МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Дальневосточный федеральный университет»

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

Кафедра прикладной математики, механики, управления и программного обеспечения

ОТЧЕТ

по лабораторной работе

на тему «АВЛ дерево»

Выполнил студент гр. Б8217

Некрасов Евгений Андреевич

Проверила:

Крестникова О.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Владивосток

2019

1.Неформальная постановка задачи:

1.1 Создать АВЛ дерево.

Примечание: узлы АВЛ дерева заполняются целочисленными значениями

1.2 Реализовать методы:

Добавления элемента, удаления элемента, печать дерева:прямого; обратного; симметричного обходов, поиск элемента, вывод: максимального; минимального; элементов, отображение высоты дерева.

2. Формальная постановка задачи:

Область исходных данных:

X-множество входных данных:

struct node {

int element;

node \*left;

node \*right;

int height;

};

typedef struct node \*nodeptr;

class bsTree

{

public:

void insert (int, nodeptr &);

void del (int, nodeptr &);

int deleteMin (nodeptr &);

void find (int, nodeptr &);

nodeptr findMin (nodeptr);

nodeptr findMax (nodeptr);

void makeEmpty (nodeptr &);

void copy (nodeptr &, nodeptr &);

nodeptr nodecopy (nodeptr &);

void preOrder (nodeptr);

void inOrder (nodeptr);

void postOrder (nodeptr);

int bsHeight (nodeptr);

nodeptr srl (nodeptr &);

nodeptr drl (nodeptr &);

nodeptr srr (nodeptr &);

nodeptr drr (nodeptr &);

int max (int, int);

int noNodes (nodeptr);

};

// Вставка узла

void bsTree::insert (int x, nodeptr &p)

{

if (p == NULL) {

p = new node;

p -> element = x;

p -> left = NULL;

p -> right = NULL;

p -> height = 0;

if (p == NULL) {

std::cout << "Садим корень" << std::endl;

}

}

else {

if (x < p -> element) {

insert (x, p -> left);

if ((bsHeight (p -> left) - bsHeight (p -> right)) == 2) {

if (x < p -> left -> element) {

p = srl (p);

}

else {

p = drl (p);

}

}

}

else if (x > p -> element) {

insert (x, p -> right);

if ((bsHeight (p -> right) - bsHeight (p -> left)) == 2) {

if (x > p -> right -> element) {

p = srr (p);

}

else {

p = drr (p);

}

}

}

else {

std::cout << "Элемент существует" << std::endl;

}

}

int m = bsHeight (p -> left);

int n = bsHeight (p -> right);

int d = max (m, n);

p -> height = d + 1;

}

// Поиск наименьшего элемента

nodeptr bsTree::findMin (nodeptr p) {

if (p == NULL) {

std::cout << "Пустое дерево" << std::endl;

return p;

}

else {

while (p -> left != NULL) {

p = p -> left;

//return p;

}

return p;

}

}

// Поиск наибольшего узла

nodeptr bsTree::findMax (nodeptr p) {

if (p == NULL) {

std::cout << "Дерево пустое" << std::endl;

return p;

}

else {

while (p -> right != NULL) {

p = p -> right;

//return p;

}

return p;

}

}

// Поиск элемента

void bsTree::find (int x, nodeptr &p) {

if (p == NULL) {

std::cout << "Искомого элемента не существует" << std::endl;

}

else {

if (x < p -> element) {

find (x, p -> left);

}

else {

if (x > p -> element) {

find (x, p -> right);

}

else {

std::cout << "Искомый элемент найден" << std::endl;

}

}

}

}

// Копирование дерева

void bsTree::copy (nodeptr &p, nodeptr &p1) {

makeEmpty (p1);

p1 = nodecopy (p);

}

// Создать пустое дерево

void bsTree::makeEmpty (nodeptr &p) {

nodeptr d;

if (p != NULL) {

makeEmpty (p -> left);

makeEmpty (p -> right);

d = p;

free (d);

p = NULL;

}

}

// Копирование узлов дерева

nodeptr bsTree::nodecopy (nodeptr &p) {

nodeptr temp;

if (p == NULL) {

return p;

}

else {

temp = new node;

temp -> element = p -> element;

temp -> left = nodecopy (p -> left);

temp -> right = nodecopy (p -> right);

return temp;

}

}

// Удаление узла

void bsTree::del (int x, nodeptr &p) {

nodeptr d;

if (p == NULL) {

std::cout << "Элемента для удаления не существует" << std::endl;

}

else if (x < p -> element) {

del (x, p -> left);

}

else if (x > p -> element) {

del (x, p -> right);

