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

Студент гр. 9304 _____ Боблаков Д.С.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург

2020

Цель работы

Научиться реализовывать иерархические списки. Получить навыки работы с иерархическими списками.

Задание

Вариант 1.

Подсчитать общий вес заданного бинарного коромысла bk, т. е. суммарный вес его гирек. Для этого ввести рекурсивную функцию unsigned int W (const БинКор bk).

Описание алгоритма работы

Сначала программа считывает строку str, включая пробелы. После чего программа проверяет корректность введенных данных используя стек. Если введенные данные некорректны, программа выведет сообщение об ошибке и завершит свою работу. Если введенные данные корректны, то программа рекурсивно строит бинарное коромысло (иерархический список) по следующему алгоритму: первое встреченное число до символа « » записывается в поле length, после этого, если встречено число, то оно записывается в поле value, если встречена открывающая скобка, то создается указатель на подсписок (левая ветвь бинарного коромысла), затем вызывается рекурсивно функция построения коромысла с новыми параметрами; если был встречен символ « », то при следующей открывающей скобке будет вызвана рекурсивно функция построения коромысла уже для правого плеча; рекурсивная функция заканчивает свою работу при извлечении закрывающей скобки. Такой алгоритм происходит до конца строки. Далее вызывается рекурсивная функция, проходящая по всему списку и подсчитывающая общую массу грузов по алгоритму ЛПК (Левый узел, Правый узел, Корень). Если хранится узел, то рекурсия продолжается, иначе хранится груз, и к

общей массе добавляется его вес. В конце работы программы выводится результат.

Разработанный код см. в приложении А.

Формат входных и выходных данных

Программа принимает на вход строку формата <((N X) (N X))>, где на месте N находится число, а на месте X находится либо число, либо строка формата <(N X)>, где на месте N также находится число, а на месте X находится либо число, либо строка аналогичного формата. Поля N и X не могут быть пустыми и/или не разделяться пробелами.

На выходе программа выведет число в случае корректного ввода данных, в ином случае программа выведет следующую строку: «Error: incorrect value».

Описание основных структур данных и функций

Class Node

Экземпляр данного класса будет являться узлом иерархического списка. Поле length хранит длину плеча, а поле value хранит либо пару умных указателй на следующие узлы, либо массу груза.

Bool isCorrect(const std::string& str)

Данная функция принимает строку str и проверяет ее на корректность записи бинарного коромысла.

unsigned int createBK(std::string & str, long unsigned int index, std::shared_ptr<Node> bk)

Данная функция принимает строку str в которой записано бинарное коромысло, индекс index в котором хранится номер элемента строки (нужен для рекурсивного вызова данной функции), умный указатель на объект класса

Node в который будет записана голова иерархического списка. Эта функция рекурсивно строит бинарное коромысло.

unsigned int W(const std::shared_ptr<Node> bk, unsigned int& count)

Данная функция принимает умный указатель на объект класса Node и счетчик массы count, а затем рекурсивно проходит по иерархическому списку и подсчитывает общую массу бинарного коромысла.

Визуализация иерархического списка

Визуализация бинарного коромысла из строки:

((5 10) (7 20))

