Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

Звіт

про виконання лабораторних та практичних робіт блоку № 5

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

з дисципліни: «Основи програмування»

до:

ВНС Лабораторної Роботи № 6
ВНС Лабораторної Роботи № 8
ВНС Лабораторної Роботи № 9
Алготестер Лабораторної Роботи №4
Алготестер Лабораторної Роботи №6
Практичних Робіт до блоку №5

Виконала:
Студентка групи ШІ - 12
Лящук Соломія

Львів 2024

EPIC 5

“Файли. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.”

Tasks:

- John Black - Epic 5 Task 1 - Theory Education Activities
- John Black - Epic 5 Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-9)
- John Black - Epic 5 Task 3 - Lab# programming: VNS Lab 6
- John Black - Epic 5 Task 4 - Lab# programming: VNS Lab 8
- John Black - Epic 5 Task 5 - Lab# programming: VNS Lab 9
- John Black - Epic 5 Task 6 - Lab# programming: Algotester Lab 4
- John Black - Epic 5 Task 7 - Lab# programming: Algotester Lab 6
- John Black - Epic 5 Task 8 - Practice# programming: Class Practice Task
- John Black - Epic 5 Task 9 - Practice# programming: Self Practice Task
- John Black - Epic 5 Task 10 - Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)
- John Black - Epic 5 Task 11 - Results Evaluation and Release

Task 1

Theory Education Activities

1. Вступ до Роботи з Файлами:
    - Основні операції з файлами: відкриття, читання, запис, закриття
    - Робота з файловими дескрипторами
    - C-style читання з файлу та запис до файлу
    - Перевірка стану файлу: перевірка помилок, кінець файлу
    - Базові приклади читання та запису в файл
2. Символи і Рядкові Змінні:
    - Робота з char та string: основні операції і методи
    - Стрічкові літерали та екранування символів
    - Конкатенація, порівняння та пошук у рядках
3. Текстові Файли:
    - Особливості читання та запису текстових файлів
    - Обробка рядків з файлу: getline, ignore, peek
    - Форматування тексту при записі: setw, setfill, setprecision
    - Парсинг текстових файлів: розділення на слова, аналіз структури
    - Обробка помилок при роботі з файлами
4. Бінарні Файли:
    - Вступ до бінарних файлів: відмінності від текстових, приклади (великі дані, ігрові ресурси, зображення)
    - Читання та запис бінарних даних
    - Робота з позиціонуванням у файлі: seekg, seekp

- Серіалізація об'єктів у бінарний формат

5. Стандартна бібліотека та робота з файлами:
   - Огляд стандартної бібліотеки для роботи з файлами
   - Потоки вводу/виводу: ifstream, ofstream, fstream
   - Обробка помилок при роботі з файлами
6. Створення й використання бібліотек:
   - Вступ до створення власних бібліотек у C++
   - Правила розбиття коду на header-и(.h) та source(.cpp) файли
   - Статичні проти динамічних бібліотек: переваги та використання
   - Інтерфейси бібліотек: створення, документування, версіонування
   - Використання сторонніх бібліотек у проектах

Sours:

https://www.youtube.com/watch?v=SSNJ7alki-E&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=171

https://www.youtube.com/watch?v=YFLRN_Gmh4o

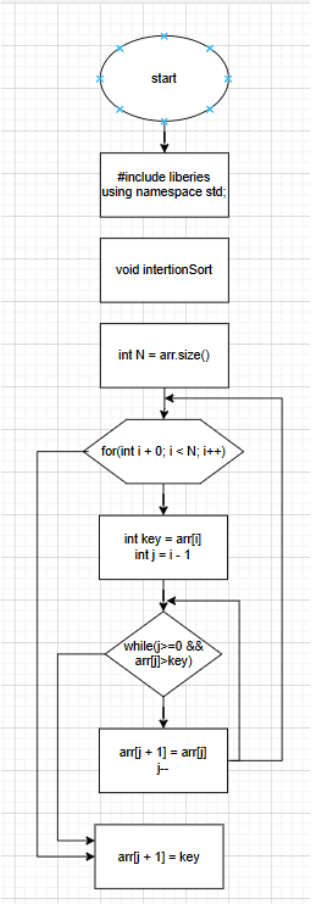
https://acode.com.ua/urok-183-shablony-klasiv/

