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

Звіт

з лабораторної роботи № 9 з дисципліни
«Основи програмування-1.
Базові конструкції»

«Рядки»

Варіант 28

Виконав студент ІП-11 Сідак Кирил Ігорович
(шифр, прізвище, ім'я, по батькові)

Перевірив _____
(прізвище, ім'я, по батькові)

Київ 2021

Мета: ознайомитися з особливостями реалізації текстових рядків, опанувати технологію їх використання, навчитися розробляти алгоритми та програми із застосуванням рядків.

Варіант 28

У рядку символів визначити кількість повторень кожного слова та видалити дублікати слів. Слова відокремлюються пробілами.

Постановка задачі:

У заданому рядку треба визначити окремі слова(додати їх у список слів), де слово – це певний набір символів, відокремлений пробілами, причому потрібно відібрати кожне слово рівно один раз, навіть якщо воно повторюється, та на відповідний індекс списку частот слів поставити одиницю, якщо це нове слово, або збільшити це значення на 1, якщо це слово вже зустрічалось. Потім треба створити новий рядок з елементів списку слів, причому кожне слово відокремити пробілами. Таким чином, отриманий рядок і буде шуканим.

Програма на C++:

```
/* Варіант 28
У рядку символів визначити кількість повторень кожного слова та
видалити дублікати слів.
Слова відокремлюються пробілами
 */

#include <iostream>
#include <string>
using namespace std;
string* get_word_list(string, int*, int&, char=' '); // функція
для отримання масиву окремих слів(без дублікатів)
bool contains(string, string*, int);  // функція для визначення
приналежності рядка масиву рядків
int find_index(string*, int, string); // функція для визначення
індексу, на якому знаходиться заданий рядок, із заданого масиву
рядків
int* replace_array(int*, int);
void display_count(string*, int*, int); // функція для виведення
слова та кількості його повторень
string join_words(string*, int, char =' ');  // фукнція для
створення рядку із заданого масиву рядків

int main() {
    int size, actual_size;
    string text;
    actual_size = 0;
```

```
    cout << "Enter a string: ";
    getline(cin, text);
    size = text.length() / 2 + 1;
    int* count_list = new int[size];
    string* words = get_word_list(text, count_list, actual_size);
    int* counts = replace_array(count_list, actual_size);
    delete[] count_list;
    display_count(words, counts, actual_size);
    text = join_words(words, actual_size);
    cout << "Input string without duplicates:\n" << text;
    delete[] words;
    delete[] counts;
    return 0;
}

string* get_word_list(string text, int* count_list, int&
actual_size, char sep) {
    int size, index;
    size = text.length() / 2 + 1;
    string* word_list = new string[size];
    string word;
    for (int i = 0; i < text.length(); i++) {
        if (text[i] != sep) {
            word += text[i];
        }
        else {
            if (!contains(word, word_list, size) && !word.empty())
{
                word_list[actual_size] = word;
                count_list[actual_size] = 1;
                actual_size++;
            }
            else if (contains(word, word_list, size) &&
!word.empty()){
                index = find_index(word_list, size, word);
                count_list[index] += 1;
            }
            word = "";
        }
    }
    if (!contains(word, word_list, size) && !word.empty()) {
        cout << "word = " << word << endl;
        word_list[actual_size] = word;
        count_list[actual_size] = 1;
        actual_size++;
    }
    else if (contains(word, word_list, size) && !word.empty()) {
        cout << "word = " << word << endl;
        index = find_index(word_list, size, word);
        count_list[index] += 1;
    }
    string* words = new string[actual_size];
    for (int i = 0; i < actual_size; i++) {
        words[i] = word_list[i];
    }
    delete[] word_list;
```

```
    return words;
}

bool contains(string item, string* list, int size) {
    int i = 0;
    bool found = false;
    while (i < size && !found) {
        if (list[i] == item) {
            found = true;
        }
        i++;
    }
    return found;
}

int find_index(string* list, int size, string item) {
    int index, i;
    i = 0;
    bool found = false;
    while(i < size && !found) {
        if (list[i] == item) {
            index = i;
            found = true;
        }
        i++;
    }
    return index;
}

int* replace_array(int *old_arr, int size) {
    int* new_arr = new int[size];
    for (int i = 0; i < size; ++i) {
        new_arr[i] = old_arr[i];
    }
    return new_arr;
}

void display_count(string* word_list, int* count_list, int size) {
    for (int i = 0; i < size; ++i) {
        cout << "Word: " << word_list[i] << ", count: " <<
count_list[i] << endl;
    }
}

string join_words(string* list, int size, char sep) {
    string text;
    for (int i = 0; i < size; i++) {
        if (i != size - 1) {
            text += list[i] + sep;
        }
        else {
            text += list[i];
        }
    }
    return text;
}
```

