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:

Практичних Робіт до блоку № 5

Виконав:

Студент групи ШІ-11

Яровой Павло Олегович

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

Мета роботи: Ознайомитися з принципами роботи з файлами в програмуванні, зокрема бінарними та текстовими файлами. Навчитися використовувати символи, рядкові змінні та текстові файли у програмах. Розглянути функціонал стандартної бібліотеки для роботи з файлами, а також методи створення та використання власних бібліотек для роботи з файлами.

Теоретичні відомості:

1)Перелік тем:

1. Вступ до Роботи з Файлами;
2. Символи і Рядкові Змінні;
3. Текстові Файли;
4. Бінарні Файли;
5. Стандартна бібліотека та робота з файлами;
6. Створення й використання бібліотек.

2)Індивідуальний план опрацювання теорії:

1. **Вступ до Роботи з Файлами**

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

2. **Символи і Рядкові Змінні**

   https://www.w3schools.com/cpp/cpp_files.asp

3. **Текстові Файли**

   https://www.geeksforgeeks.org/file-handling-c-classes/

4. **Бінарні Файли**

   https://www.geeksforgeeks.org/difference-between-cpp-text-file-and-binary-file/

5. **Стандартна бібліотека та робота з файлами**

- Unordered map: unordered_map in C++ STL - GeeksforGeeks

- Стандарті бібліотеки і їх виклик: https://en.cppreference.com/w/cpp/standard_library

6. **Створення й використання бібліотек**

   https://stackoverflow.com/questions/16693273/how-do-i-create-a-library

   https://www.geeksforgeeks.org/how-do-i-create-a-library-in-cpp/

Виконання роботи:

1)Перелік завдань:

- 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(варіант 20)

- John Black - Epic 5 Task 4 - Lab# programming: VNS Lab 8(варіант 20)

- John Black - Epic 5 Task 5 - Lab# programming: VNS Lab 9(варіант 20)

- John Black - Epic 5 Task 6 - Lab# programming: Algotester Lab 4(варіант 1)

- John Black - Epic 5 Task 7 - Lab# programming: Algotester Lab 6(варіант 2)

- 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

2)Умови завдань:

Task 3:

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.

20. Знайти для рядка довжину найдовшого слова.

Task 4:

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.

20. Структура "Держава":

- назва;
- державна мова;
- грошова одиниця;
- курс валюти відносно $.

Task 5:

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

20.

1) Скопіювати з файлу F1 у файл F2 всі рядки, у яких є однакові слова.

2) Визначити кількість голосних букв в останньому рядку файлу F2.

Task 6:

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

Task 7:

Lab 6v2

Limits: 2 sec., 256 MiB

У вас є шахова дошка розміром 8×8 та дуже багато фігур.

Кожна клітинка може мати таке значення:

- Пуста клітинка O
- Пішак P
- Тура R
- Кінь N
- Слон B
- Король K
- Королева Q

Вам дають позиції фігур на дошці (всі фігури одного кольору, кількість королів може бути > 1).

Далі йдуть Q запитів з координатами клітинки $\{x, y\}$. На кожен запит ви маєте вивести стрічку s_i - посортовані за алфавітом букви фігур, які атакують цю клітинку (пішаки атакують вниз).

У випадку, якщо на клітинці стоїть якась фігура - виведіть символ X.

У випадку, якщо клітинку не атакують - виведіть O.

Наявність фігури у певній клітинці не блокує атаку для іншої фігури. Тобто якщо між турою та клітинкою стоїть інша фігура - вважається що тура атакує цю клітинку.

Input

У перших 8 рядках стрічка row_i - стан i-го рядка дошки.

У наступному рядку ціле число Q - кількість записів

У наступних Q рядках 2 цілих числа x та y - координати клітинки

Output

Q разів відповідь у наступному форматі:

Строка $result$ - усі фігури, які атакують клітинку з запиту.

