МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



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

Кафедра «Технічна Кібернетика»

**Лабораторна робота №5**

**з дисципліни «Основи програмування»**

**на тему: «**Структурний тип даних в мові С++**»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Нежуміра О.И.

Дніпро, 2019

**Тема.** Структурний тип даних в мові С++.

**Мета.** Засвоїти поняття складеного типу даних. Отримати практичні навички роботи з інтегрованими типами даних – структурами і масивами структур – мови С++.

# **3) Постановка завдання (завдання до лабораторної роботи, індивідуальне завдання за обраним варіантом і рівнем, вимоги до програми).**

Розробити програму для роботи з даними для обраних предметної області і рівня складності. Кожен рівень передбачає виконання вказаного завдання та завдань усіх попередніх рівнів.

**Вимоги до програми:**

- всі вхідні дані вводяться з клавіатури;

- передбачити перевірку вхідних даних на відповідність діапазону значень і некоректні символи;

- введені дані зберігаються в масиві структур (масив складається з 10 елементів);

- символьні поля структури оголошуються як масив символів;

- інформація, яка зберігається у символьному полі структури, може складатися з декількох слів;

- три перши елементи масиву необхідно ініціалізувати при оголошенні масиву;

- управління виконанням програми здійснюється на основі текстового меню користувача;

- результати роботи програми виводяться на екран.

**Вимоги до тексту програми:**

- коментарі щодо призначення програми, її вхідних і вихідних даних;

- коментарі щодо призначення кожного блоку програми, дій окремих операторів для пояснення алгоритму;

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

Таблиця 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № 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 |

# **4) зовнішні специфікації програми.**

*Формат вхідних даних:*

Таблиця 2

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

# **5) метод рішення завдання.**

1. Вивести меню, в залежності від вхідних даних перейти до конкретної функції.

2.1. Додати программіста(заповнення єлементів структури програміста по черзі).

2.2. Вивести список програмістів(масив структур програмістів).

2.3. Вивести список програмістів з обраним стажем роботи(перевірити кожну структуру на віподвідність).

2.4. Вивести список програмістів з найбільшим стажем роботи(відсортувати масив, виводити усіх починаючи з найбільшого стажу, до першого у, якого стаж не дорівнює більшому).

2.5. Вивести список програмістів з вказаним іменем(перевірити кожну структуру на відповідність).

2.6. Вивести відсортований список.

2.7. Завершити виконання програми.

Алгоритм сортуваня та інші вказані у відповідній главі.

# **6) алгоритм розв’язання задачі у вигляді діаграм Н-Ш (алгоритм для реалізації завдання в цілому (меню користувача, укрупнені блоки для всіх функцій програми), окремі алгоритми введення даних і виконання задач обробки масиву структур).**

