МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



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

Кафедра «Технічна Кібернетика»

**Лабораторна робота №6**

**з дисципліни «Основи програмування»**

**на тему: «**Динамічні [масиви](http://lider.diit.edu.ua/mod/lesson/view.php?id=40269) в мові С++**»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Нежуміра О.И.

Дніпро, 2019

**Тема.** Функції. Методи передачі параметрів. Багатофайлові програми.

**Мета.**

1) Вивчити організацію функцій і механізм передачі параметрів. Отримати практичні навички розробки та використання функцій.

2) Вивчити призначення і організацію багатофайлових програми. Отримати практичні навички розробки багатофайлових програм.

# **Постановка завдання №1.**

Розробити програму для демонстрації механізму передачі параметрів. Для вказаних способів передачі параметрів і типів параметрів написати функції, в яких параметри змінюють своє значення (можна використати арифметичні операції, присвоювання тощо). Функції повинні повертати значення заданого типу. Для виклику кожної функції передбачити виведення на консоль:

1) назви способу передачі параметрів у функцію;

2) значень параметрів, що передаються у функцію, до і після її виклику;

3) значення, яке повернула функція.



Рисунок 1

# **Текст програми завдання №1.**

“main.cpp”

//Application demonstartes ways of argument passing

//function prototypes are in funcs.h file

//function definitions are in funcs.cpp file

#include "funcs.h"

#include <iostream>

int main()

{

//

char ms1[] = "MS1";

myStruct MS1{ 1, 'a', true, ms1 };//, new char[3]};

MS1.print();

MS1.Int = func\_int\_char\_bool::int\_value(MS1.Int, MS1.Char, MS1.Bool);

MS1.print();

std::cout << std::endl;

MS1.print();

char char\_example1 = func\_int\_char\_bool::char\_reference(MS1.Int, MS1.Char, MS1.Bool);//self name is a reference too

std::cout << "char example = " << char\_example1 << std::endl;

MS1.print();

std::cout << std::endl;

MS1.print();

func\_int\_char\_bool::void\_pointer(&(MS1.Int), &(MS1.Char), &(MS1.Bool));

MS1.print();

std::cout << std::endl;

MS1.print();

int\* example\_intp = func\_int\_char\_bool::intp\_value(MS1.Int, MS1.Char, MS1.Bool);

std::cout << "pointer to integer example = " << example\_intp << std::endl;

MS1.print();

std::cout << std::endl;

MS1.print();

char\* example\_charp = func\_int\_char\_bool::charp\_reference(MS1.Int, MS1.Char, MS1.Bool);//self name is a reference too

std::cout << "pointer to character example = " << static\_cast<void\*>(example\_charp) << std::endl;//to print char\* it has to be casted to other type\*

MS1.print();

std::cout << std::endl;

int Iarray[5]{3, 5, 2, 1, 4};

func\_srv::print\_int\_ar(Iarray, (sizeof(Iarray) / sizeof(\*Iarray)));

func\_num\_ar::void\_reference(Iarray);

func\_srv::print\_int\_ar(Iarray, (sizeof(Iarray) / sizeof(\*Iarray)));

std::cout << std::endl;

func\_srv::print\_int\_ar(Iarray, (sizeof(Iarray) / sizeof(\*Iarray)));

func\_num\_ar::void\_pointer(Iarray, (sizeof(Iarray) / sizeof(\*Iarray)));

func\_srv::print\_int\_ar(Iarray, (sizeof(Iarray) / sizeof(\*Iarray)));

std::cout << std::endl;

MS1.print();

myStruct MS2 = func\_struct::MS\_value(MS1);

char ms2[] = "MS2";

MS2.id = ms2;

//std::cout << std::endl;

MS2.print();

//std::cout << std::endl;

MS1.print();

std::cout << std::endl;

MS1.print();

MS2 = \*(func\_struct::MSp\_reference(MS1));

MS2.id = ms2;

MS2.print();

std::cout << std::endl;

MS1.print();

MS2 = func\_struct::MSr\_pointer(&MS1);

MS2.id = ms2;

