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

Виконав:
Студент групи ШІ-12
Іваник Тарас

Тема роботи: Файлова система в C++. Робота з бінарними файлами та текстовими файлами, маніпуляції символами й рядковими змінними, як типу std::string, так і char*. Ознайомлення з можливостями стандартної бібліотеки C++ для роботи з файлами та створенням власних бібліотек для розширення функціональності.

Мета роботи: Навчитися працювати з файлами на мові C++: зчитування та запис даних у бінарні й текстові файли. Попрактикуватися з рядковими змінними різних типів, вивчити використання стандартних методів та функцій для маніпуляцій з ними. Дослідити основи створення та застосування власних бібліотек для зручності повторного використання коду й розширення можливостей стандартної бібліотеки C++.

Джерела:

-Blogan (YouTube)
-Acode
-GeekForGeeks
-cppreference
-cplusplus
-University lectures

Виконання роботи:

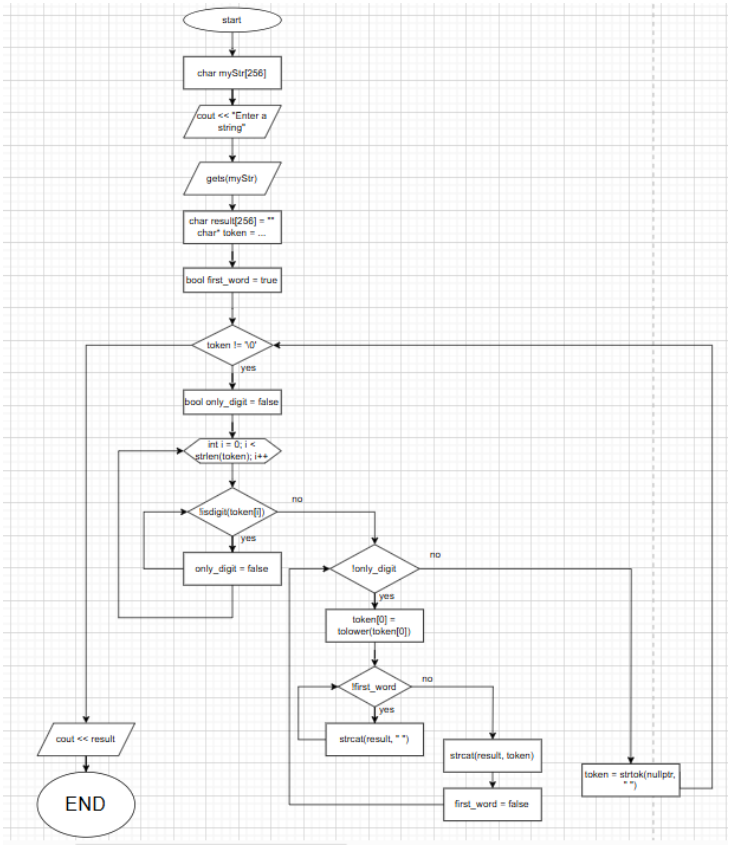
Lab# programming: VNS Lab 6 [70 хв]

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.

8. Перетворити рядок так, щоб всі слова в ньому стали ідентифікаторами, слова які складаються тільки із цифр - знищити.

```cpp
#include <iostream>
#include <stdio.h>
#include <string.h>
#include <cctype>

using namespace std;

int main() {
    char myStr[256];
    cout << "Enter a string : ";
    gets(myStr);

    char result[256] = "";
    char* token = strtok(myStr, " ");

    bool first_word = true;

    while (token != nullptr) {
        bool only_digit = true;

        for (int i = 0; i < strlen(token); i++) {
            if (!isdigit(token[i])) {
                only_digit = false;
                break;
            }
        }
        if (!only_digit) {
            token[0] = tolower(token[0]);
            if (!first_word) {
                strcat(result, " "); // додаємо пробіли
            }
            strcat(result, token);
            first_word = false;
        }
        token = strtok(nullptr, " "); //перехід до наступного слова
    }
    cout << result << endl;
    return 0;
}
```

```
Enter a string : somebody1 that 22 i used t00 know 732 svarga
somebody1 that i used t00 know svarga
PS C:\Users\User>
```



Lab# programming: VNS Lab 8 [200 хв]

Сформувати двійковий файл із елементів, заданої у варіанті структури, роздрукувати його вмістиме, виконати знищення й додавання елементів у відповідності зі своїм варіантом, використовуючи для пошуку елементів що знищуються чи додаються, функцію. Формування, друк, додавання й знищення елементів оформити у вигляді функцій. Передбачити повідомлення про помилки при відкритті файлу й виконанні операцій вводу/виводу.

8. Структура "Покупець":
   - прізвище, ім'я, по батькові;
   - домашня адреса;
   - номер телефону;
   - номер кредитної картки.

Знищити 3 елементи з початку файлу, додати 3 елементи в кінець файлу.

