Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 6, 8, 9**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Файли. Системи числення. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.»

***Виконав:***

студент групи ШІ-11

Рєвков Володимир

# **Тема роботи:**

Ознайомлення з темою «Файли. Системи числення. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.» та виконання лабораторних робіт з ВНС №6, №8 та №9, лабораторних робіт з Алготестера №4 та №6 і практичного та самостійного практичного завдання.

# **Мета роботи:**

Ознайомитись з темою «Файли. Системи числення. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.» та виконати лабораторні роботи з ВНС №6, №8 та №9, лабораторні роботи з Алготестера №4 та №6 і практичне та самостійне практичне завдання.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Системи числення.
* Тема №2: Бінарні файли.
* Тема №3: Символи і рядкові змінні.
* Тема №4: Стандартна бібліотека
* Тема №5: Методи роботи з файлами
* Тема №6: Створення й використання бібліотек

1. Індивідуальний план опрацювання теорії:

* Тема №1: Системи числення.
  + Джерела Інформації
    - Відео.

<https://youtu.be/yGmVLDenVpE?feature=shared>

* + Що опрацьовано:
    - Опрацьовано відео про системи числення.
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 13.12.2023
  + Звершення опрацювання теми: 13.12.2023
* Тема №2: Бінарні файли.
  + Джерела Інформації
    - Стаття.

<https://www.techtarget.com/whatis/definition/binary-file>

* + Що опрацьовано:
    - Опрацьовано статтю про бінарні файли.
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 13.12.2023
  + Звершення опрацювання теми: 13.12.2023
* Тема №3: Символи і рядкові змінні.
  + Джерела Інформації
    - Стаття.

<https://www.programiz.com/cpp-programming/strings>

* + Що опрацьовано:
    - Опрацьовано статтю про символи, рядкові змінні.
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 13.12.2023
  + Звершення опрацювання теми: 13.12.2023
* Тема №4: Стандартна бібліотека .
  + Джерела Інформації
    - Стаття.

<https://www.geeksforgeeks.org/functors-in-cpp/>

* + Що опрацьовано:
    - Опрацьовано статтю про стандартну бібліотеку
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 13.12.2023
  + Звершення опрацювання теми: 13.12.2023
* Тема №5: Методи роботи з файлами.
  + Джерела Інформації
    - Стаття.

<https://www.geeksforgeeks.org/basics-file-handling-c/?ref=lbp>

* + Що опрацьовано:
    - Опрацьовано статтю про методи роботи з файлами.
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 13.12.2023
  + Звершення опрацювання теми: 13.12.2023
* Тема №6: Створення та використання бібліотек.
  + Джерела Інформації
    - Стаття.

[https://learn.microsoft.com/en-us/cpp/build/walkthrough-creating-and-using-a-static-library-cpp?view=msvc-170#CreateLibProject](https://learn.microsoft.com/en-us/cpp/build/walkthrough-creating-and-using-a-static-library-cpp?view=msvc-170%23CreateLibProject)

* + Що опрацьовано:
    - Опрацьовано статтю про створення та використання бібліотек.
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 13.12.2023
  + Звершення опрацювання теми: 13.12.2023

# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

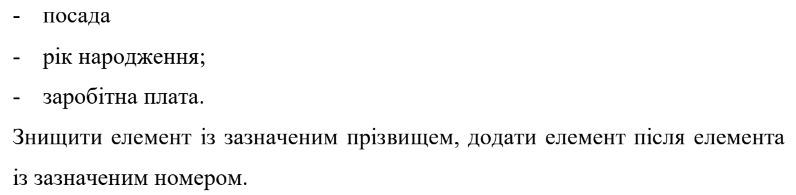
Завдання №1 Лабораторна робота ВНС №6

* Варіант завдання - **2**
* Деталі завдання: Надрукувати всі слова, які не містять голосних букв.

Завдання №2 Лабораторна робота ВНС №8

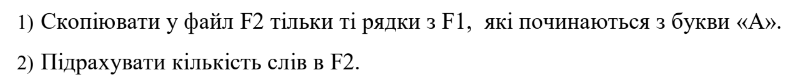
* Варіант завдання - **2**
* Деталі завдання:





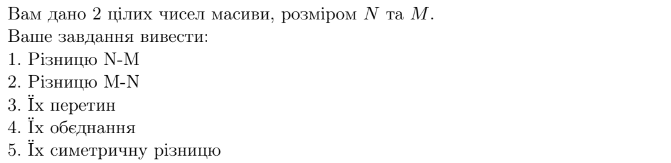
Завдання №3 Лабораторна робота ВНС №9

* Варіант завдання - **2**
* Деталі завдання:



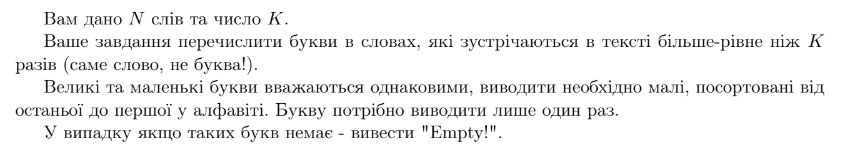
Завдання №4 Лабораторна робота Алготестер №4

* Варіант завдання - **1**
* Деталі завдання:



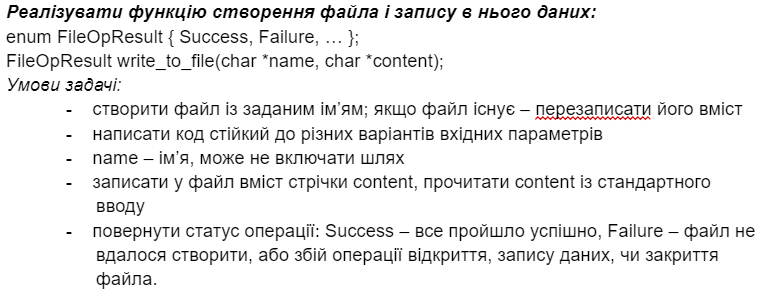
Завдання №5 Лабораторна робота Алготестер №6

* Варіант завдання - **1**
* Деталі завдання:



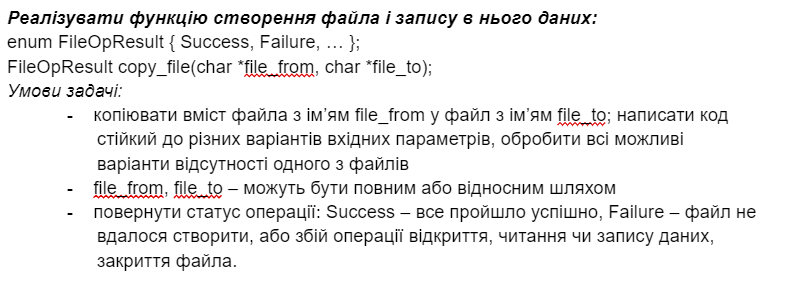
Завдання №6 Практична робота №1

* Деталі завдання:



Завдання №7 Практична робота №2

* Деталі завдання:



Завдання №8 Самостійна практична робота №1

* Деталі завдання: Застосунок для керування файлами. Програма дозволяє створювати, видаляти, очищати, редагувати та виводити файли.