MS2.print();

std::cout << std::endl;

char char\_array[]{'e', 'a', 'd', 'b', 'c'};

func\_srv::print\_char\_ar(char\_array, (sizeof(char\_array) / sizeof(\*char\_array)));

func\_char\_ar::void\_reference(char\_array);

func\_srv::print\_char\_ar(char\_array, (sizeof(char\_array) / sizeof(\*char\_array)));

std::cout << std::endl;

char char\_array2[]{'e', 'a', 'd', 'b', 'c'};

func\_srv::print\_char\_ar(char\_array2, (sizeof(char\_array2) / sizeof(\*char\_array2)));

func\_char\_ar::void\_pointer(char\_array2, (sizeof(char\_array2) / sizeof(\*char\_array2)));

func\_srv::print\_char\_ar(char\_array2, (sizeof(char\_array2) / sizeof(\*char\_array2)));

system("pause");

return 0;

}

“funcs.h”

//each class represents set of function with same arguments(name of class), but these functions vary way of passing argument to them and return type(both in the name of function)

struct myStruct {//for passing to func

int Int;

char Char;

bool Bool;

char\* id;

void print();

//add char array

//add int array

};

class func\_int\_char\_bool {

public:

static int int\_value(int Int, char Char, bool Bool);//arguments passed by value, returns int, defined in funcs\_r.cpp

static char char\_reference(int& rInt, char& rChar, bool& rBool);//arguments passed by reference, returns char, defined in funcs\_r.cpp

static void void\_pointer(int\* pInt, char\* pChar, bool\* pBool);//arguments passed by pointer, returns nothing, defined in funcs\_r.cpp

//E

static int\* intp\_value(int Int, char Char, bool Bool);//arguments passed by value, returns int\*, defined in funcs\_r.cpp

static char\* charp\_reference(int& rInt, char& rChar, bool& rBool);//arguments passed by reference, returns char\*, defined in funcs\_r.cpp

//D

};

class func\_num\_ar {

public:

//array passed by value, defined in funcs\_r.cpp

//todo: explain in report why array cant be passed by value!

static void void\_reference(int int\_array[]);//array passed by reference, defined in funcs\_r.cpp

static void void\_pointer(int\* int\_array0, int size);//array passed by pointer of first element, defined in funcs\_r.cpp

//C

};

class func\_struct {

public:

static myStruct MS\_value(myStruct myStruct1);//struct passed by value

static myStruct\* MSp\_reference(myStruct& myStruct1);//struct passed by reference, returns pointer

static myStruct& MSr\_pointer(myStruct\* myStruct);//struct passed by pointer, returns refernce

//B

};

class func\_char\_ar {

public:

//array passed by value, defined in funcs\_r.cpp

//todo: explain in report why array cant be passed by value!

static void void\_reference(char char\_array[]);//array passed by reference, defined in funcs\_r.cpp

static void void\_pointer(char\* char\_array0, int size);//array passed by point, defined in funcs\_r.cpp

//A

};

class func\_srv {

public:

static void int\_ar\_bubble\_sort(int\* pArray0, int size);//to sort arrays

static void print\_char\_ar(char\* pArray0, int size);

static void ch\_ar\_bubble\_sort(char\* pArray0, int size);

static void print\_int\_ar(int\* pArray0, int size);

};

“funcs.cpp”

//specs

#include <iostream>

#include <iomanip>

#include "funcs.h"

int func\_int\_char\_bool::int\_value(int Int, char Char, bool Bool) {

std::cout << " arguments are passed by value" << std::endl;

int temp = Int;

temp \*= 2;

std::cout << " returned int: " << temp << std::endl;

return temp;

}

char func\_int\_char\_bool::char\_reference(int& Int, char& Char, bool& Bool) {

std::cout << " arguments are passed by reference" << std::endl;

char temp = Char;

temp += 4;

Char += 2;

std::cout << " returned char: " << temp << std::endl;

return temp;

}

void func\_int\_char\_bool::void\_pointer(int\* pInt, char\* pChar, bool\* pBool) {

std::cout << " arguments are passed by pointer" << std::endl;

bool temp = \*pBool;

temp = !temp;

bool\* ptemp = pBool;

\*pBool = !(\*pBool);

}

//E

int\* func\_int\_char\_bool::intp\_value(int Int, char Char, bool Bool) {

std::cout << " arguments are passed by value" << std::endl;

std::cout << " returned pointer to int: " << &Int << std::endl;

return &Int;

}

char\* func\_int\_char\_bool::charp\_reference(int& Int, char& Char, bool& Bool) {

std::cout << " arguments are passed by reference" << std::endl;

std::cout << " returned pointer to char: " << static\_cast<void\*>(&Char) << std::endl;//to print char\* it has to be casted to other type\*

return &Char;

