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
Шийка Стефан

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: Опанувати практичні навички роботи з файлами в мові C++: створення, зчитування та запис даних у бінарні й текстові файли. Засвоїти принципи роботи з рядковими змінними різних типів (std::string і char*), 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:

CS50 course
University lectures
Google + chatGPT: string functions and memory allocation

Виконання роботи:

Lab# programming: VNS Lab 6

Time expected: 30 min

Time spent: 30 min

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.

18. Всі слова рядка, які починаються із цифри відсортувати за спаданням.

vns_lab_6_task_stefan_shyika.cpp > ...

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

using namespace std;

int main(){
    puts("Введіть строку: ");
    char s[256];
    gets(s);

    //remove the last "."
    int len = strlen(s);
    if(s[len-1] == '.') s[len-1] = '\0';

    //add words strating with a number to a vector using tockens
    vector<string> words;
    char* token = strtok(s, " ");

    while(token != nullptr){

        string word(token);

        if(isdigit(word[0])) words.push_back(word);

        token = strtok(nullptr, " ");
    }

    //sort them and cout
    sort(words.rbegin(), words.rend());

    cout << "Слова, що починаються з цифри, у спадному порядку:" << endl;
    for (const auto& word : words) {
        cout << word << endl;
    }
}
```

```
Введіть строку:
giwd 3fkelf 6ksdmflndng 9kfmglsfgnksfgnskgfj slkf 2fk
Слова, що починаються з цифри, у спадному порядку:
9kfmglsfgnksfgnskgfj
6ksdmflndng
3fkelf
2fk
```

Lab# programming: VNS Lab 8

Time expected: 1 h

Time spent: 1.5 h – 2 h

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.

18. Структура "Книга":
    - назва;
    - автор;
    - рік видання;
    - кількість сторінок.

Знищити 3 елементи з початку файлу, додати елемент перед елементом із зазначеною назвою.

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

using namespace std;

typedef struct{
    char title[50];
    char author[50];
    int year;
    int numberOfPages;
}Book;

void addInitialData(const char* filename);
void deleteThreeFirstBooks(const char* filename);
void addBeforeTheBook(const char* filename, const Book& newBook, const char* title);
void printFile(const char* filename);

int main(){
    const char* filename = "books.dat";
    addInitialData(filename);
    deleteThreeFirstBooks(filename);

    Book newBook = {"newBook", "Stefan", 2024, 999};

    addBeforeTheBook(filename, newBook, "Book6");

}


void addInitialData(const char* filename) {
    FILE* f = fopen(filename, "wb");
    if ((f == NULL)){
        cerr << "Can't opent the file." << endl;
        exit(1);
    }

    Book initialBooks[] = {
        {"Book1", "Author1", 1925, 218},
        {"Book2", "Author2", 1960, 281},
        {"Book3", "Author3", 1949, 328},
        {"Book4", "Author4", 1813, 279},
        {"Book5", "Author5", 1951, 277},
        {"Book6", "Author6", 1851, 635}
    };

    for (int i = 0; i < 6; i++) {
        if (fwrite(&initialBooks[i], sizeof(Book), 1, f) != 1) {
            cerr << "Error writing to file." << endl;
            exit(2);
        }
    }
    fclose(f);
    cout << "Initial books added successfully." << endl;
    printFile(filename);
}

void deleteThreeFirstBooks(const char* filename){
    FILE* f = fopen(filename, "rb");
    FILE* tmp = fopen("tmp.dat", "wb");

    if(f == NULL || tmp == NULL){
        cerr << "Can't open the file." << endl;
        exit(3);
    }

    Book book;
    int counter = 0;

    while(fread(&book, sizeof(Book), 1, f) == 1){
        if(counter >= 3){
            fwrite(&book, sizeof(Book), 1, tmp);
        }
        counter++;
    }

    fclose(tmp);
    fclose(f);

    remove(filename);
    rename("tmp.dat", filename);
    cout << "Deleted the first 3 books from " << filename << "." << endl;
    printFile(filename);
}

void addBeforeTheBook(const char* filename, const Book& newBook, const char* title){
    FILE* f = fopen(filename, "rb");
    FILE* tmp = fopen("tmp.dat", "wb");

    if (f == NULL || tmp == NULL) {
        cerr << "Error opening file." << endl;
        exit(4);
    }

    bool inserted = false;
    Book book;

    while(fread(&book, sizeof(Book), 1, f)){
        if(strcmp(book.title, title) == 0 && !inserted){
            fwrite(&newBook, sizeof(Book), 1, tmp);
            inserted = true;
        }
        fwrite(&book, sizeof(Book), 1, tmp);
    }

    if (!inserted) {
        cout << "No such title in the file :( Book inserted in the end." << endl;
        fwrite(&newBook, sizeof(Book), 1, tmp);
    }


    fclose(f);
    fclose(tmp);

    remove(filename);
    rename("tmp.dat", filename);
    if (inserted){
    cout << "Book added successfully before title: " << title << endl;
    }
    printFile(filename);
}

void printFile(const char* filename){
    FILE* f = fopen(filename, "rb");
    if (f == NULL) {
        cerr << "Error opening file." << endl;
        exit(5);
    }

    Book book;
    cout << "Contents of " << filename << ":" << endl;
    cout << "--------------------------------" << endl;

    while (fread(&book, sizeof(Book), 1, f) == 1) {
        cout << "Title: " << book.title << endl;
        cout << "Author: " << book.author << endl;
        cout << "Year: " << book.year << endl;
        cout << "Number of Pages: " << book.numberOfPages << endl;
        cout << "--------------------------------" << endl;
    }

    fclose(f);
}
```

