

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 № 5 з дисципліни

«Основи програмування 2. Модульне програмування»

«Дерева»

Варіант 27

Виконав студент ІП-15, Пономаренко Маргарита Альбертівна
(шифр, прізвище, ім'я, по батькові)

Перевірила Муха Ірина Павлівна
(прізвище, ім'я, по батькові)

Лабораторна робота 5

Дерева

Індивідуальне завдання

Варіант 27

27. Побудувати дерево, елементами якого є цілі числа. Визначити кількість вузлових вершин даного дерева та надрукувати їх координати (номер рівня та номер гілки).

Код

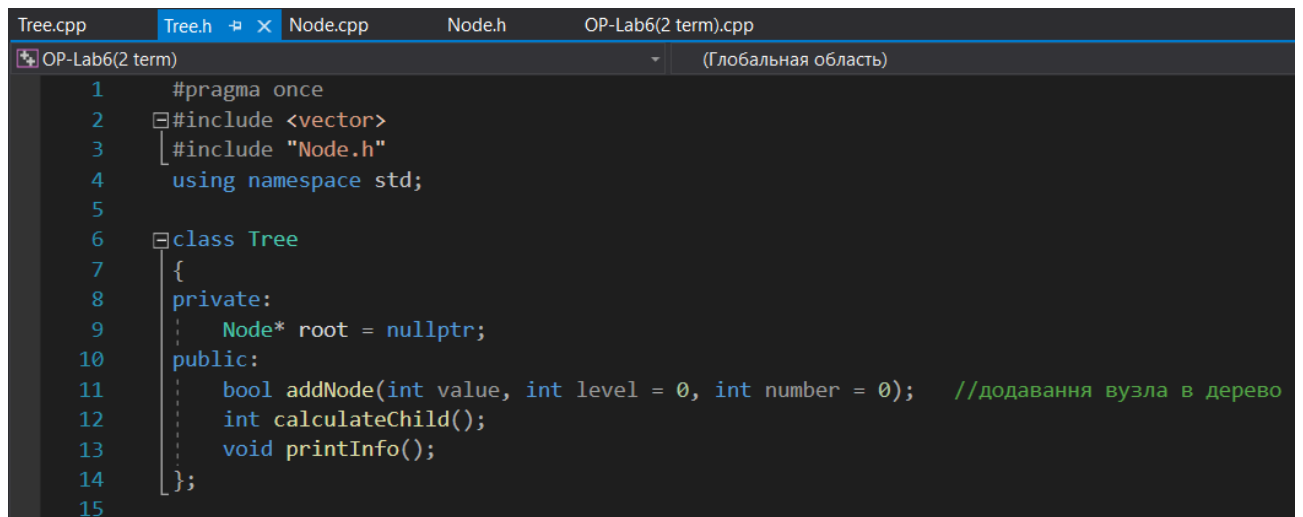
Tree.cpp :

```
Tree.cpp  Tree.h  Node.cpp  Node.h  OP-Lab6(2 term).cpp
OP-Lab6(2 term)  (Глобальная область)

1      #include "Tree.h"
2
3      bool Tree::addNode(int value, int level, int number)
4      {
5          Node* newNode = new Node(value, level, number);
6          if (root == nullptr) {
7              root = newNode; //створення кореня
8              root->setNumber(1);
9              root->printInfo();
10             cout << "\n";
11             return true;
12         }
13         else if(level == 0 && number == 0){ //припинення спроби створити більше одного кореня
14             cout << "Ви намагаєтесь додати вже існуючий корінь\n";
15             return false;
16         }
17         if (!root->addChild(newNode, 0)) {
18             cout << "Не вдалося додати вузол\n";
19             return false;
20         }
21         return true;
22     }
23
24     int Tree::calculateChild()
25     {
26         int childAmount = root->calculateChildCount();
27         return childAmount;
28     }
```

```
24     int Tree::calculateChild()
25     {
26         int childAmount = root->calculateChildCount();
27         return childAmount;
28     }
29
30     void Tree::printInfo()
31     {
32         root->printInfo();
33     }
34
```

Tree.h :



```
Tree.cpp  Tree.h  Node.cpp  Node.h  OP-Lab6(2 term).cpp
OP-Lab6(2 term) (Глобальная область)
1  #pragma once
2  #include <vector>
3  #include "Node.h"
4  using namespace std;
5
6  class Tree
7  {
8  private:
9      Node* root = nullptr;
10 public:
11     bool addNode(int value, int level = 0, int number = 0); //додавання вузла в дерево
12     int calculateChild();
13     void printInfo();
14 };
15
```

