихід з програми |
| 8 | Меню Символ | choice = f | Повідомленя про помилку, поновлення вводу |
| 9 | Меню Дріб | choice = 9.147 | Повідомленя про помилку, поновлення вводу |
| 10 | Меню не належить діапазону | choice = 8 | Повідомленя про помилку, поновлення вводу |
| 11 | Ім’я з декількох слів, загальною довжиною більше 24 символів | Vasya Peter Colin John Smithovic | Повідомленя про помилку, поновлення вводу |
| 12 |  |  | Назва компанії та інші елементи окрім елемента “Опит роботи”, вводяться тим самим чином, що й Ім'я, тож тестування цього пункту не є необхідним. |
| 13 |  |  | Елемент “Опит роботи” перевіряється тією самою функцією, що й вибір у меню, тож тестування цього пункту не є необхідним. |

5.2. набір тестів для перевірки роботи меню користувача.

Таблиця 5

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікуваний результат |
| 1 | Меню Пункт 1 | choice = 1 | Перехід до блоку програми «Додавання нового програміста» |
| 2 | Меню Пункт 2 | choice = 2 | Перехід до блоку програми «Виведення усіх програмістів» |
| 3 | Меню Пункт 3 | choice = 3 | Перехід до блоку програми «Виведення програмістів із вказаним опитом роботи» |
| 4 | Меню Пункт 4 | choice = 4 | Перехід до блоку програми «Виведення програмістів із найбільшим опитом роботи» |
| 5 | Меню Пункт 5 | choice = 5 | Перехід до блоку програми «Виведення програмістів із вказаним іменем» |
| 6 | Меню Пункт 6 | choice = 6 | Перехід до блоку програми «Виведення відсортованого за опитом роботи списку усіх програмістів» |
| 7 | Меню Пункт 7 | choice = 7 | Припинення роботи з файлом |

5.3. набори тестів для перевірки виконання задач обробки масиву структур.

Таблиця 6

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікуваний результат |
|  | Задані структури | {"George Washington", "SpaceX", 521, {"Kali linux", "C++", "SQL"}},  {"Joseph Joestar", "Atari", 176, {"Arch Linux", "Java", "MongoDB"} },  {"John Watson", "IBM", 938, {"Windows 10", "Ruby", "Oracle"} } |  |
| 1 | Сортування за опитом роботи | - | {"John Watson", "IBM", 938, {"Windows 10", "Ruby", "Oracle"}},  {"George Washington", "SpaceX", 521, {"Kali linux", "C++", "SQL"}},  {"Joseph Joestar", "Atari", 176, {"Arch Linux", "Java", "MongoDB"}} |
| 2 | Виведення програмістів із вказаним опитом роботи | chosen\_WE = 521 | {"George Washington", "SpaceX", 521, {"Kali linux", "C++", "SQL"}} |
| 3 | Виведення програмістів із вказаним іменем | Chosen\_Name = "George Washington" | {"George Washington", "SpaceX", 521, {"Kali linux", "C++", "SQL"}} |
| 4 | Виведення програмістів із найбільшим опитом роботи | - | {"John Watson", "IBM", 938, {"Windows 10", "Ruby", "Oracle"}} |

# **6. Текст програми (всі файли проекту).**

“main.cpp”

//Application behaves like a simple database it stores 3 default Software Engineer type structure with fields:

// Name, Company, Work Experience, skills structure type name software that contains next fields:

//Operating System, Programming Language, DataBase

//it has 6 main functions:

//1 - Adding new software enginner(program can store only 10 at time)

//2 - printing list of all software engineers

//3 - printing list of software engineers with chosen work expirience

//4 - printing list of software engineers with biggest work expirience

//5 - printing list of software engineers with chosen name

//6 - printing list of all softare engineers sorted by work expirience

#include <iostream>

#include <fstream>

#include <vector>

#include "SE\_DB.h"

#include "inp\_val7.h"

int main() {

bool exit;

do {

char databaseFileName[256];//database.bin

std::fstream SEdatabase;

do {

std::fill\_n(databaseFileName, 256, 0);//0 = '\0'

std::cout << "input address of file(or only name if it is in the same folder as programm) starting with disk:\\ and ending with \*.txt: " << std::endl;

std::cin.getline(databaseFileName, 255);