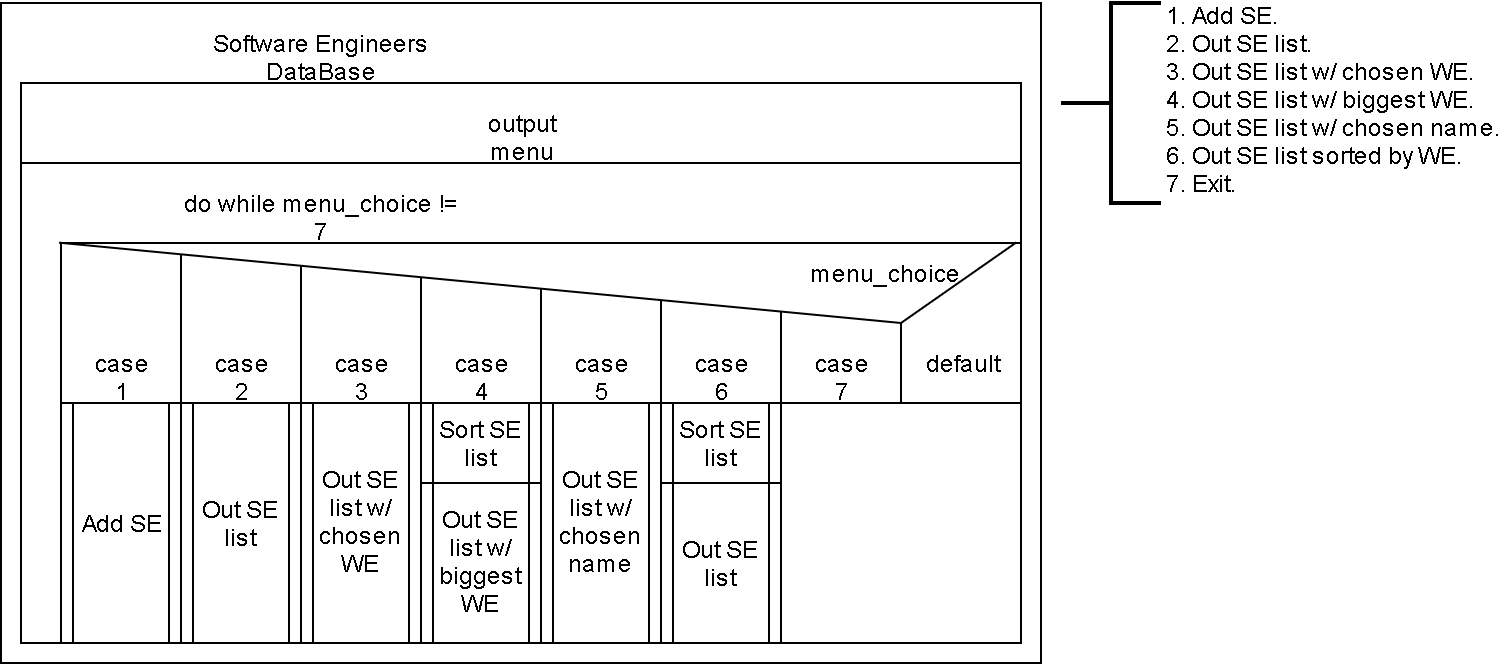


Рисунок 1

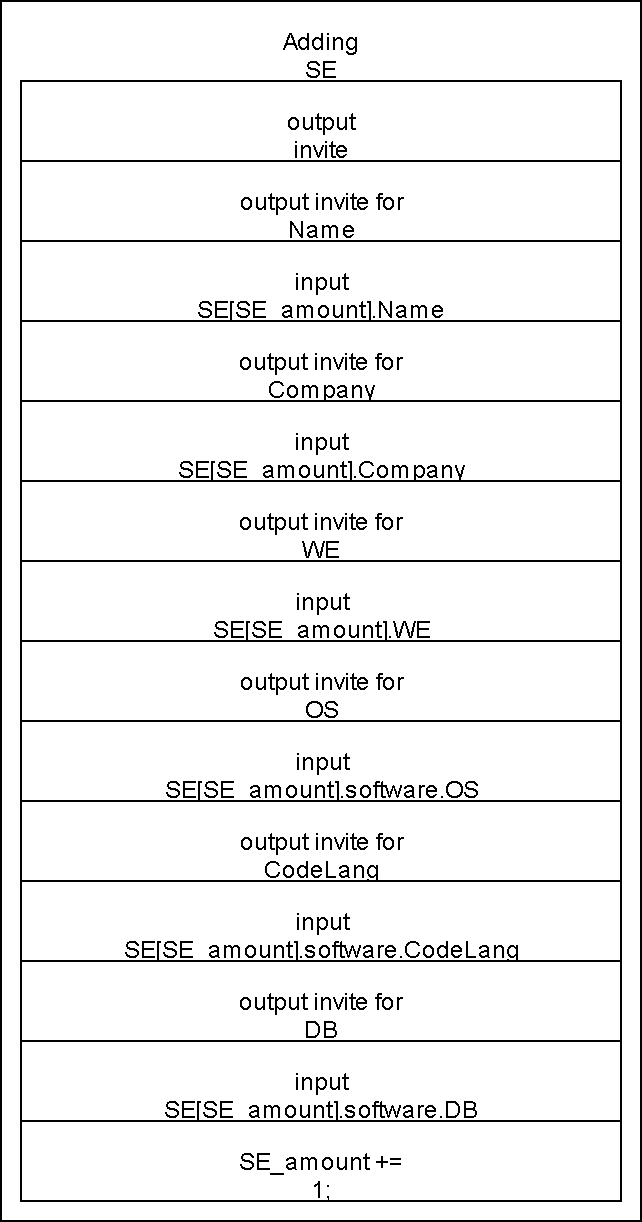


Рисунок 2

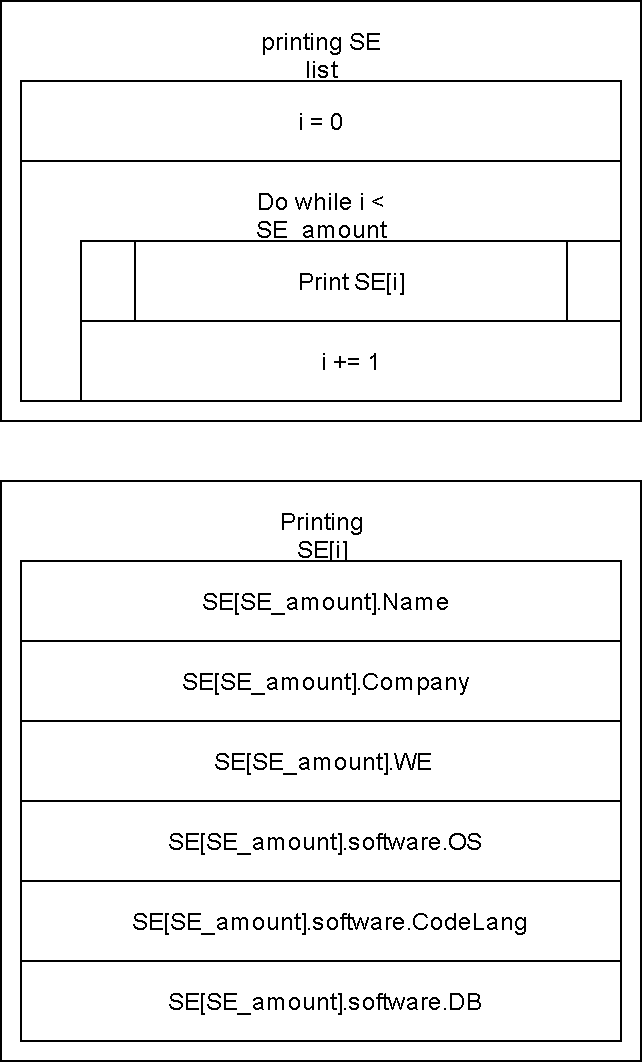


Рисунок 3

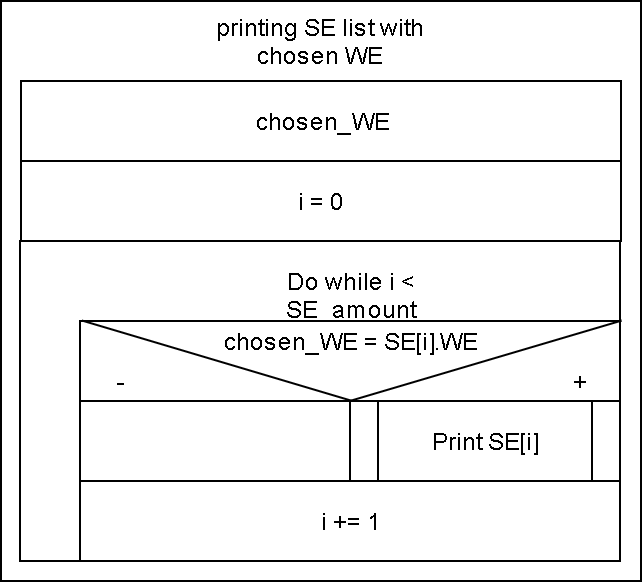


Рисунок 4

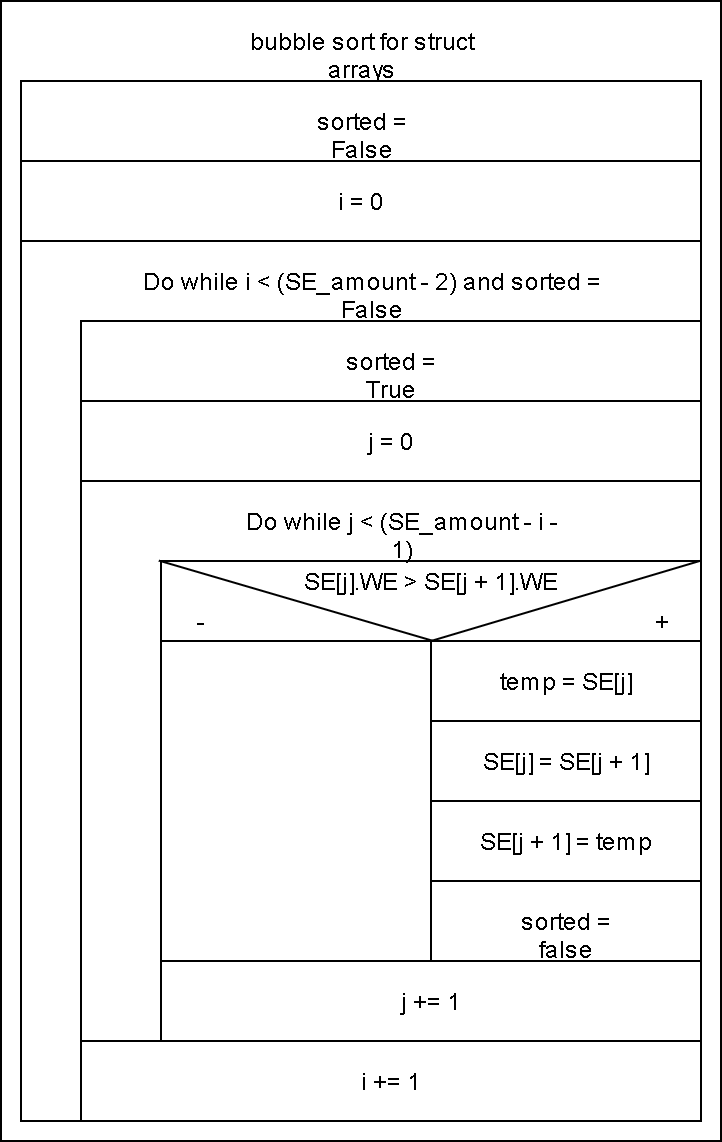


Рисунок 5

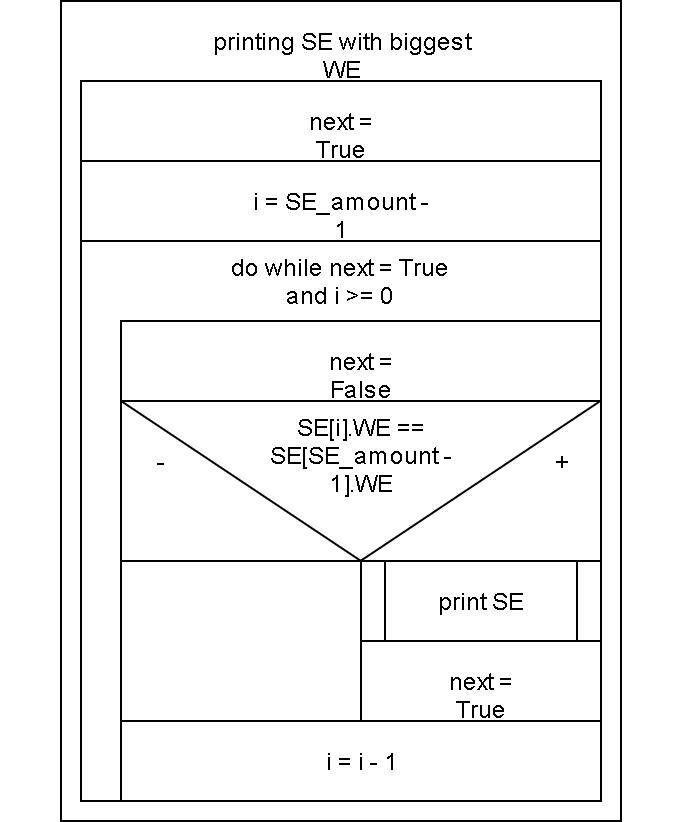


Рисунок 6

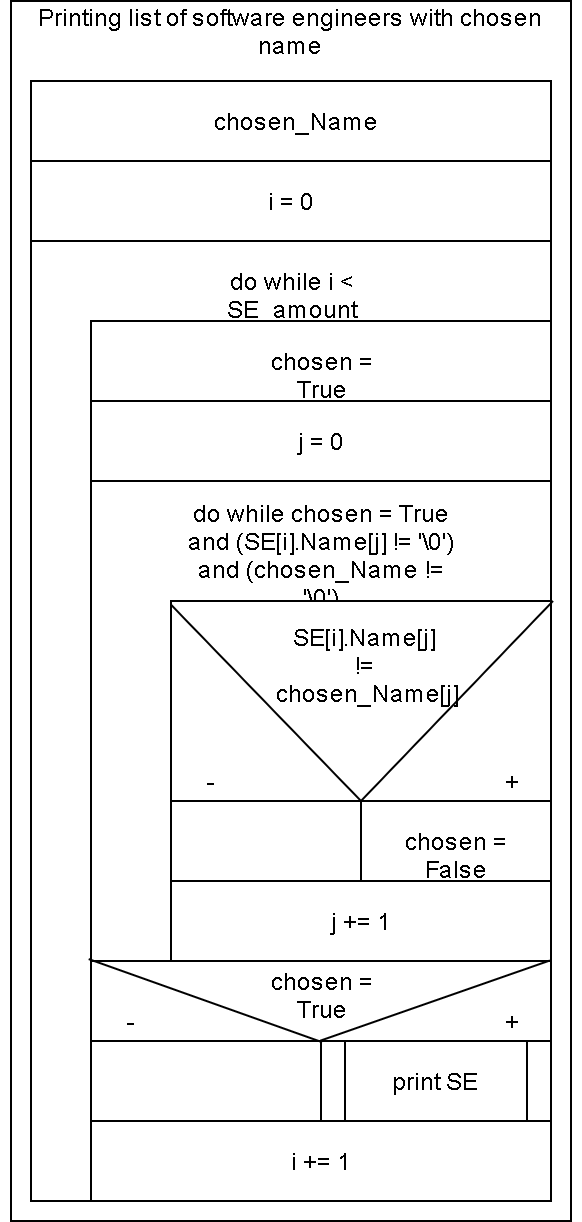


Рисунок 7

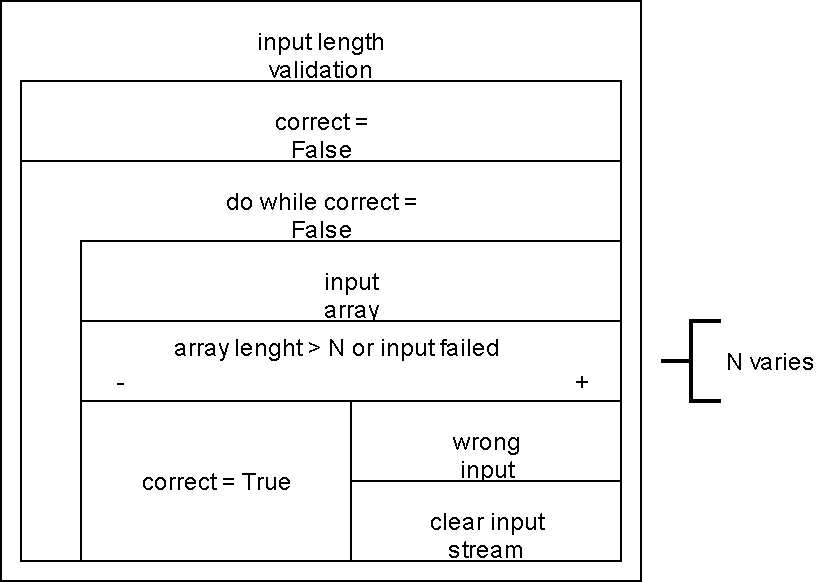


Рисунок 8

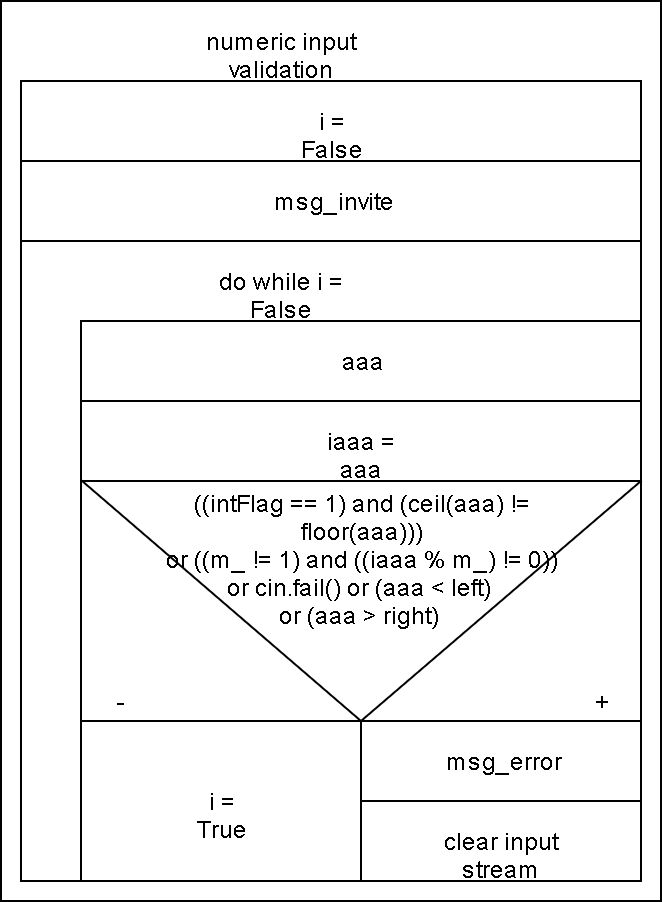


Рисунок 9

# **7) набір тестів для перевірки вводу даних (пункт меню користувача, поля структури).**

Таблиця 4

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікуваний результат |
| 1 | Меню Пункт діапазон 1-6 | menu\_choice = 1 | Перехід до відповідного блоку програми |
| 7 | Меню Пункт 7 | menu\_choice = 7 | Вихід з програми |
| 8 | Меню Символ | menu\_choice = f | Повідомленя про помилку, поновлення вводу |
| 9 | Меню Дріб | menu\_choice = 9.147 | Повідомленя про помилку, поновлення вводу |
