Мультипарадигменне Програмування

Лабораторна робота 1

Тема: імперативне програмування

Виконав: студент группи ІП-02 Красновський С. С.

Звіт

Для виконання даної лабораторної роботи мною було обрано мову програмування C#, так як вона підтримує конструкцію goto (це ключова умова для виконання даної лабораторної роботи).

Також у коді було використано тільки ключові функції для роботи зі зміною типів та обробки рядків.

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**.

```
using System;

using System.IO;

using System.Text;


namespace task1

{

class Program

{

static void Main(string[] args)

{

string[] unnecessary_words =

{"about", "above", "after", "at","between",

"for", "from", "in", "in front of", "inside", "in spite of", "instead of", "into",

"like", "near", "of", "off", "on", "onto", "out", "outside", "over", "past",
"regarding",

"to", "toward", "under", "underneath", "until", "up",

"upon", "up to", "with", "it", "is", "are", "am", "the", "a", "an", "this", "that",
"those", "theese"};
```

```
string[] rows;
using (StreamReader file = new StreamReader("Task1.txt"))
{
int counter = 0;
string row;
while1:
if ((row = file.ReadLine()) != null)
{
counter++;
goto while1;
}
rows = new string[counter];
file.DiscardBufferedData();
file.BaseStream.Seek(0, SeekOrigin.Begin);
counter = 0;
row = "";
while2:
if ((row = file.ReadLine()) != null)
{
rows[counter] = row;
counter++;
goto while2;
}
file.Close();
}
```

```
string[] unwritten_word = new string[0];
int[] NumberOfEachUnique = new int[0];

int ii = 0;
for_1:
string[] tmp = rows[ii].Split(" ");
if (tmp[0] != "")
{
int j1 = 0;
for_2:
bool FoundCase = false;
bool unnecessary_word = false;
int tolowit = 0;
byte[] asciiBytes1 = Encoding.ASCII.GetBytes(tmp[j1]);
for_Tolow:
if (tmp[j1][tolowit] >= 65 && tmp[j1][tolowit] <= 90)
{
asciiBytes1[tolowit] += 32;
}
tolowit++;
if (tolowit < tmp[j1].Length)
{
goto for_Tolow;
}
int l = 0;
string word = Encoding.ASCII.GetString(asciiBytes1);
for_3:
```

```
if (unnecessary_words[l] == word)
{
unnecessary_word = true;
}
l++;
if (l < unnecessary_words.Length)
{
goto for_3;
}
if (!unnecessary_word)
{
int k = 0;
for_4:
if (unwritten_word.Length != 0)
{
if (unwritten_word[k] == word)
{
NumberOfEachUnique[k]++;
FoundCase = true;
goto mark;
}
}
k++;
if (k < unwritten_word.Length)
{
goto for_4;
}
```

```
mark:
if (!FoundCase)
{
string[] tmpUniqueWords = new string[unwritten_word.Length];
int p1 = 0;
for_5:
if (tmpUniqueWords.Length != 0)
{
tmpUniqueWords[p1] = unwritten_word[p1];
}
p1++;
if (p1 < tmpUniqueWords.Length)
{
goto for_5;
}
unwritten_word = new string[unwritten_word.Length + 1];
int o1 = 0;
for_6:
if (tmpUniqueWords.Length != 0)
{
unwritten_word[o1] = tmpUniqueWords[o1];
}
o1++;
if (o1 < tmpUniqueWords.Length)
{
goto for_6;
}
```

```
int[] tmpUniqueWordsCount = new int[NumberOfEachUnique.Length];
int p2 = 0;
for_7:
if (tmpUniqueWordsCount.Length != 0)
{
tmpUniqueWordsCount[p2] = NumberOfEachUnique[p2];
}
p2++;
if (p2 < tmpUniqueWordsCount.Length)
{
goto for_7;
}
NumberOfEachUnique = new int[NumberOfEachUnique.Length + 1];
int o2 = 0;
for_8:

if (tmpUniqueWordsCount.Length != 0)
{
NumberOfEachUnique[o2] = tmpUniqueWordsCount[o2];
}
o2++;
if (o2 < tmpUniqueWordsCount.Length)
{
goto for_8;
}
```

```
unwritten_word[unwritten_word.Length - 1] = word;
NumberOfEachUnique[NumberOfEachUnique.Length - 1] = 1;
}
}
j1++;
if (j1 < tmp.Length)
{
goto for_2;
}
}
ii++;
if (ii < rows.Length)
{
goto for_1;
}
int i2 = 0;
for_9:
int j2 = NumberOfEachUnique.Length - 1;
for_10:
if (NumberOfEachUnique[j2] > NumberOfEachUnique[j2 - 1])
{
int temporary__ = NumberOfEachUnique[j2 - 1];
NumberOfEachUnique[j2 - 1] = NumberOfEachUnique[j2];
NumberOfEachUnique[j2] = temporary__;

string temporary_ = unwritten_word[j2 - 1];
```

```
unwritten_word[j2 - 1] = unwritten_word[j2];
unwritten_word[j2] = temporary_;
}
j2--;
if (j2 > i2)
{
goto for_10;
}
i2++;
if (i2 < NumberOfEachUnique.Length)
{
goto for_9;
}
int iter = 0;
if (unwritten_word.Length >= 25)
iter = 25;
else iter = unwritten_word.Length;
int i3 = 0;
for_11:
Console.WriteLine(unwritten_word[i3].ToString() + " - " +
NumberOfEachUnique[i3].ToString());
i3++;
if(i3 < iter)
{
goto for_11;
}


}
```

```
}
}
```

