

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

Кафедра інформатики та програмної інженерії

Мультипарадигмне програмування

## **ЗВІТ**

до лабораторної роботи №1

**Виконав**  
**студент**

Білько Олексій Євгенович  
(№ групи, прізвище, ім'я, по батькові )

**Прийняв**

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

Київ 2021

# 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**.

Для виконання даної лабораторної роботи було обрано мову C#

Програмний код:

```
// слова, які ми не вважаємо унікальними
string[] stopWords = {"about", "above", "across", "after", "against",
"along", "among", "around", "at",
"before", "behind", "below",
"beneath", "beside", "besides", "between", "beyond",
"but", "by", "concerning", "despite",
"down", "during", "except", "excepting", "for",
"from", "in", "inside", "into",
"like", "near", "of", "off", "on", "onto", "out",
"outside", "over", "past",
"regarding", "since", "through", "throughout", "to",
"toward", "under", "underneath",
"until", "up", "upon", "up to", "with", "within",
"without", "it", "is", "are", "am",
"the", "a", "an", "this", "that", "those", "these"};

string[] rows; // масив строк из текстового файла
using (StreamReader sr = new StreamReader("data_task1.txt"))
{
    int counter = 0;
    string row;
    getCountOfRowsLoop:
    if((row = sr.ReadLine()) != null)
    {
        if (row != "")
        {
            counter++;
        }
        goto getCountOfRowsLoop;
    }

    rows = new string[counter];
    sr.Close();
}
using (StreamReader sr = new StreamReader("data_task1.txt"))
{
    int counter = 0;
    string row;
    getEachRowFromFileLoop:
    if((row = sr.ReadLine()) != null)
    {
        if (row != "")
        {
            rows[counter] = row;
            counter++;
        }
    }
}
```

```

        }
        goto getEachRowFromFileLoop;
    }
    sr.Close();
}

string[] uniqueWords = new string[0]; // массив с уникальными словами
int[] countOfEachUniqueWord = new int[0]; // массив с кол-вом вхождений
каждого слова

int rowsIterator = 0;
rowsLoop:

string[] tmp = rows[rowsIterator].Split(" "); //разделяем строку на
слова

int wordsIterator = 0;
wordsLoop:

bool stopWord = false; // является ли данное слово стоп словом, если да
- на след. итерацию цикла
bool ifFound = false; // есть ли это слово в уникальных, если да -
обновляем кол-во, если нет - добавляем в уникальные

int stopWordsIterator = 0;
stopWordsLoop:

//к нижнему регистру все символы слова
char[] tmpStr = tmp[wordsIterator].ToCharArray();
for (int i = 0; i < tmpStr.Length; i++)
{
    if (tmpStr[i] >= 65 && tmpStr[i] <= 90)
        tmpStr[i] = (char)(tmpStr[i] + 32);
}
tmp[wordsIterator] = new string(tmpStr);
//
if (stopWords[stopWordsIterator] == tmp[wordsIterator]) //если нашли к
следующему слову
{
    stopWord = true;
}
stopWordsIterator++;
if (stopWordsIterator < stopWords.Length)
{
    goto stopWordsLoop;
}
if (!stopWord) // если данное слово не является стоп словом
{
    int ifAlreadyUniqueIterator = 0;
    ifAlreadyUniqueLoop:

    if (uniqueWords.Length != 0)
    {
        if (uniqueWords[ifAlreadyUniqueIterator] ==
tmp[wordsIterator])
        {
            countOfEachUniqueWord[ifAlreadyUniqueIterator]++;
            ifFound = true;
            goto IfFoundInUnique;
        }
    }
}

