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. 9381 _____ Аухадиев А.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить основные понятия иерархического списка и получить опыт работы с ним на языке программирования C++, разработать программу, использующую иерархический список.

Задание.

Вариант 3.

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

short Length (**const** БинКор bk).

Основные теоретические сведения.

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),

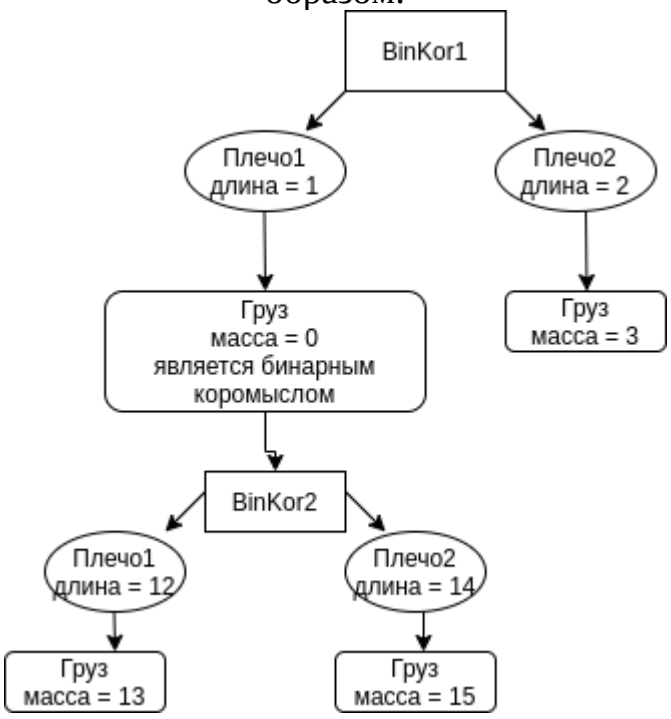
где первое плечо является левым, а второе – правым. В свою очередь Плечо будет представляться списком из двух элементов

Плечо ::= (Длина Груз),

где Длина есть натуральное число, а Груз представляется вариантами

Груз ::= (Гирька БинКор).

Коромысло вида ((1((12 13) (14 15))) (2 3)) можно представить следующим образом:



Описание алгоритма.

Для выполнения задачи были определены три структуры:

1. binKor (бинарное коромысло): два поля - shoulder* pl1 и shoulder* pl2 (указатели на два плеча);

2. shoulder (плечо): два поля - int l (длина) и cargo* mass (указатель на груз);

3. cargo (груз): три поля - bool tag (= true, если груз является бинарным коромыслом, =false, если груз является гирькой), int m - масса (=0, если tag = true) и binKor* bk - указатель на бинарное коромысло (=nullptr, если tag = false).

Функции, реализованные в программе, можно разделить на три части: сохранение введённых данных в иерархический список, обработка данных (подсчёт общей длины всех плеч) и очистка памяти.

Обработка данных включает в себя работу с тремя рекурсивными функциями:

1. short Length(const binKor* bk) - возвращает сумму результатов функций Length(bk->pl1) + Length(bk->pl2), то есть сумму длин первого и второго плеча со всеми прилегающими к ним другими бинарными коромыслами;

2. short Length(const shoulder* pl) - возвращает сумму поля l (длины плеча) и результата функции Length(pl->mass), который представляет суммарную длину плеч других бинарных коромысел, которые могут бить связаны с этим плечом.

3. short Length(const cargo* mass) - возвращает результат функции Length(mass->bk), если mass->tag = true, то есть является бинарным коромыслом, а не гирькой.

Таким образом совершается рекурсивный проход по всем плечам соединённых бинарных коромысел.

Тестирование.

