Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Алгоритми та структури даних

ЗВІТ ДО

КОМП’ЮТЕРНОГО ПРАКТИКУМУ №6

“Дерева”

Варіант № 1

Оцінка «\_\_\_\_\_\_\_\_\_»

Дата «\_\_\_» \_\_\_\_\_\_\_ 20\_\_\_

Виконав: студент 2 курсу

гр. ТВ-61

Артамонов Олексій Юрійович

Перевірила: Васильєва О.Б.

Київ – 2017

**Завдання:**

* Побудувати ідеально збалансоване дерево;
* З того ж набору ключів побудувати дерево пошуку;
* Мати можливість вилучати всі вузли;
* З того ж набору побудувати АВЛ дерево;
* Мати можливість вилучати всі вузли;
* Для заданої послідовності цілих чисел побудувати дерево пошуку та пронумерувати його вершини відповідно до їх порядку при інфіксному обході дерева.

**Лістинг програми:**

#include "stdafx.h"

#include <iostream>

#include <iomanip>

#include <conio.h>

#include <time.h>

using namespace std;

typedef int T; // тип элемента

#define compLT(a,b) (a < b)

#define compEQ(a,b) (a == b)

struct Node{

T data; // значение узла

Node \*left, \*right, \*parent;

};

Node \*root = NULL; //корень бинарного дерева поиска

Node\* Make\_BTree(int a[], int &from, int num);

Node\* Insert\_Node(T data);

void Delete\_Node(Node \*z);

Node\* Find\_Node(T data);

int Inorder(Node \*node, int f);

void Print\_Tree(Node \*node, int l = 0);

int main() {

int i, \*a, num, from = 0;

cout << "Enter count of elements: ";

cin >> num;

cout << endl;

a = new int[num];

srand(time(NULL));

for (i = 0; i < num; i++)

a[i] = rand() % 100 + 1;

cout << "All elements: ";

for (i = 0; i < num; i++)

cout << a[i] << " ";

cout << endl << endl;

Node\* Tree = Make\_BTree(a, from, num); // создание идеально сбалансированного дерева

cout << "Perfectly balanced tree" << endl;

Print\_Tree(Tree, 0);

// добавление элементов в бинарное дерево поиска

for (i = 0; i < num; i++) {

Insert\_Node(a[i]);

}

cout << "Binary search tree" << endl;

Print\_Tree(root);

cout << endl;

cout << "Order of vertices in an infix bypass" << endl;

static int f = 0;

Inorder(root, f);

Print\_Tree(root);

// поиск элементов по бинарному дереву поиска

for (i = num - 1; i >= 0; i--) {

Find\_Node(a[i]);

}

// очистка бинарного дерева поиска

for (i = 0; i < num; i++) {

Delete\_Node(Find\_Node(a[i]));

}

\_getch();

return 0;

}

Node\* Make\_BTree(int a[], int &from, int num) {

Node\* Tree;

int n1, n2;

if (num == 0) return NULL;

Tree = new Node;

Tree->data = a[from++];

n1 = num / 2;

n2 = num - n1 - 1;

Tree->left = Make\_BTree(a, from, n1);

Tree->right = Make\_BTree(a, from, n2);

return Tree;

}

//функция выделения памяти для нового узла и вставка в дерево

Node\* Insert\_Node(T data) {

Node \*x, \*current, \*parent;

current = root;

parent = 0;

while (current) {

if (data == current->data) return (current);

parent = current;

current = data < current->data ?

current->left : current->right;

}

x = new Node;

x->data = data;

x->parent = parent;

x->left = NULL;

x->right = NULL;

if (parent)

if (x->data < parent->data)

parent->left = x;

else

parent->right = x;

else

root = x;

return(x);

}

//функция удаления узла из дерева

void Delete\_Node(Node \*d) {

Node \*x, \*y;

if (!d || d == NULL) return;

if (d->left == NULL || d->right == NULL)

y = d;

else {

y = d->right;

while (y->left != NULL) y = y->left;

}

if (y->left != NULL)

x = y->left;

else

x = y->right;

if (x) x->parent = y->parent;

if (y->parent)

if (y == y->parent->left)

y->parent->left = x;

else

y->parent->right = x;

else

root = x;

if (y != d) {

y->left = d->left;

if (y->left) y->left->parent = y;

y->right = d->right;

if (y->right) y->right->parent = y;

y->parent = d->parent;

if (d->parent)

if (d == d->parent->left)

d->parent->left = y;

else

d->parent->right = y;

else

root = y;

free(d);

}

else {

free(y);

}

}

//функция поиска узла, содержащего data

Node\* Find\_Node(T data) {

Node \*current = root;

while (current != NULL)

if (compEQ(data, current->data))

return (current);

else

current = compLT(data, current->data) ?

current->left : current->right;

return(0);

}

int Inorder(Node \*node, int f)

{

if (node) {

f = Inorder(node->left, f) + 1;

node->data = f;

f = Inorder(node->right, f);

}

return f;

}

//функция вывода бинарного дерева поиска

void Print\_Tree(Node \*node, int l) {

int i;

if (node != NULL) {

Print\_Tree(node->right, l + 1);

for (i = 0; i < l; i++) cout << "\t";

cout << setw(4) << node->data;

Print\_Tree(node->left, l + 1);

}

else cout << endl;

}

**Результати:**



**Контрольні питання:**

Чим дерево відрізняється від лінійних списків?

Дерева мають іншу структуру і принцип побудови. Дерева не можуть бути циклічними (в той час як списки можуть). Списки це приклад лінійного упорядкування, а дерева – двумірного.

Дати означення листка та кореня дерева.

Кореневої вузол - найвищий вузол дерева. Корінь - одна з вершин, за бажанням спостерігача. Лист — вузол, який не має дітей.

У чому полягає особливість бінарних дерев?

Кожен вузол бінарного дерева має не більше аніж два нащадки.

Як включаються вузли у дерево?

Існує велика кількість видів дерев, і для багатьох з них існує власний спосіб включення елементу. В лабораторній роботі наявні методи для включення елементів для ідеально-збалансованого дерева, дерева пошуку та АВЛ дерево.

Як видаляються вузли в дереві?

Аналогічно, для кожного типу дерева існує свій спосіб. В лабораторній роботі наявні методи для видалення елементів для ідеально-збалансованого дерева, дерева пошуку та АВЛ дерево.