Task 8:

Задача №1 – Запис текстової стрічки у файл із заданим ім’ям

Реалізувати функцію створення файла і запису в нього даних:

enum FileOpResult { Success, Failure, … };

FileOpResult write_to_file(char *name, char *content);

Умови задачі:

- створити файл із заданим ім’ям; якщо файл існує – перезаписати його вміст
- написати код стійкий до різних варіантів вхідних параметрів
- name – ім’я, може не включати шлях
- записати у файл вміст стрічки content, прочитати content із стандартного вводу
- повернути статус операції: Success – все пройшло успішно, Failure – файл не вдалося створити, або збій операції відкриття, запису даних, чи закриття файла.

Задача №2 – Копіювання вмісту файла у інший файл

Реалізувати функцію створення файла і запису в нього даних:

enum FileOpResult { Success, Failure, … };

FileOpResult copy_file(char *file_from, char *file_to);

Умови задачі:

- копіювати вміст файла з ім’ям file_from у файл з ім’ям file_to; написати код стійкий до різних варіантів вхідних параметрів, обробити всі можливі варіанти відсутності одного з файлів
- file_from, file_to – можуть бути повним або відносним шляхом
- повернути статус операції: Success – все пройшло успішно, Failure – файл не вдалося створити, або збій операції відкриття, читання чи запису даних, закриття файла.

Task 9:

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

$1 \le N \le 9$

$|row_i| = N$

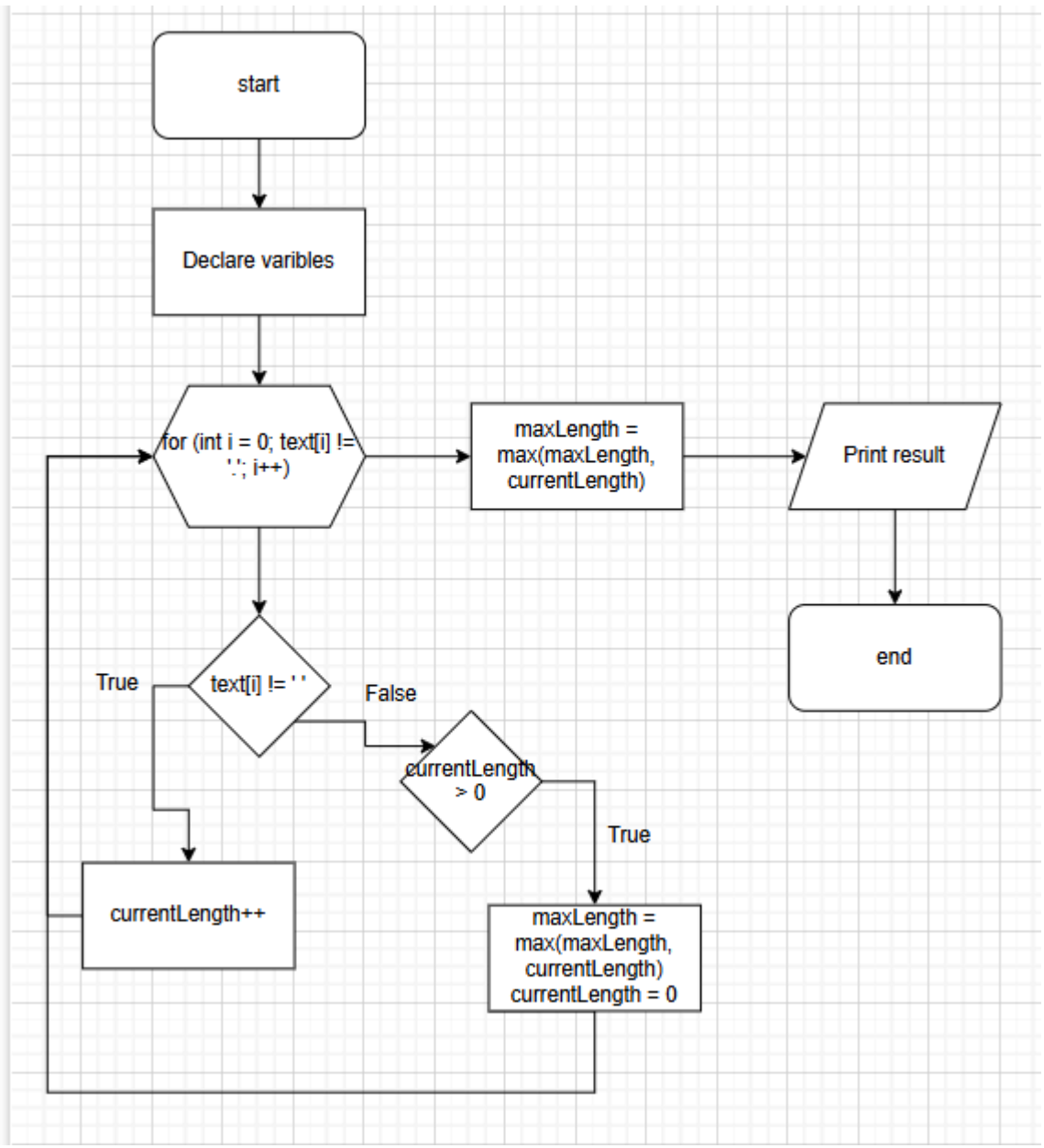
$row_i \in 1..9$

$1 \le Q \le 1000$

3)Дизайн програми

4) Код програм з посиланням на зовнішні ресурси:

Task 3 - Lab# programming: VNS Lab 6(варіант 20)


start
Declare varibles
for (int i = 0; text[i] != '.'; i++)
maxLength = max(maxLength, currentLength)
Print result
end
text[i] != ' '
True
False
currentLength > 0
True
currentLength++
maxLength = max(maxLength, currentLength) currentLength = 0

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

int main() {
    char text[256];
    cin.getline(text, 256);

    int maxLength = 0, currentLength = 0;
    for (int i = 0; text[i] != '.'; i++) {
        if (text[i] != ' ') {
            currentLength++;
        } else if (currentLength > 0) {
            maxLength = max(maxLength, currentLength);
            currentLength = 0;
        }
    }

    maxLength = max(maxLength, currentLength);

    cout << maxLength << endl;

    return 0;
}

```

Task 4 - Lab# programming: VNS Lab 8(варіант 20)

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

struct Country {
    char name[50];
    char language[50];
    char currency[50];
    double exchangeRate;
};

void createFile(const char* filename) {
    ofstream file(filename, ios::binary);
    if (!file) {
        cerr << "Error: Cannot create file." << endl;
        return;
    }

    Country countries[] = {
        {"USA", "English", "Dollar", 1.0},
        {"Japan", "Japanese", "Yen", 0.007},
        {"Germany", "German", "Euro", 1.1}
    };

    for (auto& country : countries) {
        file.write(reinterpret_cast<char*>(&country), sizeof(Country));
    }

    file.close();
}

void printFile(const char* filename) {
    ifstream file(filename, ios::binary);
    if (!file) {
        cerr << "Error: Cannot open file for reading." << endl;
        return;
    }

    Country country;
    while (file.read(reinterpret_cast<char*>(&country), sizeof(Country))) {
        cout << "Name: " << country.name
             << ", Language: " << country.language
             << ", Currency: " << country.currency
             << ", Exchange Rate: " << country.exchangeRate << endl;
```

```
46      }
47
48      file.close();
49  }
50
51  bool searchAndDelete(const char* filename, const char* targetName) {
52      ifstream file(filename, ios::binary);
53      if (!file) {
54          cerr << "Error: Cannot open file for reading." << endl;
55          return false;
56      }
57
58      ofstream tempFile("temp.bin", ios::binary);
59      if (!tempFile) {
60          cerr << "Error: Cannot create temporary file." << endl;
61          return false;
62      }
63
64      Country country;
65      bool found = false;
66
67      while (file.read(reinterpret_cast<char*>(&country), sizeof(Country))) {
68          if (strcmp(country.name, targetName) == 0) {
69              found = true;
70          } else {
71              tempFile.write(reinterpret_cast<char*>(&country), sizeof(Country));
72          }
73      }
74
75      file.close();
76      tempFile.close();
77
78      remove(filename);
79      rename("temp.bin", filename);
80
81      return found;
82  }
83
```

```cpp
void addCountry(const char* filename, const Country& newCountry) {
    ofstream file(filename, ios::binary | ios::app);
    if (!file) {
        cerr << "Error: Cannot open file for appending." << endl;
        return;
    }

    file.write(reinterpret_cast<const char*>(&newCountry), sizeof(Country));
    file.close();
}

int main() {
    const char* filename = "countries.bin";
    createFile(filename);

    cout << "Initial file contents:" << endl;
    printFile(filename);

    if (searchAndDelete(filename, "Germany")) {
        cout << "Country deleted successfully." << endl;
    } else {
        cout << "Country not found." << endl;
    }

    cout << "File contents after deletion:" << endl;
    printFile(filename);

    Country newCountry = {"India", "Hindi", "Rupee", 0.012};
    addCountry(filename, newCountry);

    cout << "File contents after addition:" << endl;
    printFile(filename);

    return 0;
}
```