```

```

        ifAlreadyUniqueIterator++;
        if (ifAlreadyUniqueIterator < uniqueWords.Length)
        {
            goto ifAlreadyUniqueLoop;
        }

    IfFoundInUnique:

        if (!ifFound) // если не нашли, добавляем новое слово в массив
        уникальных слов и устанавливаем его количество как 1
        {
            string[] tmpUniqueWords = new string[uniqueWords.Length];
            int uniqueWordsToTmpIterator = 0;

            uniqueWordsToTmpLoop:

            if (tmpUniqueWords.Length != 0)
            {
                tmpUniqueWords[uniqueWordsToTmpIterator] =
uniqueWords[uniqueWordsToTmpIterator];
            }
            uniqueWordsToTmpIterator++;
            if (uniqueWordsToTmpIterator < tmpUniqueWords.Length)
            {
                goto uniqueWordsToTmpLoop;
            }

            uniqueWords = new string[uniqueWords.Length + 1];
            int fromTmpToUniqueWordsIterator = 0;

            fromTmpToUniqueWordsLoop:

            if (tmpUniqueWords.Length != 0)
            {
                uniqueWords[fromTmpToUniqueWordsIterator] =
tmpUniqueWords[fromTmpToUniqueWordsIterator];
            }
            fromTmpToUniqueWordsIterator++;
            if (fromTmpToUniqueWordsIterator < tmpUniqueWords.Length)
            {
                goto fromTmpToUniqueWordsLoop;
            }

            int[] tmpUniqueWordsCount = new
int[countOfEachUniqueWord.Length];
            int countOfUniqueToTmpIterator = 0;

            countOfUniqueToTmpLoop:

            if (tmpUniqueWordsCount.Length != 0)
            {
                tmpUniqueWordsCount[countOfUniqueToTmpIterator] =
countOfEachUniqueWord[countOfUniqueToTmpIterator];
            }
            countOfUniqueToTmpIterator++;
            if (countOfUniqueToTmpIterator <
tmpUniqueWordsCount.Length)
            {
                goto countOfUniqueToTmpLoop;
            }

```

```

        countOfEachUniqueWord = new
int[countOfEachUniqueWord.Length + 1];
        int fromTmpToCountOfUniqueIterator = 0;

        fromTmpToCountOfUniqueLoop:

            if (tmpUniqueWordsCount.Length != 0)
            {

                countOfEachUniqueWord[fromTmpToCountOfUniqueIterator] =
tmpUniqueWordsCount[fromTmpToCountOfUniqueIterator];
                }
                fromTmpToCountOfUniqueIterator++;
                if (fromTmpToCountOfUniqueIterator <
tmpUniqueWordsCount.Length)
                {
                    goto fromTmpToCountOfUniqueLoop;
                }
                uniqueWords[uniqueWords.Length - 1] = tmp[wordsIterator];
                countOfEachUniqueWord[countOfEachUniqueWord.Length - 1] =
1;
            }
        }
        wordsIterator++;
        if (wordsIterator < tmp.Length)
        {
            goto wordsLoop;
        }
        rowsIterator++;
        if (rowsIterator < rows.Length)
        {
            goto rowsLoop;
        }

        //сортируем массив слов и их кол-ва по убыванию кол-ва для отображения
на экран
        int outerIterator = 0;
        outerLoop:
        int innerIterator = countOfEachUniqueWord.Length - 1;
        innerLoop:
        if (countOfEachUniqueWord[innerIterator] >
countOfEachUniqueWord[innerIterator - 1])
        {
            int tmp__ = countOfEachUniqueWord[innerIterator - 1];
            countOfEachUniqueWord[innerIterator - 1] =
countOfEachUniqueWord[innerIterator];
            countOfEachUniqueWord[innerIterator] = tmp__;

            string tmp_ = uniqueWords[innerIterator - 1];
            uniqueWords[innerIterator - 1] = uniqueWords[innerIterator];
            uniqueWords[innerIterator] = tmp_;
        }
        innerIterator--;
        if (innerIterator > outerIterator)
        {
            goto innerLoop;
        }
        outerIterator++;
        if (outerIterator < countOfEachUniqueWord.Length)
        {
            goto outerLoop;
        }

```

```

    }

    //вывод либо первых 25, либо всех уникальных слов если их меньше 25
    int iter = 0;
    if (uniqueWords.Length >= 25)
        iter = 25;
    else iter = uniqueWords.Length;

    int printIterator = 0;
    printLoop:
        Console.WriteLine(uniqueWords[printIterator] + " - " +
countOfEachUniqueWord[printIterator].ToString());
        printIterator++;
        if(printIterator < iter)
        {
            goto printLoop;
        }
    }
}