Node.cpp :

```

Tree.cpp  Tree.h  Node.cpp  Node.h  OP-Lab6(2 term).cpp
OP-Lab6(2 term)  (Глобальная область)

1  #include "Node.h"
2
3  bool Node::addChild(Node* child, int previous)
4  {
5      if (level < child->getLevel() - 1) {
6          cout << "перехід на рівень вище..." << endl;
7          for (size_t i = 0; i < children.size(); i++)
8          {
9              int last = i > 0 ? children.at(i - 1)->getLast() : 0;
10             if (children.at(i)->addChild(child, last)) {
11                 if (child->getLevel() == level + 2 && i < children.size() - 1) {
12                     for (size_t j = i + 1; j < children.size(); j++)
13                     {
14                         children.at(j)->increase();
15                     }
16                 }
17                 return true;
18             }
19         }
20     }

21     else if (level == child->getLevel() - 1) {
22         if (number == child->getNumber()) {
23             if (children.size() == 0) {
24                 child->setNumber(previous + 1);
25             }
26             else {
27                 child->setNumber(children.at(children.size() - 1)->getNumber() + 1);
28             }
29             children.push_back(child);
30             if (child->value == 9) {
31                 cout << " ";
32             }
33             cout << "added on level: " << child->getLevel() << " number: " << child->getNumber() << endl; //успішна спроба додавання дочірнього вузла
34             return true;
35         }
36     }
37     return false;
38 }

39
40 int Node::getLast()
41 {
42     if (children.size() > 0) {
43         return children.at(children.size() - 1)->getNumber();
44     }
45     return 0;
46 }
47

```

```
48 void Node::printInfo()
49 {
50     cout << "value: " << value << " (lev:" << level << "; num:" << number << ") - ";
51     for (auto child: children)
52     {
53         child->printInfo();
54         cout << "\n\t\t\t\t\t";
55     }
56 }
57
58 int Node::calculateChildCount()
59 {
60     int count = 1;
61     for (auto child : children)
62     {
63         count += child->calculateChildCount();
64     }
65     return count;
66 }
```

Node.h :

```
Tree.cpp  Tree.h  Node.cpp  Node.h  OP-Lab6(2 term).cpp
OP-Lab6(2 term)
1  #pragma once
2  #include <iostream>
3  #include <vector>
4  #include <string>
5  #include <sstream>
6  using namespace std;
7
8  class Node
9  {
10 private:
11     int value;
12     int level;
13     int number;
14     vector<Node*> children;
15
16 public:
17     Node(int value, int level, int number) :
18         value(value), level(level), number(number) {}
19     int getNumber() { return number; }
20     void setNumber(int number) { this->number = number; }
21     int getLevel() { return level; }
22
23     bool addChild(Node* child, int previous); //додавання дочірньої вершини
24     void increase() {
25         for (auto child : children)
26         {
27             child->number++;
28         }
29     }
```

Основи програмування 2. Модульне програмування

```
23     bool addChild(Node* child, int previous);    //додавання дочірньої вершини
24     void increase() {
25     for (auto child : children)
26     {
27         child->number++;
28     }
29     }
30     int getLast();
31     void printInfo();    //виведення інформації про дерево
32     int calculateChildCount();    //підрахунок дочірніх вершин
33
34 };
```

OP-Lab6(2 term).cpp:

```
Tree.cpp  Tree.h  Node.cpp  Node.h  OP-Lab6(2 term).cpp*  X
OP-Lab6(2 term)  (Глобальная область)
1  //Побудувати дерево, елементами якого є цілі числа. Визначити кількість вузлових вершин даного дерева та надрукувати їх координати
2  //(номер рівня та номер гілки).
3
4  #include <iostream>
5  #include "Node.h"
6  #include "Tree.h"
7  using namespace std;
8
9  int main()
10 {
11     setlocale(LC_CTYPE, "ukr");
12
13     int num;
14     cout << "Скільки елементів бажаєте створити? ";
15     cin >> num;
16     int value, level, number;
17
18     Tree tree;
19     for (size_t i = 0; i < num; i++)
20     {
21         cout << endl << "Node №" << i << endl;
22         cout << "Введіть значення: ";
23         cin >> value;
24         cout << "Введіть номер рівня: ";
25         cin >> level;
26         cout << "Введіть номер батьківського вузла: ";
27         cin >> number;
28         if (!tree.addNode(value, level, number)) {
29             i--;
30         }
31     }
32 }
```

```
18     Tree tree;
19     for (size_t i = 0; i < num; i++)
20     {
21         cout << endl << "Node №" << i << endl;
22         cout << "Введіть значення: ";
23         cin >> value;
24         cout << "Введіть номер рівня: ";
25         cin >> level;
26         cout << "Введіть номер батьківського вузла: ";
27         cin >> number;
28         if (!tree.addNode(value, level, number)) {
29             i--;
30         }
31     }
32     cout << endl << "Кількість вузлових вершин = " << tree.calculateChildCount() << endl;    //обрахунок кількості вузлових вершин
33     tree.printInfo();
34     return 0;
35 }
```

Результат виконання

```
Консоль отладки Microsoft Visual Studio
Скільки елементів бажаєте створити? 3

Node №0
Введіть значення: 2
Введіть номер рівня: 0
Введіть номер батьківського вузла: 1
value: 2 (lev:0; num:1) -

Node №1
Введіть значення: 5
Введіть номер рівня: 1
Введіть номер батьківського вузла: 1
added on level: 1 number: 1

Node №2
Введіть значення: 8
Введіть номер рівня: 2
Введіть номер батьківського вузла: 1
перехід на рівень вище...
added on level: 2 number: 1

Кількість вузлових вершин = 3
value: 2 (lev:0; num:1) - value: 5 (lev:1; num:1) - value: 8 (lev:2; num:1) -
  L
  L
```