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

struct Buyer{
    char surname[20];
    char name[20];
    char middle_name[20];
    char address[100];
    char telephone_number[20];
    char card_number[20];
};

void create_binary_file(const string& myFile, const vector<Buyer>& buyers){
    ofstream outFile(myFile, ios::binary);
    if(!outFile){
        cout << " Error, can't open a file for writing " << endl;
        return;
    }
    for (const Buyer& buyer : buyers){
        outFile.write(reinterpret_cast<const char*>(&buyer), sizeof(Buyer));
    }
    outFile.close();
    cout << " File was createn and filled with data " << endl;
}

void print_binary_file(const string& myFile){
    ifstream inFile(myFile, ios::binary);
        if(!inFile){
            cout << " Error, can't open a file for reading " << endl;
            return;
        }
        Buyer buyer;
        while (inFile.read(reinterpret_cast<char*>(&buyer), sizeof(Buyer))){
            cout << "Surname: " << buyer.surname << ", Name: " << buyer.name << endl;
            cout << "Middle name: " << buyer.middle_name << endl;
            cout << "Address: " << buyer.address << endl;
            cout << "Telephone number: " << buyer.telephone_number << endl;
            cout << "Card number: " << buyer.card_number << endl;
        }
        inFile.close();
}

void delete_first_three(const string& myFile){
    ifstream inFile(myFile, ios::binary);
    if(!inFile){
        cout << " Error, can't open a file for reading " << endl;
        return;
    }
    vector<Buyer> buyers;
    Buyer buyer;
    int count = 0;

    while (inFile.read(reinterpret_cast<char*>(&buyer), sizeof(Buyer))){
        if(count >= 3){
            buyers.push_back(buyer);
        }
        count++;
    }
    inFile.close();

    ofstream outFile(myFile, ios::binary | ios::trunc);
    if(!outFile){
        cout << " Error, can't open a file for reading " << endl;
        return;
    }
    for (const Buyer& buyer : buyers){
        outFile.write(reinterpret_cast<const char*>(&buyer), sizeof(Buyer));
    }
    outFile.close();
    cout << "First 3 elements were deleted from file " << endl;
}

void three_to_end(const string& myFile, const vector<Buyer>& newBuyers){
    ofstream outFile(myFile, ios::binary | ios::app);
    if(!outFile){
        cout << " Error, can't open a file for reading " << endl;
        return;
    }
    for(const Buyer& buyer : newBuyers){
        outFile.write(reinterpret_cast<const char*>(&buyer), sizeof(Buyer));
    }
    outFile.close();
    cout << "3 elements were added to file " << endl;
}

int main(){
    string myFile = "Buyers_Information";
```

```cpp
int main(){
    string myFile = "Buyers_Information";

    vector<Buyer> initial_buyers = {
        {"Ivanyk", "Taras", "Yurijovych", "Videnska street 25", "+380-85-465-46-57", "1111-2222-3333-4444"},
        {"Iryna", "Svyrydenko", "Serhijivna", "Videnska street 25", "+380-47-895-09-00", "1451-7822-6983-5604"},
        {"Shyika", "Stefan", "Petrovych", "Konovaltsa street 39", "+380-25-111-22-33", "2233-3778-0003-1423"},
        {"Bobr", "Bobriv", "Bobrovych", "Kohucha street 106", "+380-77-000-44-55", "5566-8764-0987-4096"},
        {"Bilyk", "Pavlo", "Ivanovych", "Pshenychnoho street 87", "+380-66-888-00-22", "3344-4646-0000-9999"},
        {"Kryvychko", "Nazar" ,"Bibkovych" ,"Fariona street 42", "+380-90-880-23-00", "9090-1010-3234-8888"},
    };

    create_binary_file(myFile, initial_buyers);

    cout << "Original file contents " << endl;
    print_binary_file(myFile);

    delete_first_three(myFile);

    vector<Buyer> newBuyers = {
        {"Sementov", "Luka", "Sementovych", "Bohuna street 16", "+48-098-563-46-57", "5555-6666-7777-8888"},
        {"Jov", "Ivan", "Velikovich", "Oksany street 6", "+889-435-436-45-65", "1000-0000-9897-0088"},
        {"Kil", "Hir", "Potapovych", "Michnovskoho street 194", "+380-98-732-36-46", "2435-8374-3782-0000"},
    };
    three_to_end(myFile, newBuyers);

    cout << "New file contents: " << endl;
    print_binary_file(myFile);

    return 0;
}
```

```
****************************************
File was createn and filled with data
Original file contents:

Surname: Ivanyk, Name: Taras, Middle name: Yurijovych
Address: Videnska street 25, Telephone number: +380-85-465-46-57, Card number: 1111-2222-3333-4444
------------------------
Surname: Iryna, Name: Svyrydenko, Middle name: Serhijivna
Address: Videnska street 25, Telephone number: +380-47-895-09-00, Card number: 1451-7822-6983-5604
------------------------
Surname: Shyika, Name: Stefan, Middle name: Petrovych
Address: Konovaltsa street 39, Telephone number: +380-25-111-22-33, Card number: 2233-3778-0003-1423
------------------------
Surname: Bobr, Name: Bobriv, Middle name: Bobrovych
Address: Kohucha street 106, Telephone number: +380-77-000-44-55, Card number: 5566-8764-0987-4096
------------------------
Surname: Bilyk, Name: Pavlo, Middle name: Ivanovych
Address: Pshenychnoho street 87, Telephone number: +380-66-888-00-22, Card number: 3344-4646-0000-9999
------------------------
Surname: Kryvychko, Name: Nazar, Middle name: Bibkovych
Address: Fariona street 42, Telephone number: +380-90-880-23-00, Card number: 9090-1010-3234-8888
------------------------
First 3 elements were deleted from file
3 elements were added to file
New file contents:

Surname: Bobr, Name: Bobriv, Middle name: Bobrovych
Address: Kohucha street 106, Telephone number: +380-77-000-44-55, Card number: 5566-8764-0987-4096
------------------------
Surname: Bilyk, Name: Pavlo, Middle name: Ivanovych
Address: Pshenychnoho street 87, Telephone number: +380-66-888-00-22, Card number: 3344-4646-0000-9999
------------------------
Surname: Kryvychko, Name: Nazar, Middle name: Bibkovych
Address: Fariona street 42, Telephone number: +380-90-880-23-00, Card number: 9090-1010-3234-8888
------------------------
Surname: Sementov, Name: Luka, Middle name: Sementovych
Address: Bohuna street 16, Telephone number: +48-098-563-46-57, Card number: 5555-6666-7777-8888
------------------------
Surname: Jov, Name: Ivan, Middle name: Velikovich
------------------------
Surname: Bilyk, Name: Pavlo, Middle name: Ivanovych
Address: Pshenychnoho street 87, Telephone number: +380-66-888-00-22, Card number: 3344-4646-0000-9999
------------------------
Surname: Kryvychko, Name: Nazar, Middle name: Bibkovych
Address: Fariona street 42, Telephone number: +380-90-880-23-00, Card number: 9090-1010-3234-8888
------------------------
Surname: Sementov, Name: Luka, Middle name: Sementovych
Address: Bohuna street 16, Telephone number: +48-098-563-46-57, Card number: 5555-6666-7777-8888
------------------------
Surname: Jov, Name: Ivan, Middle name: Velikovich
Address: Oksany street 6, Telephone number: +889-435-436-45-65, Card number: 1000-0000-9897-0088
------------------------
Surname: Kil, Name: Hir, Middle name: Potapovych
Address: Michnovskoho street 194, Telephone number: +380-98-732-36-46, Card number: 2435-8374-3782-0000
------------------------
```

Lab# programming: VNS Lab 9 [180 хв]

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

Виконати завдання.

8.

1) Скопіювати з файлу F1 у файл F2 всі рядки, які не містять цифри.

2) Підрахувати кількість рядків, які починаються на букву «А» у файлі F2.

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

bool is_digit(const string& line) {
    for (char ch : line) {
        if (isdigit(ch)) {
            return true;
        }
    }
    return false;
}

bool start_with_A(const string& line) {
    return !line.empty() && line[0] == 'A';
}

int main() {
    const string filename1 = "F1.txt";
    const string filename2 = "F2.txt";

    ofstream file1(filename1);
    if (!file1) {
        cerr << "Error: file1 не вдалося створити!" << endl;
        return 1;
    }
    file1 << "Apple is a fruit.\n";
    file1 << "123456789\n";
    file1 << "Another line starts with A.\n";
    file1 << "No digits here.\n";
    file1 << "Amazing!\n";
    file1 << "What do you eat for breakfast?.\n";
    file1 << "T0day is a good day!\n";
    file1 << "America amerikano z mlekiem\n";
    file1 << "Bruuuuuuuuuu12uuh\n";
    file1 << "A last se n t e n c e\n";
    file1.close();

    ifstream inputFile(filename1);
    if (!inputFile) {
        cerr << "Error: file1 не вдалося відкрити для читання!" << endl;
        return 1;
    }

    ofstream outputFile(filename2);
    if (!outputFile) {
        cerr << "Error: file2 не вдалося відкрити для запису!" << endl;
        return 1;
    }

    string line;
    int count_A = 0;

    while (getline(inputFile, line)) {
        if (!is_digit(line)) {
            outputFile << line << endl;
            if (start_with_A(line)) {
                count_A++;
            }
        }
    }
    outputFile << "Number of sentences that start with 'A': " << count_A << endl;

    inputFile.close();
    outputFile.close();

    ifstream readFile(filename2);
    if (!readFile) {
        cerr << "Error: file2 не вдалося відкрити для читання!" << endl;
        return 1;
    }
    cout << "Contents of F2:" << endl;
    while (getline(readFile, line)) {
        cout << line << endl;
    }
    readFile.close();

    return 0;
}
```

```
Contents of F2:
Apple is a fruit.
Another line starts with A.
No digits here.
Amazing!
What do you eat for breakfast?.
America amerikano z mlekiem
A last se n t e n c e
Number of sentences that start with 'A': 5
```

