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 №5 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

«Списокові структури даних»
Варіант 3

Виконав студент ІП-15, Борисик Владислав Тарасович
(шифр, прізвище, ім'я, по батькові)

Перевірив Соколовський Владислав Володимирович
(прізвище, ім'я, по батькові)

Київ 2022

Мета лабораторної роботи

Мета роботи – вивчити основні підходи формалізації евристичних алгоритмів і вирішення типових задач з їх допомогою.

Завдання

Розробити алгоритм розв’язання задачі відповідно до варіанту. Виконати програмну реалізацію задачі. Не використовувати вбудовані спискові структури даних (контейнери). Зробити висновок по лабораторній роботі.

3. Заданий рядок слів, які відокремлюються одне від одного пробілами. Побудувати список, елементами якого є відповідні слова. Вилучити із списку всі слова, що починаються та закінчуються на задану користувачем літеру. Надрукувати початковий і змінений списки.

Для виконання цієї лабораторної роботи я буду використовувати однобічно зв'язаний список.

Псевдокод алгоритму

```
void delete_words(char character, SinglyLinkedList &list){
    int list_size = list.get_list_size();

    Для і від 1 до list_size(включно) з кроком 1{
        // вузол
        Node *node = list.get_node(i);
        // слово із вузла
        string word = node.get_data();

        // якщо довжина слова більша за 1 символ
        якщо (word.size() > 1){
            int first_char_position = 0;
            int last_char_position = word.size() - 1;

            // якщо слово починається і закінчується заданою літерою
            якщо (word[first_char_position] == character &&
word[last_char_position] == character){
                // видаляємо літеру
                list.delete_note(i);
                // зменшуємо розмір списку на 1
                list_size--;
                // зменшуємо лічильник на 1
                i--;
            }
        }
    }
}
```

Програмна реалізація алгоритму на мові C++

```
void delete_words(char character, SinglyLinkedList &list){
    int list_size = list.get_list_size();

    for (int i = 1; i <= list_size; ++i) {
        // вузол
        Node *node = list.get_node(i);
        // слово із вузла
        string word = node->get_data();

        // якщо довжина слова більша за 1 символ
        if(word.size() > 1){
            int first_char_position = 0;
            int last_char_position = word.size() - 1;

            // якщо слово починається і закінчується заданою літерою
            if(word[first_char_position] == character && word[last_char_position] ==
character){
                // видаляємо літеру
                list.delete_note(i);
                // зменшуємо розмір списку на 1
                list_size--;
                // зменшуємо лічильник на 1
                i--;
            }
        }
    }
}
```

Вихідний код програми

main.cpp:

```
#include "functions.h"

int main() {
    string text = capture_text();
    printf("Entered text:\n%s\n\n", text.c_str());

    SinglyLinkedList list;

    fill_list(list, text);

    printf("List after filling:\n");
    list.show_list();

    char character = capture_char();
    delete_words(character, list);

    printf("\nList after deleting words with start and end character %c:\n",
character);
    list.show_list();
}
```

classes.h:

```
#ifndef ASD_LABWORK_5_CLASSES_H
#define ASD_LABWORK_5_CLASSES_H
#endif

#include <iostream>
using namespace std;

class Node{
private:
    string data;
    Node* next_node;

public:
    Node();
    Node(string data_to_add);

    string get_data();
    Node* get_next_node();
    void set_next_node(Node *node_to_add);
};


class SinglyLinkedList{
private:
    Node* head_node;

    void delete_head_node();
    bool is_empty();

public:
    SinglyLinkedList();

    int get_list_size();
    void add_node(string data_to_add);
    Node* get_node(int position);
    void delete_note(int position);
    void show_list();
};
```