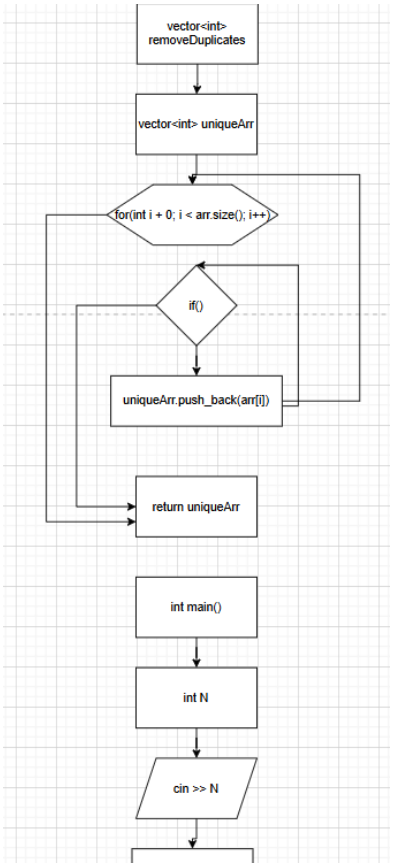
https://acode.com.ua/urok-204-standartna-biblioteka-shabloniv-stl/

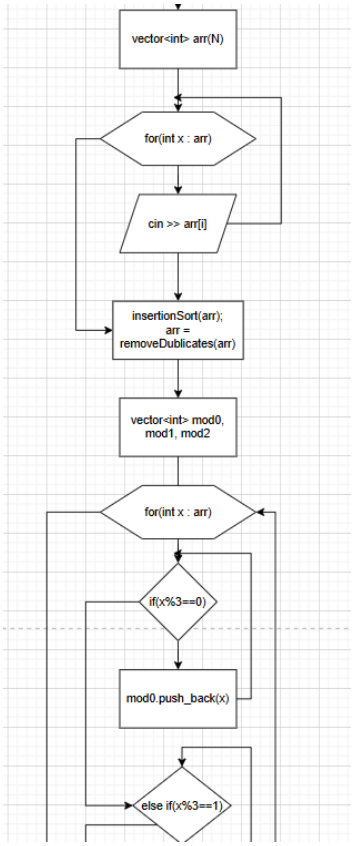
https://acode.com.ua/urok-220-bazovyj-fajlovyj-vvid-i-vyvid/