```

## Вхідний файл

Результат роботи програми:

```

63
64
65 he - 138
66 was - 107
67 and - 104
68 his - 79
69 had - 65
70 as - 43
71 harry - 40
72 dursley - 36
73 dumbledore - 26
74 mr - 25
75 were - 25
76 they - 25
77 all - 25
78 him - 24
79 didnt - 19
80 have - 19
81 been - 19
82 mrs - 18
83 them - 18
84 very - 17
85 be - 16
86 there - 16
87 looked - 16
88 back - 15
89 people - 14
90

```

Алгоритм роботи програми:

1. Зчитати дані з текстового файлу в масив рядків
2. Пробігаючись по кожному слову, перевіряти, чи є воно стоп словом, якщо є – перейти до наступного
3. Якщо слово не є стоп словом перевірити чи є воно в масиві унікальних, якщо так, збільшити кількість в масиві чисел, де відповідна цифра відповідає відповідному слову, якщо ж ще не має, то додати його в масив унікальних і в масиві чисел поставити кількість 1
4. Відсортувати за спаданням слова по кількості їх входжень
5. Вивести на екран перші 25 слів і кількість їх входжень (або всі, якщо їх менше 25)

## 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.

Програмний код:

```
// слова які ми не вважаємо унікальними
string[] stopWords = {"as", "about", "above", "across", "after",
"against", "along", "among", "around", "at",
"before", "behind", "below",
"beneath", "beside", "besides", "between", "beyond",
"but", "by", "concerning", "despite",
"down", "during", "except", "excepting", "for",
"from", "in", "inside", "into",
"like", "near", "of", "off", "on", "onto", "out",
"outside", "over", "past",
"regarding", "since", "through", "throughout", "to",
"toward", "under", "underneath",
"until", "up", "upon", "up to", "with", "within",
"without", "it", "is", "are", "am",
"the", "a", "an", "this", "that", "those", "these"};

string[] rows; // масив строк из текстового файла
using (StreamReader sr = new StreamReader("data_task2.txt"))
{
    int counter = 0;
    string row;
getCountOfRowsLoop:
    if ((row = sr.ReadLine()) != null)
    {
        if (row != "")
        {
            counter++;
        }
        goto getCountOfRowsLoop;
    }

    rows = new string[counter];
    sr.Close();
}
using (StreamReader sr = new StreamReader("data_task2.txt"))
{
    int counter = 0;
    string row;
getEachRowFromFileLoop:
    if ((row = sr.ReadLine()) != null)
    {
        if (row != "")
        {
            rows[counter] = row;
            counter++;
        }
        goto getEachRowFromFileLoop;
    }
}
```

```

        sr.Close();
    }

    string[] uniqueWords = new string[0]; // массив с уникальными словами
    string[] pageOfEachUniqueWord = new string[0]; // массив с страницами
каждого уникального слова

    int rowsIterator = 0;
rowsLoop:

    string[] tmp = rows[rowsIterator].Split(" "); //разделяем строку на
слова

    int wordsIterator = 0;
wordsLoop:

    bool stopWord = false; // является ли данное слово стоп словом, если да
- на след. итерацию цикла
    bool ifFound = false; // есть ли это слово в уникальных, если да -
обновляем кол-во, если нет - добавляем в уникальные

    int stopWordsIterator = 0;
stopWordsLoop:

    //к нижнему регистру все символы слова
    char[] tmpStr = tmp[wordsIterator].ToCharArray();
    for (int i = 0; i < tmpStr.Length; i++)
    {
        if (tmpStr[i] >= 65 && tmpStr[i] <= 90)
            tmpStr[i] = (char)(tmpStr[i] + 32);
    }
    tmp[wordsIterator] = new string(tmpStr);
    //
    if (stopWords[stopWordsIterator] == tmp[wordsIterator]) //если нашли к
следующему слову
    {
        stopWord = true;
    }
    stopWordsIterator++;
    if (stopWordsIterator < stopWords.Length)
    {
        goto stopWordsLoop;
    }
    if (!stopWord) // если данное слово не является стоп словом
    {
        int ifAlreadyUniqueIterator = 0;
        ifAlreadyUniqueLoop:

        if (uniqueWords.Length != 0)
        {
            if (uniqueWords[ifAlreadyUniqueIterator] ==
tmp[wordsIterator]) //если нашли, обновляем кол-во и выходим из массива
            {
                int page = rowsIterator / 45 + 1;
                pageOfEachUniqueWord[ifAlreadyUniqueIterator] +=
(", " + page.ToString());

                ifFound = true;
                goto IfFoundInUnique;
            }
        }
        ifAlreadyUniqueIterator++;
    }