Lab# programming: VNS Lab 9

Time expected: 1.5 h

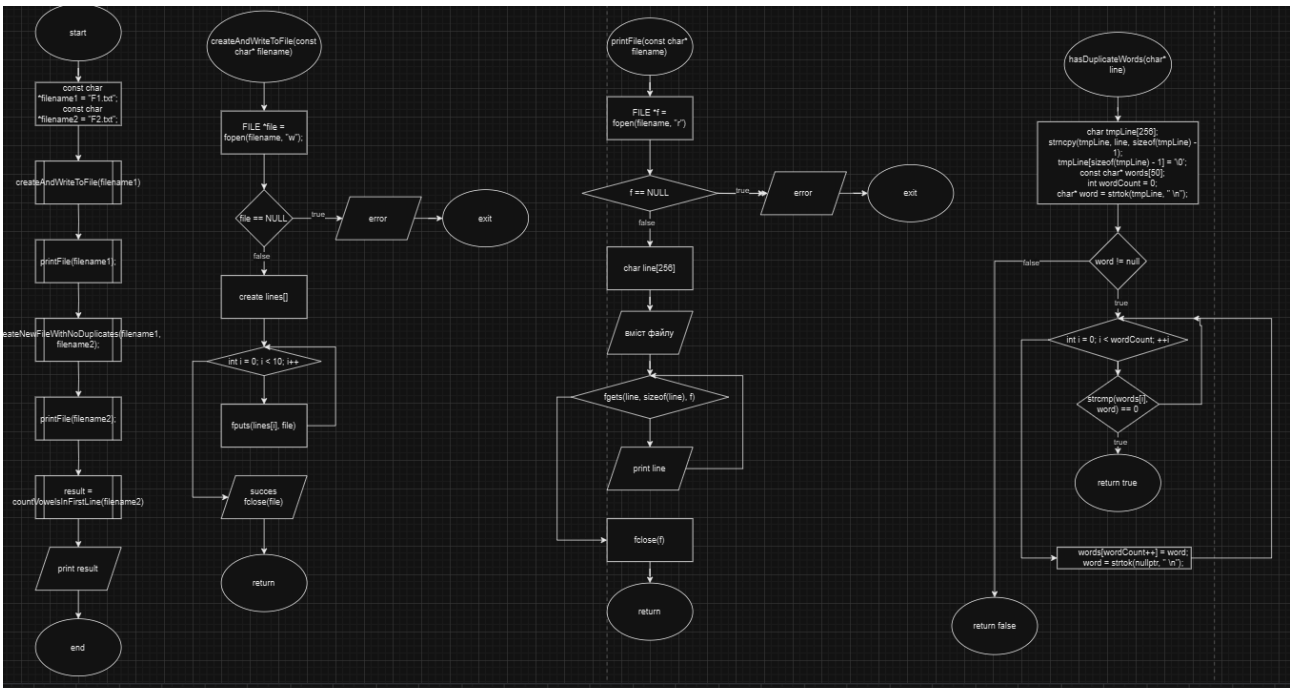
Time spent: 2.5 h