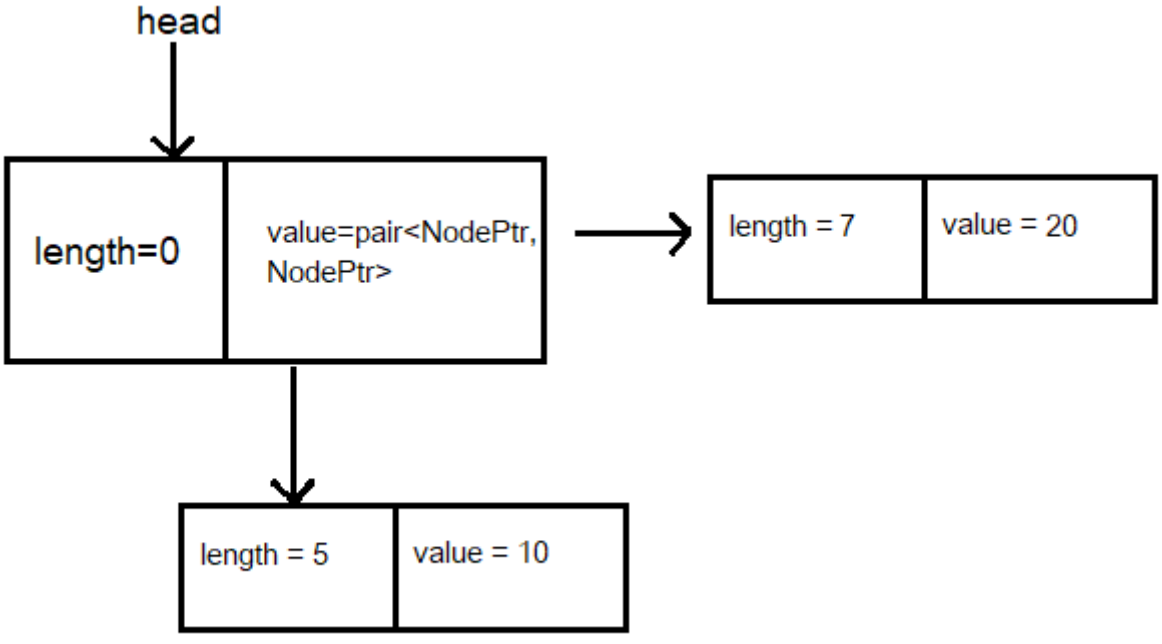

Рисунок 1 – Бинарное коромысло.

Тестирование

Тестирование программы проводится с помощью bash-скрипта tests_script. Для запуска тестирования необходимо выполнить команду ./tests_scripts, предварительно собрав программу с помощью команды make.

Для каждого теста выводится сообщение Test# <входные данные> - passed или Test# <входные данные> - failed. Также в каждом тесте выводится ожидаемый и полученный результат. Файлы с выходными данными после завершения тестирования удаляются.

Результаты тестирования см. в приложении Б.

Выводы

Было изучено понятие иерархического списка и его устройство. Были приобретены практические навыки для работы с иерархическими списками.

Была реализована программа на языке программирования C++, осуществляющая построение бинарного коромысла, а также подсчет общей массы элементов этой структуры данных. Проведено тестирование работоспособности программы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: main.cpp

```
#include <iostream>
#include <stack>
#include <memory>
#include <variant>

class Node {
    using NodePtr = std::shared_ptr<Node>;
public:
    int length{};
    std::variant<std::pair<NodePtr , NodePtr>, unsigned int >
value;
};

unsigned int createBK(std::string & str, long unsigned int index,
std::shared_ptr<Node> bk) {
    using NodePtr = std::shared_ptr<Node>;
    std::pair<NodePtr, NodePtr> side;
    if (isdigit(str[index])) {
        bk->length = std::stoi(str.substr(index));
    }
    while (isdigit(str[index])){
        index++;
    }
    if (str[index] == ' ')
        index++;
    if (str[index] == '(') {
       NodePtr left = std::make_shared<Node>();
        side.first = left;
        bk->value = side;
        while (str[index] == '(')
            index++;
```

```
            index=createBK(str, index,
std::get<std::pair<NodePtr,NodePtr>>(bk->value).first);
        }
        else {
            bk->value = std::stoi(str.substr(index));
            while (isdigit(str[index])) {
                index++;
            }
        }
        if (str[index] == ' ')
            index++;
        if (str[index] == '(') {
            NodePtr right=std::make_shared<Node>();
            side.second = right;
            bk->value = side;
            index++;
            index = createBK(str, index, std::get<std::pair<NodePtr ,
NodePtr>>(bk->value).second);
            if (index == str.length() - 1)
                return 0;
            else
                index++;
        }
        if (str[index] == ')') {
            index++;
            return index;
        }
        return 0;
    }

    bool isCorrect(const std::string& str){
        int amount=0;
        for(char i:str){
            if (isdigit(i)){
                amount++;
```

```
            }
        }
        if (amount==0){
            return false;
        }
        std::stack <char> steck;
        for (char i : str) {
            if (i == '('
                steck.push(i);
            else if (i == ')') {
                if (steck.empty()) {
                    return false;
                }
                steck.pop();
            }
            else {
                if ((i != ' ' && !isdigit(i)) || steck.empty()) {
                    return false;
                }
            }
        }
        return steck.empty();
    }

    unsigned int W(const std::shared_ptr<Node> bk, unsigned int&
count) {
        using NodePtr =std::shared_ptr<Node>;
        if (std::holds_alternative<std::pair<NodePtr, NodePtr>>(bk-
>value)) {
            if (std::get<std::pair<NodePtr , NodePtr>>(bk-
>value).first != nullptr) {
                W(std::get<std::pair<NodePtr , NodePtr>>(bk-
>value).first, count);
            }
        }
```

```
        if (std::holds_alternative<std::pair<NodePtr , NodePtr>>(bk-
>value)) {
            if (std::get<std::pair<NodePtr , NodePtr>>(bk-
>value).second != nullptr) {
                W(std::get<std::pair<NodePtr , NodePtr>>(bk-
>value).second, count);
            }
        } else {
            count += std::get<unsigned int>(bk->value);
        }

        return count;
    }
    int main()
    {
        std::string str;
        std::getline(std::cin,str);
        if (!isCorrect(str)) {
            std::cout << "Error: incorrect value\n";
            return EXIT_FAILURE;
        }
        unsigned int count=0;
        unsigned int index =0;
        using NodePtr=std::shared_ptr<Node>;
        NodePtr bk= std::make_shared<Node>();
        bk->length=0;
        createBK(str, index, bk);
        unsigned int res=W(bk,count);
        std::cout<<res<<"\n";

        return 0;
    }
```