```



```

        if (ifAlreadyUniqueIterator < uniqueWords.Length)
        {
            goto ifAlreadyUniqueLoop;
        }

    IfFoundInUnique:

        if (!ifFound) // если не нашли, добавляем новое слово в массив
        уникальных слов и устанавливаем его количество как 1
        {
            string[] tmpUniqueWords = new string[uniqueWords.Length];
            int uniqueWordsToTmpIterator = 0;

            uniqueWordsToTmpLoop:

                if (tmpUniqueWords.Length != 0)
                {
                    tmpUniqueWords[uniqueWordsToTmpIterator] =
uniqueWords[uniqueWordsToTmpIterator];
                }
                uniqueWordsToTmpIterator++;
                if (uniqueWordsToTmpIterator < tmpUniqueWords.Length)
                {
                    goto uniqueWordsToTmpLoop;
                }

                uniqueWords = new string[uniqueWords.Length + 1];
                int fromTmpToUniqueWordsIterator = 0;

            fromTmpToUniqueWordsLoop:

                if (tmpUniqueWords.Length != 0)
                {
                    uniqueWords[fromTmpToUniqueWordsIterator] =
tmpUniqueWords[fromTmpToUniqueWordsIterator];
                }
                fromTmpToUniqueWordsIterator++;
                if (fromTmpToUniqueWordsIterator < tmpUniqueWords.Length)
                {
                    goto fromTmpToUniqueWordsLoop;
                }

                string[] tmpUniqueWordsCount = new
string[pageOfEachUniqueWord.Length];
                int countOfUniqueToTmpIterator = 0;

            countOfUniqueToTmpLoop:

                if (tmpUniqueWordsCount.Length != 0)
                {
                    tmpUniqueWordsCount[countOfUniqueToTmpIterator] =
pageOfEachUniqueWord[countOfUniqueToTmpIterator];
                }
                countOfUniqueToTmpIterator++;
                if (countOfUniqueToTmpIterator <
tmpUniqueWordsCount.Length)
                {
                    goto countOfUniqueToTmpLoop;
                }

```

```

        pageOfEachUniqueWord = new
string[pageOfEachUniqueWord.Length + 1];
        int fromTmpToCountOfUniqueIterator = 0;

        fromTmpToCountOfUniqueLoop:

            if (tmpUniqueWordsCount.Length != 0)
            {

                pageOfEachUniqueWord[fromTmpToCountOfUniqueIterator] =
tmpUniqueWordsCount[fromTmpToCountOfUniqueIterator];
            }
            fromTmpToCountOfUniqueIterator++;
            if (fromTmpToCountOfUniqueIterator <
tmpUniqueWordsCount.Length)
            {
                goto fromTmpToCountOfUniqueLoop;
            }

            uniqueWords[uniqueWords.Length - 1] = tmp[wordsIterator];
            int page = rowsIterator / 45 + 1;
            if (pageOfEachUniqueWord[pageOfEachUniqueWord.Length - 1]
== null)
                pageOfEachUniqueWord[pageOfEachUniqueWord.Length -
1] = page.ToString();
            else pageOfEachUniqueWord[pageOfEachUniqueWord.Length - 1]
= ", " + page.ToString();
        }
    }
    wordsIterator++;
    if (wordsIterator < tmp.Length)
    {
        goto wordsLoop;
    }
    rowsIterator++;
    if (rowsIterator < rows.Length)
    {
        goto rowsLoop;
    }

    //сортировка по алфавиту
    int outerIterator = 0;
    outerLoop:

        int innerIterator = 0;
        innerLoop:

            bool toReplace = false; // меняем ли мы слова при сортировке
            int lengthToCompare;
            if(uniqueWords[innerIterator].Length > uniqueWords[innerIterator +
1].Length)
            {
                lengthToCompare = uniqueWords[innerIterator + 1].Length;
            }
            else lengthToCompare = uniqueWords[innerIterator].Length;

            char[] charsFromPrev = uniqueWords[innerIterator + 1].ToCharArray();
            char[] charsFromNext = uniqueWords[innerIterator].ToCharArray();

            int compareCharsIterator = 0;

