Національний технічний університет України

«Київський політехнічний інститут»

Кафедра автоматизованих систем обробки інформації та управління

Звіт про виконання лабораторної роботи №7 з дисципліни

«Алгоритми та структури даних»

Перевірено: Ковалюк Т.В. Виконав ст. 1 курсу ФОІТ

Проскура С.Л. гр. ІС-52

Дорошенко А.В.

Київ 2016

## Лабораторна робота 7

# **Динамічні структури типу дерева**

Варіант № 9

*Мета:*

1.Вивчити особливості використання  динамічних структур типу дерев

2.Навчитися застосовувати структури типу дерева в практичних задачах.

### Завдання

|  |  |
| --- | --- |
| 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-схема алгоритму



Рис.1 (Блок-схема алгоритму створення дерева)



Рис.2 (Блок-схема головного алгоритму)



Рис.3 (Блок-схема алгоритму зчитування з файла)



Рис.4 (Блок-схема алгоритму обходу дерева та збереження значень його вузлів)



Рис.5 (Блок-схема алгоритму порівняння дерев)



Рис.6 (Блок-схема алгоритму визначення кількості рівнів дерева)



Рис.7 (Блок-схема алгоритму виведення дерева)



Рис.8 (Блок-схема алгоритму запам’ятовування вузлів дерева в порядку їх знаходження у рівнях)

### Код програми

//Laboratory work

//written by student of the first curse

//of the group IC-52

//Anton Doroshenko

//2016.05.04

//==========================================================================

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

using namespace std;

struct tree

{

char data;

tree \*left,

\*right;

};

//показчики на корені дерев

tree \*root1 = NULL;

tree \*root2 = NULL;

tree \*root3 = NULL;

int numLayer = 0;//номер рівня дерева

int n1 = 0;//кількість вузлів 1 дерева

int n2 = 0;//кількість вузлів другого дерева

int \*ptrn1 = &n1;

int \*ptrn2 = &n2;

int N = 0;//допоміжна змінна

FILE \*f1, \*f2;

char fName1[10] = "f1.txt";

char fName2[10] = "f2.txt";

char fdata[100];//масив для зберігання інформації з файла

char str1[100];

char str2[100];

char strForPrint[100];

//-------Функція зчитування з файла---------------

void readFile(FILE\* f, int\* num, char name[10])

{

char c;

f = fopen(name, "r");

while (int(c = fgetc(f)) != EOF)

{

fputc(c, stdout);

fdata[\*num] = c;

(\*num)++;

}

cout << endl;

fclose(f);

}

//-------Функція створення дерева----------------

void createTree(tree \*\*ptr, int numNodes)

{

int numLeft = 0;

int numRight = 0;

if (numNodes == 0)

{

\*ptr = NULL;

}

else

{

numLeft = numNodes / 2;

numRight = (numNodes - numLeft) - 1;

\*ptr = new tree;

(\*ptr)->data = fdata[N];

fdata[N] = NULL;

N++;

(\*ptr)->left = NULL;

(\*ptr)->right = NULL;

createTree(&((\*ptr)->left), numLeft);

createTree(&((\*ptr)->right), numRight);

}

}

//-------Функція обходу дерева і збереження значень його вузлів----------

void search(tree\* ptr, char\* str)

{

if (ptr != NULL)

{

str[N] = ptr->data;

N++;

search(ptr->left, str);

search(ptr->right, str);

}

}

//-------Функція порівняння дерев-------------------

bool compare(tree\* ptr1, tree\* ptr2)

{

if (n1 != n2)

{

return false;

}

else

{

search(ptr1, str1);

N = 0;

search(ptr2, str2);

N = 0;

for (int i = 0; i <= n1; i++)

{

if (str1[i] != str2[i])

{

return false;

}

}

}

return true;

}

//------Функція визначення рівня дерева-----------

int layer(int number)

{

numLayer++;

number = number / 2;

if (number == 0)

{

return numLayer;

}

else

{

layer(number);

}

}

//---Функція внесення у масив значення коренів лівого і правого піддерева поточного вузла---

void LeftRightIn(tree\*\* ptr, int num)

{

if ((\*ptr)->left != NULL)

strForPrint[N] = (\*ptr)->left->data;

N++;

if ((\*ptr)->right != NULL)

{

strForPrint[N] = (\*ptr)->right->data;

N++;

}

}

//---Функція запам'ятовування вузлів дерева в порядку знаходження їх у рівнях---

void remNodesLev(tree\*\* ptr, int num)

{

if (num == numLayer)

{

strForPrint[N] = (\*ptr)->data;

N++;

}

num--;

if (num != 0 && ptr != 0)

{

LeftRightIn(ptr, num);

remNodesLev(&((\*ptr)->left), num);

remNodesLev(&((\*ptr)->right), num);

}

}

//-------Функція виведення дерева--------

void OutTree(int num)

