

Міністерство освіти і науки України  
Національний технічний університет України «КПІ ім. Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії  
Програмування інтелектуальних інформаційних систем

**ЗВІТ**  
до лабораторних робіт

**Виконав**  
**студент**

ІП-01 Гагарін Артем Олексійович  
(№ групи, прізвище, ім'я, по батькові )

**Прийняв**

ас. Очеретяний О. К.  
(посада, прізвище, ім'я, по батькові )

Київ 2021

# 1. Завдання лабораторної роботи

## Завдання 1:

Обчислювальна задача тут тривіальна: для текстового файлу ми хочемо відобразити N (наприклад, 25) найчастіших слів і відповідну частоту їх повторення, упорядковано за зменшенням. Слід обов'язково нормалізувати використання великих літер і ігнорувати стоп-слова, як «the», «for» тощо. Щоб все було просто, ми не піклуємося про порядок слів з однаковою частотою повторень. Ця обчислювальна задача відома як **term frequency**.

Ось такий вигляд матимуть ввід і відповідно вивід результату програми:

### Input:

```
White tigers live mostly in India
Wild lions live mostly in Africa
```

### Output:

```
live - 2
mostly - 2
africa - 1
india - 1
lions - 1
tigers - 1
white - 1
wild - 1
```

## Завдання 2:

Тепер, нам потрібно виконати задачу, що називається словниковим індексуванням. Для текстового файлу виведіть усі слова в алфавітному порядку разом із номерами сторінок, на яких Ці слова знаходяться. Ігноруйте всі слова, які зустрічаються більше 100 разів.

Припустимо, що сторінка являє собою послідовність із 45 рядків. Наприклад, якщо взяти книгу *Pride and Prejudice*, перші кілька записів індексу будуть:

```
abatement - 89
abhorrence - 101, 145, 152, 241, 274, 281
abhorrent - 253
abide - 158, 292
```

## 2. Опис алгоритму

### Завдання 1:

1. Зчитати слово, якщо слів не залишилось – перейти до кроку 5.
2. Перевірити, щоб усі символи були буквами або знаком '-'. Звести слово до нижнього регістру.
3. Якщо слово є стоп-словом – перейти до кроку 1.
4. Якщо слово вже існує, додати 1 до кількості входжень, інакше додати як нове слово.
5. Перейти до кроку 1.
6. Відсортувати вставками.
7. Записати перших 25 слів до вихідного файлу.

### Завдання 2:

1. Зчитати слово, якщо слів не залишилось – перейти до кроку 6.
2. Перевірити, щоб усі символи були буквами або знаком '-'. Звести слово до нижнього регістру.
3. Якщо слово є стоп-словом – перейти до кроку 1.
4. Якщо слово вже існує додати 1 до кількості входжень та записати номер сторінки, інакше додати як нове слово та записати номер сторінки.
5. Перейти до кроку 1.
6. Відсортувати слова в алфавітному порядку.
7. Відкинути слова, що зустрічаються більше 100 разів.
8. Записати слова, що залишилися у файл.

## 3. Вихідний код

### Завдання 1:

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

int main()
{
    const int DisplayedWordsCtn = 25;

    const int StopWordsCtn = 12;
    string stopWords[] = { "at", "for", "the", "in", "before", "on", "so", "a",
        "than", "to", "with", "by" };

    int totalRecords = 5;

    struct Record
    {
        string word = "";
        int count = 0;
    };

    Record* records = new Record[totalRecords];

    int wordsCtn = 0;
```

```

string word;

ifstream fin;
fin.open("input.txt");

loop_input:
if (fin >> word)
{
    bool isNewWord = true;

    string fixedWord = "";
    int i = 0;
process_word:
    if (word[i] != '\0')
    {
        char c = word[i];
        if ((c >= 'a' && c <= 'z') ||
            (c >= 'A' && c <= 'Z') ||
            (c == '-'))
        {
            if (c >= 'A' && c <= 'Z')
            {
                c += 'a' - 'A';
            }
            fixedWord += c;
        }
        i++;
        goto process_word;
    }
    word = fixedWord;

    if (word == "" || word == "-")
    {
        goto loop_input;
    }

    i = 0;
check_stop_words:
    if (i < StopWordsCtn)
    {
        if (stopWords[i] == word)
        {
            goto loop_input;
        }
        i++;
        goto check_stop_words;
    }

    i = 0;
check_word_entries:
    if (i < wordsCtn && isNewWord)
    {
        if (records[i].word == word)
        {
            records[i].count++;
            isNewWord = false;
        }
        i++;
        goto check_word_entries;
    }
    if (isNewWord)
    {
        records[wordsCtn].word = word;
        records[wordsCtn].count = 1;
        wordsCtn++;
    }
}

```

