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»

Варіант 22

Виконав студент _____ІП-15_Мєшков_Андрій_Ігорович_____

(шифр, прізвище, ім'я, по батькові)

Перевірив _____Вечерковська Анастасія Сергіївна_____
(прізвище, ім'я, по батькові)

Київ 2022

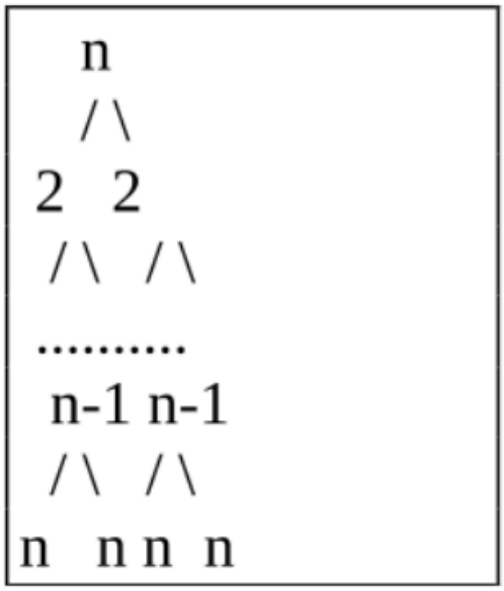
Лабораторна робота 6
Дерева

Мета – вивчити особливості організації дерев.

Варіант 22

Завдання.

22. Побудувати дерево наступного типу:



, де n -додатне ціле число

Код C++
Lab6.cpp

```cpp
#include <iostream>
#include <cmath>
#include "function.hpp"

using namespace std;

int main() {
    int n;

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

    newTree(n);
}

```

function.hpp

```
1  #ifndef function_hpp
2  #define function_hpp
3
4  #include <stdio.h>
5  #include "tree.hpp"
6
7  void space(int);
8  void newTree(int);
9
10 #endif
11
```

function.cpp

```cpp
#include "function.hpp"
#include <iostream>
using namespace std;

void space(int count) {
    for (int i = 0; i < count; i++) {
        cout << ' ';
    }
}

void newTree(int n){
    int l;
    if(n>0){
        Tree binTree(n);
        cout<<"What tree do you want?\nHorizontal(1) or Vertical(2)?\n";
        cin>>l;
        switch (l) {
            case 1:
                binTree.print_Tree(binTree.get(), 0);
                break;

            case 2:
                binTree.printLevelOrder(binTree.get(), n);
                break;

            default:
                break;
        }
        cout<<endl;

    }
    else cout<<"Impossible to create TREE"<<endl;
}

```

tree.hpp

```
#ifndef tree_hpp
#define tree_hpp
#include "function.hpp"
#include <stdio.h>
#include <cmath>

struct Node{
    int data;
    Node *left, *right;
};

class Tree{
    Node *root;
    int num;
public:
    int k;
    Tree():root(NULL){};
    Tree(int n): num(n){
        int m=0;
        for(int i=0; i<n; i++){
            m+=pow(2,i);
        }
        root = new Node [m];
        root = this->makeTree(n);
    };
    ~Tree(){delete root;};
    Node* get(){return root;}
    Node* makeTree(int n, int x=1);
    void print_Tree(Node* p, int level);

    void printCurrentLevel(Node* r, int level, int width, bool& flag, int i);
    void printLevelOrder(Node* r, int n);
};
#endif

```

tree.cpp

```cpp
#include "tree.hpp"
#include <iostream>
using namespace std;

Node* Tree::makeTree(int n, int x){
    if(n==0)return NULL;
    if(x==n+1)return NULL;
    int z;
    if(x==1)z=n;
    else z=x;
    Node* p = new Node;
    p->data = z;
    p->left = makeTree(n, x+1);
    p->right = makeTree(n, x+1);
    return p;
}

void Tree::print_Tree(Node* p, int level){
    if(p)
        {
            print_Tree(p->left,level + 1);
            for(int i = 0;i< level;i++) cout<<"    ";
            cout << p->data << endl;
            print_Tree(p->right,level + 1);
        }
}
```

```
void Tree::printCurrentLevel(Node* r, int level, int width, bool& flag, int heigh) {
    if (r == NULL) {
        space(2 * width + 1);
        if (level != 0) flag = false;
        if (!k) k = width;
        return;
    }
    if (level == 0) {
        if (!flag) {
            space(heigh * k - heigh * 3 / 2);
            cout << r->data;
            flag = true;
        }
        else {
            space(width);
            cout << r->data;
            space(width);
        }
    }
    else if (level > 0) {
        printCurrentLevel(r->left, level - 1, width, flag, heigh);
        space(1);
        printCurrentLevel(r->right, level - 1, width, flag, heigh);
    }
}
```

```
void Tree::printLevelOrder(Node* r, int n) {
    int h = n;
    int w = pow(2, h);
    int fullWidth = (2 * w - 1) / 2;
    bool flag;

    for (int i = 0; i <= h; i++) {
        flag = true;
        printCurrentLevel(r, i, fullWidth, flag, i);
        fullWidth /= 2;
        cout << endl;
    }
}
```

Результат
C++

```
Enter n: 5
What tree do you want?
Horizontal(1) or Vertical(2)?
2
                              5
              2                                   2
      3               3               3                   3
   4       4       4       4       4       4       4       4
 5   5   5   5   5   5   5   5   5   5   5   5   5   5   5   5


Program ended with exit code: 0
```

```
Enter n: 4
What tree do you want?
Horizontal(1) or Vertical(2)?
1
          4
      3
          4
   2
          4
      3
          4
4
          4
      3
          4
   2
          4
      3
          4

Program ended with exit code: 0
```