Task 5 - Lab# programming: VNS Lab 9(варіант 20)

```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <set>
#include <cctype>
using namespace std;

bool hasDuplicateWords(const string& line) {
    istringstream stream(line);
    string word;
    set<string> words;
    while (stream >> word) {
        if (words.count(word)) {
            return true;
        }
        words.insert(word);
    }
    return false;
}

int countVowels(const string& line) {
    const string vowels = "AEIOUYaeiouy";
    int count = 0;
    for (char c : line) {
        if (vowels.find(c) != string::npos) {
            count++;
        }
    }
    return count;
}

int main() {
    const char* file1 = "F1.txt";
    const char* file2 = "F2.txt";

    ofstream f1(file1);
    if (!f1) {
        cerr << "Error: Cannot create F1." << endl;
        return 1;
    }
```

```
    cout << "Enter at least 10 lines for F1. Type 'END' to finish:" << endl;
    vector<string> lines;
    string inputLine;
    while (getline(cin, inputLine)) {
        if (inputLine == "END") break;
        lines.push_back(inputLine);
    }

    if (lines.size() < 10) {
        cerr << "Error: At least 10 lines are required." << endl;
        return 1;
    }

    for (const auto& line : lines) {
        f1 << line << endl;
    }
    f1.close();

    ifstream f1Input(file1);
    ofstream f2(file2);
    if (!f1Input || !f2) {
        cerr << "Error: Cannot open files." << endl;
        return 1;
    }

    string line;
    vector<string> f2Lines;
    while (getline(f1Input, line)) {
        if (hasDuplicateWords(line)) {
            f2 << line << endl;
            f2Lines.push_back(line);
        }
    }

    f1Input.close();
    f2.close();
```

```
    if (f2Lines.empty()) {
        cout << "F2 is empty, no duplicate words found." << endl;
    } else {
        string lastLine = f2Lines.back();
        int vowelCount = countVowels(lastLine);
        cout << "Number of vowels in the last line of F2: " << vowelCount << endl;
    }

    return 0;
}
```

Task 6 - Lab# programming: Algotester Lab 4(варіант 1)

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

void displayResults(const vector<int>& vec) {
    cout << vec.size() << "\n";
    for (auto elem : vec) {
        cout << elem << " ";
    }
    cout << "\n\n";
}

int main() {
    int size1, size2;
    cin >> size1;
    vector<int> list1(size1);
    for (auto& val : list1) {
        cin >> val;
    }

    cin >> size2;
    vector<int> list2(size2);
    for (auto& val : list2) {
        cin >> val;
    }

    sort(list1.begin(), list1.end());
    sort(list2.begin(), list2.end());

    vector<int> diff1, diff2, common, combined, symDiff;
    set_difference(list1.begin(), list1.end(), list2.begin(), list2.end(), back_inserter(diff1));
    set_difference(list2.begin(), list2.end(), list1.begin(), list1.end(), back_inserter(diff2));
    set_intersection(list1.begin(), list1.end(), list2.begin(), list2.end(), back_inserter(common));
    set_union(list1.begin(), list1.end(), list2.begin(), list2.end(), back_inserter(combined));
    set_symmetric_difference(list1.begin(), list1.end(), list2.begin(), list2.end(), back_inserter(symDiff));