| 10 | Меню не належить діапазону | menu\_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. |

# **8) набір тестів для перевірки роботи меню користувача.**

Таблиця 5

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

# **9) 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"}} |

# **10) текст програми.**

//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 <iomanip>

#include "inp\_val6.h"

#include <Windows.h>

#include <string>

#include <limits>

#include "SE\_list\_manip.h"

#undef max

int main()

{

const int size = 10;

SE\_ SE[size]{ //declared in SE\_list\_manip.h

{"George Washington", "SpaceX", 521, {"Kali linux", "C++", "SQL"}},

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

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

};

int SE\_amount = 3;

int menu\_choice;

do

{

#pragma region output menu

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;

#pragma endregion

char msg\_invite[] = "Choose what you want to do:";

char msg\_error[] = "Wrong input, try again";

menu\_choice = cinum(msg\_invite, msg\_error, 1, 1, 1, 7);

switch (menu\_choice)

{

case 1:

{

system("cls");

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(SE[SE\_amount].Name, 24);

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

val\_char\_lenght(SE[SE\_amount].Company, 24);

char msg\_invite2[] = "input work expirience(in days):";

SE[SE\_amount].WE = cinum(msg\_invite2, msg\_error, 1, 1, 0);

cin.get();

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

val\_char\_lenght(SE[SE\_amount].software.OS, 12);

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

val\_char\_lenght(SE[SE\_amount].software.CodeLang, 8);

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

val\_char\_lenght(SE[SE\_amount].software.DB, 8);

SE\_amount++;

break;

}

case 2:

{

std::system("cls");

print\_heading();