## **2. Дизайн та планована оцінка часу виконання завдань:**

Завдання №4 Лабораторна робота Алготестер №4

* Блок-схема:

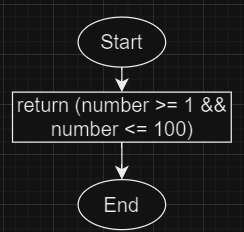
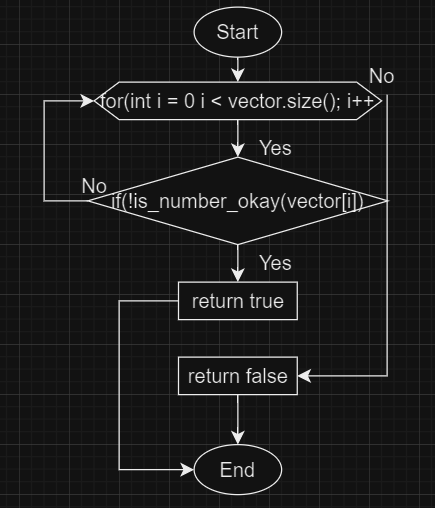
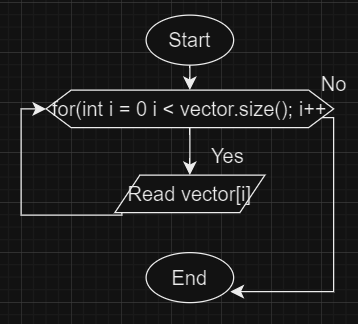
  

Figure : is\_number\_okay() Figure : is\_vector\_okay() Figure : fill\_vector()

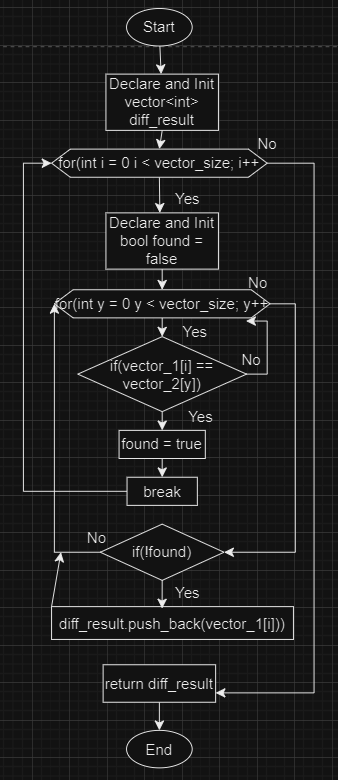
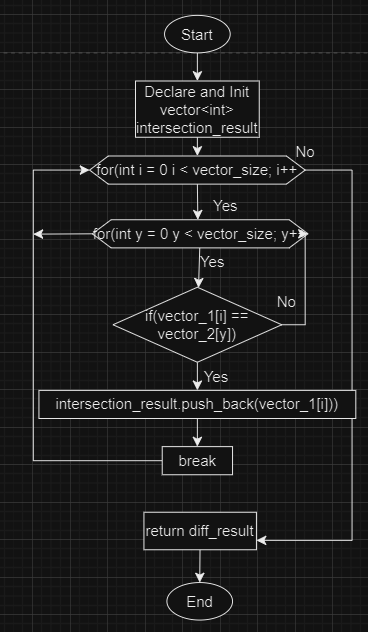
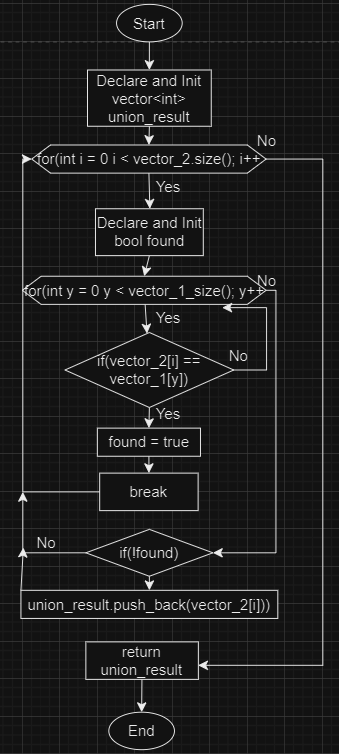
  

Figure : set\_diff() Figure : set\_intersection() Figure : set\_union

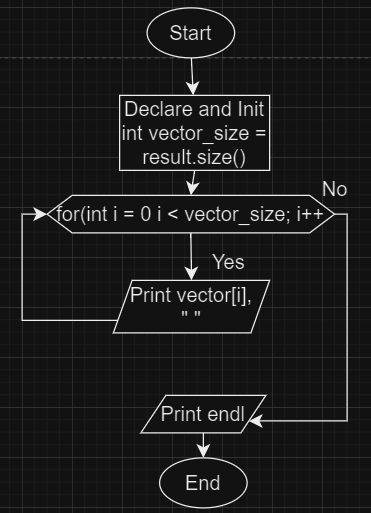


Figure : print\_result()

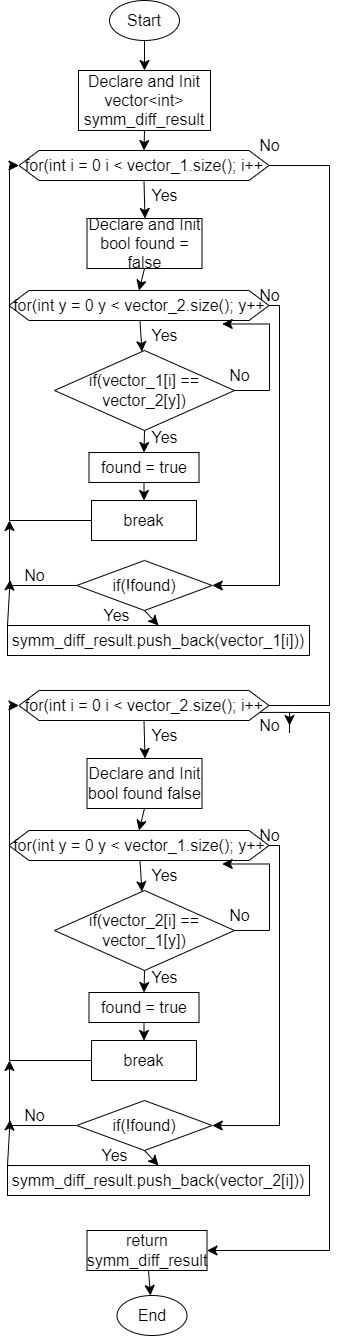
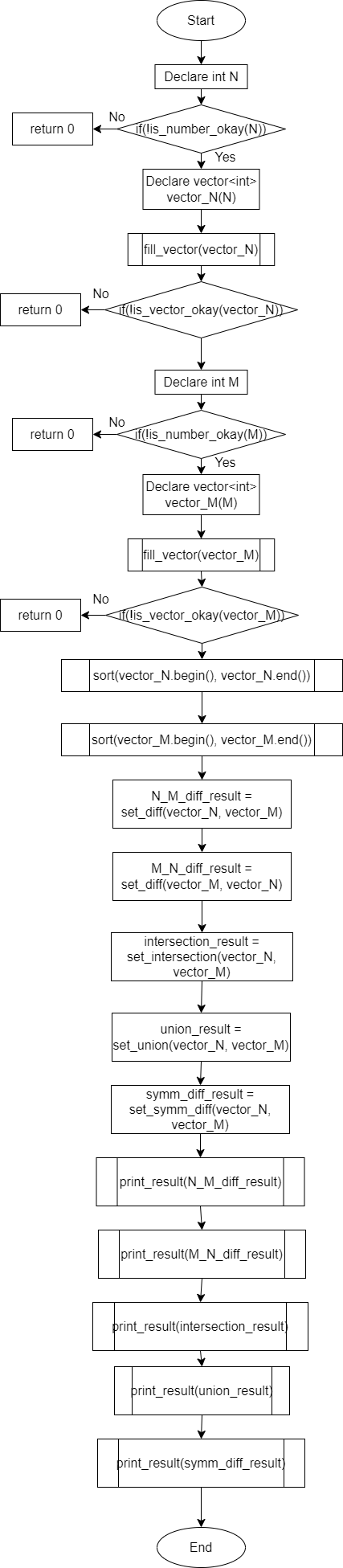
 