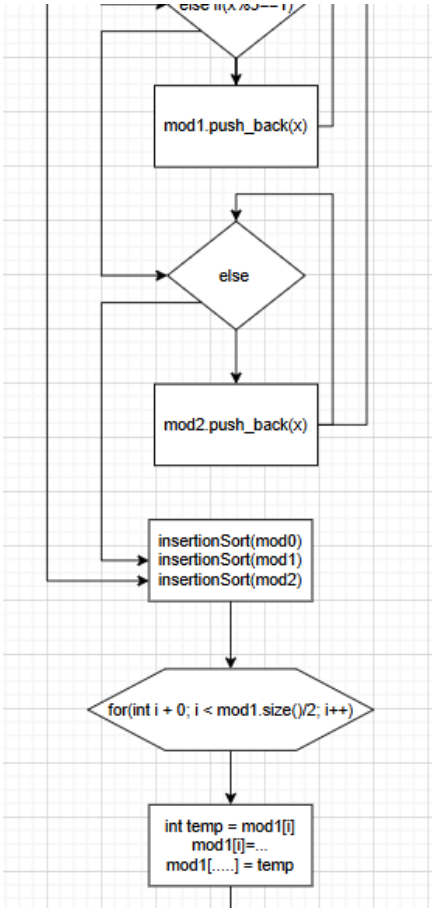
Task 2

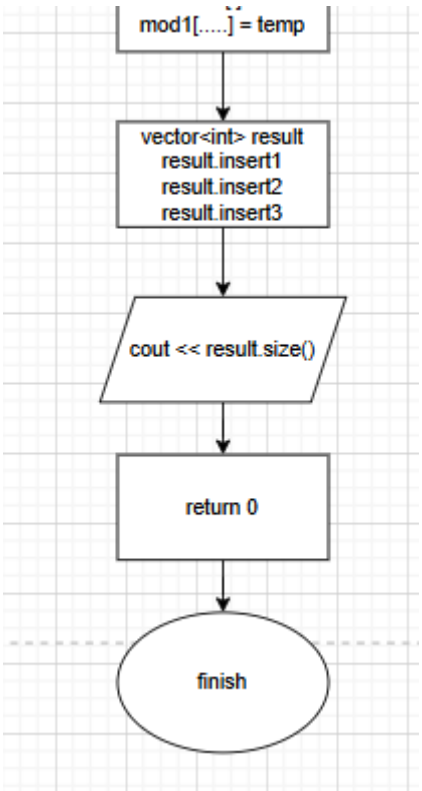
Requirements management (understand tasks) and design




vector<int>
removeDuplicates
vector<int> uniqueArr
for(int i + 0; i < arr.size(); i++)
if()
uniqueArr.push_back(arr[i])
return uniqueArr
int main()
int N
cin >> N


vector<int> arr(N)
for(int x : arr)
cin >> arr[i]
insertionSort(arr);
arr =
removeDublicates(arr)
vector<int> mod0,
mod1, mod2
for(int x : arr)
if(x%3==0)
mod0.push_back(x)
else if(x%3==1)





Task 3

VNS Lab 6

Задано рядок, що складається із символів. Символи поєднуються в слова. Слова одне від одного відокремлюються одним або декількома пробілами. Наприкінці тексту ставиться крапка. Текст містить не більше 255 символів. Виконати ввід рядка, використовуючи функцію gets(s) і здійснити обробку рядка у відповідності зі своїм варіантом.

21. Знищити всі парні слова у речені

C: > cpp > vns_lab_6_task_solomia_liashchuk.cpp > ...

```cpp
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
using namespace std;

string removeWords(const string& input)
{
    stringstream ss(input);
    string word;
    vector<string> words;

    while (ss >> word)
        words.push_back(word);

    string result;
    for (size_t i = 0; i < words.size(); i++)
    {
        if ((i + 1) % 2 != 0)
        {
            result += words[i] + " ";
        }
    }
    if (!result.empty())
    {
        result.pop_back();
    }

    return result;
}
```

```cpp

int main()
{
    string input;
    cout << "Введіть текст: " << endl;
    getline(cin, input);

    if (input.length() > 255)
    {
        cout << "Розмір тексту завеликий" << endl;
        return 1;
    }
    cout << "Текст без парних слів: " << removeWords(input) << endl;

    return 0;
}

```

```
Введіть текст:
Сьогодні гарний день! Погода похмура але тепло та без дощу.
Текст без парних слів: Сьогодні день! похмура тепло без
```