SEdatabase.open(databaseFileName, /\*std::ios::trunc | \*/std::ios::in | std::ios::out | std::ios::binary);//old file

if (!SEdatabase.is\_open())

std::cout << "Can't open that file!" << std::endl;

} while (!SEdatabase.is\_open());

//SE\_DBfill(SEdatabase);//

int choice;

do {

std::cout << "1. Add software engineer." << std::endl;

std::cout << "2. Output list of software engineers." << std::endl;//E

std::cout << "3. Output software engineers with chosen work expirience." << std::endl;//D

std::cout << "4. Output software engineers with biggest work expirience." << std::endl;//C

std::cout << "5. Output chosen software engineer." << std::endl;//B

std::cout << "6. Output list of software engineers sorted by work expirience." << std::endl;//A

std::cout << "7. Exit." << std::endl;

choice = cinum("Choose what you want to do:", "Wrong input, try again", 1, 1, 1, 7);

switch (choice) {

case 1: {

system("cls");

SE\_ SEtmp;

std::cout << "Adding new software engineer..." << std::endl;

std::cout << "input name(can't be longer than 23 characters):" << std::endl;

std::cin.get();//deleting enter from stream after menu\_choice

val\_char\_lenght(SEtmp.Name, 24);

std::cout << "input company:" << std::endl;

val\_char\_lenght(SEtmp.Company, 24);

SEtmp.WE = cinum("input work expirience(in days):", "Wrong input, try again", 1, 1, 0);

cin.get();

std::cout << "input operating system:" << std::endl;

val\_char\_lenght(SEtmp.software.OS, 12);

std::cout << "input programming language:" << std::endl;

val\_char\_lenght(SEtmp.software.CodeLang, 8);

std::cout << "input database:" << std::endl;

val\_char\_lenght(SEtmp.software.DB, 8);

addSE(SEdatabase, SEtmp);//tmp

break;

}

case 2: {

std::system("cls");

printSElist(SEdatabase);

break;

}

case 3: {

std::system("cls");

int chosenWE;//tmp

chosenWE = cinum("input what work expirience you are searching for:", "Wrong input, try again", 1, 1, 0);

std::vector<int> indexes;

searchByWE(indexes, SEdatabase, chosenWE);

printByIndexes(indexes, SEdatabase);

break;

}

case 4: {

std::system("cls");

std::vector<int> indexes;

highestWE(indexes, SEdatabase);

printByIndexes(indexes, SEdatabase);

break;

}

case 5: {

std::system("cls");

char chosenName[24];

std::cin.get();

std::cout << "input what name you are searching for:" << std::endl;

val\_char\_lenght(chosenName, 24);

std::vector<int> indexes;

searchByName(indexes, SEdatabase, chosenName);

printByIndexes(indexes, SEdatabase);

break;

}

case 6: {

std::system("cls");

sortByWE(SEdatabase);

printSElist(SEdatabase);

break;

}

case 7: {

break;

}

default: {

std::cout << "Wrong input, try again" << std::endl;

break;

}

}

} while (choice != 7);

char chExit;

std::cout << "Do you want to exit the programm?(Y/N): ";

std::cin >> chExit;

while ((chExit != 'Y' && chExit != 'y' && chExit != 'N' && chExit != 'n') || (std::cin.peek() != '\n')) {//until correct input

while (std::cin.get() != '\n');//clear istream

std::cout << "wrong input, try again." << std::endl;

std::cout << "Do you want to exit the programm?(Y/N): ";

std::cin >> chExit;

}

if (chExit == 'Y' || chExit == 'y')

exit = true;

else

exit = false;

std::cin.get();//enter

} while (!exit);

std::system("pause");

return 0;

}

“SE\_DB.h”

#ifndef \_\_SE\_DB\_H\_\_

#define \_\_SE\_DB\_H\_\_

#define SE\_size sizeof(SE\_)

#include <iomanip>

#include <iostream>

#include <fstream>

#include <random>//

#include <chrono>//

#include <vector>

struct skills {

char OS[12];

char CodeLang[8];

char DB[8];

};

struct SE\_ {//SE - SoftwareEngineer

char Name[24];

char Company[24];

int WE;//WE - WorkExpirience

skills software;

};

void addSE(std::fstream& SEdatabase, SE\_& SEtmp);

void printSElist(std::fstream& SEdatabase);

