



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «Київський
політехнічний інститут імені Ігоря Сікорського» Факультет інформатики
та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт до лабораторної роботи №1 з дисципліни «Мультипарадигменне програмування»

Виконав:
Студент групи ІТ-01
Куркін О. О.

Київ – 2022

Программный код:

```
class WordFrequency:
    def __init__(self, word, amount):
        self.word = word
        self.amount = amount

@with_goto
def main():
    # reading file
    string = []
    with open('text.txt', 'r') as file: string = list(file.read())

    # count file length
    stringLength = 0
    label .countStringLength
    try:
        string[stringLength]
        stringLength += 1
        goto .countStringLength
    except IndexError:
        pass

    # dividing file string into words
    words = [0]*(stringLength//3)
    words = []
    word = [0]*30
    lastSymbolIndex = 0
    i = 0

    label .dividingFileStringInWords
    if string[i] != '\n' and string[i] != ' ' and string[i] != '.' and string[i] !=
    ',' and string[i] != '?' and string[i] != '!':
        word[lastSymbolIndex] = string[i]
        lastSymbolIndex += 1
        if i == stringLength - 1:
            goto .addWordToTheList
    else:
        label .addWordToTheList
        wordLength = 0
        label .countWordLength
        if word[wordLength] != 0:
            wordLength += 1
            goto .countWordLength

        if wordLength > 3:
```

```

        tempWordIndex = 0
        tempWord = ""
        label .createTempWord
        asciiCharToAd = ord(word[tempWordIndex])
        if(asciiCharToAd>64 and asciiCharToAd<91):
            asciiCharToAd+=32
        tempWord += chr(asciiCharToAd)
        tempWordIndex += 1
        if tempWordIndex < wordLength:
            goto .createTempWord
        words.append(tempWord)
    word = [0]*30
    lastSymbolIndex = 0

i += 1
if i < stringLength:
    goto .dividingFileStringInWords

# count amount of words
wordsAmount = 0
label .countWordsAmount
try:
    words[wordsAmount]
    wordsAmount += 1
    goto .countWordsAmount
except IndexError:
    pass

# wordsFrequencyArray generating
wordsFrequencyArray = [0]*wordsAmount
wordIndex = 0
i = 0
label .wordsFrequencyArrayGenerating
if wordsFrequencyArray[i] == 0:
    wordsFrequencyArray[i] = WordFrequency(words[wordIndex], 1)
    wordIndex += 1
    if wordIndex >= wordsAmount:
        goto .endOfWordsFrequencyArrayGenerating
    i = 0
    goto .wordsFrequencyArrayGenerating
else:
    if wordsFrequencyArray[i].word == words[wordIndex]:
        wordsFrequencyArray[i].amount += 1
        wordIndex += 1
        if wordIndex >= wordsAmount:

```

```

        goto .endOfWordsFrequencyArrayGenerating
    i = 0
    goto .wordsFrequencyArrayGenerating

    i += 1
    if i < wordsAmount:
        goto .wordsFrequencyArrayGenerating

label .endOfWordsFrequencyArrayGenerating

# removing zeros from array
wordsFrequencyAmount = 0
label .countWordsFrequency
if wordsFrequencyArray[wordsFrequencyAmount] != 0:
    wordsFrequencyAmount += 1
    goto .countWordsFrequency

cleanWordsFrequencyArray = [0]*wordsFrequencyAmount
i = 0
label .generateCleanWordsFrequencyArray
cleanWordsFrequencyArray[i] = wordsFrequencyArray[i]
i += 1
if i < wordsFrequencyAmount:
    goto .generateCleanWordsFrequencyArray

# sorting array
i = 0
label .bubbleSortFirstStage
j = 0
label .bubbleSortSecondStage
if cleanWordsFrequencyArray[j].amount < cleanWordsFrequencyArray[j+1].amount:
    cleanWordsFrequencyArray[j], cleanWordsFrequencyArray[j+1] =
cleanWordsFrequencyArray[j+1], cleanWordsFrequencyArray[j]
j += 1
if j < wordsFrequencyAmount-1-i:
    goto .bubbleSortSecondStage
i += 1
if i < wordsFrequencyAmount-1:
    goto .bubbleSortFirstStage

# generating final string
finalString = f"{cleanWordsFrequencyArray[0].word} -
{cleanWordsFrequencyArray[0].amount}"
index = 1
label .generateFinalString

