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: Иерархический список

Студент гр. 9381 _____ Авдеев И.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Ознакомиться с основными понятиями и приемами иерархических списков , получить навыки программирования процедур и функций иерархических списков.

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. 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.

Задание.

Варинат 1

1) Подсчитать общий вес заданного бинарного коромысла bk, т. е. суммарный вес его гирек. Для этого ввести рекурсивную функцию

unsigned int W (**const** БинКор bk).

Выполнение задания

1. Был создан тип данных, описывающий элемент списка коромысла.

Класс node

Поля класса node

Int num public - номер узла

Int weight public - вес узла

struct node* left public - указатель в левое плечо

struct node* right public - указатель в правое плечо

2. Для создания списка была создана функция push, создающая узел и соединяющая его к другим.

void push(int a, node t);**

3. Функции для вывода списком или деревом

void printCor(node* t);

void printTree(node* t, int u);

4. Функции инициализации из файла или из консоли

void console(node cor);**

void file(node cor);**

5. Функция удаления структуры

void deleteCor(node* cor);

6. Для подсчета общего веса была создана функция total и totalLog (без пошагового вывода и с ним соответственно).

int totalLog(node* cor);

unsigned int total(node* t);

Описание алгоритма

Сначала проводится проверка наличия элемента струтуры. Если ее нет, то функция возвращает 0.

Создается возвращаемое значение типа unsigned int равное весу текущего узла.

Далее проводится проверка на наличие левого плеча, и в случае когда оно есть запускается еще одна сумма возвращаемого значения и возвращаемого значения функции total с левым плечом. Аналогично с правым плечом.

После чего функция возвращает результат.

Выводы

В ходе выполнения работы был ознакомлен с принципом взаимодействия с иерархическим списком. Была реализована функция, которая рекурсивно подсчитывает вес иерархического списка.

Тестирование

Входные данные	Ожидаемый результат	Результат
7 6 4 8 3 5 7 9	42	Общий вес всех гирек 42
4 47 53 4 375	479	Общий вес всех гирек 479
6 55 43 78 22 44 99	341	Общий вес всех гирек 341

3 4 5 6	15	Общий вес всех гирек 15

Приложение 1

Содержание файла Source.cpp

```
#include <iostream>
#include <conio.h>
#include <fstream>
#include "BinTree.h"
using namespace std;

int fnum = 0;

int totalLog(node* cor)
{
   // проверка не пуст ли груз
////////////////////////////////////////////////////////////////////////////////////
   if (cor == NULL)
      return 0;
   unsigned int r = 0;
   cout << "Запуск функции номер " << ++fnum << " находится в узле " << cor->num
<< "\n";
   _getch();
   int numf = fnum;
   // если есть слева груз считает его
////////////////////////////////////////////////////////////////////////////////////
   if (cor->left)
   {
      cout << "Идет в левое плечо\n";
      _getch();
      r = r + totalLog(cor->left);
      cout << "Находится в узле " << cor->num;
      cout << " сумма =" << r << "\n";
      _getch();
   }
   // если есть справа груз считает его
////////////////////////////////////////////////////////////////////////////////////
   if (cor->right)
   {
      cout << "Идет в правое плечо\n";
      _getch();
      r = r + totalLog(cor->right);
```

```
        cout << "Находится в узле " << cor->num;
        cout << " сумма =" << r << "\n";
        _getch();
    }
    // считает результат ////////////////////////////////////////////////////////////////////////////////////////
    r = r + cor->weight;
    cout << "Возвращается назад сумма =" << r << "\n";
    _getch();
    cout << "Функция номер " << numf << " завершилась\n";
    _getch();
    return r;
}

int main()
{
    setlocale(LC_ALL, "ru");
    node* cor = NULL;
    // выбор консоли или файла ////////////////////////////////////////////////////////
    char k;
    cout << "Ввод в консоль на кнопку C, из файла любая другая\n\n";
    k = _getch();
    switch (k)
    {
    case 'c':
        console(&cor);
        break;
    default:
        file(&cor);
    }
    ////////////////////////////////////////////////////////////////////////////////////
    // Выбор вывода коромысла ////////////////////////////////////////////////////////
    cout << "Вывод строкой на кнопку C, списком на кнопку S, оба варианта на
B\n\n";
    k = _getch();
    switch (k)
    {
    // Вывод списком ////////////////////////////////////////////////////////////////////////////////////////
    case 'c':
        printCor(cor);
```

```
        cout << endl;
        break;
    // Вывод списком ////////////////////////////////////////////////////////////////////////////////////
    case 's':
        printTree(cor, 0);
        break;
    // Оба варианта сразу ///////////////////////////////////////////////////////////////////////////////
    case 'b':
        printCor(cor);
        cout << endl << endl;
        printTree(cor, 0);
        break;
    default:
        break;
    }
    // Вывод общего веса ////////////////////////////////////////////////////////////////////////////////
    cout << "Пошаговое выполнение на кнопу S, обычное на любую другую\n";
    k = _getch();
    int a = 0;
    switch (k)
    {
    case 's':
        a = totalLog(cor);
        cout << "Общий вес всех гирек\t" << a;
        break;
    default:
        cout << "Общий вес всех гирек\t" << total(cor);
        break;
    }
    deleteCor(cor);
    k = _getch();
    return 0;
}
```

