

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

Кафедра інформатики та програмної інженерії

Мультипарадигменне програмування

## **ЗВІТ**

до лабораторних робіт

**Виконав**  
**студент**

Гушчін Д. О.  
(№ групи, прізвище, ім'я, по батькові )

**Прийняв**

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

Київ 2022

# 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.

## 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 250 слів.

## 2. Опис використаних технологій

Для виконання даної лабораторної роботи було використано мову C++, тому що вона задовольняє умові завдання (підтримує оператор goto). Також було використано середовище розробки Visual Studio 2022.

## 3. Опис програмного коду

### Task1.cpp:

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

struct WordFrequency {
    string word;
    int frequency = 0;
};

int main()
{
    const int WordsToShow = 25;
    int WordsAmount = 0, Size = 2;
```

```

WordFrequency* WordList = new WordFrequency[Size];

string StopWords[] = { "the", "a", "an", "for", "in", "on", "onto", "into",
"of", "to", "at", "by",
                        "are", "but", "is", "am", "and", "not", "or", "with",
"about", "towards",
                        "across", "though", "through", "up", "down", "above",
"below", "over",
                        "under", "beside", "off", "before", "after", "during",
"while", "since", "from",
                        "he", "she", "it", "i", "they", "you", "we", "that",
"as", "one", "be", "the"};

ifstream ReadingStream("text.txt");

string CurrWord;
int Counter = 0;

ReadingFile:
    if (!(ReadingStream >> CurrWord))
    {
        goto BubbleSorting;
    }
    Counter = 0;
    ToLowerCase:
        if (CurrWord[Counter] == '\\0')
        {
            goto IgnoreStopWords;
        }

        if (CurrWord[Counter] == ',' || CurrWord[Counter] == '.' ||
CurrWord[Counter] == ':' ||
            CurrWord[Counter] == '!' || CurrWord[Counter] == '?' ||
CurrWord[Counter] == '-')
        {
            CurrWord[Counter] = '\\n';
        }

        if (CurrWord[Counter] < 65 || (CurrWord[Counter] > 90 &&
CurrWord[Counter] < 97) || CurrWord[Counter] > 122)
        {
            goto ReadingFile;
        }

        if (65 <= CurrWord[Counter] && CurrWord[Counter] <= 90)
        {
            CurrWord[Counter] += 32;
        }

        Counter++;

```

```

        goto ToLowerCase;

Counter = 0;

IgnoreStopWords:
    if (Counter == 51)
    {
        goto Counting;
    }

    if (CurrWord == StopWords[Counter])
    {
        goto ReadingFile;
    }
    Counter++;
    goto IgnoreStopWords;

Counting:
    Counter = 0;

    IncrementFrequency:
        if (Counter == WordsAmount)
        {
            goto AddNewWord;
        }

        if (WordList[Counter].word == CurrWord)
        {
            WordList[Counter].frequency += 1;
            goto ReadingFile;
        }
        Counter++;
        goto IncrementFrequency;

AddNewWord:
    if (WordsAmount == Size - 1) // reallocation
    {
        Size *= 2;
        WordFrequency* TempWordList = new WordFrequency[Size];
        int i = 0;
        ArrayCopying:
            if (i == WordsAmount)
            {
                goto DeleteOldArray;
            }
            TempWordList[i] = WordList[i++];
            goto ArrayCopying;

        DeleteOldArray:
            delete[] WordList;
            WordList = TempWordList;
    }

```

```

    }

    WordList[WordsAmount].word = CurrWord;
    WordList[WordsAmount].frequency = 1;
    WordsAmount++;
    goto ReadingFile;

BubbleSorting:
    int i = 0;
    OuterLoop:
        if (i >= WordsAmount - 1)
        {
            i = 0;
            goto Output;
        }
        Counter = i + 1;

