

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 ІІІ-12

Кривичко Назар

## **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:**

Навчитись працювати з файловою системою в C++ , рядками типу `std::string` та `char*` . Використання стандартної бібліотеки та створення власних бібліотек

## **Джерела:**

книга - Stephen Prata - “ *C++ Primer Plus* ”

книга - Aditya Y.Bhargava - “ *Grokking algorithms* ”

## **Виконання роботи:**

### **Завдання № 3**

#### **Requirements :**

Vns Lab 6

#### **Time:**

**Expected: 1 hour**

**Spent: up to 1 hour**

```

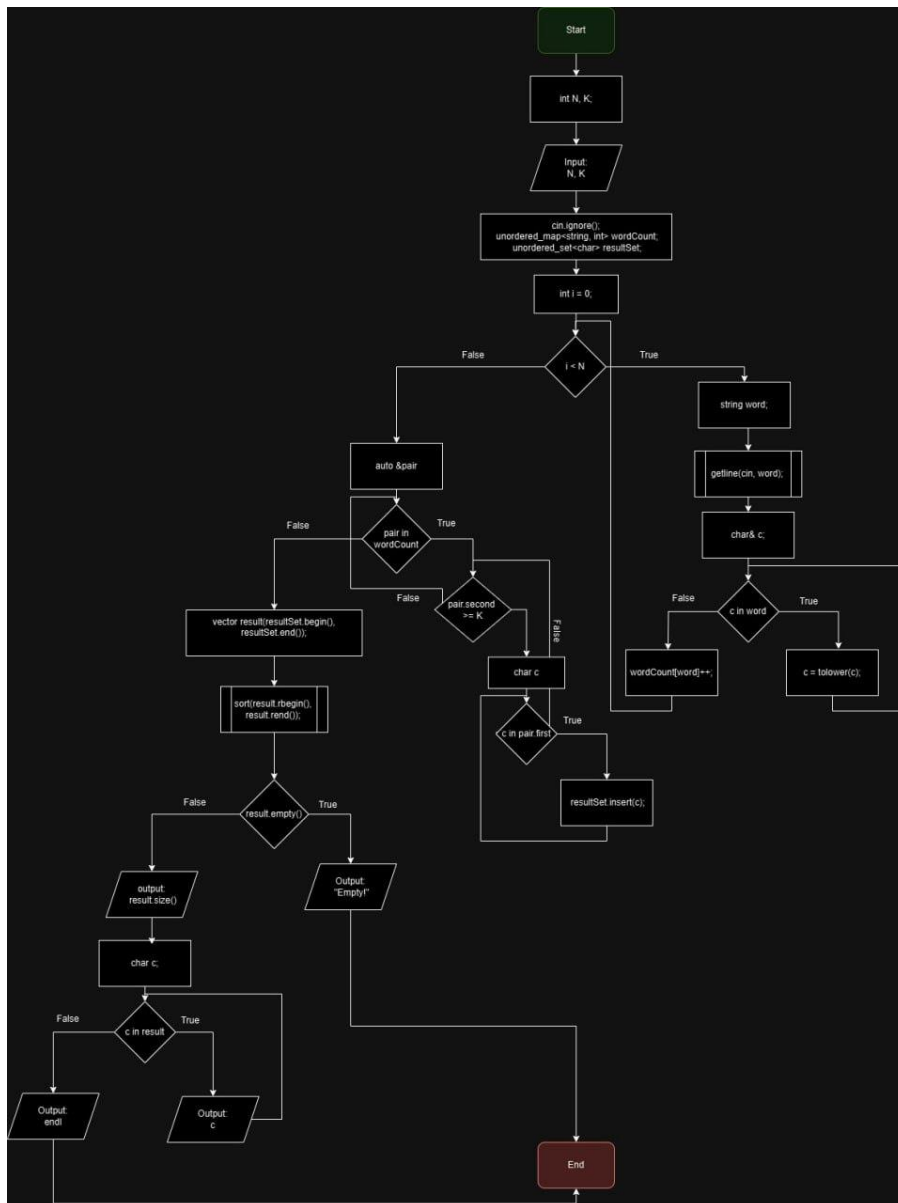
1 #include <iostream>
2 #include <cstring>
3
4 #define sLength 100
5
6 char** splitWords(const char* str, const char* delim, int& count) {
7     char* strCopy = new char[strlen(str) + 1];
8     strcpy(strCopy, str);
9
10    count = 0;
11    char* token = strtok(strCopy, delim);
12    while (token != nullptr) {
13        count++;
14        token = strtok(nullptr, delim);
15    }
16
17    char** words = new char*[count];
18    strcpy(strCopy, str);
19    token = strtok(strCopy, delim);
20    int index = 0;
21    while (token != nullptr) {
22        words[index] = new char[strlen(token) + 1];
23        strcpy(words[index], token);
24        token = strtok(nullptr, delim);
25        index++;
26    }
27
28    delete[] strCopy;
29    return words;
30 }
31
32 bool checkHas(const char* str, const char* charsSeq)
33 {
34     for (size_t i = 0; i < strlen(str); i++)
35     {
36         for (size_t j = 0; j < strlen(charsSeq); j++)
37         {
38             if (str[i] == charsSeq[j]) return true;
39         }
40     }
41     return false;
42 }
43
44 void printHasNoVowel(const char* str, const char* delim)
45 {
46     int wordCount = 0;
47     const char vowels[] = "AEIOUaeiou";
48     char** words = splitWords(str, delim, wordCount);
49     for (size_t i = 0; i < wordCount; i++)
50     {
51         if (!checkHas(words[i], vowels))
52         {
53             std::cout << words[i] << " ";
54         }
55         delete[] words[i];
56     }
57     delete[] words;
58 }
59
60 int main(void)
61 {
62     char cStr[sLength];
63     std::cout << "Enter String: ";
64     fgets(cStr, sLength, stdin);
65
66     cStr[strcspn(cStr, "\n")] = '\0';
67
68     const char delim[] = " \t, .!";
69
70     printHasNoVowel(cStr, delim);
71
72     return 0;
73 }

