

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  
«Алгоритми та структури даних-2.  
Структури даних»

«Деревовидні структури»

Варіант 5

Виконав студент

ІП-15, Буяло Дмитро Олександрович  
(шифр, прізвище, ім'я, по батькові)

Перевірив

Соколовський Владислав Володимирович  
(прізвище, ім'я, по батькові)

Київ 2022

## **Лабораторна робота 6**

### **Деревовидні структури**

**Мета** – вивчити основні підходи формалізації та імплементації алгоритмів побудови та обробки базових деревовидних структур даних.

#### **Індивідуальне завдання**

##### **Варіант 5**

#### **Завдання**

Побудувати двійкове дерево, елементами якого є символи. Визначити, чи знаходиться у цьому дереві елемент, значення якого вводиться з клавіатури. Якщо елемент знайдений, то підрахувати число його входжень.

При виконанні цієї лабораторної роботи для підрахунку кількості входжень заданого елемента, було використано зворотний симетричний обхід дерева, саме цей алгоритм буде наведений в псевдокоді.

Для реалізації завдання вистачило двох функцій (побудова дерева та пошук елемента), які наведені у файлі Template.h на 39-74 рядках коду.

Усі інші функції відносяться до вертикального та горизонтального виводу дерева.

## Виконання

### 1. Псевдокод алгоритму

Find(root, key, counter)

**Початок**

**Якщо** root = NULL

**то**

return

**Все якщо**

Find(right[root], key, counter)

**Якщо** root = key

**то**

counter ++

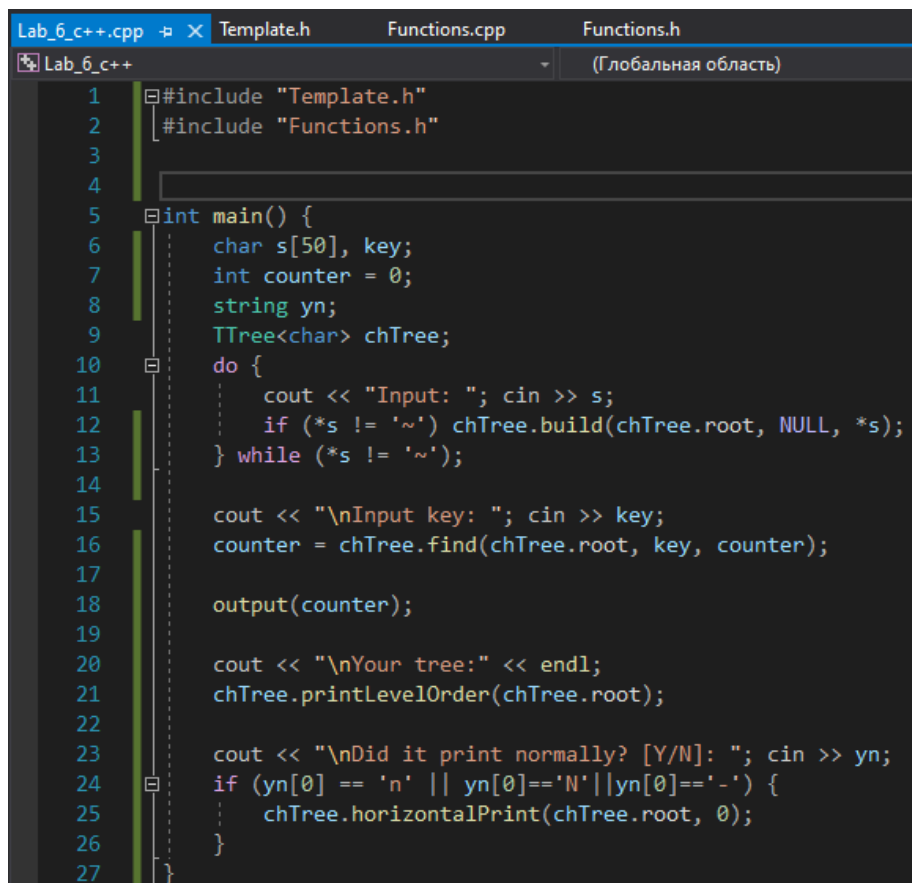
**Все якщо**

Find(left[root], key, counter)