Опис програми

1. Створюємо масив слів unnecessary_words, які треба пропустити;
2. Зчитуємо у масив rows строки, під кожну нову строку новий індекс;
3. Створюємо масив слів, у якому слова не повторюються, а також масив кількості кожного слова;
4. Створюємо масив слів з кожної нової строки і починаємо перебором проходити по кожному слову;
5. При переборі слів зводимо їх до нижнього регістру і перевіряємо слово на належність до небажаних слів;
6. Перевіряємо чи немає слова у масиві унікальних слів, якщо є, то збульшуємо його кількість, а якщо немає розширюємо масив та ставимо його кількість як 1;
7. Після перебору всіх строк, сортуємо їх бульбашкою за кількістю;
8. Виводимо перші слова, та кількість не має перевищувати 25. При перевищені вивести перші 25 слів.

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.

```
using System;
using System.IO;
using System.Text;

namespace task2
{
class Program
{
static void Main(string[] args)
{
string[] unnecessary_words =
{"about", "above", "after", "at","between",
"for", "from", "in", "in front of", "inside", "in spite of", "instead of",
"into",
"like", "near", "of", "off", "on", "onto", "out", "outside", "over", "past",
"regarding",
"to", "toward", "under", "underneath", "until", "up",
"upon", "up to", "with", "it", "is", "are", "am", "the", "a", "an", "this",
"that", "those", "theese"};

string[] rows;
using (StreamReader file = new StreamReader("task2.txt"))
{
```

```
int counter = 0;
string row;
while1:
if((row = file.ReadLine()) != null)
{
counter++;
goto while1;
}
rows = new string[counter];
file.DiscardBufferedData();
file.BaseStream.Seek(0, SeekOrigin.Begin);
counter = 0;
row = "";
while2:
if((row = file.ReadLine()) != null)
{
rows[counter] = row;
counter++;
goto while2;
}
file.Close();
}

string[] unwritten_word = new string[0];
string[] CountOfUniques = new string[0];
int i1 = 0;
for_1:
string[] tmp = rows[i1].Split(" ");
if (tmp[0]!= "")
{
int j1 = 0;
for_2:
bool FoundCase = false;
bool unnecessary_word = false;
int tolowit = 0;
byte[] asciiBytes = Encoding.ASCII.GetBytes(tmp[j1]);
for_Tolow:
if (tmp[j1][tolowit] >= 65 && tmp[j1][tolowit] <= 90)
{
asciiBytes[tolowit] += 32;
}
tolowit++;
if (tolowit < tmp[j1].Length)
{
goto for_Tolow;
}
int l1 = 0;
string word = Encoding.ASCII.GetString(asciiBytes);
for_3:
if (unnecessary_words[l1] == word)
{
unnecessary_word = true;
}
l1++;
if (l1 < unnecessary_words.Length)
{
goto for_3;
}
```

```
if (!unnecessary_word)
{
int k1 = 0;
for_4:
if (unwritten_word.Length != 0)
if (unwritten_word[k1] == word)
{
int NumberOfPage = i1 / 45 + 1;
CountOfUniques[k1] += (", " + NumberOfPage.ToString());
FoundCase = true;
goto mark;
}
k1++;
if (k1 < unwritten_word.Length)
{
goto for_4;
}
mark:
if (!FoundCase)
{
string[] tmpUniqueWords = new string[unwritten_word.Length];
int p1 = 0;
for_5:
if (tmpUniqueWords.Length != 0)
{
tmpUniqueWords[p1] = unwritten_word[p1];
}
p1++;
if (p1 < tmpUniqueWords.Length)
{
goto for_5;
}
unwritten_word = new string[unwritten_word.Length + 1];
int o1 = 0;
for_6:
if (tmpUniqueWords.Length != 0)
{
unwritten_word[o1] = tmpUniqueWords[o1];