        InnerLoop:
            if (Counter >= WordsAmount) {
                i++;
                goto OuterLoop;
            }

            if (WordList[i].frequency < WordList[Counter].frequency) {
                WordFrequency tmp = WordList[i];
                WordList[i] = WordList[Counter];
                WordList[Counter] = tmp;
            }

            Counter++;
            goto InnerLoop;

    Output:
        if (i >= WordsAmount || i >= WordsToShow)
        {
            goto Finish;
        }

        if (WordList[i].frequency)
        {
            cout << WordList[i].word << " - " << WordList[i].frequency << endl;
        }
        i++;
        goto Output;

    Finish:
        return 0;
}

```

## Task2.cpp:

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

struct WordFrequency {
    string word;
    int frequency = 0;
    int pages[100] = {-1};
    int ind = 0;
};

int main()
{
    int TotalWordsAmount = 0;
    int WordsAmount = 0, Size = 2;
    WordFrequency* WordList = new WordFrequency[Size];

    string StopWords[] = { "the", "a", "an", "for", "in", "on", "onto", "into",
"of", "to", "at", "by",
"are", "but", "is", "am", "and", "not", "or", "with",
"about", "towards",
"across", "though", "through", "up", "down", "above",
"below", "over",
"under", "beside", "off", "before", "after", "during",
"while", "since", "from",
"he", "she", "it", "i", "they", "you", "we", "that",
"as", "one", "be", "the" };

    ifstream ReadingStream("text.txt");

    string CurrWord;
    int Counter = 0;

ReadingFile:
    if (!(ReadingStream >> CurrWord))
    {
        goto BubbleSorting;
    }
    Counter = 0;
    TotalWordsAmount++;

ToLowerCase:
    if (CurrWord[Counter] == '\0')
    {
        goto IgnoreStopWords;
    }
```

```

    if (CurrWord[Counter] == ',' || CurrWord[Counter] == '.' || CurrWord[Counter]
== ':' ||
        CurrWord[Counter] == '!' || CurrWord[Counter] == '?' || CurrWord[Counter]
== '-')
    {
        CurrWord[Counter] = '\n';
    }

    if (CurrWord[Counter] < 65 || (CurrWord[Counter] > 90 && CurrWord[Counter] <
97) || CurrWord[Counter] > 122)
    {
        goto ReadingFile;
    }

    if (65 <= CurrWord[Counter] && CurrWord[Counter] <= 90)
    {
        CurrWord[Counter] += 32;
    }

    Counter++;
    goto ToLowerCase;

    Counter = 0;

IgnoreStopWords:
    if (Counter == 51)
    {
        goto Counting;
    }

    if (CurrWord == StopWords[Counter])
    {
        goto ReadingFile;
    }
    Counter++;
    goto IgnoreStopWords;

Counting:
    Counter = 0;

IncrementFrequency:
    if (Counter == WordsAmount)
    {
        goto AddNewWord;
    }

    if (WordList[Counter].word == CurrWord)
    {
        WordList[Counter].frequency += 1;
    }

```

```

        if (WordList[Counter].frequency < 100)
        {
            WordList[Counter].pages[WordList[Counter].ind++] = TotalWordsAmount /
250;

            goto ReadingFile;

        }
    }
    Counter++;
    goto IncrementFrequency;

AddNewWord:
    if (WordsAmount == Size - 1) // reallocation
    {
        Size *= 2;
        WordFrequency* TempWordList = new WordFrequency[Size];
        int i = 0;
    ArrayCopying:
        if (i == WordsAmount)
        {
            goto DeleteOldArray;
        }
        TempWordList[i] = WordList[i++];
        goto ArrayCopying;

    DeleteOldArray:
        delete[] WordList;
        WordList = TempWordList;
    }

    WordList[WordsAmount].word = CurrWord;
    WordList[WordsAmount].frequency = 1;
    WordList[WordsAmount].pages[WordList[WordsAmount].ind++] = TotalWordsAmount /
250;
    WordsAmount++;
    goto ReadingFile;

BubbleSorting:
    int i = 0;
OuterLoop:
    if (i >= WordsAmount - 1)
    {
        i = 0;
        goto Output;
    }
    Counter = i + 1;

InnerLoop:
    if (Counter >= WordsAmount) {
        i++;
        goto OuterLoop;
    }