Приложение 2

Содержание файла BinTree.cpp

```
#include <iostream>
#include <fstream>
#include "BinTree.h"
using namespace std;

int f = 0;

// Добавление элемента в список
////////////////////////////////////////////////////////////////////////////////
void push(int a, node** t)
{
   if ((*t) == NULL)
   {
      (*t) = new node;
      (*t)->num = ++f;
      (*t)->weight = a;
      (*t)->left = (*t)->right = NULL;
      return;
   }

   if (a > (*t)->weight) push(a, &(*t)->right);
   else push(a, &(*t)->left);
}
// Подсчет веса списка ////////////////////////////////////////////////////////////////////////////////
unsigned int total(node* t)
{
```

```
    // проверка не пуст ли груз
////////////////////////////////////////////////////////////////////////////////////
    if (t == NULL)
        return 0;

    unsigned int r = t->weight;
    // если есть слева груз считает его
////////////////////////////////////////////////////////////////////////////////////
    if (t->left)
        r = r + total(t->left);
    // если есть справа груз считает его
////////////////////////////////////////////////////////////////////////////////////
    if (t->right)
        r = r + total(t->right);
    // считает результат ////////////////////////////////////////////////////////////////////////////////////
    return r;
}
// Вывод строкой ////////////////////////////////////////////////////////////////////////////////////
void printCor(node* t)
{
    if (t == NULL) return;
    else
    {
        cout << "(";
        cout << t->weight;
        printCor(t->left);
    }
    printCor(t->right);
    cout << ")";
}
// Вывод списком ////////////////////////////////////////////////////////////////////////////////////
void printTree(node* t, int u)
{
    if (t == NULL) return;
    else
    {
        printTree(t->right, ++u);
        for (int i = 0; i < u - 1; ++i) cout << "\t";
        cout << t->num << "(" << t->weight << ")\n";
        u--;
    }
    printTree(t->left, ++u);
}
// Создание списка в зависимости от выбора файл/консоль
////////////////////////////////////////////////////////////////////////////////////
```

```
void console(node** cor)
{
    int n, s;
    cout << "введите количество элементов  ";
    cin >> n;
    for (int i = 0; i < n; ++i)
    {
        cout << "ведите число  ";
        cin >> s;

        push(s, &(*cor));
    }
}

void file(node** cor)
{
    int n;
    int s;
    ifstream lin("lin.txt");
    lin >> n;

    for (int i = 0; i < n; ++i)
    {
        lin >> s;

        push(s, &(*cor));
    }
    lin.close();
}
///// Удаление ////////////////////////////////////////////////////////////////
void deleteCor(node* cor)
{
    if (cor != NULL) {
        deleteCor(cor->left);
        deleteCor(cor->right);
        delete cor;
    }
}
////////////////////////////////////////////////////////////////////////////////
```