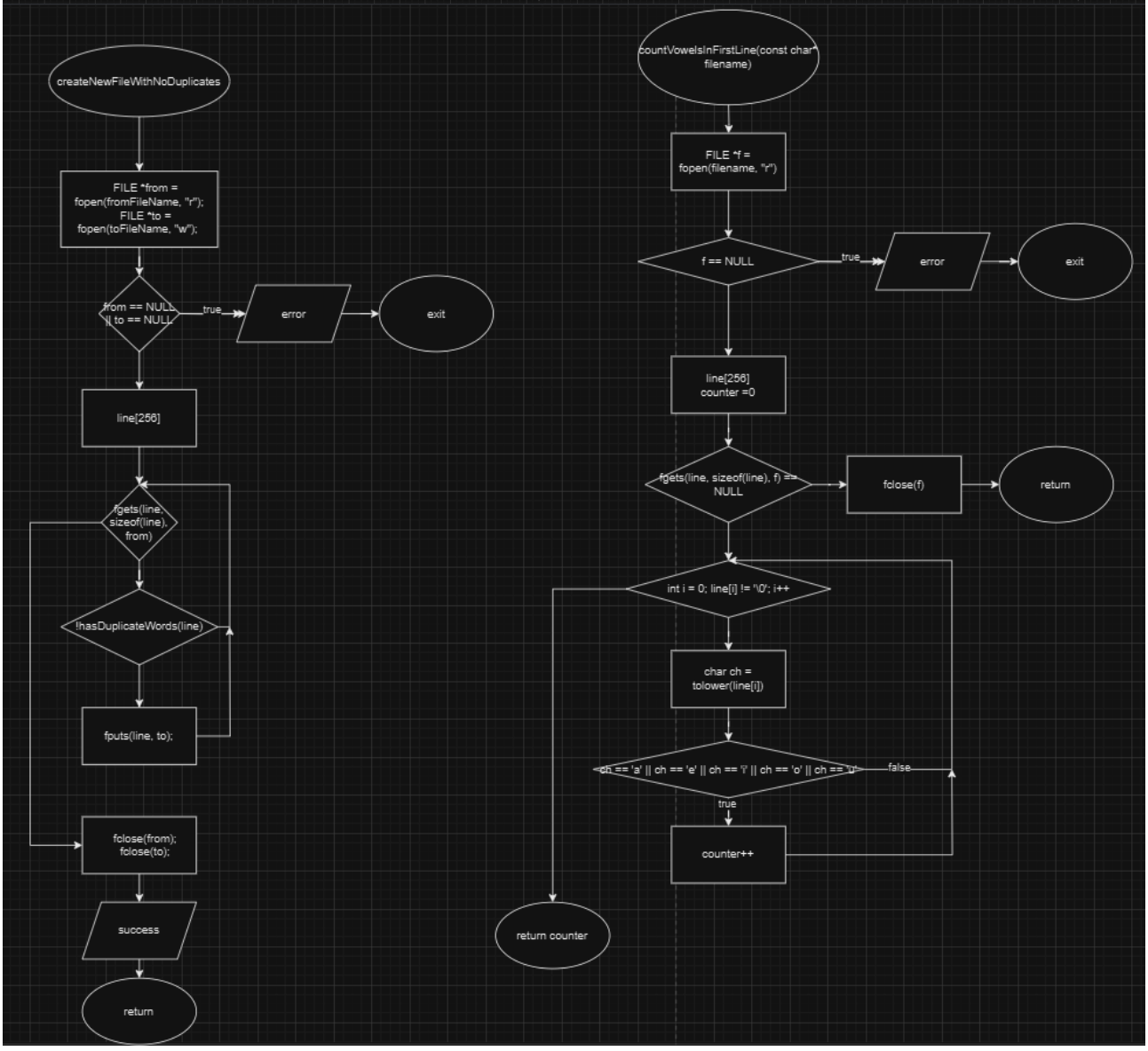
Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього інформацію