Task 4

VNS Lab 8

vns_lab_8_task_solomia_liashchuk.cpp

C: > cpp > vns_lab_8_task_solomia_liashchuk.cpp > ...

```cpp
#include <iostream>
#include <string>
#include <fstream>
#include <bitset>
#include <vector>
#include <iomanip>

using namespace std;

struct Car
{
    char brand[50];
    int serialNumber;
    char regNumber[30];
    int year;
};

void process_file(const string& filename, vector<Car>& cars)
{
    ofstream outfile(filename, ios::binary);
    if(!outfile.is_open())
    {
    cout << "Error: Unable to creat a file";
    return;
    }

    for (const Car& car : cars)
    {
        outfile.write(car.brand, sizeof(car.brand));
        outfile.write(reinterpret_cast<const char*>(&car.serialNumber), sizeof(car.serialNumber));
        outfile.write(car.regNumber, sizeof(car.regNumber));
        outfile.write(reinterpret_cast<const char*>(&car.year), sizeof(car.year));
    }

    outfile.close();
```

```
    ifstream infile(filename, ios::binary);
    if(!infile.is_open())
    {
        cout << "Error: Unable to read the file :(";
        return;
    }

    cout << left << setw(50) << "Brand" << setw(15) << "Serial Number" << setw(15) << "Reg Number" << setw(15) << "Year" << endl;
    cout << "_____" << endl;
    Car carR;
    while(infile.read(reinterpret_cast<char*>(&carR), sizeof(Car)))
    {
        cout << setw(50) << carR.brand << setw(15) << carR.serialNumber << setw(15) << carR.regNumber << setw(15) << carR.year << endl;
    }

    infile.close();

    if(cars.size() >= 3)
    {
        cars.erase(cars.begin(), cars.begin() + 3);
    }

    Car newCar = {"NewBrand", 12345, "ABC123", 2023};
    cars.push_back(newCar);

    outfile.open(filename, ios::binary | ios::trunc);
    if(!outfile.is_open())
    {
        cout << "Error: Unable to create the file" << endl;
        return;
    }
```

```
    for(const Car& car : cars)
    {
        outfile.write(car.brand, sizeof(car.brand));
        outfile.write(reinterpret_cast<const char*>(&car.serialNumber), sizeof(car.serialNumber));
        outfile.write(car.regNumber, sizeof(car.regNumber));
        outfile.write(reinterpret_cast<const char*>(&car.year), sizeof(car.year));
    }

    outfile.close();

    infile.open(filename, ios::binary);
    if(!infile.is_open())
    {
        cout << "Error: Unable to read the file :(" << endl;
        return;
    }

    cout << "\nUpdated file content:" << endl;
    cout << left << setw(50) << "Brand" << setw(15) << "Serial Number" << setw(15) << "Reg Number" << setw(15) << "Year" << endl;
    cout << "_____" << endl;

    while(infile.read(reinterpret_cast<char*>(&carR), sizeof(Car)))
    {
        cout << setw(50) << carR.brand << setw(15) << carR.serialNumber << setw(15) << carR.regNumber << setw(15) << carR.year << endl;
    }

    infile.close();
}
```

```
int main()
{
    vector<Car> cars =
    {
    {"Toyota", 12345, "AB1234AA", 2010},
    { "Honda", 67890, "KH5678KH", 2015},
    {  "Ford", 11223, "KS9101KS", 2012},
    {   "BMW", 33445, "DP1122DP", 2020}
    };

    string filename = "cars.dat.bin";
    process_file(filename, cars);

    cout << "File updated successfully!" << endl;

    return 0;
}
```

Task 5

VNS Lab 9

Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього інформацію

21. 1) Скопіювати з файлу F1 у файл F2 всі рядки, у яких більше 2 слів. 2) Визначити номер слова, у якому найбільше голосних букв.

C: > cpp > vns_lab_9_task_solomia_liashchuk.cpp > main()

```cpp
#include <iostream>
#include <fstream>
#include <cstring>
int count_words(const char* str)
{
    int word_count = 0;
    bool in_word = false;

    while (*str)
    {
        if (*str != ' ' && *str != '\n' && *str != '\t')
        {
            if (!in_word)
            {
                in_word = true;
                word_count++;
            }
        }
        else
        {
            in_word = false;
        }
        str++;
    }

    return word_count;
}

int count_vowels(const char* word)
{
    int vowels = 0;
```

```c
int count_vowels(const char* word)
{
    int vowels = 0;
    const char* vowels_set = "aeiouAEIOUаеиіоуяюєїАЕИІОУЯЮЄЇ";

    while (*word)
    {
        if (strchr(vowels_set, *word))
        {
            vowels++;
        }
        word++;
    }

    return vowels;
}

int find_word_with_most_vowels(const char* str)
{
    int max_vowels = 0;
    int word_index = 0;
    int current_index = 0;

    const char* word_start = str;
    while(*str)
    {
        if(*str == ' ' || *str == '\n' || *str == '\t' || *(str + 1) == '\0')
        {
            if(*(str + 1) == '\0')
                str++;