classes.cpp:

```cpp
#include "classes.h"

Node::Node() {
    data = "";
    next_node = nullptr;
}

Node::Node(string data_to_add) {
    data = std::move(data_to_add);
    next_node = nullptr;
}


string Node::get_data() {
    return data;
}

Node *Node::get_next_node() {
    return next_node;
}

void Node::set_next_node(Node *node_to_add){
    next_node = node_to_add;
}

SinglyLinkedList::SinglyLinkedList() {
    head_node = nullptr;
}

void SinglyLinkedList::add_node(string data_to_add) {
    Node* node_to_add = new Node(std::move(data_to_add));

    // якщо головний вузол порожній (список пустий)
    if (is_empty()) {
        // ініціалізуємо головний вуззол вузлом node_to_add
        head_node = node_to_add;
    }
    // якщо головний вузол не порожній (список не пустий)
    else{
        // тимчасовий вузол
        Node* temp_node = head_node;

        // ітеруємось поки наступний вузол не буде дорівнювати nullptr
        // тобто, ітеруємось до останнього вузла
        while (temp_node->get_next_node() != nullptr) {
            // змінюємо тимчасовий вузол наступним вузлом
            temp_node = temp_node->get_next_node();
        }

        // додаємо вузол node_to_add як останній
        temp_node->set_next_node(node_to_add);
```

```
    }

}

void SinglyLinkedList::show_list() {
    // якщо головний вузол порожній (список пустий)
    if (is_empty()) {
        cout << "List is empty" << endl;
    }
    // якщо головний вузол не порожній (список не пустий)
    else{
        Node* temp_node = head_node;

        // Ітеруємось по вузлам (ітеруємось поки останній вузол не буде дорівнювати
nullptr)
        while (temp_node != nullptr) {
            cout << temp_node->get_data() << " ";
            // змінюємо тимчасовий вузол наступним вузлом
            temp_node = temp_node->get_next_node();
        }
    }

}

int SinglyLinkedList::get_list_size() {
    // кількість елементів у списку
    int list_size = 0;
    Node *temp_node = head_node;

    while (temp_node != nullptr) {
        // змінюємо тимчасовий вузол наступним вузлом
        temp_node = temp_node->get_next_node();
        list_size++;
    }

    return list_size;
}

void SinglyLinkedList::delete_head_node() {
    // переміщуємо перший покажчик на наступний вузол
    head_node = head_node->get_next_node();
}

bool SinglyLinkedList::is_empty() {
    // якщо головний вузол порожній (список пустий)
    if (head_node == nullptr)
        return true;
    else return false;
}

void SinglyLinkedList::delete_note(int position){
    if (is_empty()) {
        cout << "List is empty" << endl;
    }
    else{
```

```
        int list_size = get_list_size();

        // перевіряємо чи позиція менша за довжину списку
        // якщо позиція більша, то виводимо повідомлення про помилку
        if (position > list_size) {
            cout << "Index is greater that list size" << endl;
        }
        // якщо ж позиція менша
        else{
            // якщо позиція == 1, то видаляємо головний вузол
            if (position == 1) {
                delete_head_node();
            }
            else{
                Node *temp_node1 = head_node;
                Node *temp_node2 = nullptr;

                // Ітеруємось по вузлам, поки не буде досягнута позиція
                while (position > 1) {
                    position--;

                    // змінюємо другий тимчасовий вузол на перший
                    temp_node2 = temp_node1;

                    // змінюємо перший вузол на наступний
                    temp_node1 = temp_node1->get_next_node();
                }

                // наступний вузол першого тимчасового вузла
                Node *next_node = temp_node1->get_next_node();
                // змінюємо покажчик другого тимчасового вузла на наступний вузол
                temp_node2->set_next_node(next_node);

                // видаляємо перший тимчасовий вузол
                delete temp_node1;
            }
        }
    }
}

Node* SinglyLinkedList::get_node(int position) {
    if (is_empty()) {
        cout << "List is empty" << endl;
    }
    else{
        int list_size = get_list_size();

        // перевіряємо чи позиція менша за довжину списку
        // якщо позиція більша, то виводимо повідомлення про помилку
        if (position > list_size) {
            cout << "Index is greater that list size" << endl;
        }
            // якщо ж позиція менша
        else{
```

```
            Node *temp_node = head_node;

            // Ітеруємось по вузлам, поки не буде досягнута позиція
            while (position > 1) {
                position--;

                // змінюємо перший вузол на наступний
                temp_node = temp_node->get_next_node();
            }

            return temp_node;
        }
    }
}
```