```

```

e-Pid-Dm'sbn2w.bxy --dbgExe=C:\msys64\
Enter String: Hll Wrld Hello World !
Hll Wrld

```



## Завдання № 4

### Requirements :

Vns Lab 8

Time:

Expected: 1 hour

Spent: up to 1 hour

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
```

```

#include <cstring>

static size_t aLength = 2;

typedef struct {
    char name[50];
    char surname[50];
    char last_name[50];
    char position[50];
    char birth_date[11];
    unsigned int salary;
} EMPLOYEE;

void print_emp(const EMPLOYEE& emp) {
    printf("Name: %s\nSurname: %s\nLast name: %s\nPosition: %s\nBirthdate: %s\nSalary: %u\n",
        emp.name, emp.surname, emp.last_name, emp.position, emp.birth_date,
        emp.salary);
}

void delete_with_surname(const char* surname) {
    EMPLOYEE* fEmps = new EMPLOYEE[aLength];
    FILE* fPtr = fopen("employees.bin", "rb");
    fread(fEmps, sizeof(EMPLOYEE), aLength, fPtr);
    fclose(fPtr);

    size_t cNewEms = aLength;
    for (size_t i = 0; i < aLength; i++) {
        if (strcmp(fEmps[i].surname, surname) == 0) {
            cNewEms--;
        }
    }

    EMPLOYEE* dEmps = new EMPLOYEE[cNewEms];
    size_t dEmpsI = 0;
    for (size_t i = 0; i < aLength; i++) {
        if (strcmp(fEmps[i].surname, surname) != 0) {
            dEmps[dEmpsI++] = fEmps[i];
        }
    }

    FILE* fnPtr = fopen("employees.bin", "wb");
    fwrite(dEmps, sizeof(EMPLOYEE), cNewEms, fnPtr);
    fclose(fnPtr);

    aLength = cNewEms;
}

```

```

    delete[] fEmps;
    delete[] dEmps;
}

void add_after(FILE* fptr, const unsigned int& number, const EMPLOYEE& newEmp)
{
    EMPLOYEE* emps = new EMPLOYEE[aLength];
    fread(emps, sizeof(EMPLOYEE), aLength, fptr);

    EMPLOYEE* newEmps = new EMPLOYEE[aLength + 1];
    size_t empsI = 0;

    for (size_t i = 0; i < number - 1; i++) {
        newEmps[empsI++] = emps[i];
    }

    newEmps[empsI++] = newEmp;

    for (size_t i = number - 1; i < aLength; i++) {
        newEmps[empsI++] = emps[i];
    }