for (int i = 0; i < SE\_amount; i++) {

print\_SE(SE, i);

}

print\_ending();

break;

}

case 3:

{

std::system("cls");

int chosen\_WE;

char msg\_invite1[] = "input what work expirience you are searching for:";

chosen\_WE = cinum(msg\_invite1, msg\_error, 1, 1, 0);

print\_heading();

for (int i = 0; i < SE\_amount; i++) {

if (SE[i].WE == chosen\_WE) {//checking if current Work Expirience == chosen\_WE

print\_SE(SE, i);

}

}

print\_ending();

break;

}

case 4:

{

SE\_list\_sort(SE, SE\_amount);

std::system("cls");

print\_heading();

bool next = true;

for (int i = SE\_amount - 1;next && i >= 0; i--) {//reading from big to small

next = false;

if (SE[i].WE == SE[SE\_amount - 1].WE) {

print\_SE(SE, i);

next = true;// if current != biggest than it's smaller, that means all next elements are smaller, so we can stop on the current one

}

}

print\_ending();

break;

}

case 5:

{

std::system("cls");

char chosen\_Name[24];

std::cin.get();

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

val\_char\_lenght(chosen\_Name, 24);

print\_heading();

for (int i = 0; i < SE\_amount; i++) {

bool chosen = true;

for (int j = 0;chosen && (SE[i].Name[j] != '\0') && (chosen\_Name[j] != '\0'); j++) {

if (SE[i].Name[j] != chosen\_Name[j]) {//cheking name one char by the tame if chars are different going to another name

chosen = false;

}

}

if (chosen) {

print\_SE(SE, i);

}

}

print\_ending();

break;

}

case 6:

{

SE\_list\_sort(SE, SE\_amount);

std::system("cls");

print\_heading();

for (int i = 0; i < SE\_amount; i++) {

print\_SE(SE, i);

}

print\_ending();

break;

}

case 7:

{

break;

}

default:

{

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

}

}

} while (menu\_choice != 7);

std::system("pause");

return 0;

}

# **10.1)Inp\_val6.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(char\* msg\_invite, 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);