{

int numNodeInLayer = 1;//кількість вузлів у поточному рівні

bool flag = true;

int index = 0;

while (flag)

{

for (int i = 0; i < numNodeInLayer; i++)

{

if (strForPrint[index] == 0)

{

flag = false;

cout << endl;

break;

}

else

{

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

{

cout << " ";

}

cout << strForPrint[index];

strForPrint[index] = 0;

index++;

}

}

cout << endl;

numNodeInLayer = numNodeInLayer \* 2;

num--;

}

}

//-----Головна функція-------

int main()

{

cout << "laboratory work number 7 made by Anton Doroshenko, IS-52" << endl;

cout << endl;

char key;

readFile(f1, ptrn1, fName1);

createTree(&root1, n1);

N = 0;

remNodesLev(&root1, layer(n1));

numLayer = 0;

N = 0;

cout << "First tree: " << endl;

OutTree(layer(n1));

numLayer = 0;

readFile(f2, ptrn2, fName2);

createTree(&root2, n2);

N = 0;

remNodesLev(&root2, layer(n2));

numLayer = 0;

N = 0;

cout << "Second tree: " << endl;

OutTree(layer(n2));

numLayer = 0;

if (compare(root1, root2))

{

cout << "Trees are simple" << endl;

cout << "Number of nodes is " << n1 << endl;

cout << "Max layer is " << layer(n1) << endl;

numLayer = 0;

system("pause");

}

else

{

cout << "Trees are different" << endl;

cout << "1. Copy first tree" << endl;

cout << "2. Copy second tree" << endl;

cout << "Enter your choose: ";

cin >> key;

switch (key)

{

case '1':{

search(root1, fdata);

N = 0;

createTree(&root3, n1);

N = 0;

remNodesLev(&root3, layer(n1));

numLayer = 0;

N = 0;

cout << "New tree: " << endl;

OutTree(layer(n1));

numLayer = 0;

system("pause");

break;

}

case '2':{

search(root2, fdata);

N = 0;

createTree(&root3, n2);

N = 0;

remNodesLev(&root3, layer(n2));

numLayer = 0;

N = 0;

cout << "New tree: " << endl;

OutTree(layer(n2));

numLayer = 0;

system("pause");

break;

}

default:{

cout << "Please try again" << endl;

system("pause");

break;