F2: Блокнот

Файл Редагування Формат Вигляд Довідка

```
Apple is a fruit.
Another line starts with A.
No digits here.
Amazing!
What do you eat for breakfast?.
America amerikano z mlekiem
A last se n t e n c e
Number of sentences that start with 'A': 5
```

Lab# programming: Algotester Lab 4 (4.1) [180 хв]

Lab 4v1

Limits: 1 sec., 256 MiB

Вам дано 2 цілих чисел масиви, розміром N та M.

Ваше завдання вивести:

1. Різницю N-M

2. Різницю M-N

3. Їх перетин

4. Їх обєднання

5. Їх симетричну різницю

Input

У першому рядку ціле число N - розмір масиву 1

У другому рядку N цілих чисел - елементи масиву 1

У третьому рядку ціле число M - розмір масиву 2

У четвертом рядку M цілих чисел - елементи масиву 2

Output

Вивести результат виконання 5 вищезазначених операцій у форматі:

У першому рядку ціле число N - розмір множини

У наступному рядку N цілих чисел - посортована у порядку зростання множина

Constraints

$1 \leq N, M \leq 100$

$1 \leq n_i, m_i \leq 100$

STL:

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <unordered_set> // використовується для зберігання унікальних значень без визначеного порядку
using namespace std;
// M - N
vector<int> differenceM_N(const vector<int>& n, const vector<int>& m) {

    vector<int> result;
    set_difference(n.begin(),n.end(),m.begin(), m.end(), back_inserter(result));
    return result;
}
// N - M
vector<int> differenceN_M(const vector<int>& m, const vector<int>& n) {

    vector<int> result;
    set_difference(m.begin(),m.end(),n.begin(), n.end(), back_inserter(result));
    return result;
}
// N / M
vector<int> intersection(const vector<int>& n, const vector<int>& m){

    vector<int> result;
    set_intersection(n.begin(),n.end(),m.begin(), m.end(), back_inserter(result));
    return result;
}
// N + M (union)
vector<int> unionN_M(const vector<int>& n, const vector<int>& m){
    vector<int> result;
    set_union(n.begin(),n.end(),m.begin(), m.end(), back_inserter(result));
    return result;
}
// -N - M (simetric difference)
vector<int> simetric_difference(const vector<int>& n, const vector<int>& m){
    vector<int> result;
    set_symmetric_difference(n.begin(),n.end(),m.begin(), m.end(), back_inserter(result));
    return result;
}
int main(){
    int N,M;

    cin >> N;
    vector<int> n(N);
    for (int i = 0; i < N; ++i){
        cin >> n[i];
    }
    cin >> M;
    vector<int> m(M);
    for (int i = 0; i < M; ++i){
        cin >> m[i];
    }
    sort(n.begin(), n.end());
    sort(m.begin(), m.end());
    vector<int> difM_N = differenceM_N(n, m);
    vector<int> difN_M = differenceN_M(m, n);
    vector<int> intersectionN_M = intersection(n, m);
```

```cpp
int main(){
    vector<int> intersectionN_M = intersection(n, m);
    vector<int> union____NM = unionN_M(n, m);
    vector<int> simDif = simetric_difference(n, m);

    cout << difM_N.size() << endl;  //кількість елементів
    for (int num : difM_N) {
        cout << num << " ";
    }
    cout << endl;
    cout << endl;

// N - M (output)
    cout << difN_M.size() << endl;
    for (int num : difN_M) {
        cout << num << " ";
    }
    cout << endl;
    cout << endl;
// N / M (output)
    cout << intersectionN_M.size() << endl;
    for (int num : intersectionN_M) {
        cout << num << " ";
    }
    cout << endl;
    cout << endl;
// N + M (output)
    cout << union____NM.size() << endl;
    for (int num : union____NM) {
        cout << num << " ";
    }
    cout << endl;
    cout << endl;
// -N - M (output)
    cout << simDif.size() << endl;
    for (int num : simDif) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

```
5
1 2 3 4 5
5
4 5 6 7 8
3
1 2 3

3
6 7 8

2
4 5

8
1 2 3 4 5 6 7 8

6
1 2 3 6 7 8
```

a day ago	C++ 23	Accepted	0.003	1.234	View

Своя реалізація:

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <unordered_set> // використовується для зберігання унікальних значень без визначеного порядку
using namespace std;
// M - N
vector<int> differenceM_N(const vector<int>& n, const vector<int>& m) {
    unordered_set<int> m_elements(m.begin(), m.end());  // Створюємо множину елементів з m
    vector<int> result;

    for (int num : n) {
        if (m_elements.find(num) == m_elements.end()) {
            result.push_back(num);  // Додаємо елемент, якщо його немає в m
        }
    }
    return result;
}
// N - M
vector<int> differenceN_M(const vector<int>& m, const vector<int>& n) {
    unordered_set<int> n_elements(n.begin(), n.end());
    vector<int> result;

    for (int num : m) {
        if (n_elements.find(num) == n_elements.end()) {
            result.push_back(num);
        }
    }
    return result;
}
// N / M
vector<int> intersection(const vector<int>& n, const vector<int>& m){
    unordered_set<int> n_elements(n.begin(), n.end());
    vector<int> result;

    for (int num : m){
        if(n_elements.find(num) != n_elements.end()){
            result.push_back(num);
        }
    }
    return result;
}
// N + M (union)
vector<int> unionN_M(const vector<int>& n, const vector<int>& m){
    unordered_set<int> elements;

    elements.insert(n.begin(), n.end());
    elements.insert(m.begin(), m.end());

    vector<int> result(elements.begin(), elements.end()); // додаємо унікальні елементи
    sort(result.begin(), result.end());
    return result;
}
// -N - M (simetric difference)
vector<int> simetric_difference(const vector<int>& n, const vector<int>& m){
    unordered_set<int> n_elements(n.begin(), n.end());
    unordered_set<int> m_elements(m.begin(), m.end());
    vector<int> result;

    for (int num : m){
        if(n_elements.find(num) == n_elements.end()){
            result.push_back(num);
        }
    }

    for (int num : n){
        if(m_elements.find(num) == m_elements.end()){
            result.push_back(num);
        }
    }
    sort(result.begin(), result.end());

    return result;
}

int main(){
    int N,M;

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

    vector<int> difM_N = differenceM_N(n, m);
    vector<int> difN_M = differenceN_M(m, n);
    vector<int> intersectionN_M = intersection(n, m);
    vector<int> union____NM = unionN_M(n, m);
    vector<int> simDif = simetric_difference(n, m);

// M - N (output)
    cout << difM_N.size() << endl;  //кількість елементів
    for (int num : difM_N) {
        cout << num << " ";
    }
    cout << endl;
    cout << endl;
// N - M (output)
    cout << difN_M.size() << endl;
    for (int num : difN_M) {
        cout << num << " ";
    }
    cout << endl;
    cout << endl;
// N / M (output)
    cout << intersectionN_M.size() << endl;
    for (int num : intersectionN_M) {
        cout << num << " ";
    }
    cout << endl;
    cout << endl;
// N + M (output)
    cout << union____NM.size() << endl;
    for (int num : union____NM) {
        cout << num << " ";
    }
    cout << endl;
    cout << endl;
// -N - M (output)
    cout << simDif.size() << endl;
    for (int num : simDif) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

```
5
1 2 3 4 5
5
4 5 6 7 8
3
1 2 3

3
6 7 8

2
4 5

8
1 2 3 4 5 6 7 8

6
1 2 3 6 7 8
```

Lab# programming: Algotester Lab 6v3 (400 хв)

Lab 6v3

Limits: 1 sec., 256 MiB

У Клінта в черговий раз виключилось світло і йому немає чим зайнятися. Так як навіть це не заставить його подивитися збережені відео про програмування на ютубі - він вирішив придумати свою гру на основі судоку.

Гра виглядає так:

Є поле розміром $N \times N$, в якому частина клітинок заповнена цифрами, а частина клітинок пусті (позначаються нулем). Також у нього є Q пар координат X та Y.

Завданням гри є написати до кожної координати скільки чисел туди можна вписати (якщо вона пуста) і які це числа (обов'язково в посортовані по зростанню!). В клітинку можна вписати лише ті числа, які не зустрічаються в рядку та стовбці, які перетинаються у цій клітинці.

Під час гри поле не міняється!

Також необовязково, щоб це було валідне судоку! Якщо є клітинка, в яку не можна вписати ніяку цифру - виведіть 0.

Також допускаються рядки та стовпці, в яких цифра записана кілька разів.

Input

У першому рядку ціле число N - розмір поля для гри

У N наступних рядках стрічка row_i яка складається з N цифер - i-й рядок.

