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. 8381 _____ Гоголев Е.Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивной обработки списков, изучить особенности реализации иерархического списка на языке программирования C++. Разработать программу, использующую иерархические списки и их рекурсивную обработку, вычисляющую длину всех плечей бинарного коромысла.

Задание.

Подсчитать общую длину всех плеч заданного бинарного коромысла bk. Для этого ввести рекурсивную функцию

```
short Length (const БинКор  bk).
```

Основные теоретические положения.

Согласно рекурсивному определению, иерархический список – такой список, элементами которого могут быть иерархические списки. Для обработки иерархического списка удобно использовать рекурсивные функции, так как он представляет собой множество линейный списков, между которыми установлены связи, иерархия.

Бинарное коромысло устроено так, что у него есть два плеча: левое и правое. Каждое плечо представляет собой (невесомый) стержень определенной длины, с которого свисает либо гирька, либо еще одно бинарное коромысло, устроенное таким же образом.

В соответствии с данным выше рекурсивным определением бинарного коромысла представим бинарное коромысло (БинКор) списком из двух элементов

```
БинКор ::= (Плечо Плечо),
```

где первое плечо является левым, а второе – правым. В свою очередь Плечо будет представляться списком из двух элементов

```
Плечо ::= (Длина Груз),
```

где Длина есть натуральное число, а Груз представляется вариантами

```
Груз ::= Гирька | БинКор,
```

где в свою очередь Гирька есть натуральное число. Таким образом, БинКор есть специального вида иерархический список из натуральных чисел.

Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в среде разработки CLion. Сборка, отладка и тестирование также производились в CLion.

Бинарное коромысло было реализовано в виде иерархического списка. Для этого создана структура BinKor, которая содержит два указателя на плечи бинарного коромысла (левое и правое). Плечо бинарного коромысла описано в структуре Side. Она содержит длину плеча (short length) и указатель на груз. Груз, в свою очередь, может быть либо гирькой, либо бинарным коромыслом. Он описан в структуре Load. Ее поле isWeight – переменная типа bool, которая указывает, что груз является гирькой. Объединение data включает в себя переменную типа int – weight, отвечающую за вес гирьки, и указатель на бинарное коромысло BinKor, что позволяет сэкономить память.

Функция Length, реализующая поставленную задачу (подсчитать общую длину всех плеч заданного бинарного коромысла), суммирует длины плеч переданного ей бинарного коромысла, и если один или оба груза являются бинарными коромыслами, рекурсивно проходит по ним, добавляя длину каждого плеча к общей сумме.

Основной проблемой при выполнении работы стало считывание и запись бинарного коромысла из введенной пользователем строки. Для этого были реализованы следующие функции:

- `void Skip (std::string& str, int n)`

  «Отрезает» от строки первые n символов
- `short GetNumber (string& str)`

  Считывает число из начала строки и возвращает его. При этом само число и последующий пробел из строки вырезаются.
- `BinKor* ScanBinKor (string& input)`

Создает бинарное коромысло, считывает в него левое и правое плечи с помощью функции ScanSide и возвращает его. Скобки по краям и пробел между плечами вырезаются из строки функцией Skip.

- `Side* ScanSide (string& input)`

  Создает плечо, считывает из строки его длину с помощью GetNumber и записывает в поле length. Затем считывает и также записывает груз, относящийся к данному плечу, используя ScanLoad.

- `Load* ScanLoad (string& input)`

  Создает груз, проверяет, является ли он гирькой, вызывает соответствующую функцию для считывания значения и записывает результат в объединение data.

Оценка эффективности алгоритма.

Алгоритм, реализованный в программе, имеет линейную зависимость от длины входной строки, то есть сложность оценивается как O(n). Рассуждения отталкиваются от того, что строка при считывании бинарного коромысла обрабатывается посимвольно, а функция Length вызывается для каждого вложенного списка (бинарного коромысла), а для отдельного бинарного коромысла выполняется за две операции.

Тестирование программы.

Ввод	Ожидаемый результат	Полученный результат
((2 2) (1 1))	3	3
((1 ((11 11) (12 12))) (2 2))	26	26
((1 ((11 11) (12((121 121) (122 122))))) (2 ((21 21) (22 22))))	212	212

Выводы.

В ходе выполнения лабораторной работы была изучена такая структура данных как иерархические списки, а также рекурсивные методы ее обработки. Была реализована программа на C++, использующая иерархические списки, которая вычисляет общую длину всех плеч заданного бинарного коромысла.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
//lr2.3
#include <iostream>
#include <string>
#include <cctype>
using namespace std;

struct BinKor;

//Груз ::= Гирька | БинКор
struct Load {
    bool isWeight;
    union {
        int weight;
        BinKor* binKor;
    } data;
};

//Плечо ::= (Длина Груз)
struct Side {
    short length;
    Load* load;
};

//БинКор ::= (Плечо Плечо)
struct BinKor {
    Side* right;
    Side* left;
};

short Length (const BinKor bk) {
    short result = 0;
    if (bk.right) {
        result += bk.right->length;
         if (bk.right->load && !bk.right->load->isWeight) result +=
Length(*bk.right->load->data.binKor);
    }
    if (bk.left) {
        result += bk.left->length;
          if (bk.left->load && !bk.left->load->isWeight) result +=
Length(*bk.left->load->data.binKor);
    }
    return result;
}
```

```
void Skip (std::string& str, int n) {
    if (str.length() >= n) {
        str = str.substr(n);
    }
}

short GetNumber (string& str) {
    string strnum = "";
    while (isdigit(str[0])) {
        strnum += str[0];
        Skip(str, 1);
    }
    return stoi(strnum);
}

BinKor* ScanBinKor (string& input);

// short|BinKor
Load* ScanLoad (string& input) {
    Load* load = new Load;
    load->isWeight = input[0] != '(';
    if (load->isWeight) {
        load->data.weight = GetNumber(input);
    }
    else {
        load->data.binKor = ScanBinKor(input);
    }
    return load;
}

// (short Load)
Side* ScanSide (string& input) {
    Side* side = new Side;
    Skip(input, 1);
    side->length = GetNumber(input);
    Skip(input, 1);
    side->load = ScanLoad(input);
    Skip(input, 1);
    return side;
}

// (Side Side)
BinKor* ScanBinKor (string& input) {
    BinKor* binKor = new BinKor;
    Skip(input, 1);
    binKor->left = ScanSide(input);
    Skip(input, 1);
    binKor->right = ScanSide(input);
    Skip(input, 1);
```

```
        return binKor;
    }

    int main () {
        string input = "((2 2) (1 1))";
        /*cout << Length(*ScanBinKor(input)) << endl;
        input = "((1 ((11 11) (12 12))) (2 2))";
        cout << Length(*ScanBinKor(input)) << endl;
         input = "((1 ((11 11) (12((121 121) (122 122))))) (2 ((21 21)
(22 22))))";
        cout << Length(*ScanBinKor(input)) << endl;*/
        cout << "Please enter binkor in the following format: ((a b) (c
d))\nwhere a, c are whole numbers\nb, d - whole numbers or binkors" <<
endl;
        cin >> input; // Ручной ввод
        cout << Length(*ScanBinKor(input)) << endl;
        return 0;
    }
```