Отчет по лабораторной работе № 23 по курсу

«Практикум программирования»

Студент группы М8О-109Б-22 Ефименко Кирилл Игоревич

Контакты: telegram @vivichv9

Работа выполнена: 27.04.2023

Преподаватель: каф.806 Сысоев Максим Алексеевич

Отчет сдан «25» июня 2023г., итоговая оценка ____

Подпись преподавателя _____

1. Тема: Динамические структуры данных. Обработка деревьев

2. Цель работы: Составить программу на C++ для построения и обработки дерева, а так же для выполнения специального действия

3. Задание (вариант № 33): Определить число вершин двоичного дерева, имеющих ровно два поддерева

4. Оборудование (студента):

Процессор AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz, ОП 16,0 Гб, SSD 512 Гб. Монитор 1920x1080 144 Hz

5. Программное обеспечение (студента):

Операционная система семейства Linux, наименование Ubuntu, версия 18.10

Интерпретатор команд: bash, версия 4.4.19

Система программирования – версия --, редактор текстов Emacs, версия 25.2.2

Утилиты операционной системы –

Прикладные системы и программы –

Местонахождение и имена файлов программ и данных на домашнем компьютере –

6. Идея, метод, алгоритм решения задачи *(в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)*

Строим дерево, используя обход находим число вершин, имеющих ровно два поддерева.

7. Сценарий выполнения работы *(план работы, первоначальный текст программы в черновике [можно на отдельном листе] и тесты, либо соображения по тестированию)*

1. Читаю про деревья
2. Делаю свою реализацию дерева на плюсах
3. Пишу код специального действия

8. Распечатка протокола *(подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем)*

Node.hpp

```hpp
#ifndef NODE_HPP_INCLUDED
#define NODE_HPP_INCLUDED

#include <iostream>

template<typename T>
class Tree;

template<typename T>
class Node {
    friend class Tree<T>;

    T data;
    Node<T>* leftNode = nullptr;
    Node<T>* rightNode = nullptr;

    Node(const T&);
    T& getData();
};

#include "../src/Node.cpp"

#endif


```

Node.cpp

```cpp
#include "../include/Node.hpp"

template<typename T>
Node<T>::Node(const T& data){
    this->data = data;
}

template <typename T>
T& Node<T>::getData() {
    return data;
}
```

Tree.hpp

```hpp
#ifndef INCLUDE_TREE_HPP
#define INCLUDE_TREE_HPP

#include "Node.hpp"

template <typename T>
class Tree {
    private:
        Node<T>* root = nullptr;
        int counter = 0;

        Node<T>* insert(const T&, Node<T>*);
        void bypass(Node<T>*);
        void clearTree(Node<T>*);

    public:
        Tree() = default;;
        Tree(const T&);
        ~Tree();

        void insert(const T&);
        void bypass();
        void task();
};

#include "../src/Tree.cpp"

#endif
```

Tree.cpp

```cpp
#include <iostream>
#include "../include/Tree.hpp"
#include "../include/Node.hpp"

template <typename T>
Tree<T>::Tree(const T& data) {
    root = new Node<T>(data);
}

template <typename T>
Node<T>* Tree<T>::insert(const T& data, Node<T>* node) {
    if(node == nullptr) {
        node = new Node<int>(data);
        node->leftNode = nullptr;
        node->rightNode = nullptr;
    }
    else if(data < node->data)
        node->leftNode = insert(data, node->leftNode);
    else if(data > node->data)
        node->rightNode = insert(data, node->rightNode);
    return node;
}

template <typename T>
void Tree<T>::insert(const T& data) {
    root = insert(data, root);
}
```

```cpp
template <typename T>
void Tree<T>::bypass(Node<T>* node) {
    if (node == nullptr) {
        return;
    }
    bypass(node->leftNode);

    if (node->leftNode != nullptr and node->rightNode != nullptr) {
        counter++;
    }

    bypass(node->rightNode);
}

template <typename T>
void Tree<T>::task() {
    bypass(root);
    std::cout << "\nthe number of vertices of a binary tree having exactly two subtrees: " << counter << '\n';
}

template <typename T>
void Tree<T>::clearTree(Node<T>* node) {
    if (node == nullptr) {
        return;
    }

    clearTree(node->leftNode);
    clearTree(node->rightNode);
    delete node;
}
```

```cpp

template <typename T>
void Tree<T>::bypass() {
    bypass(root);
}

template <typename T>
Tree<T>::~Tree() {
    clearTree(root);
}
```

main.cpp

```cpp
#include <iostream>
#include "../include/Tree.hpp"

int main() {
    Tree<int> bt;

    bt.insert(50);
    bt.insert(30);
    bt.insert(40);
    bt.insert(20);
    bt.insert(10);
    bt.insert(70);
    bt.insert(60);
    bt.insert(120);
    bt.insert(80);

    bt.task();

    return 0;
}
```

9. Дневник отладки *(дата и время сеансов отладки и основные события [ошибки в сценарии и программе, нестандартные ситуации] и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы)*

№	*Лаб. или дом*	*Дата*	*Время*	*Событие*	*Действие по исправлению*	*Примечания*

Особых проблем при выполнении лабы не возникло

10. Замечания автора *(по существу работы)*

Замечания отсутствуют

11. Вывод

Интересно было выполнять данную работу, благодаря ей я лучше понял тему “Бинарные деревья”

Подпись студента _____