```

    }
    if (wordsCtn == totalRecords)
    {
        totalRecords *= 2;
        Record* newRecords = new Record[totalRecords];

        i = 0;
resize:
        if (i < wordsCtn)
        {
            newRecords[i] = records[i];
            i++;
            goto resize;
        }
        delete[] records;
        records = newRecords;
    }
    goto loop_input;
}
fin.close();

int i = 1;
loop_i:
if (i < wordsCtn)
{
    int j = i;
loop_j:
    if (j > 0 && records[j - 1].count < records[j].count)
    {
        Record temp = records[j];
        records[j] = records[j - 1];
        records[j - 1] = temp;

        j--;
        goto loop_j;
    }
    i++;
    goto loop_i;
}

ofstream fout;;
fout.open("output.txt");

i = 0;
loop_output:
if (i < DisplayedWordsCtn && i < wordsCtn)
{
    fout << records[i].word << " - " << records[i].count << endl;
    i++;
    goto loop_output;
}
fout.close();

delete[] records;

return 0;
}

```

## Завдання 2:

```

#include <iostream>
#include <fstream>
using namespace std;

int main()

```

```

{
    const int StringsInPage = 45;

    const int StopWordsCtn = 12;
    string stopWords[] = { "at", "for", "the", "in", "before", "on", "so", "a",
        "than", "to", "with", "by" };

    const int MaxEntries = 100;

    int totalRecords = 1000;

    struct Record
    {
        string word = "";
        int count = 0;
        int* pages = new int[MaxEntries];
    };

    Record* records = new Record[totalRecords];

    string word;

    ifstream fin;
    fin.open("input.txt");

    int wordsCtn = 0,
        cutWordsCtn = 0,
        wordIdx = 0,
        currStr = 0;
loop_input:
    if (fin >> word)
    {
        loop_empty_lines:
        if (fin.peek() == '\n')
        {
            fin.get();
            currStr++;
            goto loop_empty_lines;
        }

        bool isNewWord = true;

        string fixedWord = "";
        int i = 0;
process_word:
        if (word[i] != '\0')
        {
            char c = word[i];
            if ((c >= 'a' && c <= 'z') ||
                (c >= 'A' && c <= 'Z') ||
                (c == '-'))
            {
                if (c >= 'A' && c <= 'Z')
                {
                    c += 'a' - 'A';
                }
                fixedWord += c;
            }
            i++;
            goto process_word;
        }
        word = fixedWord;

        if (word == "" || word == "-")
        {

```

```

        goto loop_input;
    }

    i = 0;
check_stop_words:
    if (i < StopWordsCtn)
    {
        if (stopWords[i] == word)
        {
            goto loop_input;
        }
        i++;
        goto check_stop_words;
    }

    i = 0;
check_word_entries:
    if (i < wordsCtn && isNewWord)
    {
        if (records[i].word == word)
        {
            if (++records[i].count == MaxEntries + 1)
            {
                cutWordsCtn++;
            }
            isNewWord = false;
            wordIdx = i;
        }
        i++;
        goto check_word_entries;
    }

    if (isNewWord)
    {
        wordsCtn++;
        wordIdx = wordsCtn - 1;

        records[wordIdx].word = word;
        records[wordIdx].count = 1;
    }
    if (wordsCtn == totalRecords)
    {
        totalRecords *= 2;
        Record* newRecords = new Record[totalRecords];

        i = 0;
copy_records:
        if (i < wordsCtn)
        {
            newRecords[i] = records[i];
            i++;
            goto copy_records;
        }
        delete[] records;

        records = newRecords;
    }
    if (records[wordIdx].count <= 100)
    {
        int pageIdx = records[wordIdx].count - 1,
            pageNum = currStr / StringsInPage + 1;
        records[wordIdx].pages[pageIdx] = pageNum;
    }
    goto loop_input;
}

```

```

    fin.close();

    int size = wordsCtn - cutWordsCtn;

    Record* newRecords = new Record[size];

    int ptr = 0,
        i = 0;
cut_words:
    if (i < wordsCtn)
    {
        if (records[i].count <= 100)
        {
            newRecords[ptr] = records[i];
            ptr++;
        }
        else
        {
            delete[] records[i].pages;
        }
        i++;
        goto cut_words;
    }

    i = wordsCtn;
free_memory:
    if (i < totalRecords)
    {
        delete[] records[i].pages;
        i++;
        goto free_memory;
    }

    delete[] records;
    records = newRecords;
    wordsCtn = size;

    i = 1;
loop_i:
    if (i < wordsCtn)
    {
        int j = i;
loop_j:
        if (j > 0)
        {
            int k = 0;
loop_char:
            if (records[j].word[k] != '\0' &&
                records[j - 1].word[k] != '\0' &&
                records[j].word[k] == records[j - 1].word[k])
            {
                k++;
                goto loop_char;
            }
            if (records[j - 1].word[k] > records[j].word[k] || records[j].word[k] ==
'\0')
            {
                Record temp = records[j];
                records[j] = records[j - 1];
                records[j - 1] = temp;
            }
            j--;
            goto loop_j;
        }
        i++;
    }

```



