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: Рекурсивна обработка иерархических списков

Студент гр. 9382 _____ Дерюгин Д.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Научится рекурсивно обрабатывать иерархические списки.

Задание.

2 вариант.

Подсчитать число всех гирек заданного бинарного коромысла bk. Для этого ввести рекурсивную функцию

unsigned int numbers (**const** БинКор bk).

Описание алгоритма.

После того, как на функции передали бинарное коромысло, функция проверяет: является ли левое плечо бинарным коромыслом(если isBinaryRocker истина, значит это коромысло, в противном случае это гирька). Если это так, то она вызывает саму себя же. Если же плечо не бинарное коромысло, то конечный результат инкрементируется. После происходит тоже самое, но уже с правым плечом.

Структура иерархического списка.

```
struct shoulder {
    bool isBinaryRocker; // true: binaryRocker, false: weight
    int length;
    int weight;
    binaryRocker* binaryRockerPtr;
}; //end shoulder

struct binaryRocker {
    shoulder *left;
    shoulder *right;
}; //end binaryRocker
```

Создано 2 структуры для реализации иерархического списка: struct shoulder, struct binaryRocker

Struct shoulder - структура, котора содержит в себе информацию о плече бинарного коромысла:

Bool isBinaryRocker - булева переменная, которая принимает истину, если плечо - бинарное коромысло;

Int length - целочисленная переменная, которая хранит длину плеча;

Int weight - целочисленная переменная, которая хранит вес гирьки(если она есть);

binaryRocker* binaryRockerPtr - указатель на бинарное коромысло, если плечо - бинарное коромысло.

Struct binaryRocker - структура которая содержит в себе 2 указателя типа Side* на левое и правое плечо коромысла.

Описание функций и остальных структур.

Перечисление Errors. Содержит в себе все возможные ошибки, которые могут произойти в ходе программы.

void error(Errors error) - на вход подаётся перечисление. Ничего не возвращает. Выводит информацию об ошибке.

shoulder* makeLength (char character) - на вход подаётся символ, который является длинной данного плеча. Возвращает указатель на плечо. Создаёт новое плечо и присваивает ему длину.

binaryRocker* readBinaryRocker(binaryRocker* binRock, char character, ifstream& fin) - на вход подаются указатель на бинарное коромысло, текущий символ из файла и сам файл. Возвращает бинарное коромысло. Создаёт для данного бинарного коромысла левое и правое плечо.

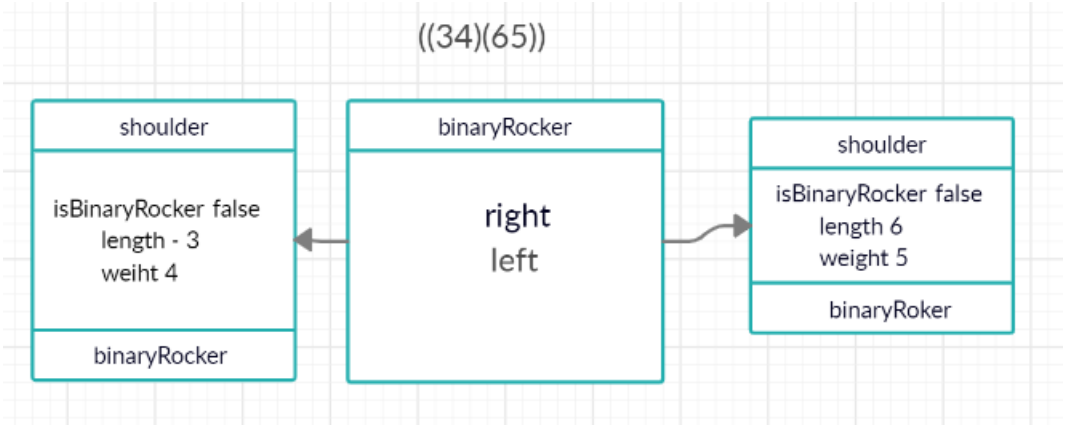
shoulder * readShoulder(char prev, binaryRocker* binRock, ifstream& fin) - на вход подаются предыдущий символ из файла, указатель на бинарное коромысло, а также файл. Возвращает указатель на плечо. Создает плечо для бинарного коромысла.

void printResult(const binaryRocker* binRock, int side, int tab) - на вход подаются бинарное коромысло, целочисленную переменную side, которая указывает, какое именно плечо нужно выводить и глубину рекурсии. Ничего не возвращает. Выводит графическое представление заданного бинарного коромысла.

unsigned int numbers(unsigned int count, const binaryRocker* binRock, int tab) - функция, которая считает количество гирек.

void destroy(binaryRocker binRock) - на вход подается бинарное коромысло. Ничего не возвращает. Освобождает память, выделенную под данное коромысло.