Ціле число Q - кількість запитань

У наступних Q рядках 2 цілих числа x_j, y_j - координати клітинок j-го запитання

Output

Q разів відповідь у наступному форматі:

Натуральне число M - кількість цифр, які можна вписати в клітинку

M цифер розділених пробілом - можливі цифри

Constraints

$1 \le N \le 9$

$|row_i| = N$

$row_i \in 1..9$

$1 \le Q \le 1000$

$1 \le x, y \le N$

17 hours ago	Lab 6v3 - Lab 6v3	C++ 23	Accepted	0.003	1.434	1864278

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

int main(){
    int N;
    cin >> N;
    vector<vector<int>> sudoku(N);
    string str;
    for(int i = 0; i < N; i++){
        cin >> str;
        for(const char& ch : str) sudoku[i].push_back(ch - '0');
    }
    int Q;
    cin >> Q;
    vector<vector<int>> results(Q);
    for(int i = 0; i < Q; i++){
        int x, y;
        cin >> x >> y;
        x--;
        y--;
        vector<int> vec(N);
        if(sudoku[x][y] == 0){
            for(int j = 0; j < N; j++){
                if(j == y || sudoku[x][j] == 0) continue;
                vec[sudoku[x][j] - 1] = sudoku[x][j];
            }
            for(int j = 0; j < N; j++){
                if(j == x || sudoku[j][y] == 0) continue;
                vec[sudoku[j][y] - 1] = sudoku[j][y];
            }

            for(int j = 0; j < N; j++){
                if(vec[j] != j + 1) results[i].push_back(j + 1);
            }
        }
        else results[i].push_back(sudoku[x][y]);
    }
    for(vector<int>& v : results){
        cout << v.size() << endl;
        for(int& val : v) cout << val << " ";
        cout << endl << endl;
    }
    return 0;
}
```

```
9
123456789
000000000
010000000
030000000
040000000
050000000
060000000
070000000
987643215
3
1 1
2 2
8 8
1
1

1
9

6
2 3 4 5 6 9
```

Practice# programming: Practice Work Team Task (3-4 год)

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

enum FileOpResult {
    Success = 0,
    Failure = 1,
};
static map<FileOpResult, string> OperationStr = {
    { Success, "Success" },
    { Failure, "Failure" } // для виведення результатів
};
bool sameEnd(const string& s_full, const string& s_end) {
    if (s_full.length() >= s_end.length()) {
        return (0 == s_full.compare(s_full.length() - s_end.length(), s_end.length(), s_end));
    } else { // перевірка чи s_full = s_end
        return false;
    }
}
FileOpResult write(const char* name, const char* content) {
    if (name == nullptr || strcmp(name, "") == 0 || !sameEnd(name, ".txt")) {
        name = "default.txt"; // якщо ім'я порожнє або не закінчується на ".txt", то йде далі
    }
    ofstream f_out(name, ios::out | ios::trunc);
    if (!f_out.is_open()) {
        return FileOpResult::Failure;
    }
    f_out << content;
    if (!f_out.good()) { // Check if writing to file was successful
        return FileOpResult::Failure;
    }
    f_out.close();
    return f_out.fail() ? FileOpResult::Failure : FileOpResult::Success;
}
unsigned int FileRead(istream& is, vector<char>& buff) {
    is.read(&buff[0], buff.size());
    return is.gcount();
}
void FileRead(ifstream& f_in, string& s) {
    const unsigned int BUFSIZE = 64 * 1024;
    vector<char> buffer(BUFSIZE);

    while (unsigned int n = FileRead(f_in, buffer)) {
        s.append(&buffer[0], n);
    }
}
FileOpResult copy(const char* file_from, const char* file_to) {
    if (file_from == nullptr || strcmp(file_from, "") == 0 ||
        file_to == nullptr || strcmp(file_to, "") == 0 ||
        !sameEnd(file_from, ".txt") || !sameEnd(file_to, ".txt")) {
        return FileOpResult::Failure;
    }
    ifstream f_in(file_from, ios::in);
    if (!f_in.is_open()) {
        return FileOpResult::Failure;
    }
    ofstream f_out(file_to, ios::out | ios::trunc);
    if (!f_out.is_open()) {
        f_in.close();
        return FileOpResult::Failure;
    }
    string fileContent;
    FileRead(f_in, fileContent);
    f_out << fileContent;
    bool failure = f_in.bad() || f_out.bad();

    f_in.close();
    f_out.close();
    return failure ? FileOpResult::Failure : FileOpResult::Success;
}
int main() {

    string contentLine, fileName;
    cout << "Enter content: ";
    getline(cin, contentLine);
    cout << "Enter file name: (*.txt) (default name = \"default.txt\"): ";
    getline(cin, fileName);
    FileOpResult writeResult = write(fileName.c_str(), contentLine.c_str());
    cout << "Operation Result: " << OperationStr[writeResult] << endl;

    string fromFile, toFile;
    cout << "Enter file name {from}: ";
    getline(cin, fromFile);
    cout << "Enter file name {to}: ";
    getline(cin, toFile);
    FileOpResult copyResult = copy(fromFile.c_str(), toFile.c_str());
    cout << "Copy Operation Result: " << OperationStr[copyResult] << endl;

    return 0;
}
```

Practice# programming: Self Practice Task(1 год)

Коля, Вася і Теніс

Limits: 2 sec., 256 MiB

Коли Коля та Вася прийшли робити ремонт на «Екстралогіку» — першим, що вони побачили в офісі, був стіл для настільного тенісу. Поки всі інші працювали, Коля та Вася вирішили пограти. Через декілька годин прийшов директор і накричав на заробітчан через те, що вони нічим не займаються. Тож Вася і Коля мусили йти працювати.

