**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Иерархические списки**.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 9304 |  | Паутова Ю.В. |
| Преподаватель |  | Филатов Ар.Ю. |

Санкт-Петербург

2020

## Цель работы.

## Ознакомиться с понятием иерархического списка, получить навыки программирования функций для работы с иерархическими списками. Реализовать иерархический список на языке программирования С++.

## Задание.

## Вариант 6

## Проверить иерархический список на наличие в нем заданного элемента (атома) x.

## Описание алгоритма работы.

Программе с помощью аргументов командной строки подаются два файла: файл1, содержащий строку-список и элемент(фтом), и файл2, в который будет записан результат работы программы. Сначала проверяем были ли переданы программе файлы, если да, то проверяем, открылись ли они для считывания/записи. Если оба файла были открыты, тогда создаются: объект класса string: list, объект класса H\_List: h\_list и переменная типа char: elem. Строка-список записывается в list, после чего проверяется на корректность функцией isCorrect().

Если строка-список корректна, то вызывается метод класса H\_List read\_h\_list(), который создает иерархический список с помощью метода read\_node, который рекурсивно пробегается по строке-списку list и создает узел (в значении лежит указатель) или атом (в значении лежит значение).

Затем считывается элемент(атом) и записывается в переменную elem, которая затем передается в метод класса H\_List set\_element(), который заполняет поля elem и is\_elem. Файл1 закрывается.

После вызывается метод класса H\_List browes(), который рекурсивно пробегается по иерархическому списку и сравнивает значения атомов с заданным значением. Если значения совпали, то переменной is\_elem присваивается значение true.

В зависимости от значения поля is\_elem(true/false) в файл2 записывается строка: “<elem> is/isn’t found”. Вызывается метод класса H\_List free\_list(), после чего файл2 закрывается и программа завершается.

**Формат входных и выходных данных.**

Входные данные представлены в виде строки, которая является краткой записью иерархического списка, и элемента (атома), наличие которого в этом списке нужно проверить.

Выходные данные представлены в виде строки, в которой говорится был найдем заданный элемент (атом) или нет.

**Описание основных структур данных и функций.**

Структуры данных:

* Class Node – элемент иерархического списка: узел или атом.
* Поле Node\* next – указатель на следующий элемент списка.
* Поле std::variant<Node\*, char> value – значение, лежащее в узле: указатель на начало списка-ответвления в иерархическом списке или символ.
* Class H\_List – реализация иерархического списока
* Поле char elem – хранит значение элемента(атома), наличие которого нужно проверить.
* Поле bool is\_elem – хранит true или false в зависимости от того, есть заданный элемент в списке или его нет.
* Поле Node\* head – указатель на голову иерархического списка.
* Метод set\_element() – принимает переменную типа char и передает ее значение полю elem, также передает полю is\_elem значение false.
* Метод get\_is\_elem() – возвращает значение поля is\_elem.
* Метод read\_node() – принимает ссылку на объект класса string и тип данных int, создает узел или атом списка.
* Метод read\_h\_list() – принимает ссылку на объект класса string и тип данных int, создает иерархический список.
* Метод print() – принимает указатель на объект класса Node, выводит иерархический список.
* Метод browse() – принимает указатель на объект класса Node, пробегается по списку и сравнивает значение атомов со значением поля elem. Если значения совпали, то переменной is\_elem присваивается значение true.
* Метод free\_list() – принимает указатель на объект класса Node, пробегается по списку и очищает память.

Функции:

* bool isCorrect() – принимает ссылку на объект класса string, и проверяет корректность строки с краткой записью списка.

## Разработанный программный код см. в приложении А.

**Тестирование.**

Тестирование происходит с помощью bash-скрипта. Он с помощью команд терминала запускает программу, подавая на вход файлы с тестами из директории Tests, и выводит результат. Также запускает программу без указания файлов и с указанием несуществующего файла test7.txt. Для запуска тестирования в консоли используется команда *make run\_tests*.