Figure 8: set\_symm\_diff() Figure 9: main()

* Планований час на реалізацію **– 2 години**

## **3. Код програм з посиланням на зовнішні ресурси:**

Завдання №1: Лабораторна робота ВНС №6

#include<iostream>

#include<string.h>

using namespace std;

const char special\_chars[] = "aeiou";

void fill\_string(char string[])

{

    cout << "Enter string (1-255 chars): ";

    gets(string);

}

bool is\_size\_problems\_with\_str(char string[], int size)

{

    if(strlen(string) < 1 || strlen(string) > size - 1)

    {

        cout << "Entered string don't reply size rules";

        cout << endl << endl;

        return true;

    }

    return false;

}

bool is\_word\_special(char\* string)

{

    bool result = true;

    for(int i = 0; i < strlen(string); i++)

    {

        if(!result)

        {

            break;

        }

        for(int y = 0; y < strlen(special\_chars); y++)

        {

            if(string[i] == special\_chars[y])

            {

                result = false;

                break;

            }

        }

    }

    return result;

}

void print\_special\_words(char string[])

{

    char\* token;

    token = strtok(string, " ");

    if(is\_word\_special(token))

    {

        puts(token);

        cout << endl;

    }

    while(token != NULL)

    {

        token = strtok(NULL, " ");

        if(token != NULL && is\_word\_special(token))

        {

            if(token[strlen(token) - 1] == '.')

            {

                token[strlen(token) - 1] = ' ';

            }

            puts(token);

            cout << endl;

        }

    }

}

int main()

{

    const int size = 255;

    char text[size];

    fill\_string(text);

    if(is\_size\_problems\_with\_str(text, size))

    {

        return 0;

    }

    print\_special\_words(text);

}

<https://github.com/volodymyr-rievkov/VolodymyrR_Reprository/blob/main/Epic%205/Lab6T1/vns_lab_6_task_1_variant_2_volodymyr_rievkov.cpp>

Завдання №2: Лабораторна робота ВНС №8

#include<iostream>

#include<string.h>

using namespace std;

const int workers\_amount = 20;

const char\* file\_name = "text.txt";

FILE\* file;

char delete\_surname[20];

struct CoWorker

{

    char name[20];

    char surname[20];

    char by\_father[20];

    char job[20];

    int birth\_year;

    int salary = 0;

    int number = 0;

};

CoWorker workers\_array[workers\_amount];

void add\_worker()

{

    if(strlen(workers\_array[workers\_amount - 1].name) == 0)

    {

        file = fopen(file\_name, "ab");

        if(file != nullptr)

        {

            cout << endl << "---Add worker---" << endl;

            cout << endl;

            int i;

            for(i = 0; i < workers\_amount; i++)

            {

                if(strlen(workers\_array[i].name) == 0)

                {

                    cout << "Enter name: ";

                    cin >> workers\_array[i].name;

                    cout << "Enter surname: ";

                    cin >> workers\_array[i].surname;

                    cout << "Enter byfather: ";

                    cin >> workers\_array[i].by\_father;

                    cout << "Enter job: ";

                    cin >> workers\_array[i].job;

                    cout << "Enter birth year: ";

                    cin >> workers\_array[i].birth\_year;

                    cout << "Enter salary: ";

                    cin >> workers\_array[i].salary;

                    workers\_array[i].number = i + 1;

                    cout << endl;

                    break;

                }

            }

            fwrite(&workers\_array[i], sizeof(CoWorker), 1, file);

            fclose(file);

        }

        else

        {

            cout << "Error: File opening is failed" << endl;

        }

    }

    else

    {

        cout << "Error: List is full" << endl;

    }

}

void print\_workers\_array()

{

    cout << "-----Workers array-----" << endl;

    cout << endl;

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

    {

        if(strlen(workers\_array[i].name) != 0)

        {

            cout << "---Worker №" << workers\_array[i].number << endl;

            cout << " - Name: " << workers\_array[i].name << endl;

            cout << " - Surname: " << workers\_array[i].surname << endl;

            cout << " - Byfather: " << workers\_array[i].by\_father << endl;

            cout << " - Job: " << workers\_array[i].job << endl;

            cout << " - Birth year: " << workers\_array[i].birth\_year << endl;

            cout << " - Salary: " << workers\_array[i].salary << endl;

            cout << endl;

        }

    }

}

void print\_workers\_file()

{

    cout << "-----Workers file-----" << endl;

    cout << endl;

    file = fopen(file\_name, "rb");

    if(file != nullptr)

    {

        CoWorker worker;

            while(fread(&worker, sizeof(CoWorker), 1, file))

            {

                cout << "---Worker №" << worker.number << endl;

                cout << " - Name: " << worker.name << endl;

                cout << " - Surname: " << worker.surname << endl;

                cout << " - Byfather: " << worker.by\_father << endl;

                cout << " - Job: " << worker.job << endl;

                cout << " - Birth year: " << worker.birth\_year << endl;

                cout << " - Salary: " << worker.salary << endl;

                cout << endl;

            }

        fclose(file);

    }

    else

    {

        cout << "Error: File opening is failed" << endl;

    }

}

void delete\_worker(char\* surname)

{

    file = fopen(file\_name, "wb");

    if(file != nullptr)

    {

        cout << endl;

        int i;

        int index;

        for(i = 0; i < workers\_amount; i++)

        {

            if(strcmp(workers\_array[i].surname, surname) == 0)

            {

                memset(&workers\_array[i], 0, sizeof(CoWorker));

                index = i;

                for(index; index < workers\_amount - 1; index++)

                {

                    workers\_array[index] = workers\_array[index + 1];

                    workers\_array[index].number--;

                }

            }

            if(strlen(workers\_array[i].name) != 0)

            {

                fwrite(&workers\_array[i], sizeof(CoWorker), 1, file);

            }

        }

        fclose(file);

    }

    else

    {

        cout << "Error: File opening is failed" << endl;

    }

}

char\* get\_surname()

{

    cout << endl << "---Delete worker---" << endl;

    cout << "Enter surname: ";

    cin >> delete\_surname;

    return delete\_surname;

}

void add\_worker\_after\_index()

{

    if(strlen(workers\_array[workers\_amount - 1].name) != 0)

    {

        cout << "Error: List is full" << endl;

    }

    else

    {

        int number;

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

        {

            if(strlen(workers\_array[i].name) == 0)

            {

                do

                {

                    cout << "Enter index(" << i << " - " << workers\_amount - 1 << "): ";

                    cin >> number;

                } while (number < i || number > workers\_amount - 1);

                break;

            }

        }

        file = fopen(file\_name, "ab");

        if(file != nullptr)

        {

            cout << endl << "---Add worker---" << endl;

            cout << endl;

            int i;

            for(i = 0; i < workers\_amount; i++)