}

# **10.2)SE\_List\_manip.h**

//LR5 custom header to make actual programm code a bit smaller

void print\_heading () {

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 print\_ending() {

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

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

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

}

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 print\_SE(struct SE\_ SE[], int i) {

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

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

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

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

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

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

std::cout << std::endl;

}

void SE\_list\_sort(struct SE\_ SE[], int SE\_amount) {//bubble sort

bool sorted = false;

for (int i = 0; !sorted && (i < (SE\_amount - 2)); i++) {

bool sorted = true;

for (int j = 0; j < (SE\_amount - i - 1); j++) {

if (SE[j].WE > SE[j + 1].WE) {

SE\_ temp;

temp = SE[j];

SE[j] = SE[j + 1];

SE[j + 1] = temp;

sorted = false;

}

}

}

}

# **11) результати тестування програми та їх аналіз.**

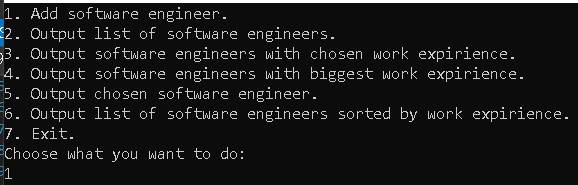


Рисунок 10

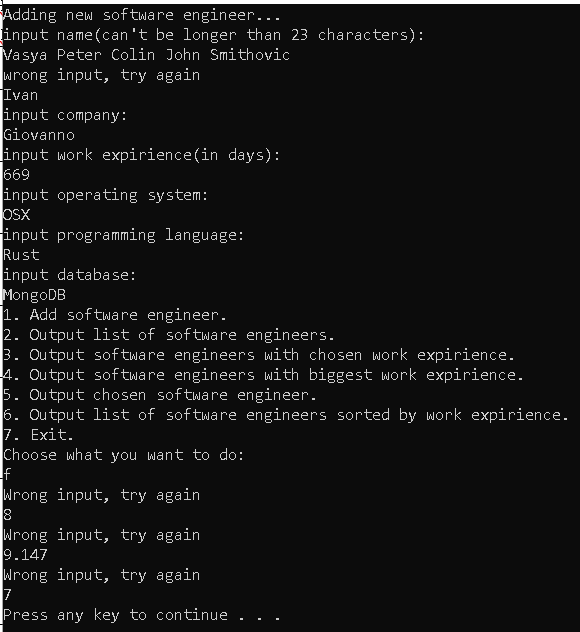


Рисунок 11

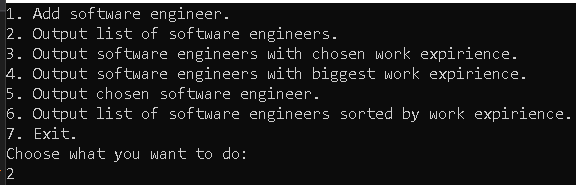