Результаты тестирования см. в приложении B.

## Выводы.

Произошло ознакомление с понятием иерархического списка, были получены навыки программирования функций для работы с иерархическими списками. Был реализован иерархический список на языке программирования С++.

Была разработана программа, считывающая краткую запись иерархического списка, создающая на её основе иерархический список и проверяющая список на вхождение заданного элемента (атома). Проведено тестирование работы программы.

# Приложение А Исходный код программы

Название файла: **source/main.cpp**

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <string>

#include <stack>

#include <variant>

#include "H\_List.h"

bool isCorrect(const std::string& list);

int main(int argc, char\*\* argv){

    if (argc < 3){

        std::cout << "Команда для запуска программы: ./lab2 /path/to/input /path/to/output" << std::endl;

        return 1;

    }

    else{

        std::ifstream in(argv[1]);

        if (!in.is\_open()){

            std::cout << argv[1] << ": No such file or directory" << std::endl;

            return 1;

        }

        std::ofstream out(argv[2]);

        if (!out.is\_open()){

            std::cout << argv[2] << ": No such file or directory" << std::endl;

            return 1;

        }

        std::string list;

        H\_List h\_list;

        char elem;

        while(getline(in, list)){

            if (list.find('(') != std::string::npos){

                if (!isCorrect(list)){

                    out << "The list isn't correct." << std::endl;

                    in.close();

                    out.close();

                    return 1;

                }

                int i = 0;

                h\_list.read\_h\_list(list,i);

            }

            else{

                elem = list[list.length()-1];

                h\_list.set\_element(elem);

            }

        }

        in.close();

        h\_list.browse(h\_list.head);

        if (h\_list.get\_is\_elem()){

            out << elem << " is found\n";

        }

        else{

            out << elem << " isn't found\n";

        }

        h\_list.free\_list(h\_list.head);

        out.close();

    }

    return 0;

}

bool isCorrect(const std::string& list){

    std::stack<char> Stack;

    if (list[0] != '('){

        return false;

    }

    for (char i : list){

        if (i == '('){

            Stack.push(i);

        }

        if (i == ')'){

            if (Stack.empty()){

                return false;

            }

            Stack.pop();

        }

    }

    return Stack.empty();

}

Название файла: **Node.h**

#ifndef NODE\_H

#define NODE\_H

#include <variant>

class Node{

public:

    Node \* next;

    std::variant<Node\*, char> value;

};

#endif

Название файла: **H\_List.h**

#ifndef H\_LIST\_H

#define H\_LIST\_H

#include <iostream>

#include <cstdlib>

#include <string>

#include <variant>

#include "Node.h"

class H\_List{

    char elem;

    bool is\_elem;

public:

    Node\* head;

    void set\_element(char elem);

    char get\_is\_elem();

    Node\* read\_node(std::string& list, int& i);

    void read\_h\_list(std::string& list, int& i);

    void print(Node\* cur);

    void browse(Node\* cur);

    void free\_list(Node\* cur);

};

#endif

Название файла: **H\_List.cpp**

#include "H\_List.h"

void H\_List::set\_element(char elem){

    this->elem = elem;

    this->is\_elem = false;

}

char H\_List::get\_is\_elem(){

    return this->is\_elem;

}

Node\* H\_List::read\_node(std::string& list, int& i){

    Node\* tmp = new Node;

    if(list[i] == ')'){

        delete tmp;

        return nullptr;

    } else

        if(list[i] == '('){

            i++;

            tmp->value = read\_node(list, i);

            if(std::get<Node\*>(tmp->value) == nullptr){

            return tmp;

            }

            i++;

            Node\* tmp1 = std::get<Node\*>(tmp->value);

            while (list[i] != ')'){

                if( list[i] != ' ') {

                    tmp1->next = read\_node(list, i);

                    tmp1->next->next = nullptr;

                    tmp1 = tmp1->next;

                }

                i++;

            }

            return tmp;

