Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи №6 з дисципліни

«Основи програмування

2. Модульне програмування»

«Дерева»

Виконав студент ІП-14 Нікулін Павло Юрійович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна
(прізвище, ім'я, по батькові)

Київ 2022

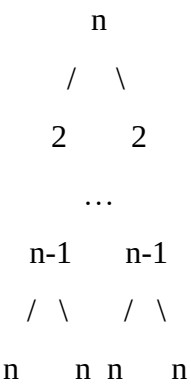
Лабораторна робота №6

Дерева

Мета: вивчити особливості організації і обробки дерев.

Хід роботи

Задача: Побудувати дерево наступного типу:



Розв'язання

1. **Постановка задачі**:

2. Побудова **математичної моделі**. Складемо таблицю імен змінних для двох мов.

C++

Змінна	Тип	Ім'я	Призначення
Хедер до файлу з функціями	Файловий	TREE.H	Файл
Файл з функціями	Файловий	TREE.CPP	Файл
Об'єкт класу Tree	Tree	T	Результат
Корінь дерева	Цілий	N	Початкове дане
Вузол дерева	Структура	NODE	Проміжне дане
Значення вузла	Цілий	DATA	Проміжне дане
Ліве листя вузла	node*	LEFT	Результат
Праве листя вузла	node*	RIGHT	Результат
Стебло дерева	Структура	TRUNK	Проміжне дане

Конструктор класу Trunk	Конструктор Trunk	TRUNK	Метод
Попереднє стебло	Trunk*	PREV_TRUNK	Проміжне дане
З’єднання у дереві	Рядковий	CONNECTION	Проміжне дане
Дерево	Клас	TREE	Результат
Конструктор класу Tree	Конструктор Tree	TREE	Метод
Метод додавання вершини (для виклику)	Універсальний	INSERT	Метод
Метод знищення дерева (для виклику)	Універсальний	DESTROY_TREE	Метод
Метод виведення дерева (для виклику)	Універсальний	OUTPUT	Метод
Метод додавання вершини	Універсальний	INSERT	Метод
Метод знищення дерева	Універсальний	DESTROY_TREE	Метод
Метод стебла дерева	Універсальний	SHOW_TRUNK	Метод
Метод виведення дерева	Універсальний	GRAPHICAL_TREE	Метод
Корінь дерева	node*	ROOT	Проміжне дане
Значення вершини	Цілий	NUM	Проміжне дане
Листя дерева	node*	LEAF	Проміжне дане
Поточне стебло	Trunk*	CURR_TRUNK	Проміжне дане
Поточний вузол/корінь	node*	CURR_ROOT	Проміжне дане
Попереднє стебло	Trunk*	PREVIOUS	Проміжне дане
Якщо рух по правому стеблу	Булевий	IS_RIGHT	Проміжне дане
Попереднє з’єднання	Рядковий	PREV_CONNECTION	Проміжне дане

Випробування коду

C++

Код

lab1.cpp

```cpp
#include <iostream>
#include "Tree.h"
using namespace std;

int main()
{
    Tree t;
    int n;

    cout << "Enter positive integer n: ";
    cin >> n;

    t.insert(n); // tree root is n
    t.insert(2); // first level siblings are 2

    if (n <= 2)
    {
        t.insert(n);
    }
    else
    {
        for (int i = 3; i <= n; i++)
        {
            t.insert(i); // insert next levels in the tree
        }
    }

    cout << "\nTree output:\n";
    t.output();
    t.destroy_tree();
}
```