}

//D

void func\_num\_ar::void\_reference(int int\_array[]) {

std::cout << " array is passed by reference" << std::endl;

int size = (sizeof(int\_array) / sizeof(\*int\_array));

double average{};

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

average += int\_array[i];

}

average /= size;

std::cout << " average = " << average << std::endl;

}

void func\_num\_ar::void\_pointer(int\* int\_array0, int size) {

std::cout << " array is passed by pointer" << std::endl;

func\_srv::int\_ar\_bubble\_sort(int\_array0, size);

}

//C

myStruct func\_struct::MS\_value(myStruct myStruct1) {

std::cout << " structure is passed by value" << std::endl;

//myStruct temp = myStruct1;

myStruct1.Int \*= -1;

if (myStruct1.Char < 'a') {

myStruct1.Char += 32;//to lowercase

}

else {

myStruct1.Char -= 32;//to upper

}

myStruct1.Bool = !myStruct1.Bool;

char MS[] = "MS\_ret";

myStruct1.id = MS;

std::cout << " returned structure:" << std::endl;

myStruct1.print();

return myStruct1;

}

myStruct\* func\_struct::MSp\_reference(myStruct& myStruct1) {

std::cout << " structure is passed by reference" << std::endl;

myStruct1.Int++;

myStruct1.Char--;

myStruct1.Bool = !(myStruct1.Bool);

std::cout << " returned pointer to structure: " << &myStruct1 << std::endl;

return &myStruct1;

}

myStruct& func\_struct::MSr\_pointer(myStruct\* myStruct1) {

std::cout << " structure is passed by pointer" << std::endl;

(\*myStruct1).Int++;

(\*myStruct1).Char--;

(\*myStruct1).Bool = !((\*myStruct1).Bool);

std::cout << " adress returned reference to structure: " << &\*myStruct1 << std::endl;

return \*myStruct1;

}

//B

void func\_char\_ar::void\_reference(char char\_array[]) {

std::cout << " array is passed by reference" << std::endl;

func\_srv::ch\_ar\_bubble\_sort(&char\_array[0], ((sizeof(char\_array) / sizeof(\*char\_array)) + 1));

}

void func\_char\_ar::void\_pointer(char\* char\_array0, int size) {

std::cout << " array is passed by pointer" << std::endl;

func\_srv::ch\_ar\_bubble\_sort(char\_array0, size);

}

//A

void myStruct::print() {

std::cout << id << ".Int = " << Int << std::endl;

std::cout << id << std::boolalpha << ".Bool = " << Bool << std::endl;//boolalpha to print True/False and not 1/0

std::cout << id << ".Char = " << Char << std::endl;

}

void func\_srv::int\_ar\_bubble\_sort(int\* pArray0, int size) {

bool sorted = false;

for (int i = 0; !sorted && (i < (size - 1)); i++) {

bool sorted = true;

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

if (pArray0[j] > pArray0[j + 1]) {

int temp = pArray0[j];

pArray0[j] = pArray0[j + 1];

pArray0[j + 1] = temp;

sorted = false;

}

}

}

}

void func\_srv::print\_char\_ar(char\* pArray0, int size) {

std::cout << "Array: ";

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

std::cout << pArray0[i] << ' ';

}

std::cout << std::endl;

}

void func\_srv::ch\_ar\_bubble\_sort(char\* pArray0, int size) {

bool sorted = false;

for (int i = 0; !sorted && (i < size - 1); i++) {

bool sorted = true;

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

if (pArray0[j] > pArray0[j + 1]) {

char temp = pArray0[j];

pArray0[j] = pArray0[j + 1];

pArray0[j + 1] = temp;

sorted = false;

}

}

}

}

void func\_srv::print\_int\_ar(int\* pArray0, int size) {

std::cout << "integer array = ";

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

std::cout << pArray0[i] << ", ";

}

std::cout << pArray0[size - 1] << ';' << std::endl;