По дорозі вони сперечалися, хто ж виграв і з яким рахунком. Оскільки вони записували результати кожної подачі, то це можна порахувати. Але оскільки гра тривала дуже довго — порахувати це вручну дуже тяжко.

Всього відбулося n подач. Про кожну з них ми знаємо, хто переміг. За виграну подачу гравець отримує одне очко. Партія вважається виграною, коли один з гравців набере не менше одинадцяти очок з перевагою щонайменше у два очки. Наприклад, за рахунків 11:9, 4:11, 15:13 партія закінчується, а за рахунків 11:10 та 99:98 — ні. Як тільки Коля і Вася закінчили одну партію — вони починають іншу.

Знаючи, хто переміг кожної подачі — виведіть загальний рахунок по партіях в грі Коля-Вася. А якщо вони не догради останню партію, то і її рахунок теж.

Input

У першому рядку задано ціле число n — загальна кількість подач.

У другому рядку задано n символів c_i. $c_i = \texttt{K}$, якщо i-ту подачу виграв Коля, та $c_i = \texttt{V}$, якщо i-ту подачу виграв Вася.

Output

У першому рядку виведіть загальний рахунок гри по партіях у форматі $k : v$, де k — кількість партій, у яких переміг Коля, а v — кількість партій, у яких переміг Вася.

Якщо вони не догради останню партію, то в другому рядку в такому ж форматі виведіть рахунок останньої партії.

Constraints

30% тестів: $1 \le n \le 10^4$

70% тестів: $1 \le n \le 10^5$

Samples

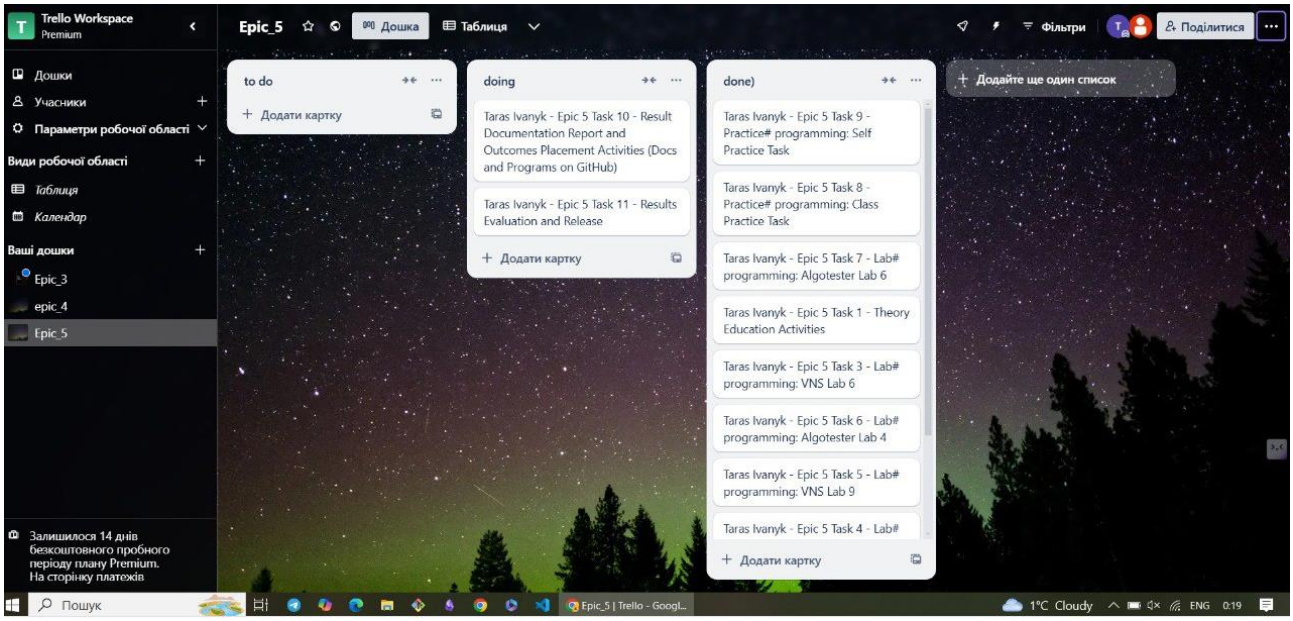
Input (stdin)	copy	Output (stdout)
30 VVKVKKVVVVKVKKKKKVVKKVKKKKVVVV		1:0 2:4