```

```

compareCharsLoop:

    if (charsFromNext[compareCharsIterator] >
charsFromPrev[compareCharsIterator])
    {
        toReplace = true;
        goto toReplaceLabel;
    }
    if (charsFromNext[compareCharsIterator] <
charsFromPrev[compareCharsIterator])
    {
        toReplace = false;
        goto toReplaceLabel;
    }
    compareCharsIterator++;
    if(compareCharsIterator < lengthToCompare)
{
        goto compareCharsLoop;
}

toReplaceLabel:

if (toReplace)
{
    string tmpUniqueWord = uniqueWords[innerIterator];
    uniqueWords[innerIterator] = uniqueWords[innerIterator + 1];
    uniqueWords[innerIterator + 1] = tmpUniqueWord;

    string tmpPagesOfUniqueWord = pageOfEachUniqueWord[innerIterator];
    pageOfEachUniqueWord[innerIterator] = pageOfEachUniqueWord[innerIterator +
1];
    pageOfEachUniqueWord[innerIterator + 1] = tmpPagesOfUniqueWord;
}
innerIterator++;
if (innerIterator < uniqueWords.Length - outerIterator - 1)
{
    goto innerLoop;
}
outerIterator++;
if (outerIterator < uniqueWords.Length)
{
    goto outerLoop;
}

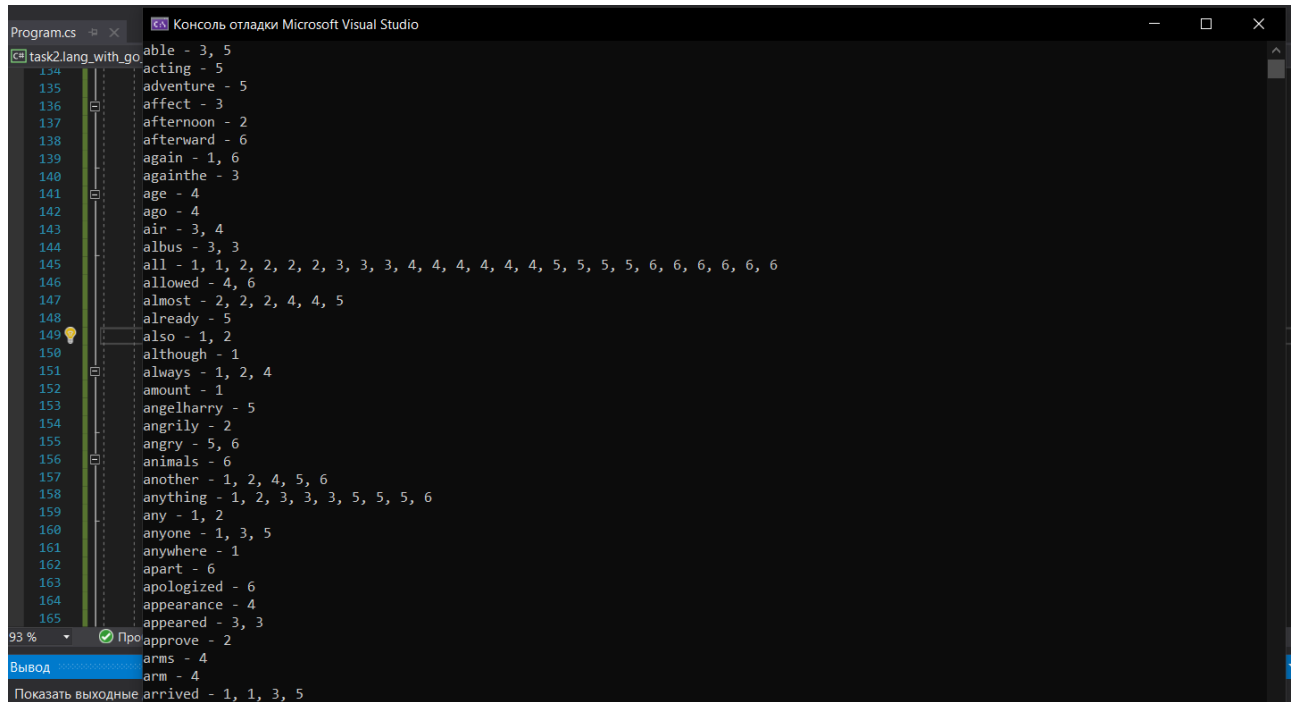
//Вывод на консоль результатов
int printIterator = 0;
printLoop:

    if (pageOfEachUniqueWord[printIterator].Split(", ").Length <= 100)
    {
        Console.WriteLine(uniqueWords[printIterator] + " - " +
pageOfEachUniqueWord[printIterator]);
    }
    printIterator++;
    if (printIterator < uniqueWords.Length)
    {
        goto printLoop;
    }