```



```

    }

    if (WordList[i].word > WordList[Counter].word) {
        WordFrequency tmp = WordList[i];
        WordList[i] = WordList[Counter];
        WordList[Counter] = tmp;
    }

    Counter++;
    goto InnerLoop;
}

Output:
    if (i >= WordsAmount)
    {
        goto Finish;
    }

    if (WordList[i].frequency && WordList[i].frequency < 100)
    {
        cout << WordList[i].word << ": ";
        Counter = 0;
        OutputPages:
        if (Counter < WordList[i].ind)
        {
            if (Counter == 0)
            {
                cout << WordList[i].pages[Counter] + 1;
            }
            else if (WordList[i].pages[Counter] != WordList[i].pages[Counter
- 1])
            {
                cout << ", " << WordList[i].pages[Counter] + 1;
            }
            Counter++;
            goto OutputPages;
        }
        cout << endl;
    }
    i++;
    goto Output;

Finish:
    return 0;
}

```

## 4. Опис алгоритму вирішення

### Task1.cpp:

1) Зчитати слово з файлу.

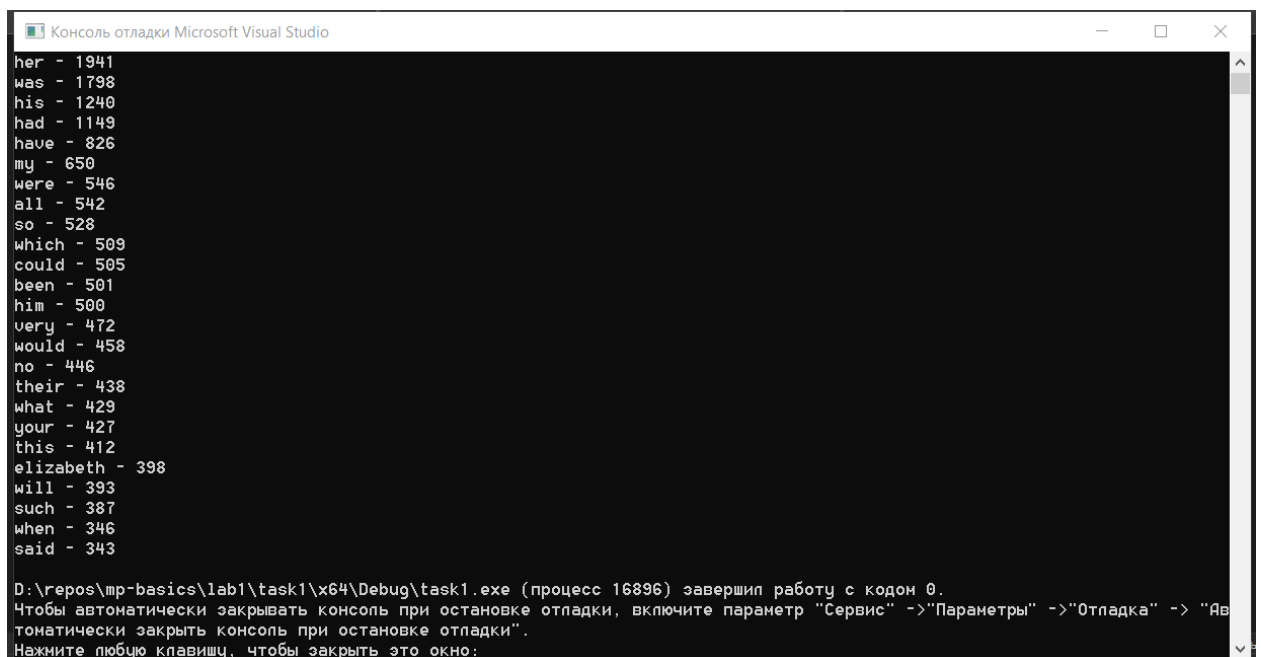
- 2) Перевести слово до нижнього регістру та прибрати знаки пунктуації.
- 3) Ігнорувати слово, якщо воно в списку стоп-слів.
- 4) Додати слово і його частоту, якщо слово було додано вперше, інакше збільшити частоту вже знайденого слова.
- 5) Відсортувати список слів за частотою за спаданням.
- 6) Вивести 25 найчастіших слів та частоту їх повторення.

