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: Рекурсивная обработка иерархических списков

Студентка гр. 9303 _____ Хафаева Н.Л.

Преподаватель _____ Филатов Ар.Ю.

Санкт-Петербург

2020

Цель работы.

Изучить принцип работы иерархических списков.

Задание.

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

unsigned int numbers (const БинКор bk).

Основные теоретические положения.

< S_expr (El) > ::= < Atomic (El) > | < L_list (S_expr (El)) >,
< Atomic (E) > ::= < El >.

Иерархический список, согласно определению, представляет собой или элемент базового типа *El*, называемый в этом случае атомом (атомарным *S*-выражением), или линейный список из S-выражений. Приведенное определение задает структуру непустого иерархического списка как элемента размеченного объединения множества атомов и множества пар «голова»–«хвост» и порождает различные формы представления в зависимости от принятой формы представления линейного списка. Традиционно иерархические списки представляют или графически, используя для изображения структуры списка двухмерный рисунок, или в виде одномерной скобочной записи.

Выполнение работы.

Были созданы следующие структуры:

BinRocker – бинарное коромысло, который имеет два плеча sh1, *sh2*

Shoulder – плечо коромысла, которое содержит в себе поля *length* (длина плеча), *cond* (определяет есть ли продолжение у плеча, *true/false*), *Cargo*.

Cargo – объект, который хранится в плече. Содержит поля *weight* (вес гири) или *binR* (указатель на еще одно бинарное коромысло).

numbers (const BinRocker rocker)* – функция, которая считает количество гирек коромысла.

detour_shoulder (Shoulder sh, int& result, int& level), detour_rocker (BinRocker* rocker, int& result, int& level)* – функции, которые проходят по всем плечам коромысла.

Destroy(BinRocker rocker), Destroy_shoulder(Shoulder* sh)* – функции, которые удаляют список.

start_read_from_file(BinRocker rocker, ifstream& input_file)* – функция, которая начинает считывать элементы с файла.

read_shoulder_from_file(ifstream& input_file) – функция, которая считывает плечо коромысла.

Выводы.

В работе был реализован иерархический список и функции для работы с ним. Ввод данных в программе осуществляется через файл. Было проведено тестирование программы, результаты тестирования содержатся в отчете.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

using namespace std;

struct binrocker;
struct shoulder;
struct cargo;

void start_read_from_file(binrocker* rocker, ifstream&
input_file);
void read_binrocker_from_file(char prev, binrocker* rocker,
ifstream& input_file);
shoulder* read_shoulder_from_file(ifstream& input_file);
void detour_shoulder(shoulder* sh, int& result, int& level);
void detour_rocker(binrocker* rocker, int& result, int&
level);
void output_result(const int level, const int result);
unsigned int numbers (const binrocker* rocker);
void destroy(binrocker* rocker);
void destroy_shoulder(shoulder* sh);
void write_shoulder(shoulder* sh);
void write_binr(binrocker* rocker);

struct binrocker {  // коромысло
    shoulder *sh1 = nullptr;
    shoulder *sh2 = nullptr;

};

struct shoulder {   // плечо
    int length = 0;
    cargo *obj = nullptr;
    bool cond = false;
};

struct cargo {    //объект
    int weight = 0;
    binrocker *binr = nullptr;
};

int main() {
    string input_file;
    binrocker *rocker = new binrocker;
    cout << "enter file name: " << endl;
    getline(cin, input_file);
    ifstream input(input_file);
```

```
        if (!input) {
            cout << "input file is invalid." << endl;
            return 0;
        } else {
            start_read_from_file(rocker, input);
        }

        write_binr(rocker);
        cout << endl;
        int k = numbers(rocker);

        cout << "numbers = " << k << endl;

        return 0;

    }

    void start_read_from_file(binrocker* rocker, ifstream&
input_file) {
        char buf;
        input_file.get(buf);
        if (buf == '(') {
            read_binrocker_from_file(buf, rocker, input_file);
        }

