

Міністерство освіти і науки України Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського" Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Лабораторна робота №1

Імперативне програмування

Мультипарадигменне програмування

Виконав студент групи IT-03:	Перевірив:
Чабан А.Є.	
	Очеретяний О. К

Київ 2021

Мета роботи:

Практична робота складається із трьох завдань, які самі по собі є досить простими. Але, оскільки задача - зрозуміти, як писали код наші славні пращури у 1950-х, ми введемо кілька обмежень:

Заборонено використовувати функції

Заборонено використовувати цикли

Для виконання потрібно взяти мову, що підтримує конструкцію GOTO

Завдання:

Код на GitHub

Алгоритм завдання 1:

Зчитуємо файл з забороненими словами поки останнім словом не буде «» (пустота – означає файл закінчився) тоді ми перейдемо на мітку readerLabel де будемо читати основний файл

Мітка readerLabel

Зчитуємо файл з текстом поки останнім словом не буде «» (пустота – означає файл закінчився) тоді ми перейдемо на мітку Finish checkWordLabel:

Якщо букви по ascii коду знаходяться в проміжку великих літер – замінюємо їх на маленькі додавши 32

Якщо останній символ зчитаного слова в проміжку розділових знаків — замінюємо його на null, а далі переписуємо у менший масив, щоб мати масив слів без розділових знаків

Якщо слово знаходиться в масиві заборонених слів — анулюємо лічильник і продовжуємо сканування наступних слів

Якщо слово вже є в масиві збільшуємо значення num (кількість слів) і продовжуємо сканування текстового файлу з наступного слова Якщо ніяка з умов під час ітерування не виконана – додаємо слово до масиву

Мітка Finish:

Відбувається сортування бульбашкою за значенням кількості слів Мітка printlabel виводимо або всі значення, або 20 найпопулярніших.

Результат виконання завдання #1

```
######
{ut 173}
{in 161}
{et 145}
{sit 143}
{amet 143}
{eu 143}
{a 136}
{quis 131}
{vel 128}
{nec 128}
{id 125}
{ac 122}
{at 121}
{nunc 118}
{non 115}
{eget 115}
{nulla 112}
{vitae 111}
{mauris 103}
{tincidunt 99}
```

Алгоритм завдання 2:

Зчитуємо файл з забороненими словами поки останнім словом не буде «» (пустота – означає файл закінчився) тоді ми перейдемо на мітку readerLabel де будемо читати основний файл

Мітка readerLabel

Зчитуємо файл з текстом поки останнім словом не буде «» (пустота – означає файл закінчився) тоді ми перейдемо на мітку Finish

Якщо лічильник слів на сторінці дорівнює константі розміру сторінки – позначаємо нову сторінку

checkWordLabel:

Якщо букви по ascii коду знаходяться в проміжку великих літер – замінюємо їх на маленькі додавши 32

Якщо останній символ зчитаного слова в проміжку розділових знаків — замінюємо його на null, а далі переписуємо у менший масив, щоб мати масив слів без розділових знаків

Якщо слово знаходиться в масиві заборонених слів — анулюємо лічильник і продовжуємо сканування наступних слів

Якщо слово вже є в масиві збільшуємо значення count (кількість слів), додаємо сторінку на якій знаходиться слово до масиву і продовжуємо сканування текстового файлу з наступного слова

Якщо ніяка з умов під час ітерування не виконана – додаємо слово до масиву та записуємо на якій воно сторінці

Мітка Finish:

Відбувається сортування бульбашкою за алфавітним порядком Мітка printlabel виводимо або всі значення, або 20 найпопулярніших, при чому кількість повторень не більше 100.

Результат виконання завдання #2

38388
ad 12 [2 4 4 5 8 11 12 13 14 14 15 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
adipiscing 4 [1 2 6 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
aenean 40 [1 1 1 1 1 2 2 2 3 4 5 6 6 7 7 7 8 8 9 9 10 10 10 11 12 12 13 14 15 16 16 16 16 16 16 16 17 17 17 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
aliquam 86 [1 1 1 2 2 2 2 2 3 3 3 3 3 3 3 4 4 4 5 5 5 5 5 6 6 6 6 6 6 6 6 7 7 8 8 9 9 9 9 9 10 10 11 11 11 11 11 12 12 12 12 12 12 12 12
aliquet 40 [1 1 1 1 2 2 3 3 3 3 3 4 4 4 4 4 4 5 5 6 6 6 7 10 10 10 12 12 13 13 14 15 16 17 17 17 18 18 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ante 74 [1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 4 4 4 4 4 5 5 5 5 5 5 5 6 6 7 7 8 8 8 8 8 9 9 9 9 10 10 10 11 11 12 12 12 13 13 13 13 13 13 14 14 15 15 15 16 16 16 16 16 16 16 16 17 17 18 19 19 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
aptent 12 [2 4 4 5 8 11 12 13 14 14 15 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
arcu 48 [1 2 2 3 4 4 5 5 5 6 6 6 7 7 7 7 8 8 9 9 9 10 10 10 11 11 11 12 12 12 12 13 13 14 14 15 16 16 17 18 18 18 19 19 19 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0
auctor 49 [1 1 2 2 2 3 4 5 6 6 6 6 6 8 8 8 9 9 9 10 10 11 11 11 11 12 12 12 13 13 13 13 14 14 14 15 16 16 17 18 18 19 19 19 20 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
augue 62 [1 1 1 1 2 2 2 3 4 4 4 4 4 5 5 6 7 7 7 7 8 8 8 9 9 9 9 10 10 10 11 11 11 11 12 12 12 12 13 13 14 14 14 15 16 16 17 17 17 17 18 18 18 19 19 19 19 19 19 19 19 0 0 0 0 0 0 0 0
bibendum 44 [1 2 3 4 4 4 4 4 5 5 5 5 6 6 7 7 7 8 8 8 9 10 11 12 12 12 13 13 13 13 13 14 14 14 14 15 16 16 16 17 17 19 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
blandit 45 [1 1 1 2 2 3 3 4 4 4 5 6 6 8 8 8 8 9 9 10 10 11 11 11 12 12 13 13 13 14 14 14 15 15 15 15 16 16 17 17 17 17 19 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
class 12 [2 4 4 5 8 11 12 13 14 14 15 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
commodo 45 [1 1 3 3 4 4 4 5 6 7 7 7 7 9 9 9 9 10 11 11 12 12 12 12 13 13 13 13 14 14 15 16 16 16 16 16 17 17 17 18 18 18 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
condimentum 42 [1 1 2 3 3 4 5 6 6 7 7 7 8 8 9 9 9 10 11 12 12 14 15 15 15 15 15 15 15 15 15 16 17 17 18 18 18 18 19 19 19 19 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
congue 42 [1 2 2 3 3 3 3 4 4 5 5 6 6 7 8 8 8 8 9 9 10 10 11 11 12 12 13 13 14 14 14 15 16 16 16 17 17 18 18 19 19 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Висновок:

Під час виконання даної роботи я ознайомився з основами імперативного програмування та конструкцією GOTO.