            {

                if(i == number && strlen(workers\_array[i].name) == 0)

                {

                    cout << "Enter name: ";

                    cin >> workers\_array[i].name;

                    cout << "Enter surname: ";

                    cin >> workers\_array[i].surname;

                    cout << "Enter byfather: ";

                    cin >> workers\_array[i].by\_father;

                    cout << "Enter job: ";

                    cin >> workers\_array[i].job;

                    cout << "Enter birth year: ";

                    cin >> workers\_array[i].birth\_year;

                    cout << "Enter salary: ";

                    cin >> workers\_array[i].salary;

                    cout << endl;

                    workers\_array[i].number = number + 1;

                    break;

                }

            }

            fwrite(&workers\_array[i], sizeof(CoWorker), 1, file);

            fclose(file);

        }

        else

        {

            cout << "Error: File opening is failed" << endl;

        }

    }

}

int main()

{

    file = fopen(file\_name, "wb");

    fclose(file);

    add\_worker();

    add\_worker();

    print\_workers\_array();

    print\_workers\_file();

    add\_worker\_after\_index();

    print\_workers\_array();

    print\_workers\_file();

    delete\_worker(get\_surname());

    print\_workers\_array();

    print\_workers\_file();

}

<https://github.com/volodymyr-rievkov/VolodymyrR_Reprository/blob/main/Epic%205/Lab8T1/vns_lab_8_task_1_variant_2_volodymyr_rievkov.cpp>

Завдання №3: Лабораторна робота ВНС №9

#include<iostream>

#include<string.h>

using namespace std;

const char\* file\_1\_name = "file\_1.txt";

const char\* file\_2\_name = "file\_2.txt";

const char\* file\_content = "Lorem Ipsum is simply dummy text of the printing and typesetting industry.\n"

                            "Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when\n"

                            "an unknown printer took a galley of type and scrambled it to make a type specimen book.\n"

                            "It has survived not only five centuries, but\n"

                            "also the leap into electronic typesetting, remaining essentially unchanged.\n"

                            "It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages,\n"

                            "and more recently with desktop publishing software like\n"

                            "Aldus PageMaker including versions of Lorem Ipsum.\0";

void init\_file()

{

    FILE\* file = fopen(file\_1\_name, "w");

    if(file != nullptr)

    {

        fprintf(file, file\_content);

        fclose(file);

    }

    else

    {

        cout << "Error: File opening is failed" << endl;

    }

}

void print\_file(const char\* file\_name)

{

    FILE\* file = fopen(file\_name, "r");

    if(file != nullptr)

    {

        char buffer[1024];

        while(fgets(buffer, sizeof(buffer), file))

        {

            cout << buffer;

        }

        cout << endl << endl;

        fclose(file);

    }

    else

    {

        cout << "Error: File opening is failed" << endl;

    }

}

void add\_a\_strings\_to\_file(const char\* file\_1\_name, const char\* file\_2\_name)

{

    FILE\* file\_1 = fopen(file\_1\_name, "r");

    FILE\* file\_2 = fopen(file\_2\_name, "w");

    if(file\_1 != nullptr && file\_2 != nullptr)

    {

        char first\_line[1024];

        while(fgets(first\_line, sizeof(first\_line), file\_1))

        {

            if(first\_line[0] == 'a' || first\_line[0] == 'A')

            {

                fprintf(file\_2, first\_line);

            }

        }

        fclose(file\_1);

        fclose(file\_2);

    }

    else

    {

        cout << "Error: File opening is failed" << endl;

    }

}

int fget\_words\_count(const char\* file\_name)

{

    int count = 0;

    FILE\* file = fopen(file\_name, "r");

    if(file != nullptr)

    {

        char buffer[1024];

        while(fgets(buffer, sizeof(buffer), file))

        {

            char\* token;

            token = strtok(buffer, " ");

            if(token != nullptr)

            {

                count++;

            }

            while(token != nullptr)

            {

                token = strtok(nullptr, " ");

                if(token != nullptr)

                {

                    count++;

                }

            }

        }

        fclose(file);

    }

    else

    {

        cout << "Error: File opening is failed" << endl;

    }

    return count;

}

int main()

{

    init\_file();

    cout << "File 1: " << endl;

    print\_file(file\_1\_name);

    cout << "File 2: " << endl;

    add\_a\_strings\_to\_file(file\_1\_name, file\_2\_name);

    print\_file(file\_2\_name);

    cout << "Words count is " << fget\_words\_count(file\_2\_name) << endl;

}

<https://github.com/volodymyr-rievkov/VolodymyrR_Reprository/blob/main/Epic%205/Lab9T1/vns_lab_9_task_1_variant_2_volodymyr_rievkov.cpp>

Завдання №4: Лабораторна робота Алготестер №4

STL

#include <iostream>

#include <vector>

#include <algorithm>

#include<iterator>

using namespace std;

vector<int> N\_M\_diff\_result;

vector<int> M\_N\_diff\_result;

vector<int> intersection\_result;

vector<int> union\_result;

vector<int> symm\_diff\_result;

bool is\_number\_okay(int number)

{

    return (number >= 1 && number <= 100);

}

bool is\_vector\_okay(vector<int>& vector)

{

    for (int i = 0; i < vector.size(); i++)

    {

        if (!is\_number\_okay(vector[i]))

        {

            return false;

        }

    }

    return true;

}

void fill\_vector(vector<int>& vector)

{

    for (int i = 0; i < vector.size(); i++)

    {

        cin >> vector[i];

    }

}

void print\_result(vector<int>& result)

{

    int vector\_size = result.size();

    cout << vector\_size << endl;

    for (int i = 0; i < vector\_size; i++)

    {

        cout << result[i] << " ";

    }

    cout << endl;

}

int main()

{

    int N;

    cin >> N;

    if (!is\_number\_okay(N))

    {

        return 0;

    }

    vector<int> vector\_N(N);

    fill\_vector(vector\_N);

    if (!is\_vector\_okay(vector\_N))

    {

        return 0;

    }

    int M;

    cin >> M;

    if (!is\_number\_okay(M))

    {

        return 0;

    }

    vector<int> vector\_M(M);

    fill\_vector(vector\_M);

    if (!is\_vector\_okay(vector\_M))

    {

        return 0;

    }

    sort(vector\_N.begin(), vector\_N.end());

    sort(vector\_M.begin(), vector\_M.end());

    set\_difference(vector\_N.begin(), vector\_N.end(), vector\_M.begin(), vector\_M.end(), back\_inserter(N\_M\_diff\_result));

    set\_difference(vector\_M.begin(), vector\_M.end(), vector\_N.begin(), vector\_N.end(), back\_inserter(M\_N\_diff\_result));

    set\_intersection(vector\_N.begin(), vector\_N.end(), vector\_M.begin(), vector\_M.end(), back\_inserter(intersection\_result));

    set\_union(vector\_N.begin(), vector\_N.end(), vector\_M.begin(), vector\_M.end(), back\_inserter(union\_result));

    set\_symmetric\_difference(vector\_N.begin(), vector\_N.end(), vector\_M.begin(), vector\_M.end(), back\_inserter(symm\_diff\_result));

    print\_result(N\_M\_diff\_result);

    print\_result(M\_N\_diff\_result);

    print\_result(intersection\_result);

    print\_result(union\_result);

    print\_result(symm\_diff\_result);

}

<https://github.com/volodymyr-rievkov/VolodymyrR_Reprository/blob/main/Epic%205/AlgtL4T1/algt_lab_4_variant_1_1_volodymyr_rievkov.cpp>