        }

        else{

            tmp->value = list[i];

            tmp->next = nullptr;

        }

        return tmp;

}

void H\_List::read\_h\_list(std::string& list, int& i){

    if(list[i] != ' ' && list[i] != ')'){

        if(list[i] == '(')

            i++;

        head = read\_node(list, i);

        Node\* tmp = head;

        i++;

        while (list[i] != ')'){

            if( list[i] != ' ') {

                tmp->next = read\_node(list, i);

                tmp->next->next = nullptr;

                tmp = tmp->next;

            }

            i++;

        }

    }

}

void H\_List::print(Node\* cur) {

    if (std::holds\_alternative<Node \*>(cur->value)) {

        std::cout << '(';

        if (std::get<Node\*>(cur->value) != nullptr) {

            print(std::get<Node\*>(cur->value));

        }

        std::cout << ')';

    }

    else{

        std::cout << std::get<char>(cur->value);

    }

    if (cur->next != nullptr) {

        print(cur->next);

    }

}

void H\_List::browse(Node\* cur){

    if (std::holds\_alternative<Node \*>(cur->value)) {

        if (std::get<Node\*>(cur->value) != nullptr) {

            browse(std::get<Node\*>(cur->value));

        }

    }

    else{

        if(std::get<char>(cur->value) == this->elem){

            this->is\_elem = true;

return;

}

    }

    if (cur->next != nullptr) {

        browse(cur->next);

    }

}

void H\_List::free\_list(Node\* cur){

    if (std::holds\_alternative<Node \*>(cur->value)) {

        if (std::get<Node\*>(cur->value) != nullptr) {

            free\_list(std::get<Node\*>(cur->value));

            delete cur;

            return;

        }

    }

    if (cur->next != nullptr) {

        free\_list(cur->next);

    }

    delete cur;

}

Название файла: **Makefile**

lab2: main.o h\_list.o

    g++ -std=c++17 main.o h\_list.o -o lab2

main.o: source/main.cpp source/H\_List.h

    g++ -std=c++17 -c source/main.cpp

h\_list.o: source/H\_List.cpp source/H\_List.h source/Node.h

    g++ -std=c++17 -c source/H\_List.cpp

run\_tests: lab2

    ./myscript

clean:

    rm -rf \*.o

Название файла: **./myscript**

#!/bin/bash

for n in {1..6}

do

    arg=$(cat Tests/test$n.txt)

    echo -e "\nTest $n"

    echo "list = $arg"

    ./lab2 Tests/test$n.txt result$n

    res=$(cat result$n)

    echo "$res"

done

echo -e "\nTest 7\nВеденная команда: ./lab2"

./lab2

echo -e "\nTest 8"

./lab2 Tests/test8.txt result8

echo -e

# Приложение B тестирование

Таблица B.1 – Результаты тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Выходные данные | Комментарии |
| 1 | (1(()23)4)  Element to search for: 2 | 2 is found | Корректный список, содержащий заданный элемент (атом) |
| 2 | (a b (c d (e f (h g))))  Element to search for: f | f is found | Корректный список, содержащий заданный элемент (атом) |
| 3 | (1(()23)(56)7)  Element to search for: 0 | 0 isn’t found | Корректный список, не содержащий заданный элемент (атом) |
| 4 | (f(t(asd)a)  Element to search for: x | The list isn't correct. | Некорректный список |
| 5 | 2(a(s(d)e)3)0)  Element to search for: w | The list isn't correct. | Некорректный список |
| 6 | (1(()23)4(25))  Element to search for: 2 | 2 is found | Корректный список, содержащий заданный элемент (атом) |
| 7 | ./lab2 | Команда для запуска программы:  ./lab2 /path/to/input /path/to/output | Программа запущена без аргументов (файлов для считывания и записи данных) |
| 8 | ./lab1 Tests/test8.txt result8 | Tests/test8.txt: No such file or directory | Программе передан несуществующий файл |