№	Входные данные	Результат
1.	((1 ((2 3) (4 ((5 6) (7 8))))) (9 ((10 11) (12 13))))	Вызов функции Length(bk0) Вызов функции Length(pl1) Вызов функции Length(mass2) Вызов функции Length(bk3) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5) Завершение функции Length(pl4) Вызов функции Length(pl4) Вызов функции Length(mass5) Вызов функции Length(bk6) Вызов функции Length(pl7) Вызов функции Length(mass8) Завершение функции Length(mass8) Завершение функции Length(pl7) Вызов функции Length(pl7) Вызов функции Length(mass8) Завершение функции Length(mass8) Завершение функции Length(pl7) Завершение функции Length(bk6) Завершение функции Length(mass5) Завершение функции Length(pl4) Завершение функции Length(bk3) Завершение функции

		Length(mass2) Завершение функции Length(pl1) Вызов функции Length(pl1) Вызов функции Length(mass2) Вызов функции Length(bk3) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5) Завершение функции Length(pl4) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5) Завершение функции Length(pl4) Завершение функции Length(bk3) Завершение функции Length(mass2) Завершение функции Length(pl1) Завершение функции Length(bk0) 50
2.	((1((12 13) (14 15))) (2 3))	Вызов функции Length(bk0) Вызов функции Length(pl1) Вызов функции Length(mass2) Вызов функции Length(bk3) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5) Завершение функции Length(pl4) Вызов функции Length(pl4) Вызов функции Length(mass5)

		Завершение функции Length(mass5) Завершение функции Length(pl4) Завершение функции Length(bk3) Завершение функции Length(mass2) Завершение функции Length(pl1) Вызов функции Length(pl1) Вызов функции Length(mass2) Завершение функции Length(mass2) Завершение функции Length(pl1) Завершение функции Length(bk0) 29
3.	((1 2)(3 4))	Вызов функции Length(bk0) Вызов функции Length(pl1) Вызов функции Length(mass2) Завершение функции Length(mass2) Завершение функции Length(pl1) Вызов функции Length(pl1) Вызов функции Length(mass2) Завершение функции Length(mass2) Завершение функции Length(pl1) Завершение функции Length(bk0) 4
4.	((1 2)(3((4 5)(6 7))))	Вызов функции Length(bk0) Вызов функции Length(pl1) Вызов функции Length(mass2) Завершение функции Length(mass2) Завершение функции Length(pl1) Вызов функции Length(pl1) Вызов функции Length(mass2) Вызов функции Length(bk3) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5)

		Завершение функции Length(pl4) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5) Завершение функции Length(pl4) Завершение функции Length(bk3) Завершение функции Length(mass2) Завершение функции Length(pl1) Завершение функции Length(bk0) 14
5.	((1((11 12)(13 14))(2((21 22)(23 24))))	Вызов функции Length(bk0) Вызов функции Length(pl1) Вызов функции Length(mass2) Вызов функции Length(bk3) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5) Завершение функции Length(pl4) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5) Завершение функции Length(pl4) Завершение функции Length(bk3) Завершение функции Length(mass2) Завершение функции Length(pl1) Вызов функции Length(pl1) Вызов функции Length(mass2)

		Вызов функции Length(bk3) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5) Завершение функции Length(pl4) Вызов функции Length(pl4) Вызов функции Length(mass5) Завершение функции Length(mass5) Завершение функции Length(pl4) Завершение функции Length(bk3) Завершение функции Length(mass2) Завершение функции Length(pl1) Завершение функции Length(bk0) 71

Вывод.

Были изучены основные понятия иерархического списка и получен опыт работы с ним на языке программирования C++. Разработана программа, использующая иерархический список.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <iostream>
#include <string>
#include <cmath>
#include <fstream>
#include <cctype>
using namespace std;
struct cargo;
struct shoulder;
struct binKor;
int global_tab = 0;
binKor* new_bk(string& str);
shoulder* new_pl(string& str);
cargo* new_mass(string& str);
void clean_memory(binKor* bk);
void clean_memory(shoulder* pl);
void clean_memory(cargo* mass);
short Length(const binKor* bk);
short Length(const shoulder* bk, int num);
short Length(const cargo* bk, int num);
struct cargo{
	bool tag; // false - m, true - bk
	int m;
	binKor *bk;
};
struct shoulder{
	int l;
	cargo* mass;
};
struct binKor{
	shoulder *pl1;
	shoulder *pl2;
};
//Считывание и сохранение
binKor* new_bk(string& str) {
	binKor* bk = new binKor;
	if(str[0] == '(' && str[1] == '(') {
		str.erase(0, 2);
		bk->pl1 = new_pl(str);
	}else{
		cout << "ERROR\n";
		exit(0);
	}
	if(str[0] == '(' && str[1] != '('){
		str.erase(0, 1);
		bk->pl2 = new_pl(str);
	}else{
		cout << "ERROR\n";
```