Виконати завдання.

18.

1) Скопіювати з файлу F1 у файл F2 всі рядки, у яких немає однакових слів.
2) Визначити кількість голосних букв у першому рядку файлу F2.


start
const char *filename1 = "F1.txt"; const char *filename2 = "F2.txt";
createAndWriteToFile(filename1)
printFile(filename1);
eateNewFileWithNoDuplicates(filename1, filename2);
printFile(filename2);
result = countVowelsInFirstLine(filename2)
print result
end
createAndWriteToFile(const char* filename)
FILE *file = fopen(filename, "w");
file == NULL
true
error
exit
false
create lines[]
int i = 0; i < 10; i++
fputs(lines[i], file)
succes fclose(file)
return
printFile(const char* filename)
FILE *f = fopen(filename, "r")
f == NULL
true
error
exit
false
char line[256]
вміст файлу
fgets(line, sizeof(line), f)
print line
fclose(f)
return
hasDuplicateWords(char* line)
char tmpLine[256]; strncpy(tmpLine, line, sizeof(tmpLine) - 1); tmpLine[sizeof(tmpLine) - 1] = '\0'; const char* words[50]; int wordCount = 0; char* word = strtok(tmpLine, " \n");
word != null
false
true
int i = 0; i < wordCount; ++i
$tromp(words[i], word) == 0
true
return true
words[wordCount++] = word; word = strtok(nullptr, " \n");
return false


createNewFileWithNoDuplicates
FILE *from = fopen(fromFileName, "r"); FILE *to = fopen(toFileName, "w");
from == NULL || to == NULL
true
error
exit
line[256]
fgets(line, sizeof(line), from)
!hasDuplicateWords(line)
fputs(line, to);
fclose(from); fclose(to);
success
return
countVowelsInFirstLine(const char* filename)
FILE *f = fopen(filename, "r")
f == NULL
true
error
exit
line[256] counter =0
fgets(line, sizeof(line), f) == NULL
fclose(f)
return
int i = 0; line[i] != '\0'; i++
char ch = tolower(line[i])
ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u'
false
true
counter++
return counter

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

using namespace std;

void createAndWriteToFile(const char* filename);
void printFile(const char* filename);
bool hasDuplicateWords(const char* line);
void createNewFileWithNoDuplicates(const char* fromFileName, const char* toFileName);
int countVowelsInFirstLine(const char* filename);

int main() {
    const char *filename1 = "F1.txt";
    const char *filename2 = "F2.txt";
    createAndWriteToFile(filename1);
    printFile(filename1);
    createNewFileWithNoDuplicates(filename1, filename2);
    printFile(filename2);
    cout << endl << "Number of vowels in the first line of F2: " << countVowelsInFirstLine(filename2) << endl;
}

void createAndWriteToFile(const char* filename) {
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        cerr << "Не вдалося відкрити файл для запису" << endl;
        exit(1);
    }