```

## Вхідний файл

Результат роботи програми:



```
Program.cs | task2.lang_with_go | Console | Консоль отладки Microsoft Visual Studio
134 | able - 3, 5
135 | acting - 5
136 | adventure - 5
137 | affect - 3
138 | afternoon - 2
139 | afterward - 6
140 | again - 1, 6
141 | againthe - 3
142 | age - 4
143 | ago - 4
144 | air - 3, 4
145 | albus - 3, 3
146 | all - 1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, 6, 6
147 | allowed - 4, 6
148 | almost - 2, 2, 2, 2, 4, 4, 5
149 | already - 5
150 | also - 1, 2
151 | although - 1
152 | always - 1, 2, 4
153 | amount - 1
154 | angelharry - 5
155 | angrily - 2
156 | angry - 5, 6
157 | animals - 6
158 | another - 1, 2, 4, 5, 6
159 | anything - 1, 2, 3, 3, 3, 5, 5, 5, 6
160 | any - 1, 2
161 | anyone - 1, 3, 5
162 | anywhere - 1
163 | apart - 6
164 | apologized - 6
165 | appearance - 4
166 | appeared - 3, 3
167 | approve - 2
168 | arms - 4
169 | arm - 4
170 | arrived - 1, 1, 3, 5
```

Алгоритм роботи програми:

1. Зчитати дані з текстового файлу в масив рядків
2. Пробігаючись по кожному слову, перевіряти, чи є воно стоп словом, якщо є – перейти до наступного
3. Якщо слово не є стоп словом перевірити чи є воно в масиві унікальних, якщо так, додати номер сторінки в масив рядків (вважаємо, що одна сторінка – 45 рядків), де відповідний рядок с номерами сторінок відповідає відповідному слову, якщо ж ще не має, то додати його в масив унікальних і в масиві рядків з сторінками додати сторінку, на якій відповідне слово розташоване
4. Відсортувати слова по алфавіту
5. Вивести на екран результат роботи у вигляді [Слово – номер сторінки, номер сторінки]

Для виконання даних завдань були використані: вбудована функція `string.ToArray` і `string.Split` для приведення типу `string` до типу `char[]` і розділення рядка на слова по пробілу відповідно, функція `int.ToString` для приведення типу `int` в тип `string`, а також клас `StreamReader` для зчитування інформації з текстового файлу.