//saves indexes of SEs with chosen WE

void searchByWE(std::vector<int>& indexes, std::fstream& SEdatabase, int chosenWE);

//saves indexes of SEs with highest WE to indexes

void highestWE(std::vector<int>& indexes, std::fstream& SEdatabase);

//saves indexes of SEs with chosen name to indexes

void searchByName(std::vector<int>& indexes, std::fstream& SEdatabase, char chosenName[24]);

void sortByWE(std::fstream& SEdatabase);

void printByIndexes(std::vector<int>& indexes, std::fstream& SEdatabase);

void printHeading();//

void printEnding();//

void printSE(SE\_& SE);

//fills file with some SEs(random)

void SE\_DBfill(std::fstream& tmp);

#endif

“SE\_DB.cpp”

#include "SE\_DB.h"

void addSE(std::fstream& SEdatabase, SE\_& SEtmp) {

SEdatabase.seekp(0, std::ios\_base::end);

SEdatabase.write(reinterpret\_cast<char\*>(&SEtmp), SE\_size);

}

void printSElist(std::fstream& SEdatabase) {

SEdatabase.seekg(0, std::ios\_base::beg);

SE\_ SEtmp;

printHeading();

SEdatabase.read(reinterpret\_cast<char\*>(&SEtmp), SE\_size);

while (!SEdatabase.eof()) {

printSE(SEtmp);

SEdatabase.read(reinterpret\_cast<char\*>(&SEtmp), SE\_size);

}

printEnding();

SEdatabase.clear();//clear eof

}

void searchByWE(std::vector<int>& indexes, std::fstream& SEdatabase, int chosenWE) {

SEdatabase.seekg(0, std::ios\_base::end);

int length = SEdatabase.tellg() / SE\_size;

int currentWE;

SEdatabase.seekg(0, std::ios\_base::beg);

for (int i = 0; i < length; i++) {//save indexes with chosen WE

SEdatabase.seekg(48, std::ios\_base::cur);

SEdatabase.read(reinterpret\_cast<char\*>(&currentWE), sizeof(int));

SEdatabase.seekg(28, std::ios\_base::cur);//read only WE

if (currentWE == chosenWE)

indexes.push\_back(i);

}

}

void highestWE(std::vector<int>& indexes, std::fstream& SEdatabase) {

SEdatabase.seekg(0, std::ios\_base::end);

int length = SEdatabase.tellg() / SE\_size;

int currentWE, highestWE = 0;

SEdatabase.seekg(0, std::ios\_base::beg);

for (int i = 0; i < length; i++) {//find highest

SEdatabase.seekg(48, std::ios\_base::cur);

SEdatabase.read(reinterpret\_cast<char\*>(&currentWE), sizeof(int));

SEdatabase.seekg(28, std::ios\_base::cur);//read only WE

if (currentWE > highestWE)

highestWE = currentWE;

}

searchByWE(indexes, SEdatabase, highestWE);

}

void searchByName(std::vector<int>& indexes, std::fstream& SEdatabase, char chosenName[24]) {

SEdatabase.seekg(0, std::ios\_base::end);

int length = SEdatabase.tellg() / SE\_size;

SEdatabase.seekg(0, std::ios\_base::beg);

char currentName[24];

for (int i = 0; i < length; i++) {//save indexes with chosen name

SEdatabase.read(reinterpret\_cast<char\*>(&currentName), 24);

SEdatabase.seekg(56, std::ios\_base::cur);//read only WE

if (!strcmp(currentName, chosenName))

indexes.push\_back(i);

}

}

void printByIndexes(std::vector<int>& indexes, std::fstream& SEdatabase) {

printHeading();

SE\_ currentSE;

for (auto index : indexes) {

SEdatabase.seekg(index \* SE\_size, std::ios\_base::beg);

SEdatabase.read(reinterpret\_cast<char\*>(&currentSE), SE\_size);

printSE(currentSE);

}

printEnding();

}

void sortByWE(std::fstream& SEdatabase) {//insertionSort

SEdatabase.seekg(0, std::ios\_base::end);

int length = SEdatabase.tellg() / SE\_size;

SE\_ SEtmp[2];

for (int i = 1; i < length; i++) {

SEdatabase.seekg((i - 1) \* SE\_size, std::ios\_base::beg);