```
            exit(0);
      }
      return bk;
}
shoulder* new_pl(string& str){
      shoulder* pl = new shoulder;
      pl->l = atoi(str.c_str());
      if(isdigit(str[0]))
            str.erase(0, log10(pl->l) + 1);
      else{
            cout << "ERROR\n";
            exit(0);
      }
      while(str[0] == ' ')
            str.erase(0, 1);
      pl->mass = new_mass(str);
      return pl;
}
cargo* new_mass(string& str){
      cargo* mass = new cargo;
      if(str[0] == '(') {
            mass->tag = true;
            mass->bk = new_bk(str);
      }else if(isdigit(str[0])){
            mass->tag = false;
            mass->m = atoi(str.c_str());
            str.erase(0, log10(mass->m)+1);
      }else{
            cout << "ERROR\n";
            exit(0);
      }
      while(str[0] == ')' || str[0] == ' ')
            str.erase(0,1);
      return mass;
}
//Очистка памяти
void clean_memory(binKor* bk){
      clean_memory(bk->pl1);
      clean_memory(bk->pl2);
      delete bk;
}
void clean_memory(shoulder* pl){
      clean_memory(pl->mass);
      delete pl;
}
void clean_memory(cargo* mass){
      if(mass->tag)
            clean_memory(mass->bk);
      delete mass;
}
```

```
//Нахождение суммарной длин всех плеч бинарных коромысел
short Length(const binKor* bk){
	int tab = global_tab;
	for(int i = 0; i<tab; i++)
		cout << "\t";
	cout << "Вызов функции Length(bk" << tab << ")\n";
	short len = Length(bk->pl1, tab+1);
	len+=Length(bk->pl2,tab+1);
	for(int i = 0; i<tab; i++)
		cout << "\t";
	cout << "Завершение функции Length(bk" << tab << ")\n";
	return len;
}
short Length(const shoulder* pl, int num){
	for(int i = 0; i<num; i++)
		cout << "\t";
	cout << "Вызов функции Length(pl" << num << ")\n";
	short len = Length(pl->mass, num+1) + pl->l;
	for(int i = 0; i<num; i++)
		cout << "\t";
	cout << "Завершение функции Length(pl" << num << ")\n";
	return len;
}
short Length(const cargo* mass, int num){
	for(int i = 0; i<num; i++)
		cout << "\t";
	cout << "Вызов функции Length(mass" << num << ")\n";
	short len = 0;
	if(mass->tag) {
		global_tab = num+1;
		len += Length(mass->bk);
	}
	for(int i = 0; i<num; i++)
		cout << "\t";
	cout << "Завершение функции Length(mass" << num << ")\n";
	return len;
}
int main(){
	string str;
	int a = 0;
	cout << "Откуда будет производиться ввод? (0 - консоль, 1 - файл)\n";
	cin >> a;
	if(a != 1 && a != 0){
		return 0;
	}
	if(a == 1){
		string filename;
		cout << "Введите название файла\n";
		cin >> filename;
		ifstream file(filename);
```

```
            if(!file.is_open()){
                  cout << "Не удалось открыть файл\n";
                  return 0;
            }
            getline(file, str);
            file.close();
      }else{
            cin.ignore();
            getline(cin, str);
      }
      if(str.empty()) {
            cout << "Вы не ввели никаких данных\n";
            return 0;
      }
      binKor* bk = new_bk(str);
      cout << Length(bk) << '\n';
      clean_memory(bk);
      return 0;
}
```