```

```
            if(*(str + 1) == '\0')
                str++;

            char word[256];
            strncpy(word, word_start, str - word_start);
            word[str - word_start] = '\0';

            int vowels = count_vowels(word);
            if(vowels > max_vowels)
            {
                max_vowels = vowels;
                word_index = current_index + 1;
            }

            current_index++;
            word_start = str + 1;
        }
        str++;
    }

    return word_index;
}

int main()
{
    FILE *file1 = fopen("file.task9.txt", "w");
    if(file1 == NULL)
    {
        printf("Error: Unable to create file F1\n");
        return 1;
    }

    fprintf(file1, "Хто тримає цей район?.\n");
```

```c

    fprintf(file1, "Хто тримає цей район?.\n");
    fprintf(file1, "Пес Патрон, пес Патрон.\n");
    fprintf(file1, "Два слова.\n");
    fprintf(file1, "Хто крутіший за iPhone?.\n");
    fprintf(file1, "Два слова.\n");
    fprintf(file1, "Пес Патрон, пес Патрон.\n");
    fprintf(file1, "Два слова.\n");
    fprintf(file1, "Хто не ходить на газон?.\n");
    fprintf(file1, "Пес Патрон, пес Патрон.\n");
    fprintf(file1, "Два слова.\n");
    fprintf(file1, "В розмінуванні чемпіон.\n");
    fprintf(file1, "Пес Патрон, пес Патрон.\n");
    fprintf(file1, "Два слова.\n");
    fprintf(file1, "Пес Патрон, пес Патрон.\n");
    fprintf(file1, "Таких нам треба батальйон.\n");

    fclose(file1);

    file1 = fopen("file.task9.txt", "r");
    if(file1 == NULL)
    {
        printf("Error: Unable to open file F1 for reading\n");
        return 1;
    }

    printf("Contents of file1.txt:\n");
    char line[256];
    while (fgets(line, sizeof(line), file1))
    {
        printf("%s", line);
    }
    fclose(file1);

```

```c
    FILE *file2 = fopen("file.task9.txt2", "w");
    if(file2 == NULL)
    {
        printf("Error: Unable to create file F2\n");
        return 1;
    }

    file1 = fopen("file.task9.txt", "r");
    if(file1 == NULL)
    {
        printf("Error: Unable to read file1\n");
        return 1;
    }

    while (fgets(line, sizeof(line), file1) != NULL)
    {
        if (count_words(line) > 2)
        {
            fputs(line, file2);
        }
    }

    fclose(file1);
    fclose(file2);

    printf("Content successfully copied from file.task9.txt to file2.task9.txt\n");

    file2 = fopen("file.task9.txt2", "r");
    if(file2 == NULL)
    {
        printf("Error: Unable to open file2 for reading\n");
        return 1;
    }
```

```c

    while(fgets(line, sizeof(line), file2))
    {
        printf("%s", line);
    }
    fclose(file2);

    file2 = fopen("file.task9.txt2", "r");
    if (file2 == NULL)
    {
        printf("Error: Unable to open file F2 for reading\n");
        return 1;
    }

    printf("\nFinding word with the most vowels in file2.txt:\n");
    while (fgets(line, sizeof(line), file2))
    {
        int max_vowel_word_index = find_word_with_most_vowels(line);
        printf("In line: \"%s\" -> Word #%d has the most vowels.\n", line, max_vowel_word_index);
    }

    fclose(file2);

    return 0;
}
```

```
Contents of file1.txt:
Хто тримає цей район?.
Пес Патрон, пес Патрон.
Два слова.
Хто крутіший за iPhone?.
Два слова.
Пес Патрон, пес Патрон.
Два слова.
Хто не ходить на газон?.
Пес Патрон, пес Патрон.
Два слова.
В розмінуванні чемпіон.
Пес Патрон, пес Патрон.
Два слова.
Пес Патрон, пес Патрон.
Таких нам треба батальйон.
Content successfully copied from file.task9.txt to file2.task9.txt
Хто тримає цей район?.
Пес Патрон, пес Патрон.
Хто крутіший за iPhone?.
Пес Патрон, пес Патрон.
Хто не ходить на газон?.
Пес Патрон, пес Патрон.
В розмінуванні чемпіон.
Пес Патрон, пес Патрон.
Пес Патрон, пес Патрон.
Таких нам треба батальйон.

Finding word with the most vowels in file2.txt:
In line: "Хто тримає цей район?.
" -> Word #2 has the most vowels.
In line: "Пес Патрон, пес Патрон.
" -> Word #2 has the most vowels.
In line: "Хто крутіший за iPhone?.
" -> Word #2 has the most vowels.
In line: "Пес Патрон, пес Патрон.
" -> Word #2 has the most vowels.
In line: "Хто не ходить на газон?.
" -> Word #3 has the most vowels.
In line: "Пес Патрон, пес Патрон.
" -> Word #2 has the most vowels.
```