SEdatabase.read(reinterpret\_cast<char\*>(&SEtmp[0]), SE\_size);//arrayToSort[i - 1]

SEdatabase.read(reinterpret\_cast<char\*>(&SEtmp[1]), SE\_size);//arrayToSort[i]

if (SEtmp[1].WE < SEtmp[0].WE) {//if !inplace

int j = i - 1;

for (j; (j >= 0) && (SEtmp[1].WE < SEtmp[0].WE); j--) {//arrayToSort[j + 1] = arrayToSort[j];//find position and shift to right elements between new and old position

SEdatabase.seekp((j + 1) \* SE\_size, std::ios\_base::beg);

SEdatabase.write(reinterpret\_cast<char\*>(&SEtmp[0]), SE\_size);

if (j > 0) {

SEdatabase.seekg((j - 1) \* SE\_size, std::ios\_base::beg);//read j-th for next iteration

SEdatabase.read(reinterpret\_cast<char\*>(&SEtmp[0]), SE\_size);

}

}

SEdatabase.seekp((j + 1) \* SE\_size, std::ios\_base::beg);//arrayToSort[j + 1] = temp;

SEdatabase.write(reinterpret\_cast<char\*>(&SEtmp[1]), SE\_size);

}

}

}

void printHeading() {//

std::cout << std::left;

std::cout << std::setw(24) << "Name" << " | ";

std::cout << std::setw(24) << "Company" << " | ";

std::cout << std::setw(20) << "Work Expirience" << " | ";

std::cout << std::setw(12) << "OS" << " | ";

std::cout << std::setw(8) << "CodeLang" << " | ";

std::cout << std::setw(8) << "DataBase" << " ] ";

std::cout << std::endl;

std::cout << std::setfill('-') << std::right;

std::cout << std::setw(26) << "|";

std::cout << std::setw(27) << "|";

std::cout << std::setw(23) << "|";

std::cout << std::setw(15) << "|";

std::cout << std::setw(11) << "|";

std::cout << std::setw(11) << "]";

std::cout << std::endl;

std::cout << std::setfill(' ') << std::left;

}

void printEnding() {//

std::cout << std::setfill('-') << std::right;

std::cout << std::setw(113) << "]" << std::endl;

std::cout << std::setfill(' ') << std::left;

}

void printSE(SE\_& SE) {//

std::cout << std::setw(24) << SE.Name << " | ";

std::cout << std::setw(24) << SE.Company << " | ";

std::cout << std::setw(20) << SE.WE << " | ";

std::cout << std::setw(12) << SE.software.OS << " | ";

std::cout << std::setw(8) << SE.software.CodeLang << " | ";

std::cout << std::setw(8) << SE.software.DB << " ] ";

std::cout << std::endl;

}

void SE\_DBfill(std::fstream& tmp) {

std::uniform\_int\_distribution<int> valueDist(0, 2);

std::default\_random\_engine gen(time(NULL));

SE\_ SEblck[3]{ {"George Washington", "SpaceX", 521, {"Kali linux", "C++", "SQL"}},

{"Joseph Joestar", "Atari", 176, {"Arch Linux", "Java", "MongoDB"} },

{"John Watson", "IBM", 938, {"Windows 10", "Ruby", "Oracle"} } };

for (int i = 0; i < 5; i++)

tmp.write(reinterpret\_cast<char\*>(&SEblck[valueDist(gen)]), SE\_size);

}

“inp\_val7.h”

//cinum - numerical only (R)

//char msg\_invite - your message for inviting user to input their number

//char msg\_error - if error input occures

//int m\_ - check if input is multiple of this number

//bool intFlag - =1(only integer input); =0(any input)

//double left - leftmost point in diapasone for input

//double right - rightmost point in diapasone for input

#include <iostream>

#include <windows.h>

#undef max

#include <cmath>

#include <climits>

using namespace std;

int cinum(const char\* msg\_invite, const char\* msg\_error, int m\_ = 1, bool intFlag = 0, float left = -2147483648.0, float right = 2147483647.0) {

bool i = 0;

float aaa;

cout << msg\_invite << endl;

do {

cin >> aaa;

int iaaa;

iaaa = (int)aaa;

if (cin.fail() || (aaa < left) || (aaa > right) || ((intFlag == 1) && (ceil(aaa) != floor(aaa))) || ((m\_ != 1) && ((iaaa % m\_) != 0))) {

cout << msg\_error << endl;

cin.clear();

