**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»**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 9381 |  | Андрух И.А. |
| Преподаватель |  | Фирсов М.А. |
|  |
|  |  |  |

Санкт-Петербург

2020

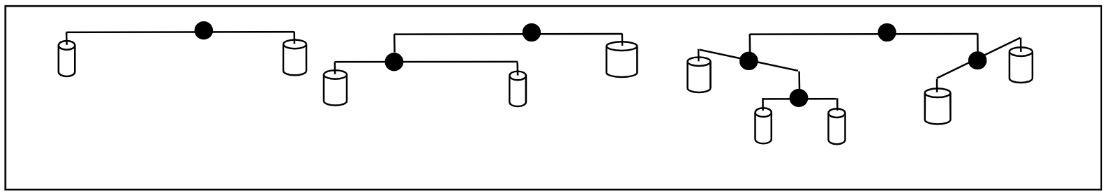
**Цель работы**

Ознакомиться с иерархическими списками и научиться применять их на практике.

**Основные теоретические положения**

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.

Возможный способ представления бинарного коромысла:



* соответствии с данным 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, бинарное коромысло есть специального вида иерархический список из натуральных чисел.

1

**Задание**

Вариант 2

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

**unsigned int** numbers (**const** БинКорbk).

**Ход работы.**

Программа написана на языке С++.

Исходный файл: main.cpp

* начале работы происходит ввод данных: строка, которая является бинарным коромыслом (пример: ((5 ((142 67) (6 7))) (1 2)) ), после чего происходит обработка строки и создание бинарного коромысла, для чего вызывается функция createBinCor(). Если входные данные некорректны, то программа выводит соответствующую ошибку и завершает работу. В случае ввода корректных данных происходит дальнейший вызов функции

numbers(), которая выводит на экран общее количество гирек в бинарном коромысле.

Для демонстрации работы было написано несколько тестов.

**Описание функций.**

1. *int createBinCor(char \* str, BinCor \*\* binCor)*

Функция принимает указатель на строку, в которую записано бинарное

коромысло и указатель на указатель на структуру BinCor (бинарное

коромысло). Функция устанавливает начальный и конечный индексы на

первый и последний символы полученной строки, после чего передаёт эти

данные функции readBinCorElement(), которая обрабатывает строку и

создаёт бинарное коромысло.

Параметры:

**str**:указатель на строку с бинарным коромыслом.

2

**binCor**:**​** указатель на указатель на структуру данныхBinCor,содержащий

адрес, куда будет заноситься результат обработки строки str. Возвращаемое значение: 0 – если данные в строке str корректны, в ином случае – 1.

1. *int***​** *readBinCorElement(char \* str, int index\_1, int index\_2, BinCor \*\* element)*

Эта функция вызывается в программе рекурсивно. Функция принимает указатель на строку, в которую записано бинарное коромысло, начальный и конечный индекс на первый и последний символы строки str, между которыми записано обрабатываемое на данном шаге бинарное коромысло и указатель на указатель на структуру BinCor(бинарное коромысло). Функция посимвольно обрабатывает два плеча бинарного коромысла, внося все данные по адресу, содержащемуся в указателе, на который указывает element. Если груз плеча является ещё одним бинарным коромыслом, то происходит рекурсивный вызов функции. При возникновении в какой-то момент ошибки вызывается

функция *errorMassage()***​**,***​***выводящай на экран сообщение об ошибке, после

чего функция завершает работу.

Параметры:

**str**:**​** указатель на строку с бинарным коромыслом.

**index\_1**:**​** индекс,с которого надо начать обрабатывать строкуstr.

**index\_2**:**​** индекс,которым надо закончить обрабатывать строкуstr.

**binCor**:**​** указатель на указатель на структуру данныхBinCor, содержащий адрес, куда будет заноситься результат обработки строки str. Возвращаемое значение: 0 – если данные в строке str корректны, в ином случае – 1.

1. *unsigned***​** *int numbers(const BinCor binCor, int deep\_of\_recursion)*

Функция принимает структуру данных BinCor (бинарное коромысло) и

3

целочисленное значение, указывающее на глубину рекурсии. Функция считает количество гирек в бинарном коромысле binCor. Параметры:

**binCor**:**​** бинарное коромысло,в котором считается количество гирек.

**deep\_of\_recursion**:**​** глубина рекурсии(необходимо для выводаработы алгоритма на экран).

Возвращаемое значение: количество гирек в бинарном коромысле binCor.