Task 6

Algotester Lab 4

V – 2

Вам дано масив a з NN цілих чисел.
Спочатку видаліть масиву a усі елементи що повторюються, наприклад масив [1, 3, 3, 4] має перетворитися у [1, 3, 4].
Після цього оберніть посортовану версію масиву a на KK, тобто при K=3K=3 масив [1, 2, 3, 4, 5, 6, 7] перетвориться на [4, 5, 6, 7, 1, 2, 3].
Виведіть результат.

C: > cpp > algotester_lab_4_task_solomia_liashchuk.cpp > main()

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
    int N, K;
    cin >> N >> K;

    vector<int> numbers(N);
    for(int i = 0; i < N; ++i)
    {
        cin >> numbers[i];
    }
    sort(numbers.begin(), numbers.end());
    auto search = unique(numbers.begin(), numbers.end());
    numbers.erase(search, numbers.end());
    N = numbers.size();

    rotate(numbers.begin(), numbers.begin() + K % N, numbers.end());

    cout << N << endl;
    for (int i = 0; i < N; i++)
    {
        cout << numbers[i] << " ";
    }
    cout << endl;
    return 0;
}
```

Algotester Lab 4

V – 3

Вам дано масив, який складається з NN додатніх цілих чисел.
Ваше завдання - розділити його на три частини, по остачі від ділення на 3, по зростанню остачі (тобто спочатку йдуть числа, у яких остача 0, далі числа з остачею 1 і тоді нарешті числа з остачею 2).

Далі необхідно ті елементи, остача від ділення на 3 яких парна посортувати по зростанню, а ті, у яких остача 1 - по спаданню.

Після цього видаліть усі дублікати з масиву.

Виведіть результуючий масив.

1) з використанням STL

algotester_lab_4_variant_2_solomia_liashchuk.cpp

C: > cpp > algotester_lab_4_variant_2_solomia_liashchuk.cpp > main()

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
    int N;
    cin >> N;

    vector<int> arr(N);
    for(int i = 0; i < N; i++)
    {
        cin >> arr[i];
    }

    sort(arr.begin(), arr.end());
    auto it= unique(arr.begin(),arr.end());
    arr.erase(it,arr.end());

    vector<int> mod0, mod1, mod2;
    for(int x : arr)
    {
        if(x % 3 == 0)
        {
            mod0.push_back(x);
        }
        else if(x % 3 == 1)
        {
            mod1.push_back(x);
        }
```

```
        else
        {
            mod2.push_back(x);
        }
    }

    sort(mod0.begin(), mod0.end());
    sort(mod1.rbegin(), mod1.rend());
    sort(mod2.begin(), mod2.end());

    arr.clear();
    arr.insert(arr.end(), mod0.begin(), mod0.end());
    arr.insert(arr.end(), mod1.begin(), mod1.end());
    arr.insert(arr.end(), mod2.begin(), mod2.end());

    cout<<arr.size()<<endl;
    for(auto i:arr)
    {
        cout<<i<<" ";
    }
}
```

2) за допомогою сортування вставками(Insertion Sort)

C: > cpp > algotester_lab_4_variant_2_solomia_liashchuk2.cpp > main()