}

}

}

}

1. **Screen Shot результатів**

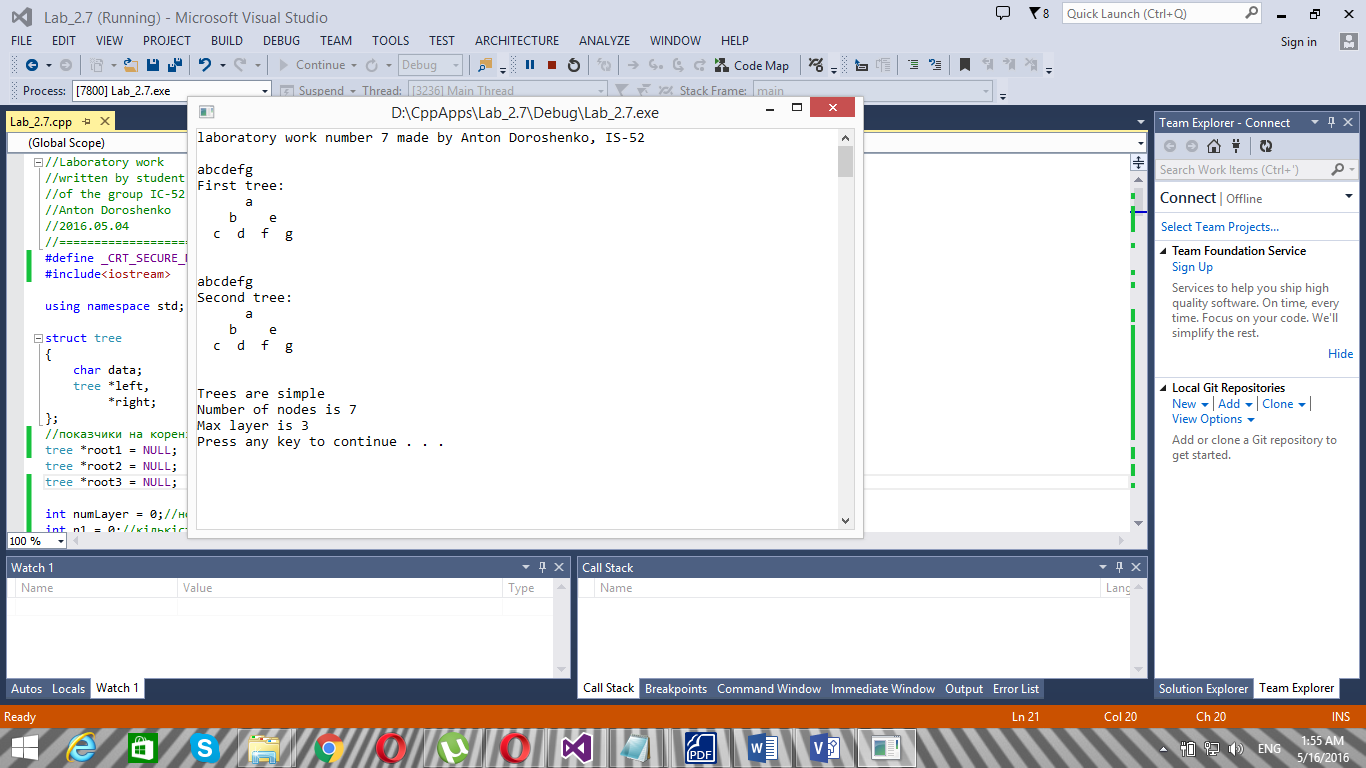


Рис.9 (Screen Shot результатів 1)

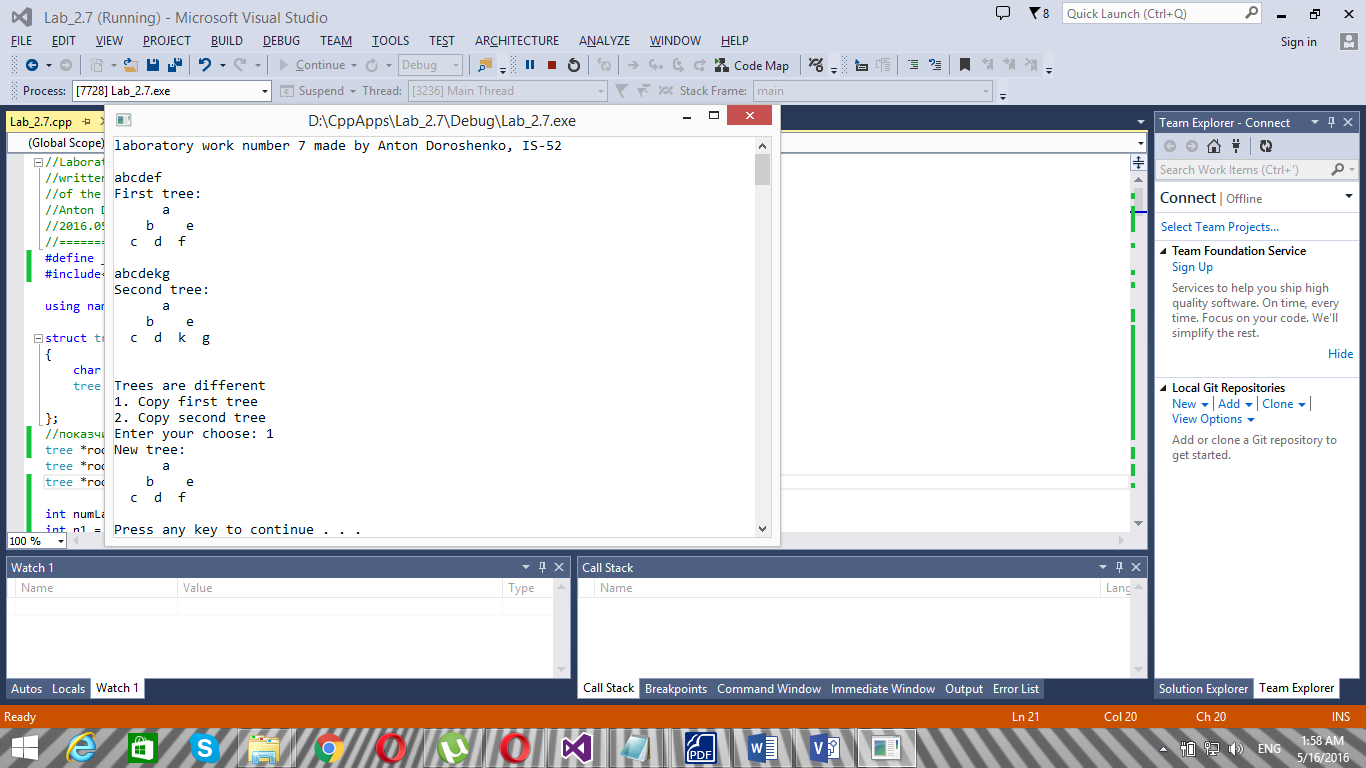


Рис.6 (Screen Shot результатів 2)

1. **Аналіз результатів**

Як ми бачимо зі ScreenShot’ів

Програма зчитує дані про вузли дерев з файлів і створює самі дерева, виводячи їх. Якщо дерева однакові, то програма виводить кількість вузлів та рівнів дерев. Якщо дерева різні, то за вибором користувача програма копіює одне з дерев.

1. **Висновок**

Програма працює правильно, про що свідчить аналіз результатів та ScreenShot’и. Програма коректно виводить результати. Програма застосовує динамічну структуру даних типу дерева.