Accepted	0.008	1.246	1864621

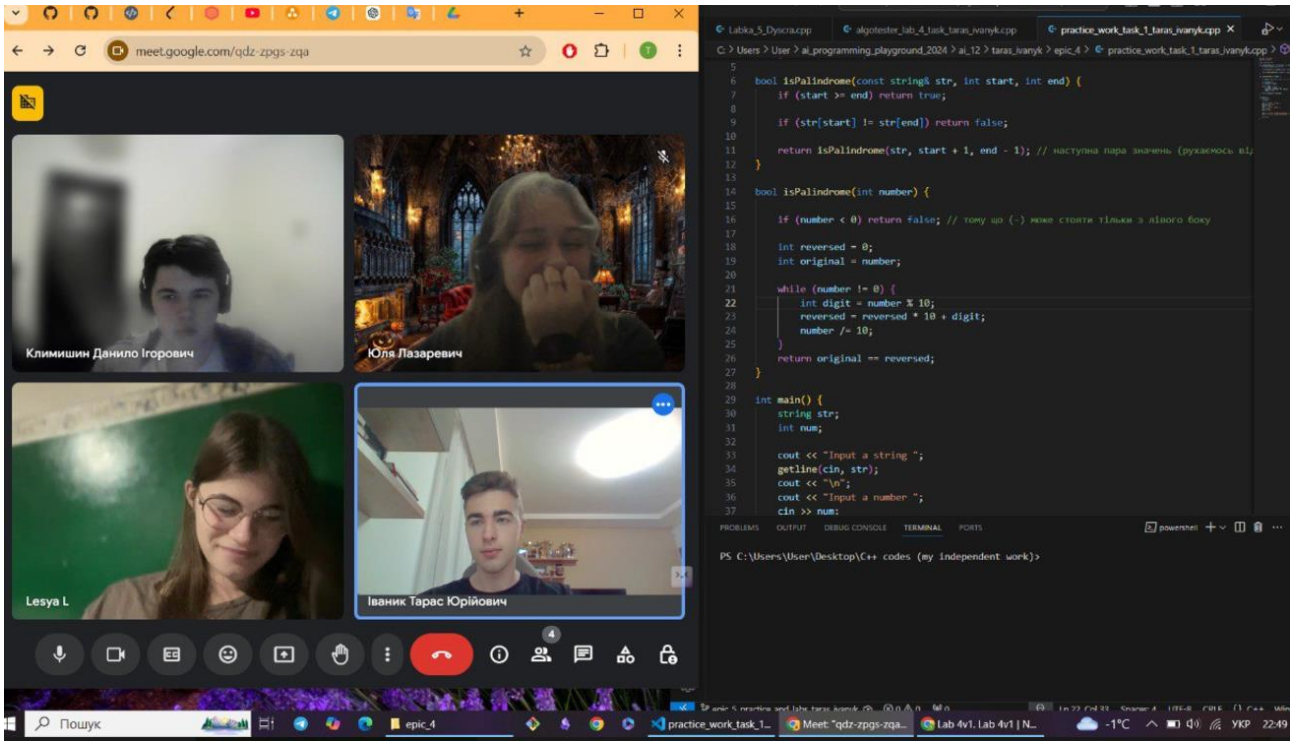
```cpp
#include <iostream>
#include <vector>
using namespace std;

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

    if(N < 1 || N > 100000){
        cout << "Error" << endl;
        return 1;
    }

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

    int count_K = 0, count_V = 0;
    int count_K_score = 0, count_V_score = 0;

    for(int i = 0; i < N; ++i){
        if (strings[i] == 'K'){
            count_K++;
        } else if (strings[i] == 'V'){
            count_V++;
        }

        if ((count_K >= 11 || count_V >= 11) && abs(count_K - count_V) >= 2) {
            if (count_K > count_V) {
                count_K_score++;
            } else {
                count_V_score++;
            }
            count_K = 0;
            count_V = 0;
        }
    }
    cout << count_K_score << ":" << count_V_score << endl;

    if (count_K > 0 || count_V > 0){
        cout << count_K << ":" << count_V;
    }
    return 0;
}
```

```
30
VVKVKKVVVVKVKKKKKVVKKVKKKKVVVV
1:0
2:4
PS C:\Users\User>
```

TRELLO:



Team meet:



[Pull Request]

Висновок: Під час виконання 5 епіку я зрозумів як працювати з файлами на мові C++, зокрема як їх відкривати-закривати, як вносити зміни, як читати файл, або щось в нього записувати. Зрозумів як

обробляти текстові і бінарні файли. Детальніше вивчив бібліотеку <iostream>, навчився використовувати деякі маніпулятори, які вже є закладені в бібліотеці <iostream>. Мав 1 зустріч з командою у форматі онлайн. Навчився використовувати Obsidian, програма для конспектів, тепер все якось більш структуровано.