[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КПІ»

Кафедра

Автоматизованих систем обробки інформації та управління

**[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[[**

Звіт №7  
з лабораторних робіт з дисципліни  
«**Алгоритми та структури даних**»

|  |  |  |
| --- | --- | --- |
| Перевірив:доц. Ковалюк Т.В. |  | Виконав: Глянько А.С. |
| ас. Якименко С.В. |  | Студент гр. ІС-42, ФІОТ |
|  |  | 1 курс,  залікова книжка № IC-4209 |

Київ 2015

Звіт з лабораторної роботи №7

Тема: «Динамічні структури типу дерева»

1.Мета роботи:

1. Вивчити особливості використання динамічних структур типу дерева.
2. Навчитися застосовувати динамічні структури типу дерева в практичних задач.

2.Умова задачі

Побудувати два бінарних дерева, зчитавши значення їх вузлів з файла. Порівняти побудовані дерева. Якщо дерева однакові, визначити їх кількість рівнів та вузлів, інакше зробити копію одного з дерев за вибором користувача.

**Блок-схеми**

Рис.1.getTree(). Рис.2.Find(). Рис.3.Copy().



Рис.4. Main().

**Лістинг програми**

/\*Lab#7 Glyanko A.S. IS-4209 \*/

#include <iostream>

#include <iomanip>

#include <conio.h>

#include <windows.h>

using namespace std;

struct TreeNode

{

TreeNode(int v)

{

value = v;

leftPtr = rightPtr = 0;

}

int value;

TreeNode \* leftPtr;

TreeNode \* rightPtr;

};

struct Tree

{

public:

Tree()

{

rootPtr = 0;

size\_ = 0;

level\_ = 0;

}

int size(){ return size\_;};

int level(){ return level\_;};

void add(int v);

void addHelper(TreeNode \* &, int v);

void print();

void Copy(Tree \*, TreeNode \*);

TreeNode \* rootPtr;

void printHelper(TreeNode \*,int);

void countLevels(TreeNode \* current, int k);

private:

int size\_ ;

int level\_;

};

void Tree::add(int v)

{

addHelper(rootPtr, v);

size\_++;

}

void Tree::print()

{

printHelper(rootPtr, 0);

}

void Tree::addHelper(TreeNode \* & current, int v)

{

if (current == 0)

{

TreeNode \* temp = new TreeNode(v);

current = temp;

}

else

{

if (current->value < v)

addHelper(current->rightPtr, v);

else

addHelper(current->leftPtr, v);

}

}

void Tree::printHelper(TreeNode \* current, int width )

{

if (current != 0)

{

printHelper(current->rightPtr, width + 4);

cout << setw(width) << current->value << endl;

printHelper(current->leftPtr, width + 4);

}

}

void Tree::countLevels(TreeNode \* current, int k)

{

if (current != 0)

{

countLevels(current->rightPtr, k + 1);

countLevels(current->leftPtr, k + 1);

if(k>level\_)

level\_=k;

}

}

Tree getTree(char \*filename)

{

Tree NewTree;

int elem;

char str[30];

FILE \*f = fopen(filename, "r");

while (!feof(f))

{

fgets(str,100,f); //зчитати рядок

char \*p=strtok(str,"\n");

elem = atoi(p);

NewTree.add(elem);

p = strtok(NULL,"\n");

}

return NewTree;

}

int i = 0;

void Find(TreeNode \*current,int \*mas)

{

if(current != 0)

{

mas[i] = current->value;

i++;

Find(current->rightPtr,mas);

Find(current->leftPtr,mas);

}

}

void Tree :: Copy(Tree \*T2,TreeNode \*current)

{

if (current != 0)

{

T2->add(current->value);

Copy(T2,current->rightPtr);

Copy(T2, current->leftPtr);

}

}

void main()

{

cout << "Lab#7. Glyanko A.S., IS-4209" << endl;

char name1[20],name2[20];

int mas1[20],mas2[20];

bool flag;

Tree NewTree1,NewTree2;

cout << "Enter the name of file with first tree:" << endl;

cin >> name1;