}
o1++;
if (o1 < tmpUniqueWords.Length)
{
goto for_6;
}

string[] tmpUniqueWordsCount = new string[CountOfUniques.Length];
int p2 = 0;
for_7:
if (tmpUniqueWordsCount.Length != 0)
{
tmpUniqueWordsCount[p2] = CountOfUniques[p2];
}
p2++;
if (p2 < tmpUniqueWordsCount.Length)
{
goto for_7;
}
CountOfUniques = new string[CountOfUniques.Length + 1];
int o2 = 0;
```

```
for_8:
if (tmpUniqueWordsCount.Length != 0)
CountOfUniques[o2] = tmpUniqueWordsCount[o2];
o2++;
if (o2 < tmpUniqueWordsCount.Length)
{
goto for_8;
}

unwritten_word[unwritten_word.Length - 1] = word;
int NumberOfPage = i1 / 45 + 1;
if (CountOfUniques[CountOfUniques.Length - 1] == null)
{
CountOfUniques[CountOfUniques.Length - 1] = NumberOfPage.ToString();
}
else
{
CountOfUniques[CountOfUniques.Length - 1] = ", " + NumberOfPage.ToString();
}
}
}
j1++;
if (j1 < tmp.Length)
{
goto for_2;
}
}
i1++;
if(i1<rows.Length)
{
goto for_1;
}
int i2 = 0;
for_9:
int j2 = unwritten_word.Length - 1;
for_10:
bool toReplace = false;
int length = unwritten_word[j2].Length > unwritten_word[j2 - 1].Length ?
unwritten_word[j2 - 1].Length : unwritten_word[j2].Length;

byte[] ByteWord1 = Encoding.ASCII.GetBytes(unwritten_word[j2 - 1]);
byte[] ByteWord2 = Encoding.ASCII.GetBytes(unwritten_word[j2]);
for (int o = 0; o < length; o++)
{
if (ByteWord1[o] > ByteWord2[o])
{
toReplace = true;
break;
}
if (ByteWord1[o] < ByteWord2[o])
{
toReplace = false;
break;
}
}
if (toReplace)
{
string s = unwritten_word[j2];
unwritten_word[j2] = unwritten_word[j2 - 1];
```

```
unwritten_word[j2 - 1] = s;

string pageInfo = CountOfUniques[j2];
CountOfUniques[j2] = CountOfUniques[j2 - 1];
CountOfUniques[j2 - 1] = pageInfo;
}
j2--;
if(j2>i2)
{
goto for_10;
}
i2++;
if(i2 < unwritten_word.Length)
{
goto for_9;
}



int iter = 0;
if (unwritten_word.Length >= 200)
iter = 200;
else iter = unwritten_word.Length;
int i3 = 0;
for11:
if (CountOfUniques[i3].Split(", ").Length <= 100 )
Console.WriteLine(unwritten_word[i3].ToString() + " - " +
CountOfUniques[i3].ToString());
i3++;
if(i3<iter)
{
goto for11;
}
}
}
}
```

Опис програми

1. Створюємо масив слів unnecessary_words, які треба пропустити;
2. Зчитуємо у масив rows строки, під кожну нову строку новий індекс;
3. Створюємо масив слів, у якому слова не повторюються, а також масив кількості кожного слова;
4. Створюємо масив слів з кожної нової строки і починаємо перебором проходити по кожному слову;
5. При переборі слів зводимо їх до нижнього регістру і перевіряємо слово на належність до небажаних слів;
6. Перевіряємо чи немає слова у масиві унікальних слів, якщо є, то записуємо сторінку на якій воно зустрілось, а якщо немає розширюємо масив та ставимо його сторінку;
7. Після перебору всіх строк, сортуємо їх за алфавіторм,;
8. виводимо до 250 перших слів.

Висновок

Під час виконання лабораторної роботи ми ознайомились з імперативним програмуванням та відчули, як це, писати код як наші пращури у 1050х роках. Також у ході виконная даної лабораторної роботи було написано дві прості программи без використання циклів та функцій, лише за допомогою операторів goto.