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

}

else

i = 1;

} while (!i);

return (int)aaa;

}

void val\_char\_lenght(char array[], int ary\_size) {//input to char array with lenght validation

bool correct = false;

do {

std::cin.getline(array, 24);

if ((strlen(array) > 23 ) || cin.fail()) {

std::cout << "wrong input, try again" << std::endl;

cin.clear();

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

}

else

correct = true;

} while (!correct);

}

# **7. Результати тестування програми та їх аналіз.**

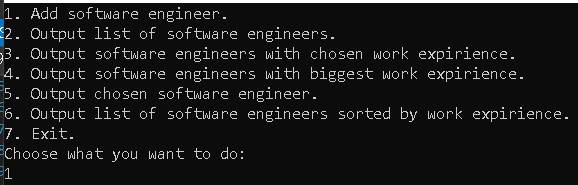


Рисунок 3



Рисунок 4

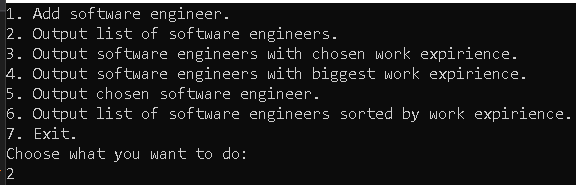


Рисунок 5

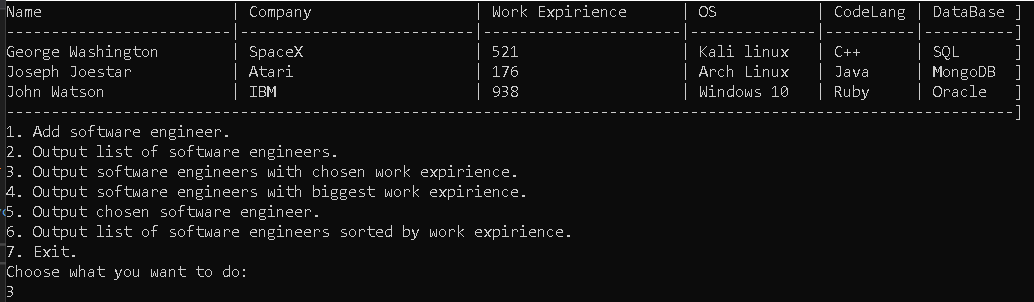


Рисунок 6

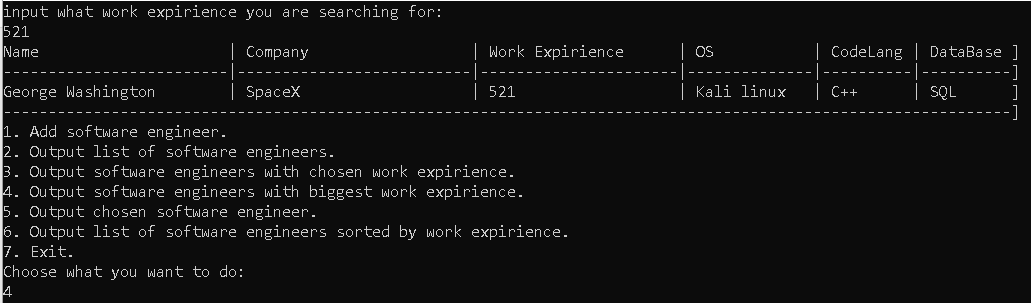


Рисунок 7

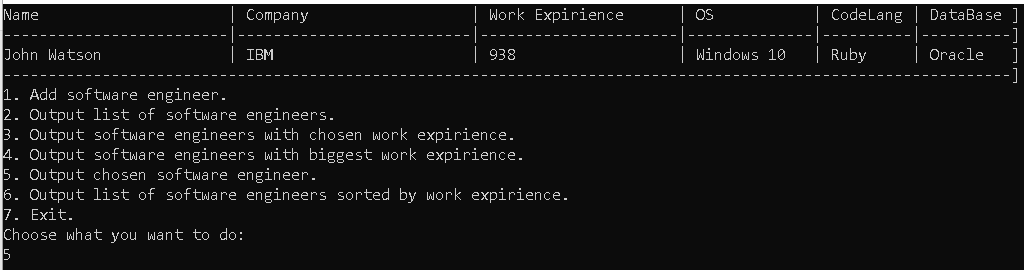


Рисунок 8

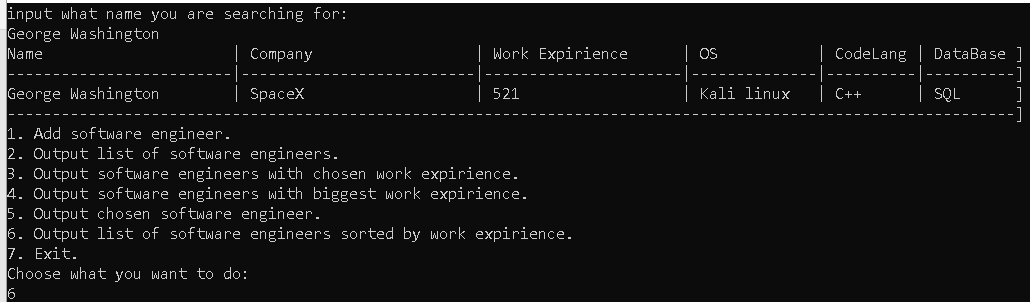


Рисунок 9

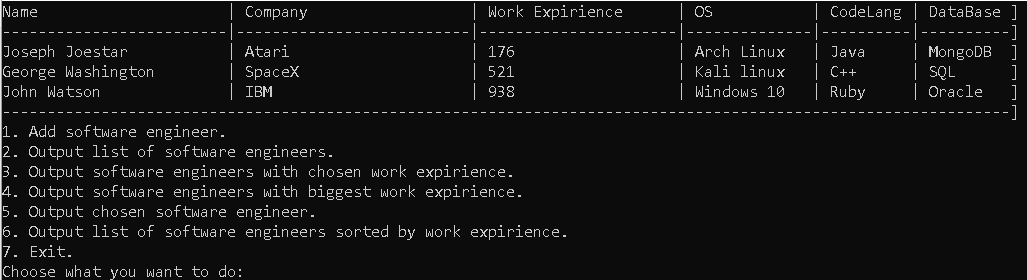


Рисунок 10