Without STL

#include <iostream>

#include<algorithm>

#include <vector>

using namespace std;

vector<int> N\_M\_diff\_result;

vector<int> M\_N\_diff\_result;

vector<int> intersection\_result;

vector<int> union\_result;

vector<int> symm\_diff\_result;

bool is\_number\_okay(int number)

{

    return (number >= 1 && number <= 100);

}

bool is\_vector\_okay(vector<int>& vector)

{

    for (int i = 0; i < vector.size(); i++)

    {

        if (!is\_number\_okay(vector[i]))

        {

            return false;

        }

    }

    return true;

}

void fill\_vector(vector<int>& vector)

{

    for (int i = 0; i < vector.size(); i++)

    {

        cin >> vector[i];

    }

}

void print\_result(vector<int>& result)

{

    int vector\_size = result.size();

    cout << vector\_size << endl;

    for (int i = 0; i < vector\_size; i++)

    {

        cout << result[i] << " ";

    }

    cout << endl;

}

vector<int> set\_diff(vector<int>& vector\_1, vector<int>& vector\_2)

{

    vector<int> diff\_result;

    for (int i = 0; i < vector\_1.size(); i++)

    {

        bool found = false;

        for (int y = 0; y < vector\_2.size(); y++)

        {

            if (vector\_1[i] == vector\_2[y])

            {

                found = true;

                break;

            }

        }

        if (!found)

        {

            diff\_result.push\_back(vector\_1[i]);

        }

    }

    return diff\_result;

}

vector<int> set\_intersection(vector<int>& vector\_1, vector<int>& vector\_2)

{

    vector<int> intersection\_result;

    for (int i = 0; i < vector\_1.size(); i++)

    {

        for (int y = 0; y < vector\_2.size(); y++)

        {

            if (vector\_1[i] == vector\_2[y])

            {

                intersection\_result.push\_back(vector\_1[i]);

                break;

            }

        }

    }

    return intersection\_result;

}

vector<int> set\_union(vector<int>& vector\_1, vector<int>& vector\_2)

{

    vector<int> union\_result = vector\_1;

    for (int i = 0; i < vector\_2.size(); i++)

    {

        bool found = false;

        for (int y = 0; y < union\_result.size(); y++)

        {

            if (vector\_2[i] == union\_result[y])

            {

                found = true;

                break;

            }

        }

        if (!found) {

            union\_result.push\_back(vector\_2[i]);

        }

    }

    return union\_result;

}

vector<int> set\_symm\_diff(vector<int>& vector\_1, vector<int>& vector\_2)

{

    vector<int> symm\_diff\_result;

    for (int i = 0; i < vector\_1.size(); i++)

    {

        bool found = false;

        for (int y = 0; y < vector\_2.size(); y++)

        {

            if (vector\_1[i] == vector\_2[y])

            {

                found = true;

                break;

            }

        }

        if (!found)

        {

            symm\_diff\_result.push\_back(vector\_1[i]);

        }

    }

    for (int i = 0; i < vector\_2.size(); i++)

    {

        bool found = false;

        for (int y = 0; y < vector\_1.size(); y++)

        {

            if (vector\_2[i] == vector\_1[y])

            {

                found = true;

                break;

            }

        }

        if (!found)

        {

            symm\_diff\_result.push\_back(vector\_2[i]);

        }

    }

    return symm\_diff\_result;

}

int main()

{

    int N;

    cin >> N;

    if (!is\_number\_okay(N))

    {

        return 0;

    }

    vector<int> vector\_N(N);

    fill\_vector(vector\_N);

    if (!is\_vector\_okay(vector\_N))

    {

        return 0;

    }

    int M;

    cin >> M;

    if (!is\_number\_okay(M))

    {

        return 0;

    }

    vector<int> vector\_M(M);

    fill\_vector(vector\_M);

    if (!is\_vector\_okay(vector\_M))

    {

        return 0;

    }

    sort(vector\_N.begin(), vector\_N.end());

    sort(vector\_M.begin(), vector\_M.end());

    N\_M\_diff\_result = set\_diff(vector\_N, vector\_M);

    M\_N\_diff\_result = set\_diff(vector\_M, vector\_N);

    intersection\_result = set\_intersection(vector\_N, vector\_M);

    union\_result = set\_union(vector\_N, vector\_M);

    symm\_diff\_result = set\_symm\_diff(vector\_N, vector\_M);

    print\_result(N\_M\_diff\_result);

    print\_result(M\_N\_diff\_result);

    print\_result(intersection\_result);

    print\_result(union\_result);

    print\_result(symm\_diff\_result);

}

<https://github.com/volodymyr-rievkov/VolodymyrR_Reprository/blob/main/Epic%205/AlgtL4T1/algt_lab_4_variant_1_2_volodymyr_rievkov.cpp>

Завдання №5: Лабораторна робота Алготестер №6

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

const int border\_1 = 1;

const int border\_2 = 10e5;

bool check\_number(int number, int lower\_bound, int upper\_bound)

{

    return (number >= lower\_bound && number <= upper\_bound);

}

void process\_input(vector<string>& all\_words\_vector, vector<int>& word\_count\_vector, int N, int K)

{

    for (int i = 0; i < N; ++i)

    {

        string word;

        cin >> word;

        for (int j = 0; j < word.size(); ++j)

        {

            word[j] = tolower(word[j]);

        }

        int index = -1;

        for (int j = 0; j < all\_words\_vector.size(); ++j)

        {

            if (all\_words\_vector[j] == word)

            {

                index = j;

                break;

            }

        }

        if (index != -1)

        {

            word\_count\_vector[index]++;

        }

        else

        {

            all\_words\_vector.push\_back(word);

            word\_count\_vector.push\_back(1);

        }

    }

}

void print\_sorted\_result(const vector<string>& all\_words\_vector, const vector<int>& word\_count\_vector, int K)

{

    vector<char> sorted\_chars;

    for (int i = 0; i < all\_words\_vector.size(); ++i)

    {

        if (word\_count\_vector[i] >= K)

        {

            for (int j = 0; j < all\_words\_vector[i].size(); ++j)

            {

                sorted\_chars.push\_back(all\_words\_vector[i][j]);

            }

        }

    }

    sort(sorted\_chars.begin(), sorted\_chars.end());

    sorted\_chars.erase(unique(sorted\_chars.begin(), sorted\_chars.end()), sorted\_chars.end());

    reverse(sorted\_chars.begin(), sorted\_chars.end());

    if (!sorted\_chars.empty())

    {

        cout << sorted\_chars.size() << endl;

        for (int i = 0; i < sorted\_chars.size(); ++i)

        {

            cout << sorted\_chars[i] << " ";

        }

        cout << endl;

    }

    else

    {

        cout << "Empty!" << endl;

    }

}

int main()

{

    int N, K;

    cin >> N >> K;

    if (!check\_number(N, border\_1, border\_2) || !check\_number(K, border\_1, N))

    {

        return 0;

    }

    vector<string> all\_words\_vector;

    vector<int> word\_count\_vector;

    process\_input(all\_words\_vector, word\_count\_vector, N, K);

    print\_sorted\_result(all\_words\_vector, word\_count\_vector, K);

}

<https://github.com/volodymyr-rievkov/VolodymyrR_Reprository/blob/main/Epic%205/AlgtL6T1/algt_lab_6_variant_1_volodymyr_rievkov.cpp>