    displayResults(diff1);
    displayResults(diff2);
    displayResults(common);
    displayResults(combined);
    displayResults(symDiff);
    return 0;
}
```

Task 7 - Lab# programming: Algotester Lab 6(варіант 2)

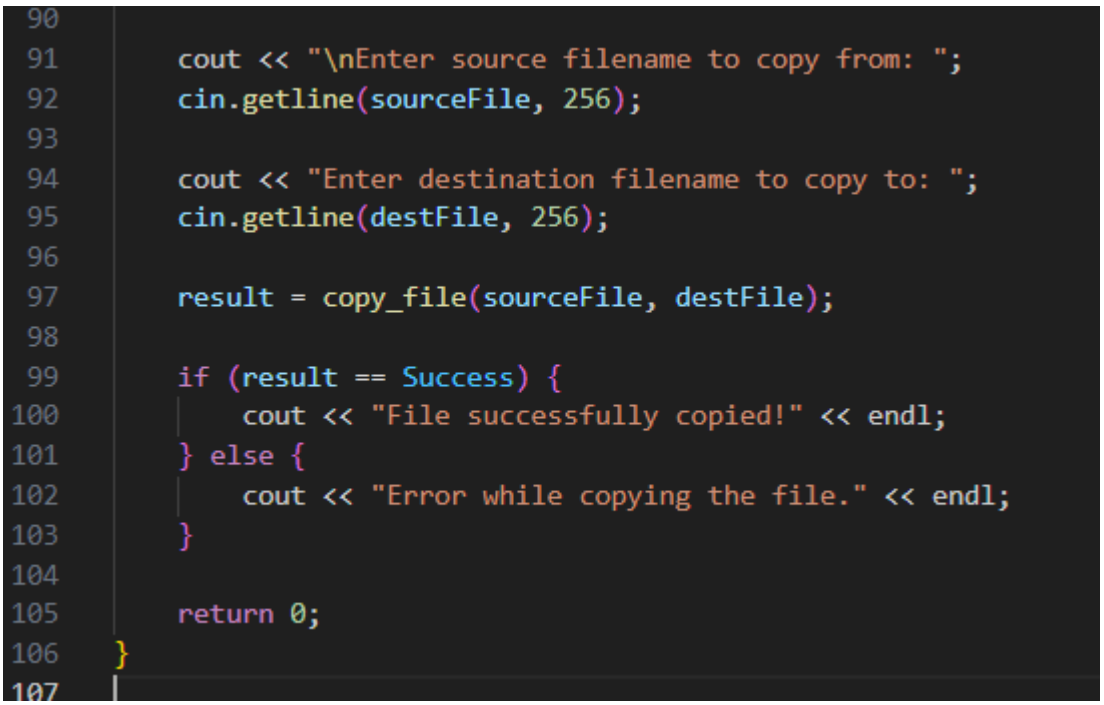
```cpp
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

const int BOARD_SIZE = 8;

bool canAttack(char pieceType, int pieceRow, int pieceCol, int targetRow, int targetCol) {
    switch (pieceType) {
        case 'R':
            return pieceRow == targetRow || pieceCol == targetCol;
        case 'B':
            return abs(pieceRow - targetRow) == abs(pieceCol - targetCol);
        case 'N':
            return (abs(pieceRow - targetRow) == 2 && abs(pieceCol - targetCol) == 1) ||
                   (abs(pieceRow - targetRow) == 1 && abs(pieceCol - targetCol) == 2);
        case 'P':
            return pieceRow + 1 == targetRow && abs(pieceCol - targetCol) == 1;
        case 'K':
            return abs(pieceRow - targetRow) <= 1 && abs(pieceCol - targetCol) <= 1;
        case 'Q':
            return (pieceRow == targetRow || pieceCol == targetCol) ||
                   (abs(pieceRow - targetRow) == abs(pieceCol - targetCol));
        default:
            return false;
    }
}

int main() {
    vector<string> chessBoard(BOARD_SIZE);
    for (int row = 0; row < BOARD_SIZE; row++) {
        cin >> chessBoard[row];
    }

    int queryCount;
    cin >> queryCount;
    vector<string> queryResults(queryCount);

    for (int query = 0; query < queryCount; query++) {
        int targetRow, targetCol;
        cin >> targetRow >> targetCol;
        targetRow--;
        targetCol--;
```