Програма на Python:

```
# Варіант 28
# У рядку символів визначити кількість повторень кожного
слова та видалити дублікати слів.
# Слова відокремлюються пробілами


def get_word_list(text: str, sep: str = ' '):  # функція для
отримання масиву окремих слів(без дублікатів)
    word_list = []
    count_list = []
    word = ''
    for symbol in text:
        if symbol != sep:
            word += symbol
        else:
            if word not in word_list and word != '':
                word_list.append(word)
                count_list.append(1)
            elif word in word_list:
                count_list[word_list.index(word)] += 1
            word = ''
    if word not in word_list and word != '':
        word_list.append(word)
        count_list.append(1)
    elif word in word_list:
        count_list[word_list.index(word)] += 1
    return word_list, count_list


def display_count(word_list, count_list):  # функція для
виведення слова та кількості його повторень
    for i in range(len(word_list)):
        word = word_list[i]
        count = count_list[i]
        if count == 1:
            print(f'The string contains "{word}" word {count}
time.')
        else:
            print(f'The string contains "{word}" word {count}
times.')


def join_words(word_list, sep: str = ' '):  # фукнція для
створення рядку із заданого масиву рядків
    text = ''
    for i in range(len(word_list)):
        if i != len(word_list) - 1:
            text += word_list[i] + sep
```