Графическая схема примера иерархического списка.



Тестирование.

№	Входные данные	Выходные данные	Комментарии
1	((1((23)(45)))(6((4((5((54)(54)))(75)))(6((54)(54))))))	Intermediate results: Left shoulder (1 ( )) Left shoulder (2 3) Right shoulder (4 5) Right shoulder (6 ( )) Left shoulder (4 ( )) Left shoulder (5 ( )) Left shoulder (5 4) Right shoulder (5 4) Right shoulder (7 5) Right shoulder (6 ( )) Left shoulder (5 4) Right shoulder (5 4) Count of weights: 7	

2	dkfjdslf	Incorrect first symbol	Действительно, первый символ должен быть «(«
3	((43)(45)	No close bracket	Нет закрывающийся скобки
4	((a5)(54))	Length of shoulder must be number	Длина должна быть числом, а не буквой.
5	((64)(65))выа	Extra symbols	Есть лишние символы
6	((64)(96))	Intermediate results: Left shoulder (6 4) Right shoulder (9 8) Count of weights: 2	Без комментариев.
7	((6((65)(87))(85))	No close bracket	Нет закрывающийся скобки после первого плеча
8	((6((65)(87)))(85))	Intermediate results: Left shoulder (6 ( )) Left shoulder (6 5) Right shoulder (8 7) Right shoulder (8 5) Count of weights: 3	Без комментариев.

Выводы.

Были изучены методы работы с иерархическим списком на языке C++, а также создана программа, которая подсчитывает, сколько гирек есть в бинарном коромысле.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
#include <iostream>
#include <fstream>

using namespace std;

enum Errors{
    INCORRECT_FIRST_SYMBOL,
    NO_CLOSE_BRACKET,
    UNFINISHED_EXPRESSION,
    NO_OPEN_BRACKET,
    INCORRECT_LENGTH,
    CANNOT_OPEN_FILE,
    EXTRA_SYMBOLS
};

struct binaryRocker; //binaryRocker struct declaration

struct shoulder {
    bool isBinaryRocker; // true: binaryRocker, false: weight
    int length;
    int weight;
    binaryRocker* binaryRockerPtr;
}; //end shoulder

struct binaryRocker {
    shoulder *left;
    shoulder *right;
}; //end binaryRocker

// Functions prototypes
void error(Errors error);
shoulder* makeLength (char character);
```

```
void makeWidth(char character, shoulder* shoulder);
binaryRocker* readBinaryRocker( binaryRocker* binRock, char character, ifstream& fin);
shoulder * readShoulder(char prev, binaryRocker* binRock, ifstream& fin);
void printResult(const binaryRocker* binRock, int side, int tab);
unsigned int numbers(unsigned int count, const binaryRocker* binRock, int tab);
void destroy(binaryRocker binRock);


void error(Errors error) {
    if (error == INCORRECT_FIRST_SYMBOL) cout<<"Incorrect first symbol\n";
    if (error == NO_CLOSE_BRACKET) cout<<"No close bracket\n";
    if (error == UNFINISHED_EXPRESSION) cout<<"Something went wrong\n";
    if (error == NO_OPEN_BRACKET) cout<<"Shoulder must start with '('\n";
    if (error == INCORRECT_LENGTH) cout<<"Length of shoulder must be number\n";
    if (error == CANNOT_OPEN_FILE) cout<<"Cannot open file\n";
    if (error == EXTRA_SYMBOLS) cout<<"Extra symbols\n";
}// end error func


shoulder* makeLength(char character) {
    auto *sh = new shoulder;
    sh->length = character - 48; // char to int
    return sh;
}// end makeLength func


void makeWeight(char character, shoulder* shoulder) {
    shoulder->weight = character - 48;// char to int
    shoulder->isBinaryRocker = false;
    shoulder->binaryRockerPtr = nullptr;
}// end makeWeight func


binaryRocker* readBinaryRocker(binaryRocker* binRock, char character, ifstream& fin) {
    if (character == '(') {
```