**Тестирование программы.**

Было создано несколько тестов для проверки работы программы. Помимо тестов, демонстрирующих работу алгоритма, были написаны тесты, содержащие некорректные данные, для демонстрации вывода сообщений об ошибках введенных данных (см. Приложение А).

**Вывод.**

В ходе выполнения работы была изучена новая структура данных: иерархические списки. Также закреплены навыки работы с рекурсивными функциями.

4

**ПРИЛОЖЕНИЕ А**

**Тестирование.**

Демонстрация работы:

input:

((5 ((142 67) (6 7))) (1 2))

output:

Программа выводит общее число гирек в указанном бинарном коромысле.

Бинарное коромысло записывается в виде:

(ПЛЕЧО ПЛЕЧО)

Плечо имеет следующий вид:

(ДЛИНА ГРУЗ)

В качестве груза может выступать ещё одно бинарное коромысло или груз (число).

Введите бинарное коромысло (не больше 500 символов): ((5 ((142 67) (6 7))) (1 2))

Введены корректные данные.

Ход работы алгоритма:

левое плечо: коромысло:

левое плечо: гиря (+1).

правое плечо: гиря (+1).

правое плечо: гиря (+1).

Общее количество гирек: 3.

Схема:

**BinKor**

((5 ((142 67) (6 7))) (1 2))

**Правое плечо**

(1 2)

**Груз :** 2 +1

**Длина :** 1

**Левое плечо**

(5 ((142 67) (6 7)))

**Груз:**

**BinKor**

((142 67) (6 7))

**Правое плечо:**

(6 7)

**Груз:** 7 +1

**Длина:** 6

**Груз:** 67 +1

**Левое плечо:**

(142 67)

**Длина:** 5

**Длина:** 142

5

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| ((5 3) (1 2)) | Введены корректные данные.  Ход работы алгоритма:  левое плечо: гиря (+1).  правое плечо: гиря (+1).  Общее количество гирек: 2. |
| ((5 6) (7 8)) | Введены корректные данные.  Ход работы алгоритма:  левое плечо: гиря (+1).  правое плечо: гиря (+1).  Общее количество гирек: 2. |
| ((2 ((13 5) (8 1))) (4 5)) | Введены корректные данные.  Ход работы алгоритма:  левое плечо: коромысло:  левое плечо: гиря (+1).  правое плечо: гиря (+1).  правое плечо: гиря (+1).  Общее количество гирек: 3. |
| ((5 ((142 67) (6 7))) (1 2)) | Введены корректные данные.  Ход работы алгоритма:  левое плечо: коромысло:  левое плечо: гиря (+1).  правое плечо: гиря (+1).  правое плечо: гиря (+1).  Общее количество гирек: 3. |
| ((13 ((11 2) (2 ((1 1) (7 3))))) (19 ((3 2) (17 ((2 2) (4 6)))))) | Введены корректные данные.  Ход работы алгоритма:  левое плечо: коромысло:  левое плечо: гиря (+1).  правое плечо: коромысло:  левое плечо: гиря (+1).  правое плечо: гиря (+1).  правое плечо: коромысло:  левое плечо: гиря (+1).  правое плечо: коромысло:  левое плечо: гиря (+1).  правое плечо: гиря (+1).  Общее количество гирек: 6. |
| ((11adsa 5) (4 3)) | Ошибка! Вы ввели некорректные данные:  Символ №5 - 'a'.  Ожидался пробел.  Программа завершила работу. |
| ((dsa1 6) (4 3)) | Ошибка! Вы ввели некорректные данные:  Символ №3 - 'd'.  Ожидалось значение от 1 до 9.  Программа завершила работу. |
| ((1 6) (4 3)))))))))) | Ошибка! Вы ввели некорректные данные:  После символа №12 присутствуют лишние символы.  Программа завершила работу. |
|  | Ошибка! Вы ввели некорректные данные:  Символ №1 - '  '.  Ожидался символ - '('.  Программа завершила работу. |

**ПРИЛОЖЕНИЕ Б**

**Исходный код программы.**

*// Задание: посчитать число всех гирек заданного бинарного коромысла.*

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#include <iostream>

#define **N** 501

*// Бинарное коромысло*

typedef struct BinCor {

unsigned int length1;

unsigned int length2;

unsigned int weight1;

unsigned int weight2;

struct BinCor\* cor\_1;

struct BinCor\* cor\_2;