    const char* lines[] = {
        "line 1\n",
        "line 2: line\n",
        "line 3\n",
        "line 4: line line\n",
        "line 5\n",
        "line 6\n",
        "line 7: line\n",
        "line 8\n",
        "line 9: line line\n",
        "line 10\n"
    };

    for (int i = 0; i < 10; i++) {
        fputs(lines[i], file);
    }

    cout << "Дані успішно записані у файл " << filename << endl;
    fclose(file);
}

void printFile(const char* filename) {
    FILE *f = fopen(filename, "r");
    if (f == NULL) {
        cerr << "Не вдалося відкрити файл для читання" << endl;
        exit(2);
    }

    char line[256];

    cout << "Вміст файлу " << filename << ":\n";
    cout << "--------------------------------\n";

    while (fgets(line, sizeof(line), f)) {
        cout << line;
    }

    fclose(f);
}

bool hasDuplicateWords(char* line){
    char tmpLine[256];
    strncpy(tmpLine, line, sizeof(tmpLine) - 1);
    tmpLine[sizeof(tmpLine) - 1] = '\0';

    const char* words[50];
    int wordCount = 0;

    char* word = strtok(tmpLine, " \n");
    while (word != nullptr) {

        for (int i = 0; i < wordCount; ++i) {
            if (strcmp(words[i], word) == 0) {
                return true;
            }
        }

        words[wordCount++] = word;
        word = strtok(nullptr, " \n");
    }
    return false;
}

void createNewFileWithNoDuplicates(const char* fromFileName, const char* toFileName){
    FILE *from = fopen(fromFileName, "r");
    FILE *to = fopen(toFileName, "w");
    if (from == NULL || to == NULL) {
        cerr << "Не вдалося відкрити файл для читання" << endl;
        exit(3);
    }

    char line[256];

    while(fgets(line, sizeof(line), from)){
        if(!hasDuplicateWords(line)){
            fputs(line, to);
        }
    }

    fclose(from);
    fclose(to);

    cout << "Lines without repeated words copied successfully." << endl;
}

int countVowelsInFirstLine(const char* filename){
    FILE* f = fopen(filename, "r");

    if(f == NULL){
        cerr << "Не вдалося відкрити файл для читання" << endl;
        exit(4);
    }

    char line[256];
    if (fgets(line, sizeof(line), f) == NULL) {
        fclose(f);
        return 0;  // No lines in file
    }

    int counter = 0;

    for(int i = 0; line[i] != '\0'; i++){
        char ch = tolower(line[i]);
        if(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u'){
            counter++;
        }
    }

    return counter;
}
```

```
Дані успішно записані у файл F1.txt
Вміст файлу F1.txt:
--------------------------------
line 1
line 2: line
line 3
line 4: line line
line 5
line 6
line 7: line
line 8
line 9: line line
line 10
Lines without repeated words copied successfully.
Вміст файлу F2.txt:
--------------------------------
line 1
line 3
line 5
line 6
line 8
line 10

Number of vowels in the first line of F2: 2
PS C:\Users\user\Desktop\c++\epic5>
```

Lab# programming: Algotester Lab 4

4.2

Lab 4v2

Limits: 1 sec., 256 MiB

Вам дано масив a з N 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 K, тобто при $K = 3$ масив [1, 2, 3, 4, 5, 6, 7] перетвориться на [4, 5, 6, 7, 1, 2, 3].
Виведіть результат.

Input

У першому рядку цілі числа N та K

У другому рядку N цілих чисел - елементи масиву a

Output

У першому рядку ціле число N - розмір множини a
У наступному рядку N цілих чисел - множина a

Time expected: 30 min

Time spent: 20 min

algotester_lab_4_task_stefan_shyika_v1.cpp > main()

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

using namespace std;

int main(){
    int N, K;
    set<int> numSet;
    cin >> N >> K;
    int tmp;

    for(int i = 0; i < N; i++){
        cin >> tmp;
        numSet.insert(tmp);
    }

    int len = numSet.size();
    cout << len << endl;

    for(int i = K; i < len + K; i++){
        int index = i % len;
        auto it  = numSet.begin();
        advance(it, index);

        if(i != len + K - 1){
        cout << *it << " ";
        }else{
            cout << *it;
        }
    }
}
```

algotester_lab_4_task_stefan_shyika_v2.cpp > main()