    FILE* fCptr = fopen("employees.bin", "wb");
    fwrite(newEmps, sizeof(EMPLOYEE), aLength + 1, fCptr);
    fclose(fCptr);

    delete[] emps;
    delete[] newEmps;
}

void read_employee_data(EMPLOYEE* emps, int count) {
    for (int i = 0; i < count; i++) {
        printf("Enter details for employee %d:\n", i + 1);
        printf("Name: ");
        fgets(emps[i].name, sizeof(emps[i].name), stdin);
        emps[i].name[strcspn(emps[i].name, "\n")] = '\0';

        printf("Surname: ");
        fgets(emps[i].surname, sizeof(emps[i].surname), stdin);
        emps[i].surname[strcspn(emps[i].surname, "\n")] = '\0';

        printf("Last Name: ");
        fgets(emps[i].last_name, sizeof(emps[i].last_name), stdin);
        emps[i].last_name[strcspn(emps[i].last_name, "\n")] = '\0';
    }
}

```

```

        printf("Position: ");
        fgets(emps[i].position, sizeof(emps[i].position), stdin);
        emps[i].position[strcspn(emps[i].position, "\n")] = '\0';

        printf("Birth Date (YYYY-MM-DD): ");
        fgets(emps[i].birth_date, sizeof(emps[i].birth_date), stdin);
        emps[i].birth_date[strcspn(emps[i].birth_date, "\n")] = '\0';

        printf("Salary: ");
        scanf("%u", &emps[i].salary);
        getchar();
    }
}

int main() {
    FILE* fPtr;

    fPtr = fopen("employees.bin", "wb");
    if (!fPtr) {
        perror("Error opening file for writing");
        return EXIT_FAILURE;
    }

    EMPLOYEE* emps = new EMPLOYEE[aLength];
    read_employee_data(emps, aLength);

    fwrite(emps, sizeof(EMPLOYEE), aLength, fPtr);
    fclose(fPtr);
    delete[] emps;

    fPtr = fopen("employees.bin", "rb");
    if (!fPtr) {
        perror("Error opening file for reading");
        return EXIT_FAILURE;
    }

    EMPLOYEE* rEmpArr = new EMPLOYEE[aLength];
    fread(rEmpArr, sizeof(EMPLOYEE), aLength, fPtr);
    fclose(fPtr);

    printf("\nEmployee Data from File:\n");
    for (int i = 0; i < aLength; i++) {
        printf("\n");
        print_emp(rEmpArr[i]);
        printf("\n");
    }
}

```

```

printf("\nAfter delete with surname Olex\n");
delete_with_surname("Olex");

fPtr = fopen("employees.bin", "rb");
if (!fPtr) {
    perror("Error opening file for reading");
    return EXIT_FAILURE;
}

EMPLOYEE* updatedEmps = new EMPLOYEE[aLength];
fread(updatedEmps, sizeof(EMPLOYEE), aLength, fPtr);
fclose(fPtr);

printf("\nUpdated Employee Data from File:\n");
for (size_t i = 0; i < aLength; i++) {
    printf("\n");
    print_emp(updatedEmps[i]);
    printf("\n");
}

EMPLOYEE newEmp;
printf("Enter details for new employee to add:\n");
printf("Name: ");
fgets(newEmp.name, sizeof(newEmp.name), stdin);
newEmp.name[strcspn(newEmp.name, "\n")] = '\0';

printf("Surname: ");
fgets(newEmp.surname, sizeof(newEmp.surname), stdin);
newEmp.surname[strcspn(newEmp.surname, "\n")] = '\0';

printf("Last Name: ");
fgets(newEmp.last_name, sizeof(newEmp.last_name), stdin);
newEmp.last_name[strcspn(newEmp.last_name, "\n")] = '\0';

printf("Position: ");
fgets(newEmp.position, sizeof(newEmp.position), stdin);
newEmp.position[strcspn(newEmp.position, "\n")] = '\0';

printf("Birth Date (YYYY-MM-DD): ");
fgets(newEmp.birth_date, sizeof(newEmp.birth_date), stdin);
newEmp.birth_date[strcspn(newEmp.birth_date, "\n")] = '\0';

printf("Salary: ");
scanf("%u", &newEmp.salary);
getchar();

```



```

    unsigned int position;
    printf("Enter the position (1 to %zu) after which to add the new employee:
", aLength);
    scanf("%u", &position);
    getchar();

    if (position > 0 && position <= aLength) {
        fPtr = fopen("employees.bin", "rb+");
        add_after(fPtr, position, newEmp);
        fclose(fPtr);
    } else {
        printf("Invalid position!\n");
    }

    fPtr = fopen("employees.bin", "rb");
    if (!fPtr) {
        perror("Error opening file for reading");
        return EXIT_FAILURE;
    }

