**Курс «Объектно-ориентированное программирование на C++»**

**Встреча №17**

Тема: Бинарное дерево

Домашнее задание

**Задание №1.** Создать класс для работы с бинарным деревом, содержащим англо-русский словарь. Каждый узел содержит счетчик обращений к слову в данном узле. В процессе эксплуатации словаря при каждом вызове слова в счетчик обращений добавляется единица.

Написать программу, которая:

a) обеспечивает начальный ввод словаря с конкретными значениями счетчика обращений;

b) позволяет отобразить слово и его перевод

с) позволяет добавить,заменить,удалить перевод слова

d) отображает топ-3 самых популярных слов (определяем популярность на основании счетчика обращений)

e) отображает топ-3 самых непопулярных слов (определяем непопулярность на основании счетчика обращений)

#include<iostream>

#include<windows.h>

#include<string.h>

using namespace std;

struct node

{

char en[50];

char ukr[50];

int count;

node\* left, \* right, \* parent;

};

class tree

{

node\* root;

public:

tree()

{

root = NULL;

}

node\* getroot()

{

return root;

}

void add(char\* en, char\* ukr,int k)

{

if (root == NULL)

{

root = new node;

strcpy(root->en, en);

strcpy(root->ukr,ukr);

root->count = k;

root->left = NULL;

root->right = NULL;

root->parent = NULL;

}

else

{

node\* temp = NULL;

node\* y = root;

while (y != NULL)

{

temp = y;

if (strcmp(en , temp->en) == 1)

{

y = y->right;

}

else y = y->left;

}

node\* temp1 = new node;

strcpy(temp->en , en);

strcpy(temp->ukr , ukr);

temp->count = k;

temp1->left = NULL;

temp1->right = NULL;

if (strcmp(en , temp->en) == 1)

temp->right = temp1;

else temp->left = temp1;

}

}

node\* minn(node \*a)

{

while (a->left != NULL)

{

a = a->left;

}

return a;

}

node\* maxx(node\* a)

{

while (a->right != NULL)

{

a = a->right;

}

return a;

}

void print(node \* y)

{

if (y != NULL)

{

print(y->left);

cout << y->en << " " << y->ukr << " " << y->count << endl;

print(y->right);

}

}

node\* search(node \* a, char\* k)

{

node\* temp = a;

while (strcmp(temp->en, k) != 0 || temp == NULL)

{

if (strcmp(k, temp->en) == 1)

{

temp = temp->right;

}

else temp = temp->left;

}

temp->count += 1;

return temp;

}

node\* next(node \* y)

{

node\* temp = NULL;

if (y != NULL)

{

if (y->right != NULL)

return minn(y->right);

temp = y->parent;

while (temp != NULL && y == temp->right)

{

y = temp;

temp = temp->parent;

}

}

}

node\* previous(node \* y)

{

node\* temp = 0;

if (y != 0)

{

if (y->left != 0)

return maxx(y->left);

temp = y->parent;

while (temp != 0 && y == temp->left)

{

y = temp;

temp = temp->parent;

}

}

return temp;

}

void del(node\* z = NULL)

{

if (z != NULL)

{

node\* temp, \* y;

if (z->left == 0 || z->right == 0)

y = z;

else

y = next(z);

if (y->left != 0)

temp = y->left;

else

temp = y->right;

if (temp != 0)

temp->parent = y->parent;

if (y->parent == 0)

root = temp;

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

y->parent->left = temp;

else

y->parent->right = temp;

if (y != 0)

{

strcpy(z->en , y->en);

strcpy(z->ukr , y->ukr);

z->count = y->count;

}

delete y;

}

else

{

while (root != 0)

del(root);

}

}

};

void main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

tree a;

a.add((char\*)"Hello", (char\*)"Привет", 0);

a.add((char\*)"Pig", (char\*)"Свинья", 0);

a.add((char\*)"Table", (char\*)"Стол", 0);

a.add((char\*)"Phone", (char\*)"Телефон", 0);

a.add((char\*)"Lesson", (char\*)"Урок", 0);

a.add((char\*)"Money", (char\*)"Деньги", 0);

a.add((char\*)"Pillow", (char\*)"Подушка", 0);

a.add((char\*)"School", (char\*)"Школа", 0);

a.add((char\*)"Mouse", (char\*)"Мишь", 0);

a.add((char\*)"Sofa", (char\*)"Диван", 0);

node\* root = a.getroot();

a.print(root);

node \* res = a.search(root,(char\*)"Sofa");

cout << res->en << " " << res->ukr << " " << res->count << endl;

node\* del = a.search(root, (char\*)"Table");

a.del(del);

}