}

# **Результати виконання програми для завдання №1.**

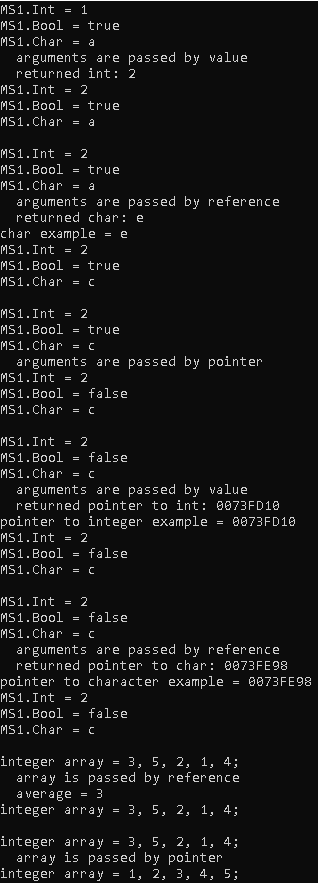


Рисунок 2

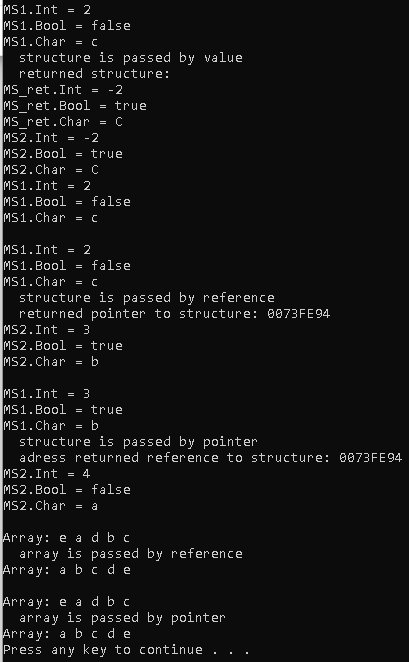


Рисунок 3

# **Аналіз результатів виконання програми для завдання №1 на основи опису механізму передачі параметрів.**

При передачі параметрів за значенням насправді створюються та іниціалізуються нові локальні змінні, всі маніпуляції проводяться з ними. Цей тип передачі належно використовувати, коли потрібно зберегти початкове значення параметра, а в функції він використовуеться, і там його значення змінюється. Але це е правдою тільки для відносно не великих об’ємів даних. У деяких випадках легше передати вказівник або посилання, і копіювати потрібний елемент структури даних. Масиви передати за значенням неможливо, через те, що масив насправді просто вказівник на перший елемент, а під значенням можна розуміти наприклад усі елементи, в такому разі потрібно було б вказувати фіксовану довжину масива для передачі в функцію, а це було б не універсально.

Передача за вказівником та за посиланням дещо схожі, наприклад в обох випадках модифікуються значення глобальних/локальних рівнем вище змінних. Але у той самий час присутні відмінності, наприклад якщо масив передається за вказівником, то передається і розмір масиву, а з вказівником – ні, хоча в обох випадках можливо ітерувати по елементам.

# **5. Постановка завдання №2.**

Переробити завдання з лабораторної роботи №5 в багатофайлову програму. Для кожної задачі розробити функцію (її специфікацію записати в коді програми). Розбити програму з трьох файлів: головний файл, файл заголовків і файл реалізації (або більше за функціоналом програми). В головному файлі реалізувати меню користувача. В файл заголовків перенести константи, опис типів даних, прототипи функцій. У файлі реалізації розмістити визначення функцій.

# **Текст програми для завдання №2 (головний файл, файл заголовків і файл реалізації).**

“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 <iomanip>

#include <Windows.h>

#include <string>

#include <limits>

#include "SE\_list\_manip.h"

#include "inp\_val6.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 = input\_validation::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

input\_validation::val\_char\_lenght(SE[SE\_amount].Name, 24);

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

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

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

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

std::cin.get();

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

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

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

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

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

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

SE\_amount++;

break;

}

case 2:

{

std::system("cls");

SE\_list\_manip::print\_heading();

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

SE\_list\_manip::print\_SE(SE, i);

}

SE\_list\_manip::print\_ending();

break;

}

case 3:

{

std::system("cls");

int chosen\_WE;

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

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

SE\_list\_manip::print\_heading();

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

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

SE\_list\_manip::print\_SE(SE, i);

}

}

SE\_list\_manip::print\_ending();

break;

}

case 4:

{

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

std::system("cls");

SE\_list\_manip::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) {

SE\_list\_manip::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

}

}

SE\_list\_manip::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;

input\_validation::val\_char\_lenght(chosen\_Name, 24);