NewTree1 = getTree(name1);

NewTree1.print();

cout << "Enter the name of file with second tree:" << endl;

cin >> name2;

NewTree2 = getTree(name2);

NewTree2.print();

for (int k = 0; k<20; k++)

mas1[k]=mas2[k]=NULL;

i = 0;

Find(NewTree1.rootPtr,mas1);

i=0;

Find(NewTree2.rootPtr,mas2);

for (int j=0; j<20; j++)

{

if (mas1[j] != NULL && mas2[j] != NULL)

{

if (mas1[j] == mas2[j]) continue;

else

{

cout << "Trees are not equal" << endl;

flag = false;

break;

}

}

else

{

if (mas1[j]==NULL && mas2[j] == NULL)

{

cout << "Trees are equal"<< endl;

cout<<"size1 = "<<NewTree1.size()<<endl;

cout<<"size2 = "<<NewTree2.size()<<endl;

NewTree1.countLevels(NewTree1.rootPtr,0);

NewTree2.countLevels(NewTree2.rootPtr,0);

cout<< "Level1 = "<< NewTree1.level()<<endl;

cout<< "Level2 = "<< NewTree2.level()<<endl;

}

else

{

cout << "Trees are not equal"<< endl;

flag = false;

}

break;

}

}

char answer;

Tree T3;

if (!flag)

{

cout<<"What tree do you want to copy?(press '1' or '2')"<<endl;

cin >> answer;

if (answer == '1')

{

NewTree1.Copy(&T3,NewTree1.rootPtr);

cout<<"Copy of the first tree:"<<endl;

T3.print();

}

else

if (answer == '2')

{

NewTree2.Copy(&T3,NewTree2.rootPtr);

cout<<"Copy of the first tree:"<<endl;

T3.print();

}

else cout << "Error.Sorry :("<<endl;

}

system("pause");

}

**Screenshots**

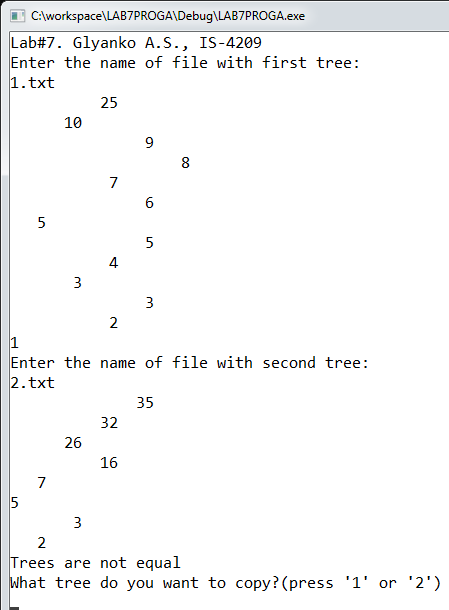


Рис. 5. Результат роботи програми при різних деревах.

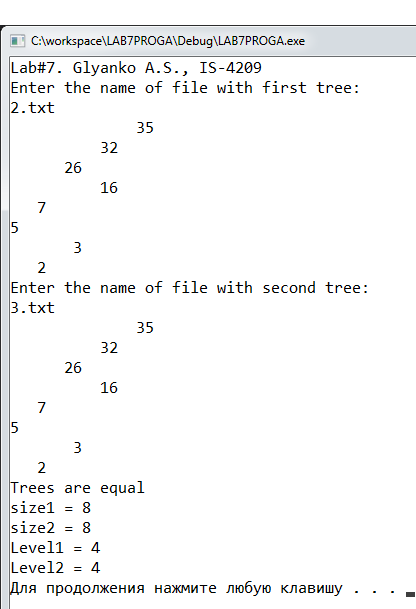


Рис. 6. Результат роботи програми при однакових деревах.

**Висновок**

В процесі виконання лабораторної роботи я вивчив особливості використання динамічних структур типу дерева та роботи з ними (додавання елементів, виведення дерева, копіювання дерева). За результатами роботи програми можна вважати, що програма працює вірно.