```
46
47          if (chessBoard[targetRow][targetCol] != 'O') {
48              queryResults[query] = "X";
49              continue;
50          }
51
52          string threats;
53          for (int row = 0; row < BOARD_SIZE; row++) {
54              for (int col = 0; col < BOARD_SIZE; col++) {
55                  char currentPiece = chessBoard[row][col];
56                  if (currentPiece != 'O' && canAttack(currentPiece, row, col, targetRow, targetCol)) {
57                      if (threats.find(currentPiece) == string::npos) {
58                          threats += currentPiece;
59                      }
60                  }
61              }
62          }
63
64          if (threats.empty()) {
65              queryResults[query] = "O";
66          } else {
67              sort(threats.begin(), threats.end());
68              queryResults[query] = threats;
69          }
70      }
71
72      for (const auto& result : queryResults) {
73          cout << result << endl;
74      }
75
76      return 0;
77  }
78
```

Task 8 - Practice# programming: Class Practice Task

```cpp
#include <iostream>
#include <fstream>
#include <cstring>

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult write_to_file(const char* name, const char* content) {
    if (name == nullptr || content == nullptr) {
        return Failure;
    }

    ofstream outFile(name);

    if (!outFile.is_open()) {
        return Failure;
    }

    outFile << content;

    if (outFile.fail()) {
        outFile.close();
        return Failure;
    }

    outFile.close();

    if (outFile.fail()) {
        return Failure;
    }

    return Success;
}

FileOpResult copy_file(const char* file_from, const char* file_to) {
    if (file_from == nullptr || file_to == nullptr) {
        return Failure;
    }

    ifstream inFile(file_from);

    if (!inFile.is_open()) {
        return Failure;
    }
```

```
47      ofstream outFile(file_to);
48
49      if (!outFile.is_open()) {
50          inFile.close();
51          return Failure;
52      }
53
54      string line;
55      while (getline(inFile, line)) {
56          outFile << line << endl;
57      }
58
59      if (inFile.fail() || outFile.fail()) {
60          inFile.close();
61          outFile.close();
62          return Failure;
63      }
64
65      inFile.close();
66      outFile.close();
67
68      return Success;
69  }
70
71  int main() {
72      char filename[256];
73      char content[1024];
74
75      cout << "Enter filename to write: ";
76      cin.getline(filename, 256);
77
78      cout << "Enter content to write into the file: ";
79      cin.getline(content, 1024);
80
81      FileOpResult result = write_to_file(filename, content);
82
83      if (result == Success) {
84          cout << "File successfully created and content written!" << endl;
85      } else {
86          cout << "Error while creating or writing to the file." << endl;
87      }
88
89      char sourceFile[256], destFile[256];
```



```
90
91      cout << "\nEnter source filename to copy from: ";
92      cin.getline(sourceFile, 256);
93
94      cout << "Enter destination filename to copy to: ";
95      cin.getline(destFile, 256);
96
97      result = copy_file(sourceFile, destFile);
98
99      if (result == Success) {
100         cout << "File successfully copied!" << endl;
101     } else {
102         cout << "Error while copying the file." << endl;
103     }
104
105     return 0;
106 }
107
```

Task 9 - Practice# programming: Self Practice Task#1

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

using namespace std;

vector<int> getPossibleNumbers(const vector<string>& grid, int row, int col) {
    vector<int> availableNumbers;
    int size = grid.size();

    if (grid[row][col] != '0') {
        availableNumbers.push_back(grid[row][col] - '0');
        return availableNumbers;
    }

    set<int> usedDigits;
    for (int i = 0; i < size; ++i) {
        if (grid[row][i] != '0') usedDigits.insert(grid[row][i] - '0');
        if (grid[i][col] != '0') usedDigits.insert(grid[i][col] - '0');
    }

    for (int num = 1; num <= size; ++num) {
        if (usedDigits.find(num) == usedDigits.end()) {
            availableNumbers.push_back(num);
        }
    }
    return availableNumbers;
}

int main() {
    int gridSize;
    cin >> gridSize;
    vector<string> sudokuGrid(gridSize);

    for (int i = 0; i < gridSize; ++i) {
        cin >> sudokuGrid[i];
    }

    int numQueries;
    cin >> numQueries;
    vector<pair<int, int>> queries(numQueries);

    for (int i = 0; i < numQueries; ++i) {
        int row, col;
```

```
        int row, col;
        cin >> row >> col;
        queries[i] = {row - 1, col - 1};
    }

    for (const auto& query : queries) {
        int row = query.first, col = query.second;
        vector<int> possibleNumbers = getPossibleNumbers(sudokuGrid, row, col);

        cout << possibleNumbers.size() << endl;
        for (int num : possibleNumbers) {
            cout << num << " ";
        }
        cout << endl;
    }

    return 0;
}
```