}

else if ((p -> left == NULL) && (p -> right == NULL)) {

d = p;

free (d);

p = NULL;

std::cout << "Элемент удален" << std::endl;

}

else if (p -> left == NULL) {

d = p;

free (d);

p = p -> right;

std::cout << "Элемент удален" << std::endl;

}

else if (p -> right == NULL) {

d = p;

p = p -> left;

free (d);

std::cout << "Элемент удален" << std::endl;

}

else {

p -> element = deleteMin (p -> right);

}

}

//Удаление минимального элемента

int bsTree::deleteMin (nodeptr &p) {

int c;

std::cout << "Удаление минимального элемента" << std::endl;

if (p -> left == NULL) {

c = p -> element;

p = p -> right;

return c;

}

else {

c = deleteMin (p -> left);

return c;

}

}

//Прямой обход

void bsTree::preOrder (nodeptr p) {

if (p != NULL) {

std::cout << p -> element << std::endl;

preOrder (p -> left);

preOrder (p -> right);

}

}

// Симметричный обход

void bsTree::inOrder (nodeptr p) {

if (p != NULL) {

inOrder (p -> left);

std::cout << p -> element << std::endl;

inOrder (p -> right);

}

}

// Обратный обход

void bsTree::postOrder (nodeptr p) {

if (p != NULL) {

postOrder (p -> left);

postOrder (p -> right);

std::cout << p -> element << std::endl;

}

}

// Расчёт разницы высот

int bsTree::max (int value1, int value2) {

return ((value1 > value2) ? value1 : value2);

}

//Расчёт высоты

int bsTree::bsHeight (nodeptr p) {

int t;

if (p == NULL) {

return -1;

}

else {

t = p -> height;

return t;

}

}

//Поворот право-лево

nodeptr bsTree::srl (nodeptr &p1) {

nodeptr p2;

p2 = p1 -> left;

p1 -> left = p2 -> right;

p2 -> right = p1;

p1 -> height = max (bsHeight (p1 -> left), bsHeight (p1 -> right)) + 1;

p2 -> height = max (bsHeight (p2 -> left), p1 -> height) + 1;

return p2;

}

//Поворот право-право

nodeptr bsTree::srr (nodeptr &p1) {

nodeptr p2;

p2 = p1 -> right;

p1 -> right = p2 -> left;

p2 -> left = p1;

p1 -> height = max (bsHeight (p1 -> left), bsHeight (p1 -> right)) + 1;

p2 -> height = max (p1 -> height, bsHeight (p2 -> right)) + 1;

return p2;

}

// Поворот лепо-право

nodeptr bsTree::drl (nodeptr &p1) {

p1 -> left = srr (p1 -> left);

return srl (p1);

}

//Поворот лево-лево

nodeptr bsTree::drr (nodeptr &p1) {

p1 -> right = srl (p1 -> right);

return srr (p1);

}

// Проверка отсутствующих узлов

int bsTree::noNodes(nodeptr p) {

int count = 0;

if (p != NULL) {

noNodes (p -> left);

noNodes (p -> right);

count++;

}

return count;

}

3. Спецификация данных:

INPUT:

Данные считываются из потока ввода std::in;.

Формат входного значения: int

OUTPUT

Данные отправляются в поток вывода std::out

Формат выходного значения: int

4. Тесты

(Вывод из консоли перенаправлен в файл командой "avl | tee conclusion.txt")

АВЛ Дерево

:

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Добавление нового узла

Введите элемент:

Новый элемент добавлен

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Добавление нового узла

Введите элемент:

Новый элемент добавлен

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Добавление нового узла

Введите элемент:

Новый элемент добавлен

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Добавление нового узла

Введите элемент:

Новый элемент добавлен

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Добавление нового узла

Введите элемент:

Новый элемент добавлен

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Добавление нового узла

Введите элемент:

Новый элемент добавлен

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Минимальный элемент в дереве: 0

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Максимальный элемент в дереве: 9

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Введите искомый элемент: Искомый элемент найден

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Введите искомый элемент: Искомого элемента не существует

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Какой узел удалять? : Элемент удален

0

1

2

5

6

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Вариант прямого обхода

2

1

0

6

5

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Вариант симметричного обхода

0

1

2

5

6

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Вариант обратного обхода

0

1

5

6

2

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Высота дерева

TДерево имеет высоту: 2

:1 Вставить новый узел

:2 Найти минимальный элемент

:3 Найти максимальный элемент

:4 Поиск по значению

:5 Удалить элемент

:6 Вариант прямого обхода

:7 Вариант симметричного обхода

:8 Вариант обратного обхода

:9 Показать высоту дерева

:0 Выход

Выберите нужное действие и нажмите Enter

Выход