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

using namespace std;

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

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

    sort(a.begin(), a.end());

    auto last = unique(a.begin(), a.end());
    a.erase(last, a.end());

    int len = a.size();
    K = K % len;
    rotate(a.begin(), a.begin() + K, a.end());

    cout << len << endl;
    for(int i = 0; i < len; i++){
        if(i != len-1){
            cout << a[i] << " ";
        }else{
            cout << a[i];
        }
    }
}
```

2 days ago	Lab 4v2 - Lab 4v2	C++ 23	Accepted	0.003	1.219	1858136

4.3

Lab 4v3

Limits: 2 sec., 256 MiB

Вам дано масив, який складається з N 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.

Input

У першому рядку N - кількість чисел.

У другому рядку N чисел a_i - елементи масиву.

Output

У першому рядку M - кількість чисел у масиву

У другому рядку M посоротованих за умовою чисел.

Time expected: 45 min

Time spent: 1 h

algotester_lab_4_variant_2_stefan_shyika_v1.cpp > main()

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

using namespace std;

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

    vector<int> vec(N);

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

    sort(vec.begin(), vec.end());
    auto last = unique(vec.begin(), vec.end());
    vec.erase(last, vec.end());

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

    sort(mod0.begin(), mod0.end());
    sort(mod2.begin(), mod2.end());
    sort(mod1.begin(), mod1.end(), greater<int>());

    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());

    int len = result.size();
    cout << len << endl;
    for (int i = 0; i < len; i++) {
        if(i != len -1){
            cout << result[i] << " ";
        }else{
            cout << result[i];
        }
    }
}
```

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

using namespace std;

int partition(vector<int>& arr, int start, int end) {
    int pivot = arr[end];
    int i = start - 1;
    for (int j = start; j < end; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[end]);
    return i + 1;
}

void quickSort(vector<int>& arr, int start, int end) {
    if (start < end) {
        int pi = partition(arr, start, end);
        quickSort(arr, start, pi - 1);
        quickSort(arr, pi + 1, end);
    }
}

vector<int> removeDuplicatesAfterSorting(const vector<int>& arr) {
    vector<int> uniqueArr;
    for (size_t i = 0; i < arr.size(); i++) {
        if (i == 0 || arr[i] != arr[i - 1]) {
            uniqueArr.push_back(arr[i]);
        }
    }
    return uniqueArr;
}


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

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

    quickSort(vec, 0, N - 1);

    vec = removeDuplicatesAfterSorting(vec);


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

    //mod0 and mod1 are already sorted
    for (size_t i = 0; i < mod1.size() / 2; i++) {
        swap(mod1[i], mod1[mod1.size() - 1 - i]);
    }


    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());