} BinCor;

*// вывод сообщений об ошибках*

void errorMassage(int error\_number, char\* str, int index) {

printf("\nОшибка! Вы ввели некорректные данные:\n");

switch (error\_number) {

case 1:

printf("Символ №%d - '%c'.\n", index + 1, str[index]);

printf("Ожидался символ - '('.\n");

break;

case 2:

printf("Символ №%d - '%c'.\n", index + 1, str[index]);

printf("Ожидался символ - ')'.\n");

break;

case 3:

printf("Символ №%d - '%c'.\n", index + 1, str[index]);

printf("Ожидалось значение от 1 до 9.\n");

break;

case 4:

printf("Символ №%d - '%c'.\n", index + 1, str[index]);

printf("Ожидался пробел.\n");

break;

case 5:

printf("Символ №%d - '%c'.\n", index + 1, str[index]);

printf("Отсутствует закрывающая скобка.\n");

break;

case 6:

printf("Символ №%d - '%c'.\n", index + 1, str[index]);

printf("Ожидалось значение от 1 до 9 или '('.\n");

break;

case 7:

printf("После символа №%d присутствуют лишние символы.\n", index + 1);

break;

}

}

BinCor\* initBinCorElement() { *// Инициализация элемента списка*

BinCor\* element = (BinCor\*)malloc(sizeof(BinCor));

element->length1 = 0;

element->length2 = 0;

element->weight1 = 0;

element->weight2 = 0;

element->cor\_1 = **NULL**;

element->cor\_2 = **NULL**;

return element;

}

*// считывание и создание бинарного коромысла*

*// функция возвращает 1, если возникла ошибка*

int readBinCorElement(char\* str, int index\_1, int index\_2, BinCor\*\* element) {

\*element = initBinCorElement();

if (str[index\_1++] != '(') {

errorMassage(1, str, index\_1 - 1);

return 1;

}

if (str[index\_2--] != ')') {

errorMassage(2, str, index\_2 + 1);

return 1;

}

*// считывание левого плеча*

if (str[index\_1++] != '(') { *// открывающая скобка левого плеча*

errorMassage(1, str, index\_1 - 1);

return 1;

}

*// первое число(длина)*

if (!isdigit(str[index\_1]) || str[index\_1] == '0') {

errorMassage(3, str, index\_1);

return 1;

}

while (1) { *// считывание числа*

if (isdigit(str[index\_1])) {

(\*element)->length1 = (\*element)->length1 \* 10 + str[index\_1] - '0';

index\_1++;

}

else {

break;

}

}

if (str[index\_1++] != ' ') { *// пробел после первого числа*

errorMassage(4, str, index\_1 - 1);

return 1;

}

if (isdigit(str[index\_1]) && str[index\_1] != 0) { *// случай, если гирька*

while (1) {

if (isdigit(str[index\_1])) {

(\*element)->weight1 = (\*element)->weight1 \* 10 + str[index\_1] - '0';

index\_1++;

}

else {

break;

}

}

}

else if (str[index\_1] == '(') { *// случай, если ещё одно коромысло*

int bracket\_count = 0;

int index;

*// поиск закрывающей скобки*

for (index = index\_1; index < index\_2; index++) {

if (str[index] == '(')

bracket\_count++;

if (str[index] == ')')

bracket\_count--;

if (bracket\_count == 0) {

if (readBinCorElement(str, index\_1, index, &((\*element)->cor\_1))) {

return 1;

}

index\_1 = index + 1;

break;

}

}

if (bracket\_count != 0) {

errorMassage(5, str, index\_1);

return 1;

}

}

else {

errorMassage(6, str, index\_1);

return 1;

}

if (str[index\_1++] != ')') { *// закрывающая скобка левого плеча*

errorMassage(2, str, index\_1 - 1);

return 1;

}

if (str[index\_1++] != ' ') { *// пробел между плечами*

errorMassage(4, str, index\_1 - 1);

return 1;

}

*// считывание правого плеча*

if (str[index\_1++] != '(') { *// открывающая скобка прового плеча*

errorMassage(1, str, index\_1 - 1);

return 1;