    EMPLOYEE* finalEmps = new EMPLOYEE[aLength + 1];
    fread(finalEmps, sizeof(EMPLOYEE), aLength + 1, fPtr);
    fclose(fPtr);

    printf("\nFinal Employee Data from File:\n");
    for (size_t i = 0; i < aLength + 1; i++) {
        printf("\n");
        print_emp(finalEmps[i]);
        printf("\n");
    }

    delete[] rEmpArr;
    delete[] updatedEmps;
    delete[] finalEmps;

    return 0;
}

```

# Final Employee Data from File:

Name: Sirko  
Surname: Kozak  
Last name: Kush  
Position: Kozar-Sichi  
Birthdate: 2004-11-23  
Salary: 19022

Name: nazar  
Surname: nazar2  
Last name: lastname2  
Position: server-engineer  
Birthdate: 2006-11-23  
Salary: 1200

Name: Olena  
Surname: Kshuk  
Last name: Lastname3  
Position: Backend-developer  
Birthdate: 2005-11-22  
Salary: 1900

After delete with surname Olex

Updated Employee Data from File:

Name: nazar

Surname: nazar2

Last name: lastname2

Position: server-engineer

Birthdate: 2006-11-23

Salary: 1200

Name: Olena

Surname: Kshuk

Last name: Lastname3

Position: Backend-developer

Birthdate: 2005-11-22

Salary: 1900

Enter details for new employee to add:

Name: Sirko

Surname: Kozak

Last Name: Kush

Position: Kozar-Sichi

Birth Date (YYYY-MM-DD): 2004-11-23

Salary: 19022

Enter the position (1 to 2) after which to add the new employee: 1

```
Enter details for employee 1:
Name: nazar
Surname: nazar2
Last Name: lastname2
Position: server-engineer
Birth Date (YYYY-MM-DD): 2006-11-23
Salary: 1200
Enter details for employee 2:
Name: Olena
Surname: Kshuk
Last Name: Lastname3
Position: Backend-developer
Birth Date (YYYY-MM-DD): 2005-11-22
Salary: 1900

Employee Data from File:

Name: nazar
Surname: nazar2
Last name: lastname2
Position: server-engineer
Birthdate: 2006-11-23
Salary: 1200

Name: Olena
Surname: Kshuk
Last name: Lastname3
Position: Backend-developer
Birthdate: 2005-11-22
Salary: 1900
```

## Завдання № 5

### Requirements :

Vns Lab 9

### Time:

**Expected: 1 hour**

**Spent: up to 1 hour**

```
int main() {
    FILE *inputFile = fopen("F1.txt", "r");
    FILE *outputFile = fopen("F2.txt", "w");
    FILE *oFRptr = fopen("F2.txt", "r");

    if (inputFile == NULL) {
        perror("Error opening input file");
        return EXIT_FAILURE;
    }
    if (outputFile == NULL) {
        perror("Error opening output file");
        fclose(inputFile);
        return EXIT_FAILURE;
    }

    char word[100];

    while (fscanf(inputFile, "%99s", word) == 1) {
        if (word[0] == 'A') {
            fprintf(outputFile, "%s\n", word);
        }
    }
    fclose(inputFile);
    fclose(outputFile);

    int count = countWordsInFile("F2.txt");

    printf("Words in F2: %u", count);

    return EXIT_SUCCESS;
}
```

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>

int countWordsInFile(const char* filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        perror("Error opening file");
        return -1;
    }

    int wordCount = 0;
    char line[100];

    while (fgets(line, sizeof(line), file) != NULL) {
        line[strcspn(line, "\n")] = '\0';

        if (line[0] != '\0') {
            wordCount++;
        }
    }

    fclose(file);
    return wordCount;
}
```

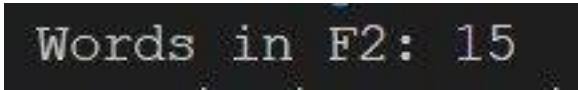
F2:

Apple  
Almonds.  
Art  
An  
Adventure  
Astronomy  
Apricot  
After  
Asphalt  
Anna  
Artist  
Amanda  
Aquarium  
A  
Atmosphere

F1:

The Apple was fresh, and it paired well with Almonds.  
She visited the Art museum and admired the beautiful paintings.  
An amazing Adventure awaited them as they reached the mountain.  
Learning Astronomy can be fascinating and rewarding for curious minds.  
They shared an Apricot pie while sitting under the big tree.  
After the rain, the Asphalt on the road looked dark and clean.  
Anna loves to read, especially novels about ancient civilizations.  
The Artist painted a vibrant sunset across the sky.  
Amanda went to the Aquarium with her little brother last week.  
A sudden change in the Atmosphere made the evening cooler.

Result:



Words in F2: 15

## Завдання № 6

### Requirements :

Algotester Lab 4

Time:

**Expected: 1 hour**

**Spent: up to 1 hour**

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

using namespace std;

// Function to print the size and sorted elements of a vector
void printSetResult(const vector<int>& result) {
    cout << result.size() << endl;
    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;
}

// Function to perform the set operations
void manualSetOperations() {
    int N, M;

    // Input for the first array
    cin >> N;
    vector<int> array1(N);
    for (int i = 0; i < N; ++i) {
        cin >> array1[i];
    }

    // Input for the second array
    cin >> M;
    vector<int> array2(M);
    for (int i = 0; i < M; ++i) {
```



```

    cin >> array2[i];
}

// Use unordered_map to count occurrences of each element
unordered_map<int, int> count1, count2;
for (int num : array1) {
    count1[num]++;
}
for (int num : array2) {
    count2[num]++;
}

// Calculate difference N - M
vector<int> difference_N_M;
for (const auto& pair : count1) {
    int num = pair.first;
    int count = pair.second;
    if (count2.find(num) == count2.end()) {
        difference_N_M.insert(difference_N_M.end(), count, num);
    } else {
        // If it exists in both, add the difference in counts
        int diffCount = count - count2[num];
        if (diffCount > 0) {
            difference_N_M.insert(difference_N_M.end(), diffCount, num);
        }
    }
}

// Calculate difference M - N
vector<int> difference_M_N;
for (const auto& pair : count2) {
    int num = pair.first;
    int count = pair.second;
    if (count1.find(num) == count1.end()) {
        difference_M_N.insert(difference_M_N.end(), count, num);
    } else {
        // If it exists in both, add the difference in counts
        int diffCount = count - count1[num];
        if (diffCount > 0) {
            difference_M_N.insert(difference_M_N.end(), diffCount, num);
        }
    }
}

// Calculate intersection
vector<int> intersection;
for (const auto& pair : count1) {

```

```

        int num = pair.first;
        if (count2.find(num) != count2.end()) {
            int minCount = min(pair.second, count2[num]);
            intersection.insert(intersection.end(), minCount, num);
        }
    }

    // Calculate union
    vector<int> unionSet;
    for (const auto& pair : count1) {
        int num = pair.first;
        int count = pair.second;
        unionSet.insert(unionSet.end(), count, num);
    }
    for (const auto& pair : count2) {
        int num = pair.first;
        int count = pair.second;
        if (count1.find(num) == count1.end()) {
            unionSet.insert(unionSet.end(), count, num);
        }
    }

    // Calculate symmetric difference
    vector<int> symmetric_difference;
    for (const auto& pair : count1) {
        int num = pair.first;
        if (count2.find(num) == count2.end()) {
            symmetric_difference.insert(symmetric_difference.end(),
pair.second, num);
        } else {
            int diffCount = pair.second - count2[num];
            if (diffCount > 0) {
                symmetric_difference.insert(symmetric_difference.end(),
diffCount, num);
            }
        }
    }
    for (const auto& pair : count2) {
        int num = pair.first;
        if (count1.find(num) == count1.end()) {
            symmetric_difference.insert(symmetric_difference.end(),
pair.second, num);
        } else {
            int diffCount = pair.second - count1[num];
            if (diffCount > 0) {
                symmetric_difference.insert(symmetric_difference.end(),
diffCount, num);
            }
        }
    }

```