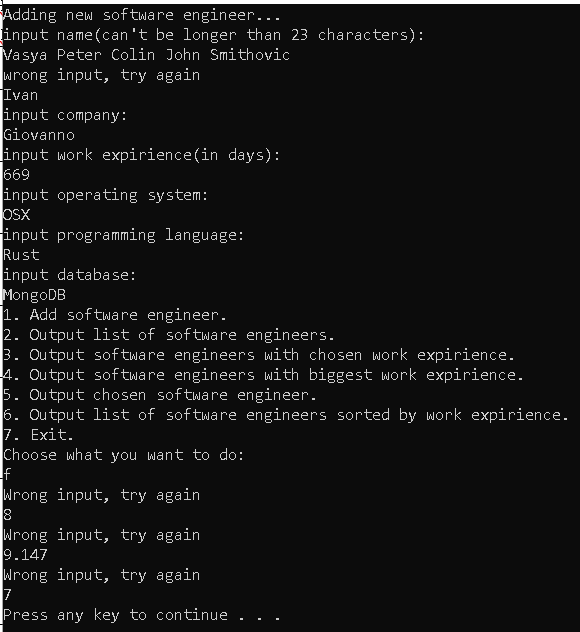


Рисунок 11

# **8. Висновки щодо роботи з бінарними файлами і реалізації алгоритмів пошуку/сортування даних у файлах.**

Робота з бінарними файлами ще більше схожа на роботу із масивами ніж з текстовими. Можна сказати, що бінарний файл – динамічний масив байтів, розташований на носії. Файли потрібні для зберігання довгострокової інформації, яка може знадобитися при наступних використаннях програми.

Бібліотека <fstream> дозволяє відкрити файл в наступних режимах(вони можуть комбінуватися):

In – Для читання.

Out – Для запису.

Binary – Як бінарний.

Ate – Вказівник для запису встановлений на кінець.

App – Усі операції запису проводяться в кінець файлу.

Trunc – Початковий розмір файлу 0.

Ifstream відкривається з модіфікатором in, якщо не написати інших.

Ofstream з out, а fstream з обома.

Також на відміну від <stdio>, <fstream> має роздільні і незалежні вказівники і буфери для читання та запису.