Завдання №6: Практична робота №1

#include<iostream>

using namespace std;

const char\* file\_name = "text.txt";

enum FileResult

{

    Success,

    Failure

};

void set\_content(string& content)

{

    cout << "Enter content to write to file: ";

    getline(cin, content);

}

FileResult write\_to\_file(const char\* file\_name, const char\* content)

{

    FILE\* file = fopen(file\_name, "w");

    if(file == nullptr)

    {

        return Failure;

        cout << "Error: Content writing is failed" << endl;

    }

    else

    {

        if(fprintf(file, content) < 0)

        {

            cout << "Error: File writing is failed" << endl;

            return Failure;

        }

        if(fclose(file) != 0)

        {

            cout << "Error: File closing is failed" << endl;

            return Failure;

        }

        return Success;

    }

}

void print\_file(const char\* file\_name)

{

    FILE\* file = fopen(file\_name, "r");

    if(file != nullptr)

    {

        cout << endl << "File content: " << endl << endl;

        char buffer[1024];

        while(fgets(buffer, sizeof(buffer), file))

        {

            cout << buffer;

        }

        cout << endl << endl;

        fclose(file);

    }

    else

    {

        cout << "Error: File  reading is failed" << endl;

    }

}

int main()

{

    string content;

    set\_content(content);

    if(write\_to\_file(file\_name, content.c\_str()) == Success)

    {

        cout << endl << "Content is successfully written" << endl;

        print\_file(file\_name);

    }

}

<https://github.com/volodymyr-rievkov/VolodymyrR_Reprository/blob/main/Epic%205/PractiseTask1/practise_task_1_volodymyr_rievkov.cpp>

Завдання №7: Практична робота №2

#include<iostream>

using namespace std;

const char\* file\_from\_name = "D:\\Programming\\C,C++Applications\\Epic 5\\PractiseTask2\\file\_from.txt";

const char\* file\_to\_name = "file\_to.txt";

enum FileResult

{

    Success,

    Failure

};

FileResult copy\_file(const char\* file\_from\_name, const char\* file\_to\_name)

{

    FILE\* file\_from = fopen(file\_from\_name, "r");

    FILE\* file\_to = fopen(file\_to\_name, "w");

    if(file\_from == nullptr && file\_to == nullptr)

    {

        cout << "Error: Files opening is failed" << endl;

        return Failure;

    }

    if(file\_from == nullptr)

    {

        cout << "Error: " << file\_from\_name << " opening is failed" << endl;

        return Failure;

    }

    if(file\_to == nullptr)

    {

        cout << "Error: " << file\_to\_name << " opening is failed" << endl;

        return Failure;

    }

    char buffer[1024];

    while(fgets(buffer, sizeof(buffer), file\_from))

    {

        if (buffer[strlen(buffer) - 1] != '\n')

        {

            cout << "Error: File reading is failed" << endl;

            return Failure;

        }

        if(fprintf(file\_to, buffer) < 0)

        {

            cout << "Error: File copying is failed" << endl;

            return Failure;

        }

    }

    int close\_f1 = fclose(file\_from);

    int close\_f2 = fclose(file\_to);

    if(close\_f1 != 0 && close\_f2 != 0)

        {

            cout << "Error: Files closing is failed" << endl;

            return Failure;

        }

    if(close\_f1 != 0)

        {

            cout << "Error: " << file\_from\_name << "closing is failed" << endl;

            return Failure;

        }

    if(close\_f2 != 0)

        {

            cout << "Error: " << file\_to\_name << "closing is failed" << endl;

            return Failure;

        }

    return Success;

}

void print\_file(const char\* file\_name)

{

    FILE\* file = fopen(file\_name, "r");

    if(file != nullptr)

    {

        cout << "File content: " << endl << endl;

        char buffer[1024];

        while(fgets(buffer, sizeof(buffer), file))

        {

            cout << buffer;

        }

        cout << endl;

        fclose(file);

    }

    else

    {

        cout << "Error: File  reading is failed" << endl;

    }

}

int main()

{

    if(copy\_file(file\_from\_name, file\_to\_name) == Success)

    {

        cout << endl << "From ";

        print\_file(file\_from\_name);

        cout << endl << "To ";

        print\_file(file\_from\_name);

    }

}

<https://github.com/volodymyr-rievkov/VolodymyrR_Reprository/blob/main/Epic%205/PractiseTask2/practise_task_2_volodymyr_rievkov.cpp>

Завдання №8: Самостійна практична робота

#include <iostream>

#include <string.h>

using namespace std;

const int array\_size = 10;

string file\_names[array\_size];

bool app\_exit = false;

string get\_file\_name()

{

    string file\_name;

    cout << "Enter file name: ";

    cin >> file\_name;

    cout << endl;

    return file\_name;

}

string get\_content()

{

    cin.ignore();

    string content;

    cout << "Enter content to write to file: ";

    getline(cin, content);

    cout << endl;

    return content;

}

bool is\_existed(const string& file\_name)

{

    for(int i = 0; i < array\_size; i++)

    {

        if(file\_names[i] == file\_name)

        {

            return true;

        }

    }

    return false;

}

bool is\_filled()

{

    return !file\_names[array\_size - 1].empty();

}

void create\_file(const string& file\_name)

{

    if (!is\_filled())

    {

        if (!is\_existed(file\_name))

        {

            FILE\* file = fopen(file\_name.c\_str(), "w");

            if (file != nullptr)

            {

                fclose(file);

                cout << "File '" << file\_name << "' has been successfully created" << endl << endl;

                for (int i = 0; i < array\_size; i++)

                {

                    if (file\_names[i].empty())

                    {

                        file\_names[i] = file\_name;

                        break;

                    }

                }

            }

            else

            {

                cout << "Error: File '" << file\_name << "' opening is failed" << endl<< endl;

            }

        }

        else

        {

            cout << "Error: File '" << file\_name << "' already exists" << endl<< endl;

        }

    }

    else

    {

        cout << "Error: File array is filled" << endl<< endl;

    }

}

void remove\_file(const string& file\_name)

{

    if (is\_existed(file\_name))

    {

        int index;

        for (int i = 0; i < array\_size; ++i)

        {

            if (file\_names[i] == file\_name)

            {

                index = i;

                break;

            }

        }

        if (index >= 0)

        {

            file\_names[index].clear();

            for (int i = index; i < array\_size - 1; ++i)

            {

                file\_names[i] = file\_names[i + 1];

            }

        }

        if (remove(file\_name.c\_str()) == 0)

        {

            cout << "File '" << file\_name << "' has been successfully removed" << endl << endl;

        }

        else

        {

            cout << "Error: File '" << file\_name << "' removing is failed" << endl << endl;

        }

    }

    else

    {

        cout << "Error: File '" << file\_name << "' doesn't exist" << endl << endl;

    }

}

void print\_file\_names()

{

    cout << endl << "----Existing files----" << endl;

    for (int i = 0; i < array\_size; i++)

    {

        if (file\_names[i].empty())

        {

            break;

        }

        else

        {

            cout << " - " << file\_names[i] << endl;

        }

    }

    cout << endl;

}

void clear\_file(const string& file\_name)

{

    FILE\* file = fopen(file\_name.c\_str(), "w");

    if (file != nullptr)

    {

        fclose(file);

        cout << "File '" << file\_name << "' has been cleared" << endl << endl;

    }

    else

    {

        cout << "Error: File '" << file\_name << "' opening is failed" << endl << endl;