    int len = result.size();
    cout << len << endl;
    for (int i = 0; i < len; i++) {
        if(i != len -1){
            cout << result[i] << " ";
        }else{
            cout << result[i];
        }
    }
}
```

19 hours ago	Lab 4v3 - Lab 4v3	C++ 23	Accepted	0.003	1.242	1858386

Lab# programming: Algotester Lab 6

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.

17 hours ago	Lab 6v2 - Lab 6v2	C++ 23	Accepted	0.003	1.332	1858492

Time expected: 4h

Time spent: 4h

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

using namespace std;

bool validCoordinates(int x, int y) {
    return x >= 0 && x < 8 && y >= 0 && y < 8;
}

void addPawnAttacks(vector<set<char>>& attacks, int x, int y) {
    if (validCoordinates(x + 1, y - 1)) attacks[(x + 1) * 8 + (y - 1)].insert('P');
    if (validCoordinates(x + 1, y + 1)) attacks[(x + 1) * 8 + (y + 1)].insert('P');
}

void addRookAttacks(vector<set<char>>& attacks, int x, int y, char piece) {
    for (int i = 0; i < 8; i++) {
        if (i != y) attacks[x * 8 + i].insert(piece);
        if (i != x) attacks[i * 8 + y].insert(piece);
    }
}

void addBishopAttacks(vector<set<char>>& attacks, int x, int y, char piece) {
    for (int i = -7; i <= 7; i++) {
        if (i != 0) {
            if (validCoordinates(x + i, y + i)) attacks[(x + i) * 8 + (y + i)].insert(piece);
            if (validCoordinates(x + i, y - i)) attacks[(x + i) * 8 + (y - i)].insert(piece);
        }
    }
}

void addKnightAttacks(vector<set<char>>& attacks, int x, int y) {
    int knightMoves[8][2] = {{-2, -1}, {-2, 1}, {2, -1}, {2, 1}, {-1, -2}, {1, -2}, {-1, 2}, {1, 2}};
    for (auto& move : knightMoves) {
        int x1 = x + move[0], y1 = y + move[1];
        if (validCoordinates(x1, y1)) attacks[x1 * 8 + y1].insert('N');
    }
}

void addKingAttacks(vector<set<char>>& attacks, int x, int y) {
    for (int dx = -1; dx <= 1; dx++) {
        for (int dy = -1; dy <= 1; dy++) {
            if (dx != 0 || dy != 0) {
                int x1 = x + dx, y1 = y + dy;
                if (validCoordinates(x1, y1)) attacks[x1 * 8 + y1].insert('K');
            }
        }
    }
}

int main(){
    vector<string> board(8);
    for (int i = 0; i < 8; i++) {
        cin >> board[i];
    }

    vector<set<char>> attacks(64);

    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            char piece = board[i][j];
            switch (piece) {
                case 'P':
                    addPawnAttacks(attacks, i, j);
                    break;
                case 'R':
                    addRookAttacks(attacks, i, j, 'R');
                    break;
                case 'N':
                    addKnightAttacks(attacks, i, j);
                    break;
                case 'B':
                    addBishopAttacks(attacks, i, j, 'B');
                    break;
                case 'K':
                    addKingAttacks(attacks, i, j);
                    break;
                case 'Q':  //queen = rook + bishop
                    addRookAttacks(attacks, i, j, 'Q');
                    addBishopAttacks(attacks, i, j, 'Q');
                    break;
                default:
                    break;
            }
        }
    }

    int Q;
    cin >> Q;
    for(int i = 0; Q-i > 0; i++) {
        int x, y;
        cin >> x >> y;
        x--; y--;

        char cell = board[x][y];
        if (cell != '0') {
            cout << "X\n";
        } else {
            if (attacks[x * 8 + y].empty()) {
                cout << "O\n";
            } else {
                vector<char> attackers(attacks[x * 8 + y].begin(), attacks[x * 8 + y].end());
                sort(attackers.begin(), attackers.end());
                for (char attacker : attackers) {
                    cout << attacker;
                }
                cout << "\n";
            }
        }
    }
}
```

Practice# programming: Class Practice Task

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.

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.