SE\_list\_manip::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) {

SE\_list\_manip::print\_SE(SE, i);

}

}

SE\_list\_manip::print\_ending();

break;

}

case 6:

{

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

std::system("cls");

SE\_list\_manip::print\_heading();

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

SE\_list\_manip::print\_SE(SE, i);

}

SE\_list\_manip::print\_ending();

break;

}

case 7:

{

break;

}

default:

{

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

}

}

} while (menu\_choice != 7);

std::system("pause");

return 0;

}

“SE\_list\_manip.h”

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

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;

};

class SE\_list\_manip {

public:

static void print\_heading();

static void print\_ending();

static void print\_SE(struct SE\_ SE[], int i);

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

};

“SE\_list\_manip.cpp”

#include "SE\_list\_manip.h"

#include <iostream>

#include <iomanip>

void SE\_list\_manip::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 SE\_list\_manip::print\_ending() {

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

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

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

}

void SE\_list\_manip::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\_manip::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;

}

}

}

}

“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

class input\_validation {

public:

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

static void val\_char\_lenght(char array[], int ary\_size);

};

“inp\_val6.cpp”

//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>

#include <cmath>

#include <climits>

#include "inp\_val6.h"

#undef max

int input\_validation::cinum(char\* msg\_invite, char\* msg\_error, int m\_, bool intFlag, float left, float right)

{

bool i = 0;

float aaa;

std::cout << msg\_invite << std::endl;

do

{

std::cin >> aaa;

int iaaa;

iaaa = (int)aaa;

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

std::cout << msg\_error << std::endl;

std::cin.clear();

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

}

else {

i = 1;

}

} while (!i);

return (int)aaa;

}

void input\_validation::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) || std::cin.fail()) {

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

std::cin.clear();

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

}

else {

correct = true;

}

} while (!correct);