    }

}

void write\_file(const string& file\_name)

{

    string content = get\_content();

    FILE\* file = fopen(file\_name.c\_str(), "a");

    if (file != nullptr)

    {

        if (fprintf(file, content.c\_str()) >= 0)

        {

            cout << "File '" << file\_name << "' has been successfully edited" << endl << endl;

        }

        else

        {

            cout << "Error: File '" << file\_name << "' hasn't been edited" << endl << endl;

        }

        fclose(file);

    }

    else

    {

        cout << "Error: File '" << file\_name << "' opening is failed" << endl << endl;

    }

}

void print\_file\_content(const string& file\_name)

{

    if (is\_existed(file\_name))

    {

        FILE\* file = fopen(file\_name.c\_str(), "r");

        if (file != nullptr)

        {

            cout << endl << "File '" << file\_name << "' content: " << endl << endl;

            char buffer[1024];

            while (fgets(buffer, sizeof(buffer), file))

            {

                cout << buffer;

            }

            cout << endl << endl;

            fclose(file);

        }

        else

        {

            cout << "Error: File '" << file\_name << "' opening is failed" << endl;

        }

    }

    else

    {

        cout << "Error: File '" << file\_name << "' doesn't exist" << endl;

    }

}

void clear\_all\_files()

{

    for (int i = 0; i < array\_size; i++)

    {

        if (file\_names[i].empty())

        {

            break;

        }

        remove(file\_names[i].c\_str());

        file\_names[i].clear();

    }

}

void exit\_app()

{

    app\_exit = true;

}

int get\_action()

{

    int action;

    int action\_count = 7;

    do

    {

        cout << " - Print files(1)" << endl

            << " - Create file(2)" << endl

            << " - Remove file(3)" << endl

            << " - Clear file(4)" << endl

            << " - Write file(5)" << endl

            << " - Print file content(6)" << endl

            << " - Exit(7)" << endl

            << "Enter action: ";

        cin >> action;

    } while (action < 1 || action > action\_count);

    return action;

}

void File\_Managment\_App()

{

    while (!app\_exit)

    {

        switch (get\_action())

        {

        case 1:

            print\_file\_names();

            break;

        case 2:

            create\_file(get\_file\_name());

            break;

        case 3:

            print\_file\_names();

            remove\_file(get\_file\_name());

            break;

        case 4:

            print\_file\_names();

            clear\_file(get\_file\_name());

            break;

        case 5:

            print\_file\_names();

            write\_file(get\_file\_name());

            break;

        case 6:

            print\_file\_names();

            print\_file\_content(get\_file\_name());

            break;

        case 7:

            exit\_app();

            break;

        }

    }

    clear\_all\_files();

}

int main()

{

    File\_Managment\_App();

}

<https://github.com/volodymyr-rievkov/VolodymyrR_Reprository/blob/main/Epic%205/SelfPractiseTask/self_practise_task_volodymyr_rievkov.cpp>

## **4. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1 Лабораторна робота ВНС №6

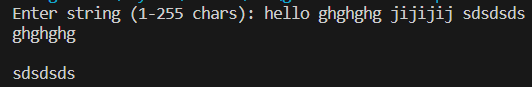


Figure 10: Результат ЛБ№6

Час затрачений на виконання завдання – **1 година**

Завдання №2 Лабораторна робота ВНС №8

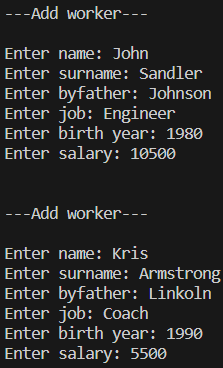
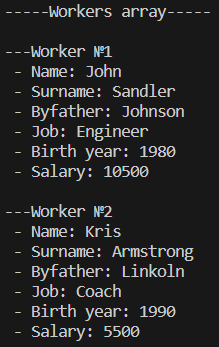
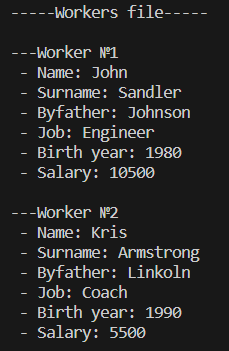
  

Figure 11: Результат ЛБ№8 Figure 12: Результат ЛБ№8 Figure 13: Результат ЛБ№8

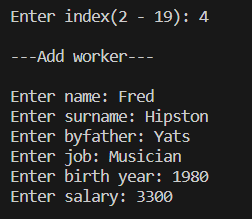
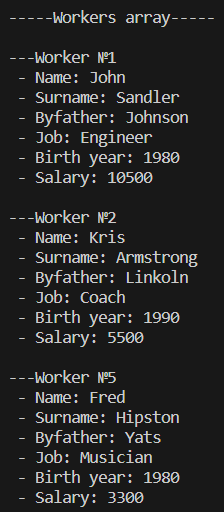
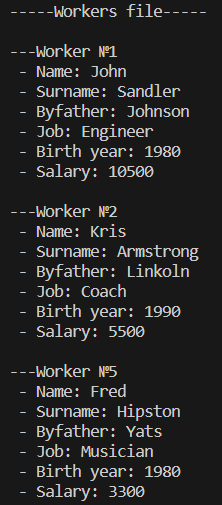
  

Figure 14: Результат ЛБ№8 Figure 15: Результат ЛБ№8 Figure 16: Результат ЛБ№8

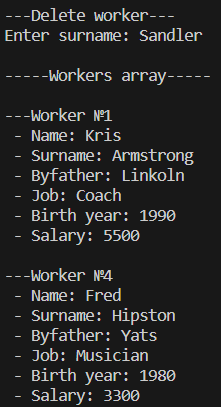
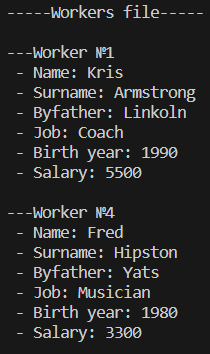
 