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

void insertionSort(vector<int>& arr) //сортування вставками
{
    int N = arr.size();
     for(int i = 1; i < N; i++)
    {
        int key = arr[i];
        int j = i - 1;

        while(j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }

}

vector<int> removeDuplicates(vector<int>& arr) //видалення дублікатів
{
    vector<int> uniqueArr;
    for(int i = 0; i < arr.size(); i++)
    {
        if(i == 0 || arr[i] != arr[i - 1])
        {
            uniqueArr.push_back(arr[i]);
        }
    }
    return uniqueArr;
}
```

```cpp
int main()
{
    int N;
    cin >> N;

    vector<int> arr(N);
    for(int i = 0; i < N; i++)
    {
        cin >> arr[i];
    }

    insertionSort(arr);
    arr = removeDuplicates(arr);

    vector<int> mod0, mod1, mod2;
    for(int x : arr)
    {
        if(x % 3 == 0)
        {
            mod0.push_back(x);
        }
        else if(x % 3 == 1)
        {
            mod1.push_back(x);
        }
        else
        {
            mod2.push_back(x);
        }
    }
```

```cpp
    insertionSort(mod0);
    insertionSort(mod1);
    insertionSort(mod2);

    for(int i = 0; i < mod1.size() / 2; i++)
    {
        int temp = mod1[i];
        mod1[i] = mod1[mod1.size() - 1 - i];
        mod1[mod1.size() - 1 - i] = temp;
    }

    vector<int> result;
    result.insert(result.end(), mod0.begin(), mod0.end());
    result.insert(result.end(), mod1.begin(), mod1.end());
    result.insert(result.end(), mod2.begin(), mod2.end());

    cout << result.size() << endl;
    for (int i : result)
    {
        cout << i << " ";
    }

    return 0;
}
```

Task 7

Algotester Lab 6

C: > cpp > algotester_lab_6_task_solomia_liashchuk.cpp > findPossibleValues(vector<vector<int>>&, int, int, int)

```cpp
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
#include <set>

using namespace std;

vector<int> findPossibleValues(vector<vector<int>> &board, int rowIdx, int colIdx, int size)
{
    vector<int> possibleValues;

    if(board[rowIdx][colIdx] != 0)
    {
        possibleValues.push_back(board[rowIdx][colIdx]);
        return possibleValues;
    }

    set<int> restrictedValues;

    for(int i = 0; i < size; i++)
    {
        if(board[i][colIdx] != 0)
        {
            restrictedValues.insert(board[i][colIdx]);
        }
    }

    for(int i = 0; i < size; i++)
    {
        if(board[rowIdx][i] != 0)
        {
            restrictedValues.insert(board[rowIdx][i]);
        }
    }
```

```
    for(int i = 1; i <= size; i++)
    {
        if(restrictedValues.find(i) == restrictedValues.end())
        {
            possibleValues.push_back(i);
        }
    }

    return possibleValues;
}

int main()
{
    int N, Q, x, y;
    string rowData;
    cin >> N;

    if(N < 1 || N > 9)
    {
        return 0;
    }

    vector<vector<int>> board(N, vector<int>(N));
    for(int i = 0; i < N; i++)
```

```
    for(int i = 0; i < N; i++)
    {
        cin >> rowData;

        if(rowData.length() != N)
        {
            return 0;
        }

        for(int j = 0; j < N; j++)
        {
            board[i][j] = rowData[j] - '0';

            if(board[i][j] < 0 || board[i][j] > 9)
            {
                return 0;
            }
        }
    }

    cin >> Q;

    if(Q < 1 || Q > 1000)
    {
        return 0;
    }

    vector<pair<int, int>> queries;
    for(int i = 0; i < Q; i++)
    {
        cin >> x >> y;
        queries.push_back({x - 1, y - 1});
    }

    for(const auto& query : queries)
    {
        int row = query.first;
        int col = query.second;

        vector<int> possibleValuesList = findPossibleValues(board, row, col, N);

        cout << possibleValuesList.size() << endl;

        for(int val : possibleValuesList)
        {
            cout << val << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Task 8

Class Practice Task

Задача №1 – Запис текстової стрічки у файл із заданим ім’ям
Задача №2 – Копіювання вмісту файла у інший файл

