Міністерство освіти і науки України
Черкаський державний технологічний університет
Кафедра програмного забезпечення автоматизованих систем

Звіт

Про виконання лабораторної роботи №3
з дисципліни “Проектний практикум”

Перевірив:
Асистент Кафедри ПЗАС
Півень О. Б.

Виконав:
Студент 3-го курсу
Групи ПЗС-1644
Близнюк А. О.

Черкаси 2017

Лабораторна робота № 3

Тема роботи: Бінарні дерева.

Мета роботи: Набути навиків роботи з бінарними деревами.

Завдання(Варіант 2):

Написати функцію, яка знаходить найбільший елемент дерева

Теоретичні відомості

Двійкове (або Бінарне) де́рево по́шуку (англ. binary search tree, BST) в інформатиці — двійкове дерево, в якому кожній вершині x зіставлене певне значення val[x].

Бінарні дерева пошуку набагато ефективніші в операціях пошуку, аніж лінійні структури, в яких витрати часу на пошук пропорційні O(n), де n — розмір масиву даних, тоді як в повному бінарному дереві цей час пропорційний в середньому O(log2n) або O(h), де h — висота дерева (хоча гарантувати, що h не перевищує log2n можна лише для збалансованих дерев, які є ефективнішими в алгоритмах пошуку, аніж прості бінарні дерева пошуку).

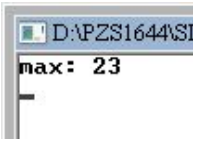
Найпоширенішою операцією, яка виконується з бінарним деревом пошуку, є пошук в ньому певного ключа. Крім того, бінарні дерева пошуку підтримують такі запити, як пошук мінімального і максимального елемента, а також попереднього і наступного.

Процедура пошуку починається з кореня дерева і проходить вниз по дереву. Для кожного вузла x на шляху вниз його ключ key[x] порівнюється з переданим як параметр ключем k. Якщо ключі однакові, пошук завершується. Якщо k менше key[x], пошук триває в лівому піддереві x; якщо більше — то пошук переходить в праве піддерево. Ту ж процедуру можна записати ітеративно, «розгортаючи» рекурсію в цикл while.

Лістинг програми

```
#include <iostream.h>
#include <conio.h>

struct treenode{
    int val;
    treenode* left;
    treenode* right;
};

//добавление элемента
void add(treenode*& p, int val){
    if(p == NULL){
        p = new treenode();
        if(p != NULL){
            p->left = p->right = NULL;
            p->val  = val;
        }
        return;
    }
    if(val < p->val)
        add(p->left, val);
    else
        add(p->right, val);
}

//удаление дерева
void treenode_clear(treenode* p){
    if(p != NULL){
        if(p->left != NULL)
            treenode_clear(p->left);
        if(p->right != NULL)
            treenode_clear(p->right);
        delete p;
    }
}

//максимальный элемент
int treenode_max(const treenode* p){
    if(p == NULL)
        return 0;
    while(p->right != NULL)
        p = p->right;
    return p->val;
}

int main(void){
    treenode* t = NULL;

    int A[] = { 5, 4, 6, 9, 23, 3, 8, 6, 1 };
    for(unsigned i = 0; i < sizeof(A)/sizeof(A[0]); ++i)
        add(t, A[i]);

    cout << "max: " << treenode_max(t) << std::endl;
    treenode_clear(t);
    getch();
    return 0;
                                        }
```

Результат роботи



Висновок

На даній лабораторній роботі набув навики роботи з бінарними деревами. Набув навичок реалізації пошуку елементу та визначення глибини елемента в бінарному дереві. Програма містить меню вибору.