НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота № 3

з дисципліни «Алгоритми та структури даних»
На тему: «Структури даних: масиви, зв'язні списки»

Виконав:
студент групи ФІ-12
Завалій Олександр

Київ-2023

Реалізація завдання

Варіант №5

Надрукувати всі слова, у яких парна кількість літер. Перед друком подвоїти останню літеру кожного слова, а також видалити зі слів всі літери "о".
Реалізувати завдання за допомогою:

1. Двозв'язного (двонаправленого) списку.
2. Використовуючи динамічний масив.

Task I

Двозв'язний (двонаправлений) список.

```
#include <algorithm>
#include <iostream>
#include <string>
using namespace std;

struct Node {
    string data;
    Node* prev;
    Node* next;
};

class DoublyLinkedList {
private:
    Node* head;
    Node* tail;
public:
    DoublyLinkedList() {
        head = nullptr;
        tail = nullptr;
    }
    ~DoublyLinkedList() {
        Node* current = head;
        while (current != nullptr) {
            Node* next = current->next;
            delete current;
            current = next;
        }
    }

    void push_back(string value);
    void print();
    void pop_item();
    void show_item();
};
```

```
void DoublyLinkedList::push_back(string value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->prev = nullptr;
    newNode->next = nullptr;
    if (head == nullptr) {
        head = newNode;
        tail = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        tail = newNode;
    }
}
void DoublyLinkedList::print() {
    Node* current = head;
    cout << endl;
    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}
void DoublyLinkedList::pop_item() {
    Node* current = head;
    while (current != nullptr) {
        string word = current->data;
        if (word.length() % 2 != 0) {
            if (current == head) {
                head = current->next;
                head->prev = nullptr;
            }
            else if (current == tail) {
                tail = current->prev;
                tail->next = nullptr;
            }
            else {
                current->prev->next = current->next;
                current->next->prev = current->prev;
            }
            Node* temp = current->next;
            delete current;
            current = temp;
        }
        else {
            word += word[word.length()-1];
            word.erase(remove(word.begin(), word.end(), 'o'), word.end());
            current->data = word;
            current = current->next;
        }
    }
}
```

```
void DoublyLinkedList::show_item() {
    cout << '\n';
    Node* current = head;
    while (current != nullptr) {
        string word = current->data;
        if (word.length() % 2 == 0) {
            word += word[word.length()-1];
            word.erase(remove(word.begin(), word.end(), 'o'), word.end());
            cout << word << ' ';
        }
        current = current->next;
    }
}

int main() {
    DoublyLinkedList list;
    string temp;

    do {
        cin >> temp;
        if (temp != "q"){
            list.push_back(temp);
        } else{
            break;
        }
    } while(true);

    list.print();
    list.show_item();
    list.pop_item();
    list.print();
    return 0;
}
```

Task II

Динамічний масив.

```
#include <algorithm>
#include <iostream>
#include <string>
using namespace std;

class DynamicArray{
    string* arr;
    int size;
    int capacity;
public:
    DynamicArray() {
        size = 0;
        capacity = 1;
        arr = new string[capacity];
    }

    ~DynamicArray() {
        delete[] arr;
    }

    void push_back(const string& word);
    void print();
    void pop_item(int index);
    void task();
};

void DynamicArray::push_back(const string& word) {
    if (size == capacity) {
        capacity *= 2;
        string* newArr = new string[capacity];
        for (int i = 0; i < size; i++) {
            newArr[i] = arr[i];
        }
        delete[] arr;
        arr = newArr;
    }
    arr[size++] = word;
}
void DynamicArray::print() {
    cout << endl;
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
void DynamicArray::pop_item(int index) {
    if (index < 0 || index >= size) {
        return;
    }
    for (int i = index; i < size - 1; i++) {
        arr[i] = arr[i + 1];
    }
    size--;
}
```

```
void DynamicArray::task(){
    string temp;
    for (int i = 0; i < size; i++) {
        temp = arr[i];
        if (temp.length() % 2 == 0) {
            temp += temp[temp.length()-1];
            temp.erase(remove(temp.begin(), temp.end(), 'o'), temp.end());
            arr[i] = temp;
        } else {
            pop_item(i);
            i--;
        }
    }
}

int main() {
    DynamicArray arr;
    string temp;

    do {
        cin >> temp;
        if (temp != "q"){
            arr.push_back(temp);
        } else{
            break;
        }
    } while(true);

    arr.print();
    arr.task();
    arr.print();
    return 0;
}
```