

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Алгоритмы и структуры данных»
Тема: Иерархические списки

Студент гр. 9304

Силкин В.А.

Преподаватель

Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Изучить иерархические списки, способы их построения и вывода.

Научиться обрабатывать данные в иерархических списках.

Задание.

Вариант 8.

Заменить в иерархическом списке все вхождения заданного элемента (атома) *x* на заданный элемент (атом) *y*.

Выполнение работы.

При подаче данных, входная строка проверяет, является ли первый и последний символы соответственно '(' и ')', попутно проверяя, что каждая открывающая скобка имеет свою закрывающую, наличие в ней пробелов. Если есть пробелы, а первые 2 условия не соблюдаются, программа не будет обрабатывать входящую строку как иерархический список.

Программа рекурсивно просматривает каждый символ строки с помощью итератора. Если итератор стоит на '(', он записывает в значение ноды указатель на новую ноду, которая является головой нового подсписка. Если итератор стоит на ')', функция перестаёт вызываться рекурсивно. В остальных случаях, в ноду записывается значение итератора, и переходит на следующую ноду.

Помимо основной строки, в программу подаётся 2 символа: тот который нужно заменить, и тот, на который нужно заменить. Программа выводит сначала список до изменения, через пробел, а затем на следующей строке список после изменения.

Разработанный программный код см. в приложении А.

Тестирование.

Компиляция выполняется командой `make` (или `make lab2`). Для тестирования написан `bash`-скрипт, лежащий в корневой папке лабораторной работы, он запускается через `make testing`, либо вручную через файл `test_script`. Если до этого компиляции не было, программа сама скомпилирует файл для

ТЕСТОВ.

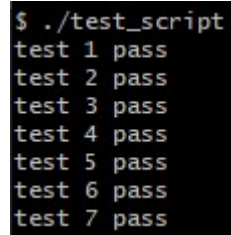
A screenshot of a terminal window with a black background and white text. The first line shows a prompt character followed by the command './test_script'. The subsequent seven lines show the output of the script, each consisting of 'test' followed by a number from 1 to 7 and the word 'pass'.

Рисунок 1 - Вывод тестирующего bash-скрипта

Результаты тестирования см. в приложении В.

Выводы.

Была проведена работа по изучению иерархических списков. Были изучены способы их считывания, обработки и вывода.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

//Задание 8: Заменить все вхождения элемента x на элемент y

```
#include <iostream>
#include <string>
#include <iterator>
#include "list.h"

bool isCorrectStr(std::string str) {
    if(str[0] != '(' || str[str.length()-1] != ')') {
        return false;
    }
    int count = 0;
    for(int i = 0; i < str.size(); i++) {
        if(str[i] == '(') {
            count++;
        }
        if(str[i] == ')') {
            count--;
        }
        if(count < 0) {
            return false;
        }
    }
    if(!count) {
        return true;
    } else {
        return false;
    }
}

int main() {
    std::string expression;
    std::cin >> expression;
    char Old, New;
    std::cin >> Old >> New;
    if(!isCorrectStr(expression)) {
        std::cout << "Incorrect string\n";
        return 0;
    }
    List* list = new List(expression, Old, New);
    return 0;
}
```

Название файла: list.h

```

#pragma once
#include <variant>
#include <string>

struct Node {
    std::variant<Node*, char> value;
    Node* next = nullptr;
};

class List {
    Node* head = new Node;
    void createList(Node**, std::string&, std::string::iterator&);
public:
    List(std::string&, char, char);
    void print(Node*);
    void replaceElem(Node*, char, char);
    ~List();
    void deliting(Node*);
};

```

Название файла: list.cpp

```

#include "list.h"
#include <iostream>

List::List(std::string& str, char Old, char New){
    auto iter = str.begin();
    createList(&head, str, iter);
    print(head);
    std::cout << "\n";
    replaceElem(head, Old, New);
    print(head);
    std::cout << "\n";
}

void List::createList(Node**ptr, std::string& str, std::string::iterator& iter)
{
    if(iter < str.end()) {
        if(*iter == '(') {
            (*ptr)->value = new Node;
            iter++;
            if(*iter != ')') {
                createList(&(std::get<Node*>((*ptr)->value)), str, iter);
            } else {
                delete std::get<Node*>((*ptr)->value);
                (*ptr)->value = nullptr;
            }
        } else if(*iter == ')') {
            return;
        } else {
            Node* next_node = new Node;
            next_node->value = "\0";
            (*ptr)->next = next_node;
            (*ptr)->value = *iter;
        }
    }
}

```

```

        iter++;
        createList(&(*ptr)->next, str, iter);
    }
    iter++;
    createList(&(*ptr)->next, str, iter);
}

void List::print(Node *ptr) {
    Node* tmp = ptr;
    while (tmp != nullptr)
    {
        if(std::holds_alternative<Node*>(tmp->value)) {
            if(std::get<Node*>(tmp->value) != nullptr) {
                print(std::get<Node*>(tmp->value));
            }
        } else {
            std::cout << std::get<char>(tmp->value) << ' ';
        }
        tmp = tmp->next;
    }
}

void List::replaceElem(Node* ptr, char Old, char New) {
    Node* tmp = ptr;
    while (tmp != nullptr)
    {
        if(std::holds_alternative<Node*>(tmp->value)) {
            if(std::get<Node*>(tmp->value) != nullptr) {
                replaceElem(std::get<Node*>(tmp->value), Old, New);
            }
        } else {
            if(std::get<char>(tmp->value) == Old) {
                tmp->value = New;
            }
        }
        tmp = tmp->next;
    }
}

List::~List() {
    deliting(head);
}

void List::deliting(Node* to_delete) {
    if(std::holds_alternative<Node*>(to_delete->value)) {
        if(std::get<Node*>(to_delete->value) != nullptr) {
            deliting(std::get<Node*>(to_delete->value));
        }
    }
    if(to_delete->next != nullptr) {
        deliting(to_delete->next);
    }
}

```

```
delete to_delete;  
}
```

ПРИЛОЖЕНИЕ В

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

№	Входные данные	Выходные данные	Комментарии
1	(abcde) cx	a b c d e a b x d e	Замена c на x в простом списке
2	(a((bc)n)de) bx	a b c n d e a x c n d e	Также замена b на x в дважды вложенном списке
3	(234) 25	2 3 4 5 3 4	Обработка цифр, замена 2 на 5
4	(a))((b) an	Incorrect string	Даже если количество '(' и ')' одинаково, но они не парные, выражение не будет обрабатываться
5	(abc25(i78)(lun1q(bsv) mn)rmc) b i	a b c 2 5 i 7 8 l u n 1 q b s v m n r m c a i c 2 5 i 7 8 l u n 1 q i s v m n r m c	Пример обработки длинной строки с буквами и цифрами
6	(isndilfn273487((alb)nek)nmsldfm)	i s n d i l f n 2 7 3 4 8 7 a l b n e k n m s l d f m	Замена строчной буквы на

	mS	i s n d i l f n 2 7 3 4 8 7 a l b n e k n S s l d f S	заглавную
7	(jaskldfj(imenjkn23874(j kd)imf)jksldjl) jX	j a s k l d f j i m e n j k n 2 3 8 7 4 j k d i m f j k s l d j l X a s k l d f X i m e n X k n 2 3 8 7 4 X k d i m f X k s l d X l	Также обработка длинной строки