### **Task2.cpp:**

- 1) Зчитати слово з файлу.
- 2) Перевести слово до нижнього регістру та прибрати знаки пунктуації.
- 3) Ігнорувати слово, якщо воно в списку стоп-слів.
- 4) Додати слово, частоту і сторінку, на якому воно зустрічається, якщо слово було додано вперше, інакше збільшити частоту і додати нову сторінку.
- 5) Відсортувати список слів за алфавітом.
- 6) Вивести усі слова разом зі списком сторінок, якщо слово зустрічається менше 100 разів.

## **5. Скріншоти роботи програмного застосунку**

### **Task1.cpp:**



```
Консоль отладки Microsoft Visual Studio
her - 1941
was - 1798
his - 1240
had - 1149
have - 826
my - 650
were - 546
all - 542
so - 528
which - 509
could - 505
been - 501
him - 500
very - 472
would - 458
no - 446
their - 438
what - 429
your - 427
this - 412
elizabeth - 398
will - 393
such - 387
when - 346
said - 343

D:\repos\mp-basics\lab1\task1\x64\Debug\task1.exe (процесс 16896) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:
```

## Task2.cpp:

Консоль отладки Microsoft Visual Studio

```
abatement: 144
abhorrence: 253, 407, 470
abhorrent: 427
abide: 263, 489
abiding: 268
abilities: 102, 156, 232
able: 23, 47, 79, 110, 118, 123, 125, 132, 141, 146, 157, 161, 162, 177, 187, 193, 194, 214, 217, 227, 233, 261, 268, 270, 278, 282, 284, 296, 312, 333, 337, 347, 349, 359, 367, 374, 378, 389, 390, 401, 402, 407, 408, 413, 415, 436, 442, 456, 459, 474, 486
ablution: 176
abode: 80, 81, 161, 181, 401
abominable: 42, 99, 180, 241
abominably: 62, 198, 415, 460
abominate: 407, 456
abound: 147
abroad: 294
abrupt: 308
abruptly: 53, 231
abruptness: 301
absence: 72, 75, 108, 109, 129, 144, 146, 155, 156, 163, 188, 225, 300, 311, 315, 357, 437
absolute: 109, 349, 390, 473
absolutely: 19, 31, 41, 133, 136, 186, 220, 250, 252, 259, 287, 308, 373, 402, 415, 467
absurd: 82, 465
absurdities: 333
absurdity: 287
abundant: 349
abundantly: 92, 121, 186
abuse: 4, 251
abused: 271, 299
abusing: 41, 460
abusive: 280
accede: 250
acceded: 316
acceding: 383
accent: 285, 309, 324, 341
accents: 358
accept: 8, 9, 39, 106, 133, 136, 156, 237, 239, 240, 262, 325, 445, 447, 461, 488
acceptable: 81, 132, 145, 214, 395
acceptance: 192, 235
accepted: 37, 83, 87, 90, 107, 110, 143, 153, 177, 208, 210, 214, 221, 250, 430, 498
accepting: 138, 139, 150, 471, 498
access: 489, 490, 491, 492, 493
accessible: 497
accident: 104, 175, 183
```