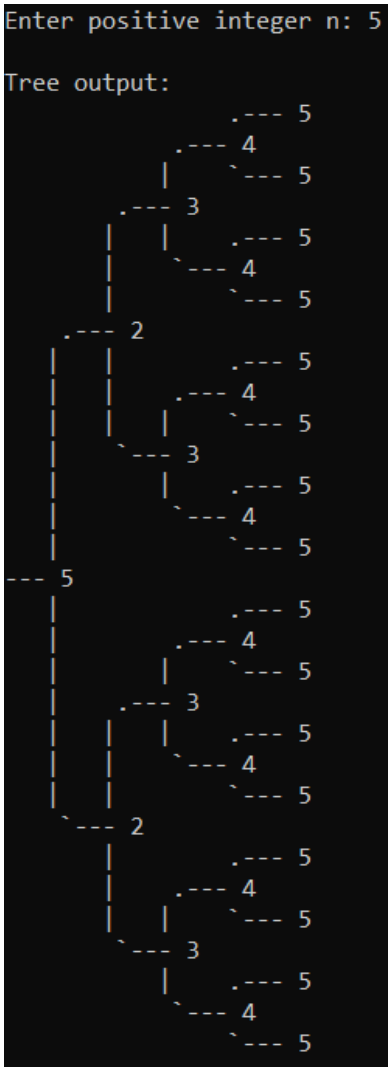
Tree.h

```cpp
#pragma once
#include <iostream>
#include <iomanip>
#include <string>
using namespace std;

struct node
{
    int data;
    node* left;
    node* right;
};

struct Trunk
{
    Trunk* prev_trunk;
    string connection;

    Trunk(Trunk* prev_trunk, string connection)
    {
        this->prev_trunk = prev_trunk;
        this->connection = connection;
    }
};

class Tree
{
public:
    Tree();

    void insert(int);
    void destroy_tree();
    void output();

private:
    void insert(int, node*);
    void destroy_tree(node*);


    void show_trunk(Trunk*);
    void graphical_tree(node*, Trunk*, bool);

    node* root;
};
```

Tree.cpp

```cpp
#include "Tree.h"

Tree::Tree()
{
    root = NULL;
}

void Tree::insert(int num) // initialize creating new level
{
    if (root != NULL)
        insert(num, root);
    else
    {
        root = new node;
        root->data = num;
        root->left = NULL;
        root->right = NULL;
    }
}

void Tree::insert(int num, node* leaf) // insert num siblings to the new level
{
    if (leaf->left != NULL && leaf->right != NULL)
    {
        insert(num, leaf->left);
        insert(num, leaf->right);
    }
    else
    {
        leaf->left = new node;
        leaf->left->data = num;
        leaf->left->left = NULL;
        leaf->left->right = NULL;

        leaf->right = new node;
        leaf->right->data = num;
        leaf->right->left = NULL;
        leaf->right->right = NULL;
    }
}

void Tree::destroy_tree() // initialize destroying tree
{
    destroy_tree(root);
}

void Tree::destroy_tree(node* leaf) // destroy tree
{
    if (leaf != NULL)
    {
        destroy_tree(leaf->left);
        destroy_tree(leaf->right);
        delete leaf;
    }
}

void Tree::output() // initialize tree output
{
    graphical_tree(root, NULL, false);
}

void Tree::show_trunk(Trunk* curr_trunk) // output trunk
{
    if (curr_trunk == NULL) {
        return;
    }

    show_trunk(curr_trunk->prev_trunk);
    cout << curr_trunk->connection;
}

void Tree::graphical_tree(node* curr_root, Trunk* previous, bool is_right) // output tree horizontally
{
    if (curr_root == NULL) {
        return;
    }

    string prev_connection = "    ";
    Trunk* curr_trunk = new Trunk(previous, prev_connection);

    graphical_tree(curr_root->right, curr_trunk, true);

    if (!previous) {
        curr_trunk->connection = "---";
    }
    else if (is_right)
    {
        curr_trunk->connection = ".---";
        prev_connection = "    |";
    }
    else if (previous)
    {
        curr_trunk->connection = "`---";
        previous->connection = prev_connection;
    }

    show_trunk(curr_trunk);
    cout << " " << curr_root->data << endl;

    if (previous) {
        previous->connection = prev_connection;
    }
    curr_trunk->connection = "    |";

    graphical_tree(curr_root->left, curr_trunk, false);
}
```

Результат



Висновок

Під час виконання лабораторної роботи було досліджено особливості організації і обробки дерев. Було створено бінарне дерево за макетом з умови. Коренем (нульовий рівень) є ціле число n, його дитини на першому рівні – число 2, наступні рівні – числа від 2 до n, і на останьому рівні всі вершини є числом n. Було реалізовано горизонтальне виведення дерева. Роботу було виконано на мові програмування C++ з використанням ООП (класи та об'єкти). Програма працює коректно, відповідно до умови задачі.