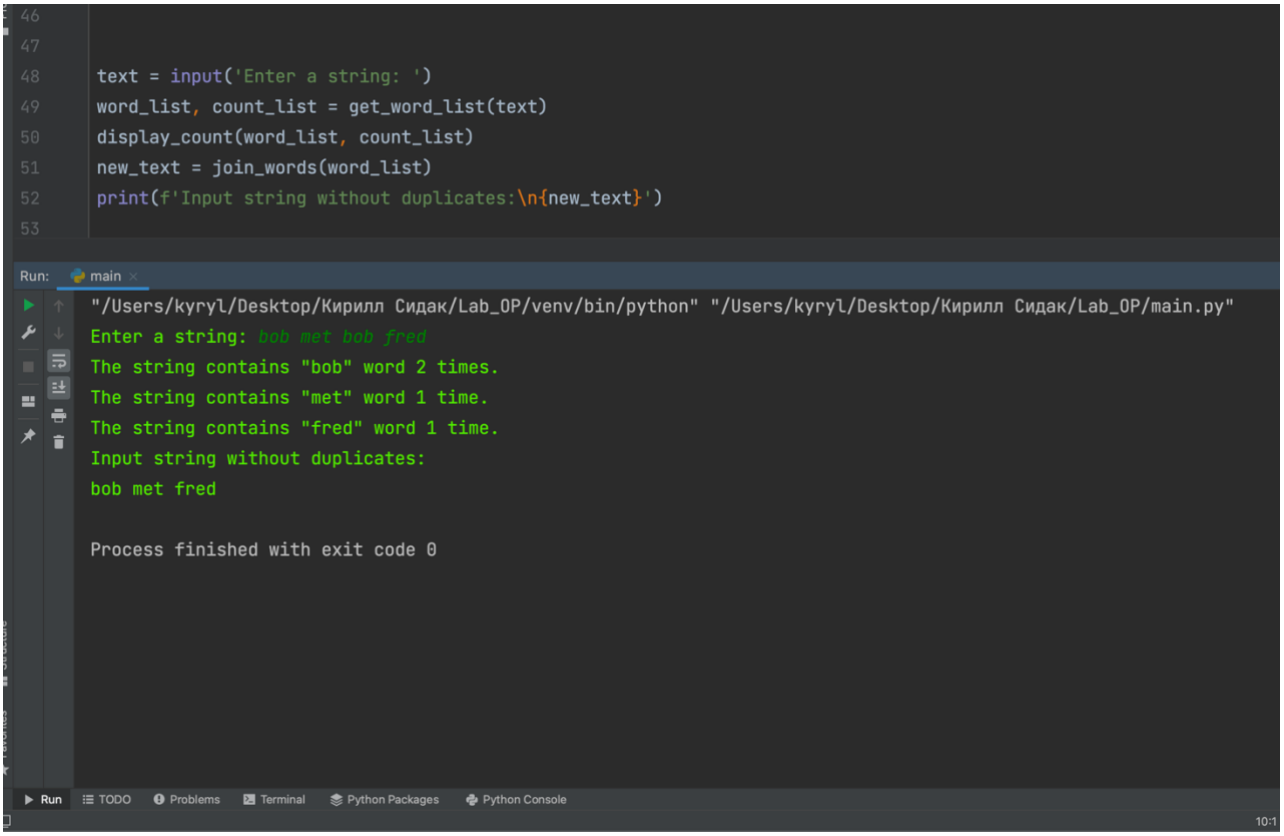
```
        else:
            text += word_list[i]
    return text


text = input('Enter a string: ')
word_list, count_list = get_word_list(text)
display_count(word_list, count_list)
new_text = join_words(word_list)
print(f'Input string without duplicates:\n{new_text}')
```

Результат на C++:

```
17      actual_size = 0;
18      cout << "Enter a string: ";
19      getline( &: cin,  &: text);
20      size = text.length() / 2 + 1;
21      int* count_list = new int[size];
22      string* words = get_word_list( text: text, count_list,  &: actual_size);
        main
Run:  Lab_9
/Users/kyryl/Desktop/Lab_9/cmake-build-debug/Lab_9
Enter a string: this is new lab lab of mine lab
Word: this, count: 1
Word: is, count: 1
Word: new, count: 1
Word: lab, count: 3
Word: of, count: 1
Word: mine, count: 1
Word: , count: 0
Input string without duplicates:
this is new lab of mine
Process finished with exit code 0
```

```
13
14  int main() {
15      int size, actual_size;
16      string text;
17      actual_size = 0;
18      cout << "Enter a string: ";
19      getline( &: cin,  &: text);
20      size = text.length() / 2 + 1;
21      int* count_list = new int[size];
22      string* words = get_word_list( text: text, count_list,  &: actual_size);
23      int* counts = new int[actual_size];
24      for (int i = 0; i < actual_size; ++i) {
25          counts[i] = count_list[i];
Run:  Lab_9
/Users/kyryl/Desktop/Lab_9/cmake-build-debug/Lab_9
Enter a string: hello man hello friend man
Word: hello, count: 2
Word: man, count: 2
Word: friend, count: 1
Word: , count: 0
Input string without duplicates:
hello man friend
Process finished with exit code 0
```

Результат на Python:



Висновок

Отже, я ознайомився з особливостями реалізації текстових рядків, зокрема на мовах C++ та Python та опанував технологію їх використання, розробивша алгоритм і створивши програму для визначення кількості повторень кожного слова в рядку символів та видалення дублікатів, й отримав коректний результат.