5) Результати виконання завдань та фактично затрачений час

Task 3 - Lab# programming: VNS Lab 6(варіант 20)

```
Line starts with dot dot starts with universe creation univerce creation starts with big boom big boom starts with dot.
8
```

Task 4 - Lab# programming: VNS Lab 8(варіант 20)

```
Initial file contents:
Name: USA, Language: English, Currency: Dollar, Exchange Rate: 1
Name: Japan, Language: Japanese, Currency: Yen, Exchange Rate: 0.007
Name: Germany, Language: German, Currency: Euro, Exchange Rate: 1.1
Country deleted successfully.
File contents after deletion:
Name: USA, Language: English, Currency: Dollar, Exchange Rate: 1
Name: Japan, Language: Japanese, Currency: Yen, Exchange Rate: 0.007
File contents after addition:
Name: USA, Language: English, Currency: Dollar, Exchange Rate: 1
Name: Japan, Language: Japanese, Currency: Yen, Exchange Rate: 0.007
Name: India, Language: Hindi, Currency: Rupee, Exchange Rate: 0.012
```

Task 5 - Lab# programming: VNS Lab 9(варіант 20)

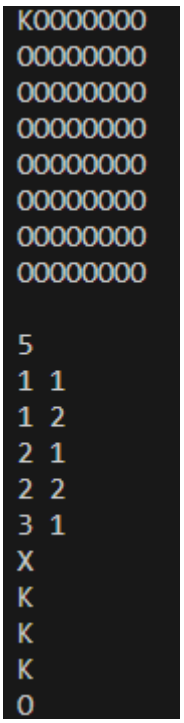
```
Enter at least 10 lines for F1. Type 'END' to finish:
I wish I knew about my ability to lose deadlines,
The way time slips through cracks and thin lines.
Each plan a castle built on shifting sand,
A grand design undone by a faltering hand.
Minutes turn to hours, days to weeks,
Dreams of progress drowned in time’s vast creeks.
Promises made to calendars that weep,
As alarms cry out, yet I find no sleep.
A skillless art, this dance with delay,
A masterpiece of moments, wasting away.
END
Number of vowels in the last line of F2: 12
```

Task 6 - Lab# programming: Algotester Lab 4(варіант 1) 100%

```
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
```

4 hours ago	C++ 23	Accepted	0.003	1.488 View

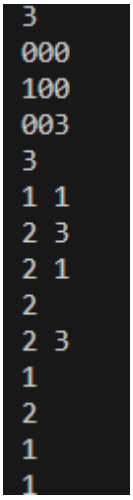
Task 7 - Lab# programming: Algotester Lab 6(варіант 2)



4 hours ago	C++ 23	Accepted	0.003	1.367	View

Task 8 - Practice# programming: Class Practice Task

Task 9 - Practice# programming: Self Practice Task#1



3 hours ago	C++ 23	Accepted	0.003	1.289	View

6) Робота з комадою

Відео-зустріч:

Висновок: Робота з файлами, як текстовими, так і бінарними, є важливим елементом програмування, що дозволяє зберігати, обробляти та передавати дані. Стандартна бібліотека надає зручний набір інструментів для роботи з файлами, які спрощують операції читання, запису та обробки даних. Володіння цими навичками є важливим для створення ефективного та масштабованого програмного забезпечення, а також для розробки власних бібліотек, що адаптовані до специфічних потреб.