}

*// первое число(длина)*

if (!isdigit(str[index\_1]) || str[index\_1] == 0) {

errorMassage(3, str, index\_1);

return 1;

}

while (1) { *// считывание числа*

if (isdigit(str[index\_1])) {

(\*element)->length2 = (\*element)->length2 \* 10 + str[index\_1] - '0';

index\_1++;

}

else {

break;

}

}

if (str[index\_1++] != ' ') { *// пробел после первого числа*

errorMassage(4, str, index\_1 - 1);

return 1;

}

if (isdigit(str[index\_1]) && str[index\_1] != '0') { *// случай, если гирька*

while (1) {

if (isdigit(str[index\_1])) {

(\*element)->weight2 = (\*element)->weight2 \* 10 + str[index\_1] - '0';

index\_1++;

}

else {

break;

}

}

}

else if (str[index\_1] == '(') { *// случай, если ещё одно коромысло*

int bracket\_count = 0;

int index;

*// поиск закрывающей скобки*

for (index = index\_1; index < index\_2; index++) {

if (str[index] == '(')

bracket\_count++;

if (str[index] == ')')

bracket\_count--;

if (bracket\_count == 0) {

if (readBinCorElement(str, index\_1, index, &((\*element)->cor\_2))) {

return 1;

}

index\_1 = index + 1;

break;

}

}

if (bracket\_count != 0) {

errorMassage(5, str, index\_1);

return 1;

}

}

else {

errorMassage(6, str, index\_1);

return 1;

}

if (str[index\_1] != ')') { *// закрывающая скобка левого плеча*

errorMassage(2, str, index\_1);

return 1;

}

*// проверка на лишние символы*

if (index\_2 != index\_1) {

errorMassage(7, str, index\_1);

return 1;

}

return 0;

}

*// создание бинарного коромысла*

*// функция возвращает 1, если возникла ошибка*

int createBinCor(char\* str, BinCor\*\* binCor) {

int index\_1 = 0;

int index\_2 = strlen(str) - 2;

return readBinCorElement(str, index\_1, index\_2, binCor);

}

*// Возвращаемое значение равно количеству всех*

*// гирек в заданном бинарном коромысле.*

unsigned int numbers(const BinCor binCor, int deep\_of\_recursion) {

int result = 0;

for (int i = 0; i < deep\_of\_recursion; i++)

printf(" ");

printf("левое плечо: ");

if (binCor.cor\_1 == **NULL**) {

printf("гиря (+1).\n");

result++;

}

else {

printf("коромысло:\n");

result += numbers(\*(binCor.cor\_1), deep\_of\_recursion + 1);

}

for (int i = 0; i < deep\_of\_recursion; i++)

printf(" ");

printf("правое плечо: ");

if (binCor.cor\_2 == **NULL**) {

printf("гиря (+1).\n");

result++;

}

else {

printf("коромысло:\n");

result += numbers(\*(binCor.cor\_2), deep\_of\_recursion + 1);

}

return result;

}

*// очистка памяти*

void free\_memory(BinCor\* binCor) {

if (binCor != **NULL**) {

free(binCor->cor\_1);

free(binCor->cor\_2);

}

free(binCor);

}

int main(void)

{

setlocale(**LC\_ALL**, "Russian");

char str[**N**]; *// строка для ввода*

BinCor\* binCor = **NULL**; *// бинарное коромысло*

*// начало считывания и обработки данных*

printf("\nПрограмма выводит общее число гирек в указанном бинарном коромысле.\n");

printf("\nБинарное коромысло записывается в виде:\n");

printf("(ПЛЕЧО ПЛЕЧО)\n");

printf("Плечо имеет следующий вид:\n");

printf("(ДЛИНА ГРУЗ)\n");

printf("В качестве груза может выступать ещё одно бинарное коромысло или груз (число).\n");

printf("\nВведите бинарное коромысло (не больше 500 символов): ");

fgets(str, **N**, **stdin**);

*// обработка данных и проверка на ошибки*

if (createBinCor(str, &binCor)) {

printf("Программа завершила работу.\n\n");

free\_memory(binCor);

return 0;

}

printf("\nВведены корректные данные.\n\n");

*// конец считывания и обработки данных*

*// вывод результата*

printf("Ход работы алгоритма:\n\n");

printf("\nОбщее количество гирек: %u.\n\n", numbers(\*binCor, 1));

free\_memory(binCor);

return 0;

}