Файл: Makefile

```
        g++ -std=c++17 Source/main.cpp -o lab2
```

```
run_tests: lab2
        ./tests_script
```

Файл: tests_scripts.sh

```
#! /bin/bash

printf "\nRunning tests...\n\n"
for n in {1..10}
do
    ./lab2 < "./Tests/tests/test$n.txt" > "./Tests/out/out$n.txt"
    printf "Test$n: "
    cat "./Tests/tests/test$n.txt" | tr -d '\n'
    if cmp "./Tests/out/out$n.txt"
"./Tests/true_results/true_out$n.txt" > /dev/null; then
            printf " - Passed\n"
    else
            printf " - Failed\n"
    fi
    printf "Desired result:\n"
    cat "./Tests/true_results/true_out$n.txt"
    printf "Actual result:\n"
    cat "./Tests/out/out$n.txt"
    printf "==============\n"
done
rm ./Tests/out/out*
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Результаты тестирования представлены в таблице Б.1

Таблица Б.1 — Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	((123 423) (456 789))	1212	Бинарное коромысло с двумя грузами. (Пример 1 из исходного задания)
2.	((10 ((10 25) (10 25))) (10 25))	75	Бинарное коромысло с допонительным коромыслом слева. (Пример 2 из исходного задания)
3.	((1 ((11 11) (12 ((121 121) (122 122))))) (2 ((21 21) (22 22))))	297	Сложное бинарное коромысло. (Пример 3 из исходного задания)
4.	(( ) ( ))	Error: incorrect value	Пустое бинарное коромысло.
5.	((asd qwe) (qbgf qw))	Error: incorrect value	Коромысло с некорректными значениями длины плеча и массой грузов.
6.	CONSERVATIVE ANIME REVOLUTION	Error: incorrect value	Строка не являющаяся бинарным коромыслом.
7.	((60 (90 45)) ((78 81) (133 28)))	154	Корректное бинарное коромысло.
8.	()()()()()()	Error: incorrect value	Пустые скобки не являющиеся бинарным коромыслом

9.	((7 90) (((100 2) (64 46)) ((222 50) (39 91))))	138	Корректное бинарное коромысло.
10.	((12309qwert 213) (878 zxc))	Error: incorrect value	Бинарное коромысло с некорректными параметрами.