```

    }
}

// Sort results for output
sort(difference_N_M.begin(), difference_N_M.end());
sort(difference_M_N.begin(), difference_M_N.end());
sort(intersection.begin(), intersection.end());
sort(unionSet.begin(), unionSet.end());
sort(symmettric_difference.begin(), symmettric_difference.end());

// Print results
printSetResult(difference_N_M);
printSetResult(difference_M_N);
printSetResult(intersection);
printSetResult(unionSet);
printSetResult(symmettric_difference);
}

int main() {
    manualSetOperations();
    return 0;
}

```

6 днів тому

[Lab 4v1 - Lab 4v1](#)

C++ 17

Зараховано

0.003

1.262

[1846215](#)

## [6.2]

```

#include <iostream>
#include <vector>
#include <algorithm>
#include <unordered_set>

void printsc(const std::vector<int>& v) {
    std::cout << v.size() << std::endl;
    for (const auto& el : v) {
        std::cout << el << ' ';
    }
    std::cout << std::endl;
}

int main() {
    size_t N, M;
    std::cin >> N;
    std::vector<int> v1(N);
    for (size_t i = 0; i < N; i++) {
        std::cin >> v1[i];
    }
}

```

```

}

std::cin >> M;
std::vector<int> v2(M);
for (size_t i = 0; i < M; i++) {
    std::cin >> v2[i];
}

std::sort(v1.begin(), v1.end());
std::sort(v2.begin(), v2.end());

// N - M
std::vector<int> NdM;
std::set_difference(v1.begin(), v1.end(), v2.begin(), v2.end(),
std::back_inserter(NdM));
printsc(NdM);
std::cout << std::endl;

// M - N
std::vector<int> MnD;
std::set_difference(v2.begin(), v2.end(), v1.begin(), v1.end(),
std::back_inserter(MnD));
printsc(MnD);
std::cout << std::endl;

// M with N
std::vector<int> MwithN;
std::set_intersection(v1.begin(), v1.end(), v2.begin(), v2.end(),
std::back_inserter(MwithN));
printsc(MwithN);
std::cout << std::endl;

// M and N
std::vector<int> MandN;
std::set_union(v1.begin(), v1.end(), v2.begin(), v2.end(),
std::back_inserter(MandN));
printsc(MandN);
std::cout << std::endl;

// M <-> N
std::vector<int> MsN;
std::set_symmetric_difference(v1.begin(), v1.end(), v2.begin(), v2.end(),
std::back_inserter(MsN));
printsc(MsN);

return 0;

```

```
}
```

## Завдання № 7

### Requirements :

Algotester Lab 6

Time:

Expected: 1 hour

Spent: up to 1 hour

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include <unordered_set>
#include <algorithm>
#include <cctype>

using namespace std;

int main() {
    int N, K;
    cin >> N >> K;
    cin.ignore(); // Ignore the newline after reading integers

    unordered_map<string, int> wordCount;
    unordered_set<char> resultSet;

    // Read words and count their occurrences
    for (int i = 0; i < N; ++i) {
        string word;
        getline(cin, word);

        // Convert to lowercase
        for (char &c : word) {
            c = tolower(c);
        }

        // Increment the count for the word
        wordCount[word]++;
    }

    // Check which words meet the threshold K
    for (const auto &pair : wordCount) {
        if (pair.second >= K) {
            // Add letters of the word to the result set
            for (char c : pair.first) {
                resultSet.insert(c);
            }
        }
    }

    // Prepare the result
    vector<char> result(resultSet.begin(), resultSet.end());
    sort(result.rbegin(), result.rend()); // Sort in reverse order

    // Output results
    if (result.empty()) {
        cout << "Empty!" << endl;
    } else {
        cout << result.size() << endl;
        for (char c : result) {
            cout << c << " ";
        }
        cout << endl;
    }

    return 0;
}
```

2 години тому	Lab 6v1 - Lab 6v1	C++ 17	Зараховано	0.035	1.246	1858400
---------------	-------------------	--------	------------	-------	-------	---------

## Завдання № 8

### Requirements :

Class Practice Task

Time:

Expected: 1 hour

Spent: up to 1 hour

```
#include <iostream>
#include <fstream>
#include <cstring>
#include <map>
#include <vector>
#include <string>

enum FileOpResult { Success = 0, Failure = 1 };

static std::map<FileOpResult, std::string> OperationStr = {
    { Success, "Success" },
    { Failure, "Failure" }
};

bool hasEnding(const std::string& fullString, const std::string& ending) {
    if (fullString.length() >= ending.length()) {
        return (0 == fullString.compare(fullString.length() - ending.length(),
ending.length(), ending));
    } else {
        return false;
    }
}

FileOpResult write_to_file(const char* name, const char* content) {
    if (name == nullptr || strcmp(name, "") == 0 || !hasEnding(name, ".txt"))
    {
        name = "default.txt";
    }

    std::ofstream fileOut(name, std::ios::out | std::ios::trunc);
    if (!fileOut.is_open()) {
        return FileOpResult::Failure;
    }
}
```

```

        fileOut << content;
        if (!fileOut.good()) { // Check if writing to file was successful
            return FileOpResult::Failure;
        }

        fileOut.close();
        return fileOut.fail() ? FileOpResult::Failure : FileOpResult::Success;
    }

    unsigned int FileRead(std::istream& is, std::vector<char>& buff) {
        is.read(&buff[0], buff.size());
        return is.gcount();
    }

    void FileRead(std::ifstream& ifs, std::string& s) {
        const unsigned int BUFSIZE = 64 * 1024;
        std::vector<char> buffer(BUFSIZE);

        while (unsigned int n = FileRead(ifs, buffer)) {
            s.append(&buffer[0], n);
        }
    }

    FileOpResult copy_file(const char* file_from, const char* file_to) {
        if (file_from == nullptr || strcmp(file_from, "") == 0 ||
            file_to == nullptr || strcmp(file_to, "") == 0 ||
            !hasEnding(file_from, ".txt") || !hasEnding(file_to, ".txt")) {
            return FileOpResult::Failure;
        }

        std::ifstream fileIn(file_from, std::ios::in);
        if (!fileIn.is_open()) {
            return FileOpResult::Failure;
        }

        std::ofstream fileOut(file_to, std::ios::out | std::ios::trunc);
        if (!fileOut.is_open()) {
            fileIn.close();
            return FileOpResult::Failure;
        }

        std::string fileContent;
        FileRead(fileIn, fileContent);
        fileOut << fileContent;
    }

```

```

    bool failure = fileIn.bad() || fileOut.bad();
    fileIn.close();
    fileOut.close();

    return failure ? FileOpResult::Failure : FileOpResult::Success;
}

int main() {
    // Task [1]
    std::string contentLine, fileName;
    std::cout << "Enter content: ";
    std::getline(std::cin, contentLine);
    std::cout << "Enter file name: (*.txt) (DEFAULT NAME = \"default.txt\"): ";
    std::getline(std::cin, fileName);
    FileOpResult writeResult = write_to_file(fileName.c_str(),
contentLine.c_str());
    std::cout << "Operation Result: " << OperationStr[writeResult] <<
std::endl;

    // Task [2]
    std::string fromFile, toFile;
    std::cout << "Enter file name {from}: ";
    std::getline(std::cin, fromFile);
    std::cout << "Enter file name {to}: ";
    std::getline(std::cin, toFile);
    FileOpResult copyResult = copy_file(fromFile.c_str(), toFile.c_str());
    std::cout << "Copy Operation Result: " << OperationStr[copyResult] <<
std::endl;

    return 0;
}

```



From.txt:

```
From Content 1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

To.txt:

```
From Content 1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

Test1.txt:

```
JWT token
```

```
Enter content: JWT Token
Enter file name: (*.txt) (DEFAULT NAME = "default.txt"): test1.txt
Operation Result: Success
Enter file name {from}: from.txt
Enter file name {to}: to.txt
Copy Operation Result: Success
```

## Завдання № 9

### Requirements :

Class Practice Task

Time:

Expected: 1 hour

Spent: up to 1 hour

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

int main(void)
{
    int N;
    std::cin >> N;
    std::vector<int> cats(N);
    for (int i = 0, sz = cats.capacity(); i < sz; i++)
    {
        std::cin >> cats[i];
    }
    std::cout << *std::max_element(cats.begin(), cats.end()) - *std::min_element(cats.begin(), cats.end())
    return 0;
}
```

33 хвилини тому	0114 - Тренер слонів	C++ 17	Зараховано	0.047	1.449	1858481
--------------------	----------------------	--------	------------	-------	-------	---------

Pull Request: [Link](#)

Висновок :

Я навчився працювати з рядками типу char\* та std::string , закріпив свої знання роботи з файлами на C та C++.

