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-14 Демченко Філіпп Ігорович
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

Лабораторна робота 5

Дерева

Мета – вивчити особливості організації та обробки дерев.

Завдання.

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 програми

BinaryTree.hpp

```
#ifndef BINARY_TREE_H
#define BINARY_TREE_H

struct Node {
private:
    char letter;
    Node *right;
    Node *left;
public:
    Node() { letter = 0; right = 0; left = 0; }

    void Letter(char i_letter) { letter = i_letter; }
    void RightSubTree(Node *i_right) { right = i_right; }
    void LeftSubTree(Node *i_left) { left = i_left; }

    char Letter() const { return letter; }
    Node *RightSubTree() const { return right; }
    Node *LeftSubTree() const { return left; }
};
```

```
class BinaryTree {
    Node *root;
public:
    BinaryTree() { root = 0; }
    ~BinaryTree();

    Node *GetRoot() const { return root; };


    void AddNewNode(char new_letter);
    void ShowTree(Node *current_root, int depth = 0) const;

private:
    BinaryTree(const BinaryTree &ref);
    BinaryTree operator=(const BinaryTree &ref);
    void ClearTree(Node *current_root);
};
```

Реалізація методів класу BinaryTree

```
#include "BinaryTree.hpp"
#include <cstdio>


void print_spaces(int n)
{
    for (int i = 0; i < n; i++)
        putchar(' ');
}


void BinaryTree::ClearTree(Node *current_root)
{
    if (!current_root)
        return;

    ClearTree(current_root->LeftSubTree());
    ClearTree(current_root->RightSubTree());
    delete current_root;
    current_root = 0;
}


BinaryTree::~BinaryTree()
{
    ClearTree(root);
}
```

```
void BinaryTree::AddNewNode(char new_letter)
{
    Node *new_node = new Node;

    new_node->Letter(new_letter);
    new_node->RightSubTree(0);
    new_node->LeftSubTree(0);

    if (!root){
        root = new_node;
        return;
    }

    Node *current_node = root;
    while (current_node->RightSubTree()){
        current_node = current_node->RightSubTree();
    }

    current_node->RightSubTree(new_node);
}

void BinaryTree::ShowTree(Node *current_root, int depth) const
{
    if (!current_root)
        return;

    print_spaces(depth * 4);
    printf("%c\n", current_root->Letter() == '\0'? '*' : current_root->Letter());
    ShowTree(current_root->RightSubTree(), depth + 1);
    ShowTree(current_root->LeftSubTree(), depth + 1);
}
```

Код головної програми

```
#include <cstdio>
#include "BinaryTree.hpp"
#include <cstring>
#include <cstdlib>

// Max string len
constexpr int STRING_LEN = 1024;

// Creating buffer for a string
static char string[STRING_LEN];


// Count characters in string
int count_char(const char *string, char c);
// Comparator function
int comp(const void *p1, const void *p2);
```

```
int main(int argc, char *argv[])
{
    printf("Enter some text: ");
    fgets(string, STRING_LEN, stdin);
    string[STRING_LEN - 1] = '\0';

    char *new_line_pos = strchr(string, '\n');
    if (new_line_pos)
        *new_line_pos = '\0';

    qsort(string, strlen(string), sizeof(char), comp);

    BinaryTree tree;
    tree.AddNewNode('\0');

    // Counter for letters
    int counter = 0;

    for (int i = 0; i < strlen(string); i++){
        if (i == 0 || string[i] != string[i - 1]){
            tree.AddNewNode(string[i]);
            printf("%c - %d\n", string[i], count_char(string, string[i]));
        }
    }

    putchar('\n');
    tree.ShowTree(tree.GetRoot());

    return 0;
}
```

```
int count_char(const char *string, char c)
{
    int result = 0;
    while (*string){
        if (*string == c)
            ++result;
        string++;
    }
    return result;
}

int comp(const void *p1, const void *p2)
{
    if (count_char(string, *(char *)p1) > count_char(string, *(char *)p2))
        return 1;
    if (count_char(string, *(char *)p1) < count_char(string, *(char *)p2))
        return -1;

    if (*(char *)p1 < *(char *)p2)
        return -1;
    else if (*(char *)p1 > *(char *)p2)
        return 1;
    else
        return 0;
}
```

Результат виконання коду

Висновок. В цій лабораторній роботі було досліджено декілька фундаментальних структур данних, а саме списки, дерева і тд. Було вивчено особливості побудови дерев та роботу з ними.