```

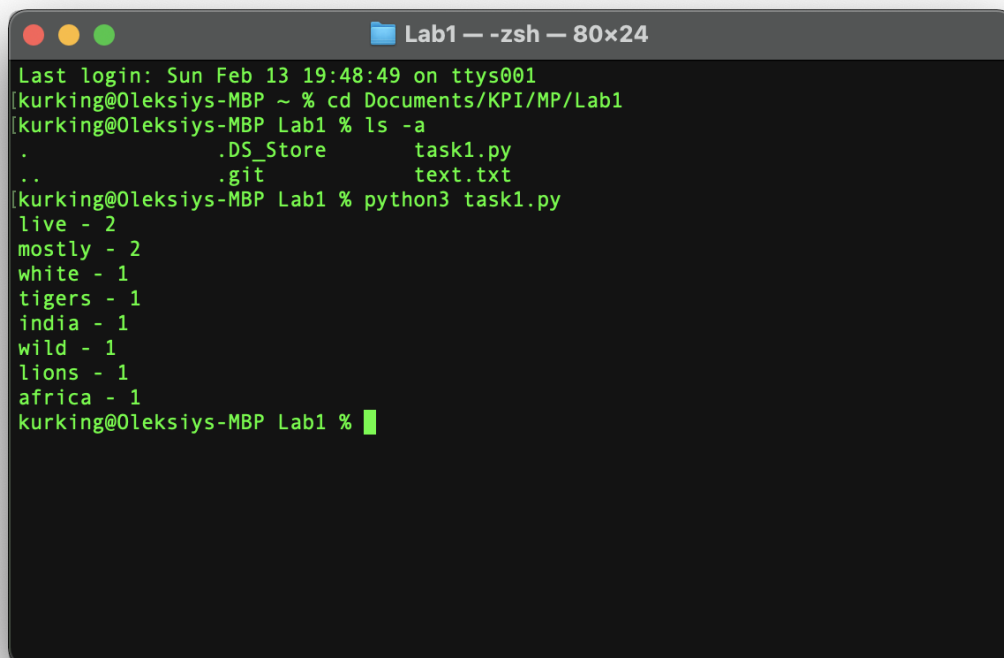
```
        finalString += f"\n{cleanWordsFrequencyArray[index].word} -  
{cleanWordsFrequencyArray[index].amount}"  
        index += 1  
        if index < wordsFrequencyAmount:  
            goto .generateFinalString  
  
        # printing result  
        print(finalString)  
  
main()
```

Перевірка роботи програми

Тестові дані:



Результат роботи:

A screenshot of a terminal window titled 'Lab1 — -zsh — 80x24'. The terminal shows the following commands and output:

```
Last login: Sun Feb 13 19:48:49 on ttys001
[kurking@Oleksiy-MBP ~ % cd Documents/KPI/MP/Lab1
[kurking@Oleksiy-MBP Lab1 % ls -a
.
..
.DS_Store
.git
task1.py
text.txt
[kurking@Oleksiy-MBP Lab1 % python3 task1.py
live - 2
mostly - 2
white - 1
tigers - 1
india - 1
wild - 1
lions - 1
africa - 1
kurking@Oleksiy-MBP Lab1 %
```

Опис алгоритму:

1. Рахуємо кількість символів у текстовому файлі
2. Розділяємо даний рядок на слова:
 - a. Використовуючи GOTO імітуємо цикл while
 - b. Проходимося по рядку
 - c. Якщо символ не є символом пунктуації, пробілом чи переносом на новий рядок, то запам'ятовуємо його в масиві `word`
 - d. Якщо знаходимо один із символів, що не може бути записаний в масив `word`, то формуємо з масиву рядок для спрощення процесу порівняння слів.
 - e. Додаємо виділене з рядка слово в масив `words`.
3. Рахуємо кількість слів
4. Створюємо масив, що містить інформацію про частоту використання слів:
 - a. Послідовно вибираємо слова загального масиву
 - b. Перевіряємо чи є обране слово в масиві `wordsFrequencyArray`. Якщо:
 - i. ТАК: збільшуємо на 1 змінну `amount` елементарної структури `WordFrequency`, що містить також саме слово
 - ii. НІ: Створюємо нову елементарну структуру `WordFrequency` в кінці масиву `wordsFrequencyArray`
5. Використовуючи `Bubble sort` сортуємо `wordsFrequencyArray` за `WordFrequency.amount`
6. Формуємо з відсортованого масиву фінальний рядок
7. Виводимо результат в консоль