**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по лабораторной работе №2**

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

**Тема:** **Бинарное коромысло**

|  |  |  |
| --- | --- | --- |
| Студент гр. 7381 |  | Судакова П.С. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2018

**Цель работы.**

Ознакомиться с основными приёмами реализации иерархических списков, получить навыки программирования иерархических списков на языке программирования C++.

**Формулировка задания.**

Подсчитать общий вес заданного бинарного коромысла bk, т. е. суммарный вес его гирек.

**Описание входных и выходных данных.**

Входные данные: Входные данные содержит файл “text.txt”, содержащий последовательность скобок, пробелов и цифр, задающих бинарное коромысло.

Выходные данные:Бинарное коромысло в иерархическом виде вместе с общим его весом.

**Реализация.**

Функции bk() и shoulder() анализируют содержимое БК и плеча соответственно, в процессе занося результаты в подобие иерархического списка. Далее, при помощи функции printNode рекурсивно выводит на экран содержимое этого списка.

**Тестирование программы.**

|  |  |
| --- | --- |
| Входные данные | Результат работы программы |
| ((5 10) (6 ((2 ((58 789) (11 ((5 16) (77 99))))) (8 228)))) |  |
| ((1 5) (5 1)) |  |

По результатам тестирования программа показала хорошие результаты.

**Выводы.**

В результате выполнения данной лабораторной работы была написана программа, использующая бинарное коромысло.

**Приложение А  
Основной код программы**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Core;

struct Branch;

Core\* bk(char s);

Branch\* shoulder(char s);

char newSym();

int number(int s);

int toInt(char s);

bool isInt(char s);

void log(string msg, int depth);

void terminate(string msg);

void printNode(Core \*node);

void reverse();

struct Core {

int weight;

Branch \*left;

Branch \*right;

};

struct Branch {

int length;

Core \*end;

};

string line;

int linePos = 0;

int nodeDepth = 0;

ifstream myfile("text.txt");

// Отдает следующий символ из строки

char newSym() {

if (linePos < line.length()) {

return line.at(linePos++);

} else {

return '-';

}

}

// Выводит сообщение с отступом

void log(string msg, int depth) {

cout << depth << " ";

for(int i = 0; i < depth\*3; ++i) {

cout << " ";

}

cout << msg << endl;

}

// Получает первый символ описания БК

// и ожидает, что это оно

Core\* bk(char s) {

if(s != '(') { terminate("bk - missing '('"); }

Branch\* leftBranch;

leftBranch = shoulder(newSym()); // Получаем левое плечо

if(newSym() != ' ') { terminate("bk - missing ' '"); }

Branch\* rightBranch;

rightBranch = shoulder(newSym()); // Получаем правое плечо

if(newSym() != ')') { terminate("bk - missing ')'"); }

Core \*node = new Core();

node->left = leftBranch;

node->right = rightBranch;

node->weight = leftBranch->end->weight +

rightBranch->end->weight;

return node;

}

// Получает первый символ описания плеча

// и ожидает, что это оно

Branch\* shoulder(char s) {

if(s != '(') { terminate("sh - missing '('"); }

// Начинаем получать первое число - длину плеча

char p = newSym();

int currentLength = number(toInt(p));

Branch \*br = new Branch();

br->length = currentLength;

if(newSym() != ' ') { terminate("sh - missing ' '"); }

Core \*target = new Core();

p = newSym();

if(isInt(p)) { // Если на этом плече висит груз

int currentWeight = number(toInt(p));

target->weight = currentWeight;

target->left = NULL;

target->right = NULL;

} else { // Если на этом плече висит еще одно БК

delete target;

target = bk(p);

}

if(newSym() != ')') { terminate("sh - missing ')'"); }

br->end = target;

return br;

}

// Склеивает число посимвольно,

// пока не встретит другой символ

int number(int s) {

char nxt = newSym(); // Получаем символ, следующий за числом

if (isInt(nxt)) { // Если это была цифра

int newNum = s\*10 + toInt(nxt); // Приклееваем ее в конец числа

return number(newNum); // Проверяем дальше на цифры

} else { // Если не цифра

reverse(); // Откатываем курор, чтобы потом получить этот символ

return s;

}

}

int main(int argc, char \*argv[])

{

if(myfile.is\_open()) {

cout << "Reading..." << endl;

getline (myfile,line); // Читаем строки заданием из файла

cout << line << endl;

myfile.close();

Core \*root = new Core(); // Создаём корневой элемент бинарного коромысла

root->left = NULL;

root->right = NULL;

root->weight = 0;

cout << "Parsing..." << endl << endl;

log("Ceiling", 0);

root = bk(newSym()); // Рекурсивно парсим строку

printNode(root); // Печатаем БК

}

return 0;

}

// Преобразует символ цифры в целое число

int toInt(char s) {

switch(s) {

case '0': return 0;

case '1': return 1;

case '2': return 2;

case '3': return 3;

case '4': return 4;

case '5': return 5;

case '6': return 6;

case '7': return 7;

case '8': return 8;

case '9': return 9;

}

return 0;

}

// В символе хранится цифра?

bool isInt(char s) {

if ( (s >= '0') && (s <= '9') ) {

return true;

} else {

return false;

}

}

// Когда случилась ошибка

void terminate(string msg) {

cout << endl << "=================================" << endl;

cout << "PROGRAM HAS BEEN TERMINATED: " << msg;

int i;

cin >> i;

exit(1);

}

void printNode(Core \*node) {

nodeDepth++;

if(node->left == NULL || node->right == NULL) { // Если нода пустая (висит груз)

log("\xC0 weight " + std::to\_string(node->weight), nodeDepth);

} else { // Если на ноде висит еще одно БК

log("\xC3 Node weight " + std::to\_string(node->weight), nodeDepth);

log("\xC0 left " + std::to\_string(node->left->length), nodeDepth);

printNode(node->left->end);

log("\xC0 right " + std::to\_string(node->right->length), nodeDepth);

printNode(node->right->end);

}

nodeDepth--;

}

// Откатываем следующий символ строки

void reverse() {

linePos--;

}