```
        binRock->left = readShoulder(character, binRock, fin);
        binRock->right = readShoulder(character, binRock, fin);
    }
    else {
        error(INCORRECT_FIRST_SYMBOL);
        exit(1);
    }
    fin>>character;
    if (character != ')') {
        error(NO_CLOSE_BRACKET);
        exit(1);
    }
    return binRock;
} //end readBinaryRocker func


shoulder* readShoulder(char prev, binaryRocker* binRock, ifstream& fin) { //prev - previous character
        auto* sh = new shoulder;
        if (!(fin>>prev)) {
            error(UNFINISHED_EXPRESSION);
            exit(1);
        }
        if (prev != '(') {
            error(NO_OPEN_BRACKET);
            exit(1);
        }
        fin>>prev;
        if (isdigit(prev)) sh = makeLength(prev);
        else{
            error(INCORRECT_LENGTH);
            exit(1);
        }
        fin>>prev;
        if (isdigit(prev)){
```

```
            makeWeight(prev, sh);
        }
        else{
            auto* br = new binaryRocker;
            sh->binaryRockerPtr = readBinaryRocker(br, prev, fin);
            sh->isBinaryRocker = true;
            sh->weight = 0;
        }
        fin>>prev;
        if (prev != ')') {
            error(NO_CLOSE_BRACKET);
            exit(1);
        }
        return sh;

}// end readShoulder func


void printResult(const binaryRocker* binRock, int side, int tab) {//side 1 - left, side 2 - right
    if (side == 1) {
        if (!binRock->left->isBinaryRocker) {
            for (int i = 0; i < tab; i++) cout<<"\t";
            cout<<"Left shoulder ("<<binRock->left->length<<" "<<binRock->left->weight<<")\n";
        }
        else {
            for (int i = 0; i < tab; i++) cout<<"\t";
            cout<<"Left shoulder ("<<binRock->left->length<<" (  ))\n";
        }
    }
    else {
        if (!binRock->right->isBinaryRocker) {
            for (int i = 0; i < tab; i++) cout<<"\t";
            cout<<"Right shoulder ("<<binRock->right->length<<" "<<binRock->right-
>weight<<")\n";
        }
```

```
        else {
            for (int i = 0; i < tab; i++) cout<<"\t";
            cout<<"Right shoulder ("<<binRock->right->length<<" (  ))\n";
        }
    }
}


unsigned int numbers(unsigned int count, const binaryRocker* binRock, int tab) {
    if (binRock->left->isBinaryRocker) {
        printResult(binRock, 1, tab);
        count = numbers(count, binRock->left->binaryRockerPtr, tab +1);
    }
    else {
        printResult(binRock, 1, tab);
        count++;
    }
    if (binRock->right->isBinaryRocker) {
        printResult(binRock, 2, tab);
        count = numbers(count, binRock->right->binaryRockerPtr, tab + 1);
    }
    else {
        printResult(binRock, 2, tab);
        count++;
    }

    return count;
}// end numbers func



void destroy(binaryRocker* binRock) {
    if (binRock->left->weight == 0) destroy(binRock->left->binaryRockerPtr);
    else delete binRock->left;
```

```
    if (binRock->right->weight == 0) destroy(binRock->right->binaryRockerPtr);
    else delete binRock->right;
}// end destroy func


int main ( ) {
    char character;// previous character
    string path = "input.txt";// path to input file
    unsigned int result;//answer on question

    ifstream fin;
    fin.open(path);
    if (!fin.is_open()){
        error(CANNOT_OPEN_FILE);
        exit(1);
    }

    auto* binRock = new binaryRocker;
    fin>>character;
    readBinaryRocker(binRock, character,  fin);
    if (fin>>character){
        error(EXTRA_SYMBOLS);
        exit(1);
    }
    cout<<"Intermediate results:\n";
    result = numbers(0, binRock, 0);
    cout<<"Count of weights: "<<result;//write answer
    destroy(binRock);
    delete binRock;
    fin.close();//close file
    return 0;
}// end main func
```