}

# **Результати тестування програми для завдання №2.**

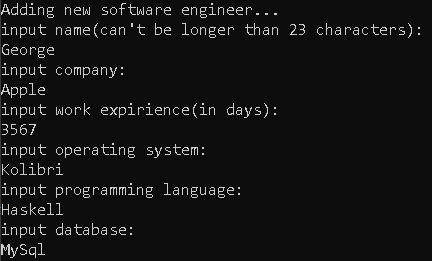


Рисунок 4

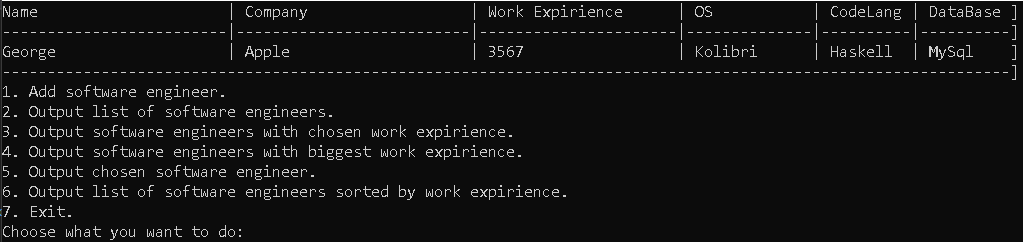


Рисунок 5

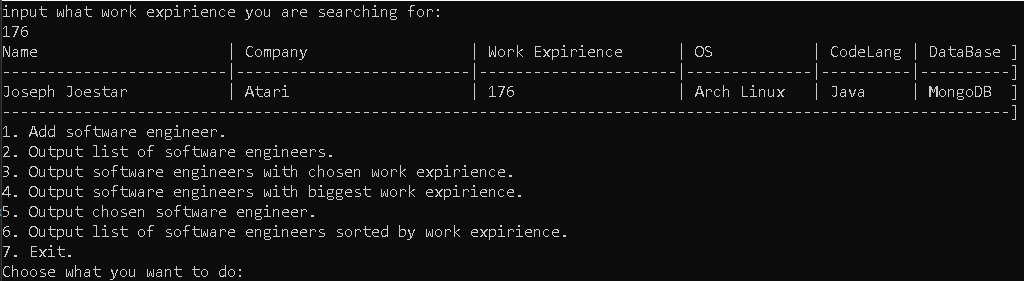


Рисунок 6

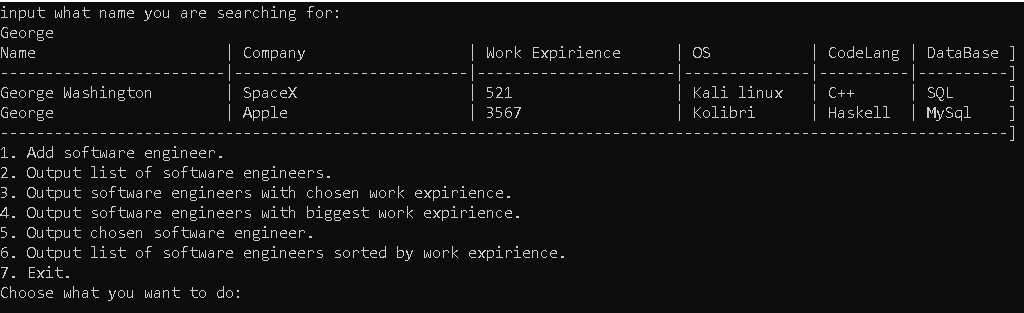


Рисунок 7

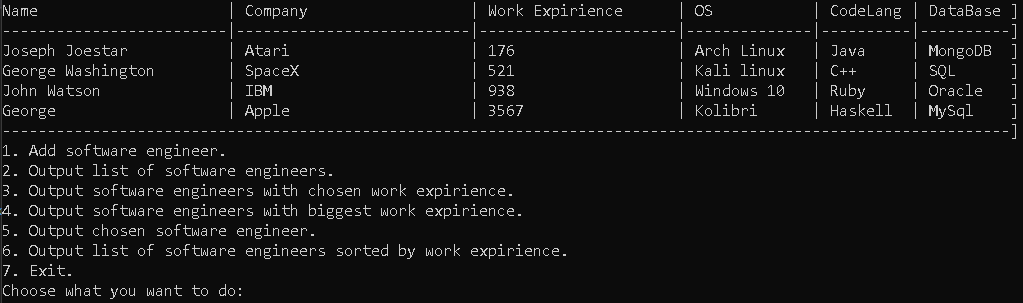


Рисунок 8

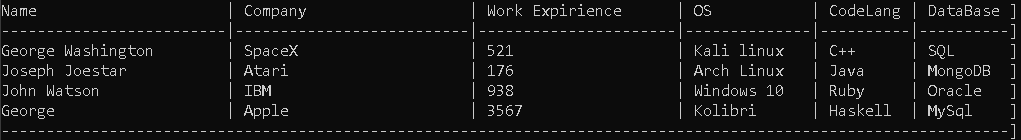


Рисунок 9

Програма працює корректно, через те, що вона перетерпіла мінімальну кількість модифікацій(Ще перша версія була розроблена із використанням функцій).

# **8. Висновки щодо призначення функцій, багатофайлових програм і механізму передачі параметрів.**

Код програми розбивається на окремі файли для того, щоб можна було легше розробляти/модифікувати програму. По перше можна виділити три основних типи файлів: головний файл(сама програма з якої визиваються окремі функції, тільки один на програму, містить точку входу та точку виходу з програми), інтерфейс(файли з прототипами функції), реалізація(файли з реалізацією прототипів).

Розбиття на інтерфейс та реалізацію існує по перше для того, щоб компілятор побачив усі зовнішні символи в файлі з прототипами, також набагато легше орієнтуватись дивлячись тільки на назви функцій, ніж на функції та їх реалізацію, інколи це може дуже заважати.

Також зазвичай усі пари прототип-реалізація зберігаються не в одному файлі, а розділенні за призначенням, бо так легше використовувати код в нових програмах(програми можуть використовувати лише одну спільну функцію, а всі інші різні, тоді було б потрібно кожного разу модифікувати файли).

При використанні функцій авжеж потрібно обирати з трьох типів передачі параметрів(в окремих випадках з двох), від цього залежить ефективність використання пам’яті, швидкість виконання програми та зручність використання функції. Наприклад, якщо потрібно зберегти значення аргументи, а функція приймає його за вказівником/посиланням, то потрібно буде перед викликом функції виділяти пам’ять під нову змінну, щоб тимчасово зберігати значення, а потім повертати, це буде не універсально. Але цей приклад демонструє один з найкращих випадків, існують сценарії з набагато більшими проблемами. Тож при обиранні типу передачі треба спиратися на тип змінної, її використання в функції, до її виклику та після.