```

        goto loop_i;
    }

    ofstream fout;
    fout.open("output.txt");

    i = 0;
loop_output:
    if (i < wordsCtn)
    {
        fout << records[i].word << " - ";

        int j = 0;
        fout << records[i].pages[j];
        j++;
loop_pages:
        if (j < records[i].count)
        {
            fout << ", " << records[i].pages[j];
            j++;
            goto loop_pages;
        }
        fout << endl;

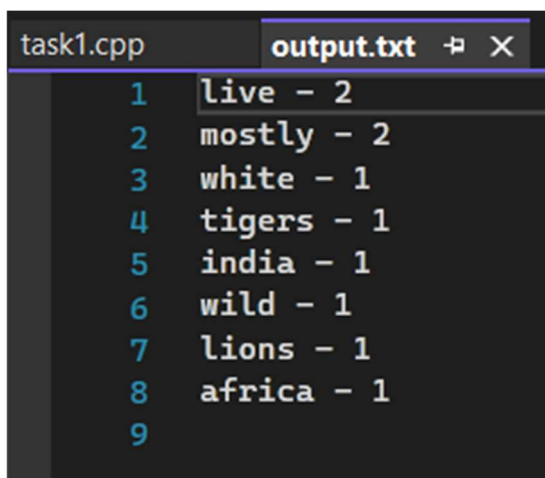
        i++;
        goto loop_output;
    }
    fout.close();

    i = 0;
clean_memory:
    if (i < wordsCtn)
    {
        delete[] records[i].pages;
        i++;
        goto clean_memory;
    }
    delete[] records;

    return 0;
}

```

#### 4. Скріншоти роботи програмного застосунку



```

task1.cpp  output.txt
1 live - 2
2 mostly - 2
3 white - 1
4 tigers - 1
5 india - 1
6 wild - 1
7 lions - 1
8 africa - 1
9

```

```
task2.cpp  output.txt  # X
1  a-shooting - 305
2  abatement - 99
3  abhorrence - 111, 160, 167, 263, 299, 306
4  abhorrent - 276
5  abide - 174, 318
6  abiding - 177
7  abilities - 72, 74, 107, 155, 171, 194
8  able - 19, 37, 58, 78, 84, 86, 88, 91, 98, 101, 107, 107, 109, 110, 120, 126, 130, 131, 144, 145, 152, 156, 172, 177, 178, 184, 186, 187, 195, 205, 211
9  ablution - 119
10 abode - 59, 60, 66, 110, 122, 130, 176, 260
11 abominable - 32, 51, 71, 71, 122, 161
12 abominably - 48, 133, 269, 299
13 abominate - 263, 296
14 abound - 101
15 above - 11, 11, 32, 153, 179, 195, 202, 210, 212, 213, 214, 214, 218, 220, 232, 237, 256, 257, 262, 278, 284
16 abroad - 194, 196, 233, 288
17 abrupt - 203
18 abruptly - 41, 155
19 abruptness - 198, 198
20 absence - 54, 56, 64, 77, 78, 78, 90, 99, 99, 100, 106, 106, 110, 111, 127, 150, 172, 172, 194, 195, 197, 205, 207, 224, 232, 238, 283
21 absent - 31, 199, 225, 229
22 absolute - 78, 227, 253, 308
23 absolutely - 17, 25, 32, 92, 94, 125, 147, 166, 167, 171, 190, 203, 242, 260, 269, 299, 304
24 absurd - 61, 163, 171, 296, 302
25 absurdities - 127, 217
26 absurdity - 189
27 abundant - 227
28 abundantly - 67, 85, 125
29 abuse - 6, 166
30 abused - 179, 197
31 abusing - 31, 299, 299
32 abusive - 184, 316
33 accede - 166
34 acceded - 207
35 acceding - 249
36 accent - 188, 204, 212, 222
37 accents - 192, 233
38 accept - 10, 10, 31, 76, 92, 92, 94, 107, 158, 160, 161, 173, 213, 283, 289, 291, 300, 318
39 acceptable - 60, 92, 94, 100, 143, 256
40 acceptance - 94, 129, 157, 213

100 %  No issues found  Ln: 1  Ch: 1  SPC  CRLF
```