Рисунок 12

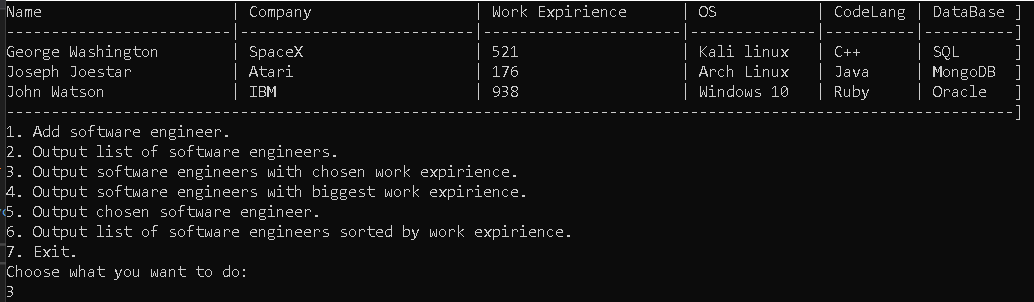


Рисунок 13

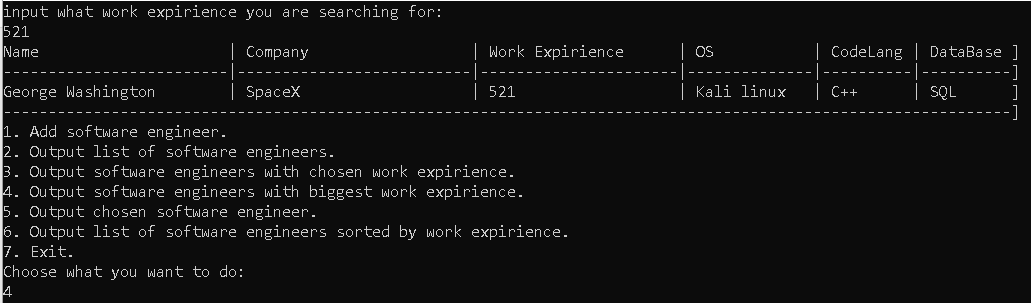


Рисунок 14

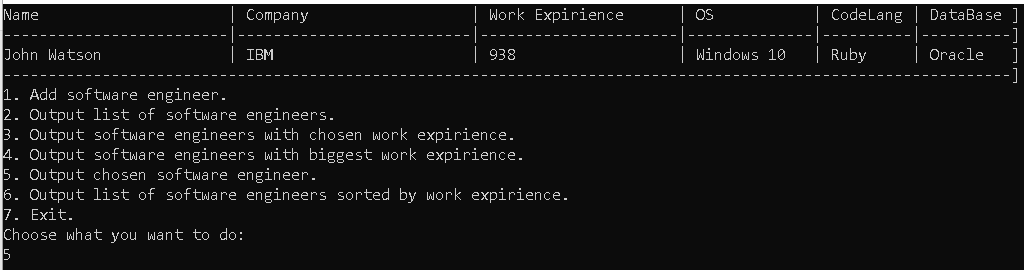


Рисунок 15

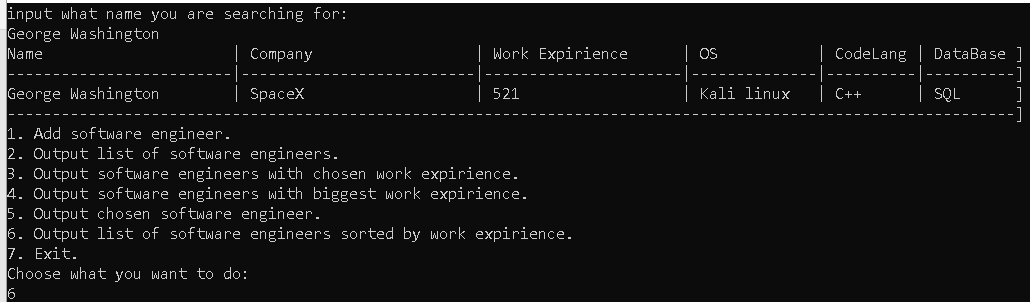


Рисунок 16

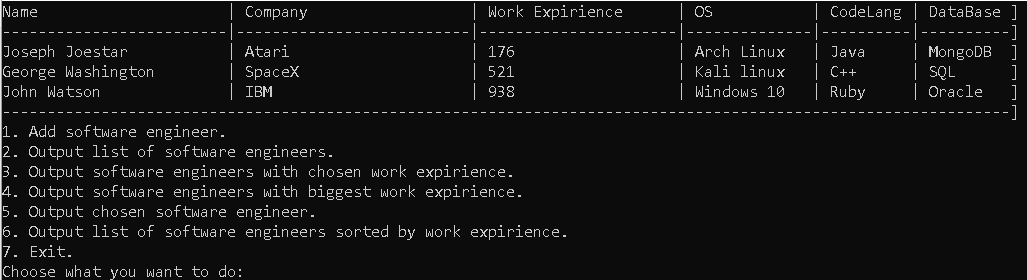


Рисунок 17

# **12) висновки щодо розробки програм з використанням структур і масивів.**

Структура – складений тип даних, шаблон якого задається програмістом. Структури дуже схожі з масивами, за тим винятком, що в них елементи можуть мати різні типи даних, також елементами можуть бути масиви та структури.

Іниціалізуються структури так само, як і масив, за виключенням того, що елементи різні.

Структури потрібні для того, щоб об’єднувати різні за типами та розмірами дані, які мають щось загальне, наприклад опис якогось об’єкта – назва(масив символів), розміри(цілі числа). Авжеж можна задати змінні з іменами, які б казали про належність до одного об’єкту. Але коли з’явиться потреба наприклад відсортувати декілька об’єктів одного типу за одним елементом відповідної структури, набагато простіше копіювати усю структуру за раз, а ніж кожен єлемент.

Масив символів стане в нагоді, коли з’явиться потреба маніпулювати структурами одного типу, наприклад вивести усі елементи для кожної структури з масиву, тобто вивести списки відповідних елементів.