C: > cpp > practice_work_team_tasks_solomia_liashchuk.cpp > copy_file(const string &, const string &)

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

enum FileOpResult{Success, Failure};
FileOpResult write_to_file(const string& content)
{

ofstream outfile("nameFile.txt");
if(!outfile.is_open())
{
    cout << "Error: Unable to creat a file";
    return FileOpResult::Failure;
}

outfile << content;
outfile.close();

ifstream infile("nameFile.txt");
if(!infile.is_open())
{
    cout << "Error: Unable to read the file";
    return FileOpResult::Failure;
}

char c;
while(infile.get(c))
{
    cout << c;
}

return FileOpResult::Success;
}
```

```
35
36  FileOpResult copy_file(const string& sourcePath, const string& destinationPath)
37  {
38
39  ifstream sourceFile(sourcePath);
40  if(!sourceFile.is_open())
41  {
42      cout << "Error: Unable to open a file";
43      return FileOpResult::Failure;
44  }
45
46  ofstream destinationFile(destinationPath);
47  if(!destinationFile.is_open())
48  {
49      cout << "Error: Unable to creat a file";
50      return FileOpResult::Failure;
51  }
52
53  string readLine;
54  while(getline(sourceFile, readLine))
55  {
56      destinationFile << readLine << '\n';
57
58  }
59
60  destinationFile.close();
61
62  ifstream readfile(destinationPath);
63  if(!readfile.is_open())
64  {
65      cout << "Error: Unable to craeat a file";
66      return FileOpResult::Failure;
67  }
68
```

```
68
69  char ch;
70  while(readfile.get(ch))
71  {
72      cout << ch;
73  }
74
75  cout << "Content of the file " << destinationPath << ":\n";
76
77  readfile.close();
78  sourceFile.close();
79
80  cout << "Content copied from " << sourcePath << " to " << destinationPath << "\n";
81  return FileOpResult::Success;
82
83  }
84
```

```
85
86  int main()
87  {
88      string content;
89      cout << "Enter content to write to the file: ";
90      getline(cin, content);
91
92      FileOpResult result = write_to_file(content);
93      if (result == FileOpResult::Success)
94      {
95          cout << "\nFile operation completed successfully.\n";
96      } else
97      {
98          cout << "\nFile operation failed.\n";
99      }
100
101     FileOpResult copyResult = copy_file("nameFile.txt", "fileCopy.txt");
102     if (copyResult == FileOpResult::Success)
103     {
104     std::cout << "File copied successfully.\n";
105     } else
106     {
107     std::cerr << "File copy failed.\n";
108     }
109
110     return 0;
111
112 }
113
```

```
Enter content to write to the file: Hello! It`s my first try to creat, copy and read file:)
Hello! It`s my first try to creat, copy and read file:)
File operation completed successfully.
Hello! It`s my first try to creat, copy and read file:)
Content of the file fileCopy.txt:
Content copied from nameFile.txt to fileCopy.txt
File copied successfully.
PS C:\Users\olesi>
```

Task 9

Self practice

Найбільша зростаюча підпослідовність

Вам задано послідовність із n цілих чисел a. Ваша задача — знайти довжину найбільшої зростаючої підпослідовності заданої послідовності.

cpp > Epicu > ai_programming_playground_2024 > ai_12 > solomia_liashchuk > epic_5 > practice

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{

    short numsnum;
    cin >> numsnum;
    vector<int> LIS(numsnum, 1);
    vector<int> nums(numsnum);

    for(int i = 0; i < numsnum; i++)
        cin >> nums[i];

    for(int i = numsnum - 1; i >= 0; i--)
    {
        for(int j = i; j < numsnum; j++)
        {
            if(nums[i] < nums[j])
            LIS[i] = max(LIS[i], 1 + LIS[j]);
        }
    }
    auto result = max_element(LIS.begin(), LIS.end());
    cout << *result;
    return 0;
}
```

Робота в команді:

Висновок: в цьому епіку я навчилася як працювати з файлами, вивчила нові алгоритми та самостійно спробувала на практиці.