**Кінець**

### 2. Програмна реалізація

#### 2.1. Вихідний код



```
Lab_6_c++.cpp  Template.h  Functions.cpp  Functions.h
Lab_6_c++ (Глобальная область)
1  #include "Template.h"
2  #include "Functions.h"
3
4
5  int main() {
6      char s[50], key;
7      int counter = 0;
8      string yn;
9      TTree<char> chTree;
10     do {
11         cout << "Input: "; cin >> s;
12         if (*s != '~') chTree.build(chTree.root, NULL, *s);
13     } while (*s != '~');
14
15     cout << "\nInput key: "; cin >> key;
16     counter = chTree.find(chTree.root, key, counter);
17
18     output(counter);
19
20     cout << "\nYour tree:" << endl;
21     chTree.printLevelOrder(chTree.root);
22
23     cout << "\nDid it print normally? [Y/N]: "; cin >> yn;
24     if (yn[0] == 'n' || yn[0] == 'N' || yn[0] == '-') {
25         chTree.horizontalPrint(chTree.root, 0);
26     }
27 }
```

```

Lab_6_c++.cpp  Template.h  Functions.cpp  Functions.h
Lab_6_c++      (Глобальная область)

1  #pragma once
2  #include <iostream>
3  #include <iomanip>
4  // ...
5  #include "Functions.h"
6
7
8  using namespace std;
9
10 template <class T>
11 class TTree {
12     T info; // текущий элемент
13     TTree* left;
14     TTree* right;
15     int height;
16     int width;
17     int k; // счетчик пробелов
18 public:
19     TTree* root; // указатель на корень
20     TTree() { root = NULL; }
21     ~TTree(); // деструктор для удаления динамически выделенной памяти
22     void build(TTree<T>* r, TTree<T>* previous, T info);
23     int find(TTree<T>* r, T key, int& counter);
24     int findHeight(TTree<T>* r);
25     void printCurrentLevel(TTree<T>* r, int level, int width, bool& flag, int i);
26     void printLevelOrder(TTree<T>* r);
27     void horizontalPrint(TTree<T>* r, int l);
28 };
29
30 //для шаблонов методы не нужно выносить в отдельный файл
31
32 template<class T>
33 TTree<T>::~~TTree() {
34     delete left;
35     delete right;
36     delete root;
37 }
38

```

```

Lab_6_c++.cpp  Template.h  Functions.cpp  Functions.h
Lab_6_c++  (Глобальная область)

39  template<class T>
40  void TTree<T>::build(TTree<T>* r, TTree<T>* previous, T info) { // строим дерево
41  if (!r) {
42      r = new TTree<T>;
43      r->left = NULL;
44      r->right = NULL;
45      r->info = info;
46      if (!root) {
47          root = r;
48      }
49      else {
50          if (info < previous->info) {
51              previous->left = r;
52          }
53          else {
54              previous->right = r;
55          }
56      }
57  }
58  else if (info < r->info) {
59      build(r->left, r, info);
60  }
61  else {
62      build(r->right, r, info);
63  }
64  }

65
66  template<class T>
67  int TTree<T>::find(TTree<T>* r, T key, int& counter) { // ищем заданный элемент
68  if (!r) {
69      return counter;
70  }
71  find(r->right, key, counter); // симметричный обход дерева
72  if (r->info == key) counter++;
73  find(r->left, key, counter);
74  }
75
76  template<class T>
77  void TTree<T>::printLevelOrder(TTree<T>* r) { // рекурсивный обход дерева в ширину
78      int h = findHeight(r);
79      width = pow(2, height);
80      int fullWidth = (2 * width - 1) / 2;
81      bool flag;
82
83      for (int i = 0; i <= h; i++) {
84          flag = true;
85          printCurrentLevel(r, i, fullWidth, flag, i);
86          fullWidth /= 2;
87          cout << endl;
88      }
89  }