Figure 17: Результат ЛБ№8 Figure 18: Результат ЛБ№8

Час затрачений на виконання завдання – **10 годин**

Завдання №3 Лабораторна робота ВНС №9

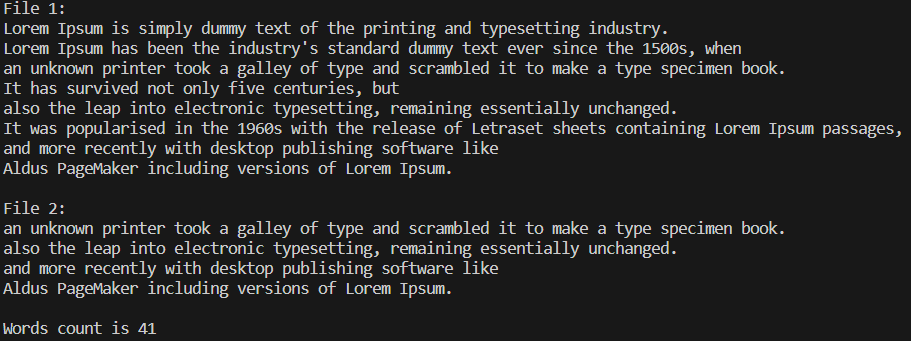


Figure 19: Результат ЛБ№9

Час затрачений на виконання завдання – **4 години**

Завдання №4 Лабораторна робота Алготестер №4

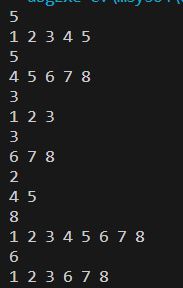


Figure 20: Результат ЛБ№4

Час затрачений на виконання завдання – **6 годин**

Завдання №5 Лабораторна робота Алготестер №6

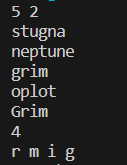


Рисунок 21: Результат ЛБ№6

Час затрачений на виконання завдання – **4 години**

Завдання №6 Практична робота №1

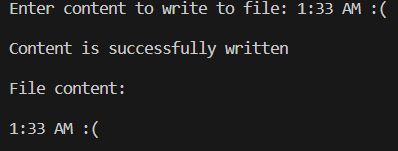


Figure 22: Результат ПР№1

Час затрачений на виконання завдання – **4 години**

Завдання №7 Практична робота №2

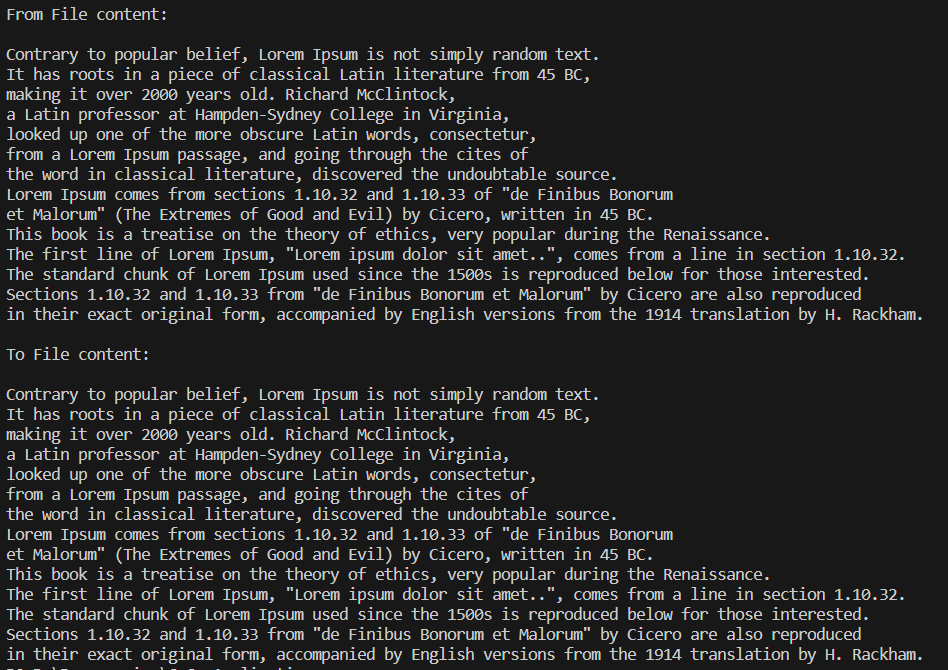


Figure 23: Результат ПР№2

Час затрачений на виконання завдання – **3 години**

Завдання №8 Самостійна практична робота №1

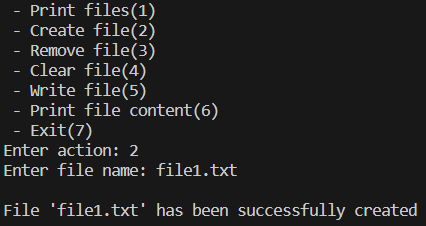
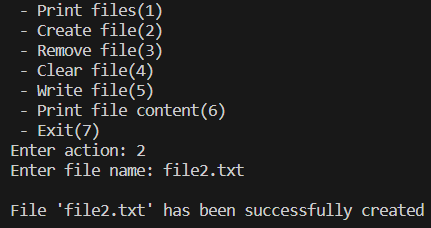
 

Figure 24: Результат СПР№1 Figure 25: Результат СПР№1

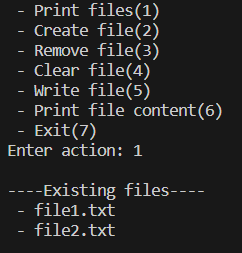
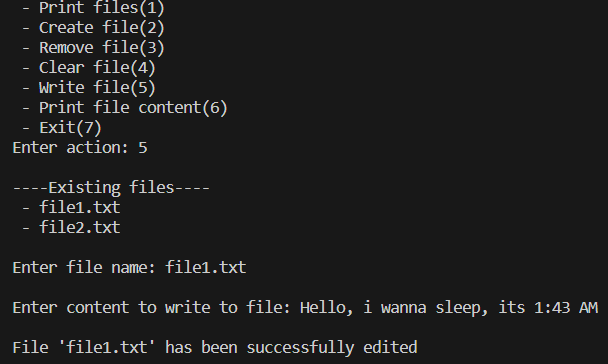
 

Figure 25: Результат СПР№1 Figure 26: Результат СПР№1

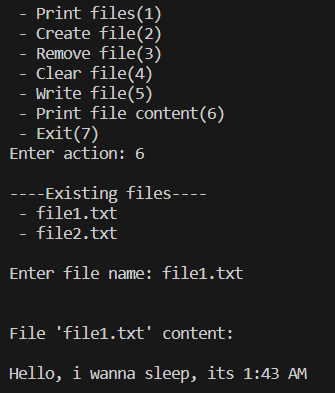
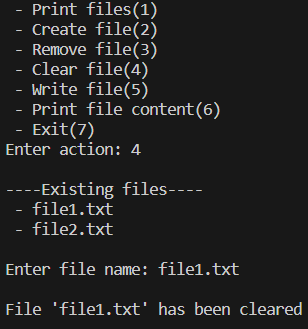
 

Figure 27: Результат СПР№1 Figure 28: Результат СПР№1

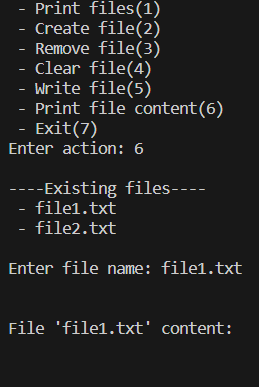
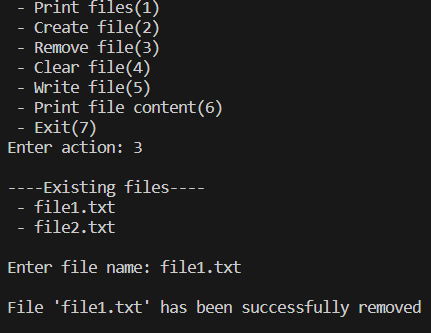
 

Figure 29: Результат СПР№1 Figure 30: Результат СПР№1

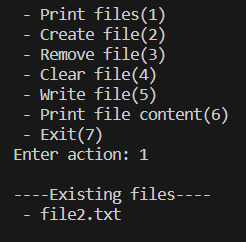
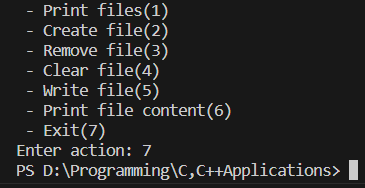
 

Figure 31: Результат СПР№1 Figure 32: Результат СПР№1

Час затрачений на виконання завдання – **2 години**

# **Висновки:**

Виконавши завдання Епіку №5, я дуже багато вивчив нового, а саме: символи, рядкові змінні, методи роботи з файлами, а також пригадав системи числення. А також отримав цінний досвід під час виконання практичних завдань.