functions.h:

```
#ifndef ASD_LABWORK_5_FUNCTIONS_H
#define ASD_LABWORK_5_FUNCTIONS_H

#endif

#include "classes.h"

string capture_text();
char capture_char();
void fill_list(SinglyLinkedList &list, const string& text);
void delete_words(char character, SinglyLinkedList &list);
```

functions.cpp:

```
#include "functions.h"
#include <vector>

string capture_text(){
    string text;

    cout << "Enter text separated by space: ";
    getline(cin, text);

    return text;
}

char capture_char(){
    char character;

    cout << "\n\nEnter the character with which words will be deleted: ";
    cin >> character;

    return character;
}

vector<string> split_words(const string& text){
    // вектор зі словами
    vector<string> lines;
    // позиція минулого пробілу
    int previous_position = 0;
    // лічильник (рахує символи тексту)
    int counter = 0;

    while(counter <= text.size()){
        // якщо символ дорівнює пробілу або counter дорівнює останньому символу тексту
        if(text[counter] == ' ' || counter == text.size()){
            // рядок (підрядок, який починається з позиції минулого пробілу довжиною
counter-previous_position)
            string line = text.substr(previous_position, counter-previous_position);
            // добавляємо це слово у вектор
            lines.push_back(line);
            // оновлюємо позицію пробілу
            previous_position = counter+1;
        }
        counter++;
    }

    return lines;
}

void fill_list(SinglyLinkedList &list, const string& text){
    vector<string> words = split_words(text);

    for (const auto& word : words) {
        list.add_node(word);
```

```
    }
}

void delete_words(char character, SinglyLinkedList &list){
    int list_size = list.get_list_size();

    for (int i = 1; i <= list_size; ++i) {
        // вузол
        Node *node = list.get_node(i);
        // слово із вузла
        string word = node->get_data();

        // якщо довжина слова більша за 1 символ
        if(word.size() > 1){
            int first_char_position = 0;
            int last_char_position = word.size() - 1;

            // якщо слово починається і закінчується заданою літерою
            if(word[first_char_position] == character && word[last_char_position] ==
character){
                // видаляємо літеру
                list.delete_note(i);
                // зменшуємо розмір списку на 1
                list_size--;
                // зменшуємо лічильник на 1
                i--;
            }
        }
    }
}
```

Приклад роботи програми:

```
Enter text separated by space:test testing hello ttt world
 Entered text:
test testing hello ttt world

List after filling:
test testing hello ttt world

Enter the character with which words will be deleted:t

List after deleting words with start and end character t:
testing hello world
Process finished with exit code 0
```

```
Enter text separated by space:aaa abb bba asomethinga something aba
 Entered text:
aaa abb bba asomethinga something aba

List after filling:
aaa abb bba asomethinga something aba

Enter the character with which words will be deleted:a

List after deleting words with start and end character a:
abb bba something
Process finished with exit code 0
```

```
Enter text separated by space:bfsdfsb hgh bbb another word
 Entered text:
bfsdfsb hgh bbb another word

List after filling:
bfsdfsb hgh bbb another word

Enter the character with which words will be deleted:b

List after deleting words with start and end character b:
hgh another word
Process finished with exit code 0
```

Бачимо, що програма коректно видалила усі слова, що починались і закінчувались заданою літерою, отже, вона працює правильно.

Висновок

Під час виконання цієї лабораторної роботи я вивчив основні підходи формалізації евристичних алгоритмів і вирішив типову задачу з їх допомогою. В результаті виконання лабораторної роботи я отримав програму, яка вилучає із однобічно зв'язаного списку всі слова, що починаються та закінчуються на задану користувачем літеру.