    }

    void read_binrocker_from_file(char prev, binrocker* rocker,
ifstream& input_file) {
        if (prev == '(') {
            rocker->sh1 = read_shoulder_from_file(input_file);
            if (rocker->sh1 == nullptr) {
                cout << "error." << endl;
                exit(0);
            }
            char buf;
            input_file.get(buf);

            rocker->sh2 = read_shoulder_from_file(input_file);
            if (rocker->sh2 == nullptr) {
                cout << "error." << endl;
                exit(0);
            }
            input_file.get(buf);
            if (buf != ')') {
                cout << "error." << endl;
                exit(0);
            }
        }
        else {
            cout << "error." << endl;
            exit(0);
```

```
        }
    }

    shoulder* read_shoulder_from_file(ifstream& input_file) {
        char buf;
        shoulder* sh = new shoulder;
        input_file.get(buf);
        if (buf == '(') {
            input_file >> sh->length;
            input_file.get(buf);
            if (buf == '(') {
                sh->cond = true;
                cargo* cr = new cargo;
                sh->obj = cr;
                binrocker* binr = new binrocker;
                sh->obj->binr = binr;
                read_binrocker_from_file(buf, sh->obj->binr,
input_file);
            }
            else if (buf == ',') {
                sh->cond = false;
                cargo* cr = new cargo;
                sh->obj = cr;
                input_file >> sh->obj->weight;
            }
            input_file.get(buf);

            if (buf != ')') {
                return nullptr;
            }
        }
        else {
            return nullptr;
        }

        return sh;
    }

    void detour_shoulder(shoulder* sh, int& result, int& level) {
        level += 1;
        if (!sh->cond) {
            result += 1;
        }
        output_result(level,  result);
        if (sh->cond) {
            detour_rocker(sh->obj->binr, result, level);
        }
        level -= 1;
    }

    void detour_rocker(binrocker* rocker, int& result, int&
level) {
        detour_shoulder(rocker->sh1, result, level);
```

```
    detour_shoulder(rocker->sh2, result, level);
}

unsigned int numbers (const binrocker* rocker) {
    int result = 0;
    int level = 0;
    detour_shoulder(rocker->sh1, result, level);
    detour_shoulder(rocker->sh2, result, level);
    return result;
}

void output_result(const int level, const int result) {
    for (int i = 0; i < level; i++)
        cout << ' ';
    cout << "rocker depth = " << level << endl;
}

void destroy(binrocker* rocker) {
     if (rocker->sh1 != nullptr)
          destroy_shoulder(rocker->sh1);
     if (rocker->sh2 != nullptr)
          destroy_shoulder(rocker->sh2);
     delete rocker;
}

void destroy_shoulder(shoulder* sh) {
     if (sh->cond == true)
          destroy(sh->obj->binr);
     delete sh->obj;
     delete sh;
}

void write_shoulder(shoulder* sh) {
     if (sh == nullptr) {
          cout << "error." << endl;
          return;
     }
     cout << '(';
     cout << sh->length;
     if (sh->cond == false) {
          cout << ',';
          cout << sh->obj->weight;
     }
     else {
          write_binr(sh->obj->binr);
    }
     cout << ')';
}

void write_binr(binrocker* rocker) {
     if (rocker == nullptr) {
        cout << "error." << endl;
        return;
```

```
    }
     cout << '(';
     write_shoulder(rocker->sh1);
     cout << ' ';
     write_shoulder(rocker->sh2);
     cout << ')';
}
```

ПРИЛОЖЕНИЕ В
ТЕСТИРОВАНИЕ

№	Входные данные	Результат
1	((2((3,10),(1,0))),(4,-1))	((2((3,10) (1,0))) (4,-1)) rocker depth = 1 rocker depth = 2 rocker depth = 2 rocker depth = 1 numbers = 3
2	((4((3,1),(2,3))),(4((3,5),(7,9))))	((4((3,1) (2,3))) (4((3,5) (7,9)))) rocker depth = 1 rocker depth = 2 rocker depth = 2 rocker depth = 1 rocker depth = 2 rocker depth = 2 numbers = 4
3	((2,3),(8,4))	((2,3) (8,4)) rocker depth = 1 rocker depth = 1 numbers = 2
4	(2,3)	Error.