Time expected: 1 h

Time spent: 40 min

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

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult write_to_file(const char *name, const char *content);
FileOpResult copy_file(const char *file_from, const char *file_to);

int main() {
    const char* filename1 = "file1.txt";
    const char* filename2 = "file2.txt";
    char content[256];

    cout << "Введіть вміст для запису у файл: ";
    cin.getline(content, sizeof(content));

    FileOpResult result1 = write_to_file(filename1, content);
    if (result1 == Success) {
        cout << "Success. File created." << endl;
    } else {
        cout << "Failure. File could not be created." << endl;
    }

    FileOpResult result2 = copy_file(filename1, filename2);
    if (result2 == Success) {
        cout << "Success. File is copied." << endl;
    } else {
        cout << "Failure coping file." << endl;
    }
}

FileOpResult write_to_file(const char *name, const char *content) {
    if (name == nullptr || strlen(name) == 0) {
        return FileOpResult::Failure;
    }

    FILE* f = fopen(name, "w");
    if (f == NULL) {
        return FileOpResult::Failure;
    }

    int len = strlen(content);
    size_t written = fwrite(content, sizeof(char), len, f);

    if (written != len) {
        fclose(f);
        return FileOpResult::Failure;
    }

    if (fclose(f) != 0) {
        return FileOpResult::Failure;
    }

    return FileOpResult::Success;
}

FileOpResult copy_file(const char *file_from, const char *file_to){
    if (file_from == nullptr || strlen(file_from) == 0 || file_to == nullptr || strlen(file_to) == 0) {
        return FileOpResult::Failure;
    }

    FILE* f1 = fopen(file_from, "r");
    FILE* f2 = fopen(file_to, "w");

    if (f1 == NULL || f2 == NULL) {
        return FileOpResult::Failure;
    }

    char buffer[512];
    while(fgets(buffer, sizeof(buffer), f1) != NULL){
        if(fputs(buffer, f2) == EOF){
            fclose(f1);
            fclose(f2);
            return FileOpResult::Failure;
        }
    }

    if (fclose(f1) != 0 || fclose(f2) != 0) {
        return FileOpResult::Failure;
    }

    return FileOpResult::Success;

}
```

```
Введіть вміст для запису у файл: roses are red, violets are blue
Success. File created.
Success. File is copied.
```

Practice# programming: Self Practice Task

Числа для малят

Limits: 2 sec., 256 MiB

Малята, напишіть на клаптику паперу ціле додатне число n. Тепер уявіть собі, що Ви можете переставляти цифри у його десятковому записі як завгодно. Єдина умова — не повинно бути нулів на початку запису числа. Вам потрібно визначити мінімальне та максимальне числа, які можна отримати таким способом.

Бажаю успіху! І не забувайте, що ви все знаєте, просто можливо щось забули.

Input

У єдиному рядку задано одне ціле число n.

Output

У єдиному рядку виведіть два цілих числа — мінімальне та максимальне числа, які можна отримати перестановкою цифр числа n.

14 hours ago	0889 - Числа для малят	C++ 23	Accepted	0.003	1.215	1858604

Time expected: 30 min

Time spent: 30 min

practice_work_self_algotester_tasks_stefan_shyika.cpp > ...

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

using namespace std;

int main() {
    string n;
    cin >> n;

    string minNumber = n;
    sort(minNumber.begin(), minNumber.end());

    // Переміщуємо першу ненульову цифру на початок, якщо є нуль на початку
    if (minNumber[0] == '0') {
        for (int i = 1; i < minNumber.size(); i++) {
            if (minNumber[i] != '0') {
                swap(minNumber[0], minNumber[i]);
                break;
            }
        }
    }

    string maxNumber = n;
    sort(maxNumber.begin(), maxNumber.end(), greater<char>());

    cout << minNumber << " " << maxNumber;

}
```

Pull

Висновок: У ході роботи було вивчено основи роботи з файловою системою в C++: опрацьовано принципи обробки текстових і бінарних файлів, включаючи процеси запису, зчитування й редагування даних. Завдяки використанню різних типів рядкових змінних (std::string та char*) вдалося ознайомитися з різними підходами до зберігання й обробки текстових даних. Використання стандартної бібліотеки значно спростило роботу з файлами, дозволяючи зосередитися на вирішенні основних завдань.