```

```

Lab_6_c++.cpp  Template.h  Functions.cpp  Functions.h
Lab_6_c++  (Глобальная область)

91  template<class T> // печатаем все элементы на уровне
92  void TTree<T>::printCurrentLevel(TTree<T>* r, int level, int width, bool& flag, int heigh) {
93      if (r == NULL) {
94          space(2 * width + 1);
95          if (level != 0) flag = false;
96          if (!k) k = width;
97          return;
98      }
99      if (level == 0) {
100         if (!flag) {
101             space(heigh * k - heigh * 3 / 2);
102             cout << r->info;
103             flag = true;
104         }
105         else {
106             space(width);
107             cout << r->info;
108             space(width);
109         }
110     }
111     else if (level > 0) {
112         printCurrentLevel(r->left, level - 1, width, flag, heigh);
113         space(1);
114         printCurrentLevel(r->right, level - 1, width, flag, heigh);
115     }
116 }

118  template<class T>
119  int TTree<T>::findHeight(TTree<T>* r) { // поиск высоты дерева
120      if (!(r->left) && !(r->right)) height = 0;
121      else if (!(r->right)) {
122          height = findHeight(r->left) + 1;
123      }
124      else if (!(r->left)) {
125          height = findHeight(r->right) + 1;
126      }
127      else {
128          height = max(findHeight(r->left), findHeight(r->right)) + 1;
129      }
130      return height;
131  }

132
133  template <class T> // на всякий случай
134  void TTree<T>::horizontalPrint(TTree<T>* r, int l) { // горизонтальный вывод
135      int i;
136      if (!r)
137      {
138          return;
139      }
140      horizontalPrint(r->right, l + 1);
141      space(1);
142      cout << r->info << endl;
143      horizontalPrint(r->left, l + 1);
144  }

```

```
Lab_6_++.cpp  Template.h  Functions.cpp  Functions.h  [X]
Lab_6_++
1  #pragma once
2  #include <iostream>
3  #include <iomanip>
4
5  using namespace std;
6
7  void space(int);
8  void output(int);
```

```
Lab_6_++.cpp  Template.h  Functions.cpp  Functions.h
Lab_6_++
1  #include "Functions.h"
2
3
4  void space(int count) { // вывод заданного количества пробелов
5      for (int i = 0; i < count; i++) {
6          cout << ' ';
7      }
8  }
9
10 void output(int count) {
11     if (count == 0) cout << "\nNot found(" << endl;
12     else {
13         cout << "\nNumber of occuences = " << count; cout << endl;
14     }
15 }
```

## 2.2. Приклади роботи

```

Input: 4
Input: 6
Input: 7
Input: 2
Input: 5
Input: 3
Input: 3
Input: 6
Input: 4
Input: 1
Input: 1
Input: 5
Input: 0
Input: 7
Input: .
Input: .
Input: ~

Input key: 1

Number of occurrences = 2

Your tree:
      4
     / \
    2   6
   / \ / \
  1  3 5  7
 / \
0  1
.
.

Did it print normally? [Y/N]: y
C:\Users\Пользователь\OneDrive\Документы\КПИ\ОП\ОП-1\ОП-1-2\Л

```

```

Input: f
Input: R
Input: j
Input: 7
Input: :
Input: W
Input: w
Input: .
Input: 9
Input: A
Input: w
Input: ~

Input key: w

Number of occurrences = 2

Your tree:
      f
     / \
    R   j
   / \ / \
  7  W  :  w
 / \
.  9 A

Did it print normally? [Y/N]: yes
C:\Users\Пользователь\OneDrive\Док

```

```

Input: 1
Input: 2
Input: 3
Input: 4
Input: 5
Input: ~

Input key: Q
Not found(

Your tree:
      1
     / \
    2   3
   / \ / \
  4  5 4  5

Did it print normally? [Y/N]: +
C:\Users\Пользователь\OneDrive\Д

```

## 3. Висновок

Під час виконання шостої лабораторної роботи, розглянули на практиці роботу з деревами. В результаті була створена програма, яка може визначити, чи знаходиться елемент у цьому дереві, якщо так, то виводить кількість входжень.

Також вивчили основні підходи формалізації та імплементації алгоритмів побудови та обробки базових деревовидних структур даних.