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 9

Виконав студент ІП-15, Дзюбенко Даниїл Дмитрович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 5

Дерева

Мета – вивчити особливості організації і обробки дерев.

Варіант 9

Задача

9. Заданий рядок символів латинського алфавіту. Побудувати дерево, в якому значеннями вершин є символи, що розміщуються на рівнях відповідно до кількості їх повторень у рядку.

Код

Main.cpp

```cpp
#include "Tree.h"
#include "func.h"

int main() {

    Tree* tree;
    tree = createTree();

    cout << "\nResulting binary tree:\n";
    printTree(tree, 0);

    system("pause");
    return 0;
}
```

Tree.h

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

class Tree {
public:

    char data;
    int count;
    Tree* left, * right;

    Tree(char ch, int c, Tree* l = nullptr, Tree* r = nullptr) :data(ch), count(c), left(l), right(r) {};
    ~Tree();

};
```

Tree.cpp

```cpp
#include "Tree.h"
#include "func.h"

Tree::~Tree() {
    recursionDestructor(left);
    recursionDestructor(right);
}
```

Func.h

```cpp
#pragma once
#include "Tree.h"

Tree* createTree();
void insert(Tree*&, char, int);
void printTree(Tree*, int);
bool contains(Tree*, char);
void recursionDestructor(Tree*);
bool validateString(string);
```

Func.cpp

```cpp
#include "func.h"

Tree* createTree() {
    int count;
    Tree* root = nullptr;
    string str;
    cout << "Enter string of symbols separated by one space:\n";
    getline(cin, str);
    while (!validateString(str)) {
        cout << "\nInvalid input string!\nEnter string again:\n";
        getline(cin, str);
    }
    for (int i = 0; i < str.length(); i++) {
        count = 0;
        if (str[i] != ' ' && !contains(root, str[i])) {
            for (int j = 0; j < str.length(); j++)
            {
                if (str[i] == str[j]) {
                    count++;
                }
            }
            insert(root, str[i], count);
        }
    }
    return root;
}

bool validateString(string str) {
    if (str.length() >= 1) {
        for (int i = 0; i < str.length(); i++) {
            if (i % 2 == 0 && (str[i] == ' ' || str[i] < 65 || str[i] > 122 || (str[i] > 90 && str[i] < 97))) {
                return false;
            }
            else if (i % 2 != 0 && str[i] != ' ') {
                return false;
            }
        }
        return true;
    }
    else {
        return false;
    }
}
```

```
void insert(Tree*& tree, char ch, int count) {
    if (tree == nullptr) tree = new Tree(ch, count);
    else {
        if (tree->data < ch) insert(tree->right, ch, count);
        else insert(tree->left, ch, count);
    }
}

bool contains(Tree* tree, char ch) {
    if (tree == nullptr) return false;
    if (tree->data == ch) return true;
    if (tree->data < ch) return contains(tree->right, ch);
    if (tree->data > ch) return contains(tree->left, ch);
}

void printTree(Tree* tree, int count) {
    if (tree != nullptr) {
        string space = " ";
        for (int i = 0; i < count; i++)
        {
            space += "    ";
        }
        printTree(tree->right, count + 1);
        cout << space << tree->count << " " << tree->data << endl;
        printTree(tree->left, count + 1);
    }
}

void recursionDestructor(Tree* current) {
    if (current->left) recursionDestructor(current->left);
    if (current->left) recursionDestructor(current->left);
    delete current;
}
```

Результат роботи програми

```
Enter string of symbols separated by one space:
d d b a c c d f e d g e e g

Resulting binary tree:
        2 g
    1 f
        3 e
 4 d
        2 c
    1 b
        1 a
Для продолжения нажмите любую клавишу . . .
```