

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования «Тульский государственный
университет»

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

ДРЕВОВИДНЫЕ СТРУКТУРЫ

отчет о
лабораторной работе №10

по дисциплине
ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

ВАРИАНТ 8

Выполнила:

ст. гр. 230711

Павлова В.С.

Проверил:

асс. каф. ИБ

Курбаков М.Ю.

Тула, 2022 г.

ЦЕЛЬ И ЗАДАЧА РАБОТЫ

Цель: ознакомиться с древовидными структурами, такими как деревья, бинарные и сбалансированные деревья. Узнать основные понятия и научиться выполнять операции поиска, включения и удаления узлов, как с бинарными, так и со сбалансированными деревьями.

Задача: в данной работе требуется написать программу, демонстрирующую использование изученных принципов.

ЗАДАНИЕ НА РАБОТУ

Написать программу, реализующую операцию включения узла в бинарном дереве.

СХЕМА ПРОГРАММЫ

Схема алгоритма программы представлена на рисунке 1.

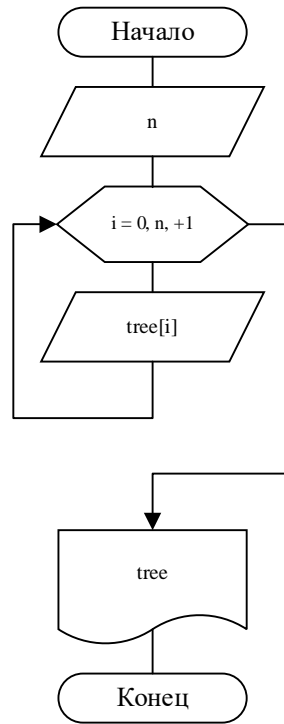


Рисунок 1 – Схема алгоритма программы

ТЕКСТ ПРОГРАММЫ

Текст программы на языке программирования C++ для включения узла в бинарном дереве представлен в листинге 1.

Листинг 1. Текст программы

```
#include <iostream>
#include <string>
using namespace std;

struct Node
{
    int value;
    Node* left, * right;

    Node() {};
    Node(int data)
    {
        this->value = data;
        this->left = this->right = nullptr;
    }
};
```

Листинг 1. Текст программы (продолжение)

```
    }
    ~Node() {}
};

class BinaryTree
{
public:
    Node* root;
    BinaryTree()
    {
        root = NULL;
    }

    void Add(int key, Node* leaf)
    {
        if (key < leaf->value) {
            if (leaf->left != NULL) {
                Add(key, leaf->left);
            }
            else {
                leaf->left = new Node;
                leaf->left->value = key;
                leaf->left->left = NULL;
                leaf->left->right = NULL;
            }
        }
        else if (key >= leaf->value) {
            if (leaf->right != NULL) {
                Add(key, leaf->right);
            }
            else {
                leaf->right = new Node;
                leaf->right->value = key;
                leaf->right->right = NULL;
                leaf->right->left = NULL;
            }
        }
    }

    void Add(int key)
    {
        if (root != NULL) {
            Add(key, root);
        }
        else {
            root = new Node;
            root->value = key;
            root->left = NULL;
            root->right = NULL;
        }
    }
};

struct Trunk //Вспомогательная структура
{
    Trunk* prev;
    string str;

    Trunk(Trunk* prev, string str)
    {
        this->prev = prev;
        this->str = str;
    }
};
```

Листинг 1. Текст программы (продолжение)

```
void showTrunks(Trunk* p)           //Вспомогательная функция для печати ветвей
двоичного дерева
{
    if (p == nullptr) {
        return;
    }

    showTrunks(p->prev);
    cout << p->str;
}

void printTree(Node* root, Trunk* prev, bool isLeft)
{
    if (root == nullptr) {
        return;
    }

    string prev_str = " ";
    Trunk* trunk = new Trunk(prev, prev_str);

    printTree(root->right, trunk, true);

    if (!prev) {
        trunk->str = "——";
    }
    else if (isLeft)
    {
        trunk->str = ".——";
        prev_str = " |";
    }
    else {
        trunk->str = "`——";
        prev->str = prev_str;
    }

    showTrunks(trunk);
    cout << " " << root->value << endl;

    if (prev) {
        prev->str = prev_str;
    }
    trunk->str = " |";

    printTree(root->left, trunk, false);
}

int main()
{
    setlocale(LC_ALL, "RUSSIAN");
    int n, x;
    cout << "Введите количество элементов дерева: "; cin >> n;

    BinaryTree tree;

    for (size_t i = 0; i < n; i++)
    {
        cout << "Введите число: "; cin >> x;
        tree.Add(x);
    }
    cout << "\n\nПолученное дерево:\n\n";
    printTree(tree.root, nullptr, false);

    return 0;}
```

ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЯ

Данная программа предназначена для включения узла в бинарное дерево. Пользователю предлагается ввести количество узлов и содержащиеся в них числа. В конце выполнения программы выводится полученное дерево.

ИНСТРУКЦИЯ ПРОГРАММИСТА

Структуры данных, используемые в программе, приведены в таблицах 1-3.

Таблица 1 – Структуры данных в программе

Имя	Тип (класс)	Предназначение
n	int	Количество узлов
x	int	Текущее число
tree	Node*	Полученное дерево

Для хранения узла бинарного дерева была создана вспомогательная структура Node, описание которой представлено в таблице 2.

Таблица 3 – Описание разработанной структуры Node

struct Node	
Поля/свойства (элементы данных) структуры	
Название и тип	Описание
char data	Значение, которое хранится в узле
Node* left	Ссылка на левое поддерево
Node* right	Ссылка на правое поддерево

Для работы с бинарным деревом был написан класс BinaryTree, описание которого приведено в таблице 3.

Таблица 3 – Описание разработанного класса BinaryTree

Class BinaryTree	
Поля/свойства (элементы данных) структуры	
Название и тип	Описание
Node* root	Ссылка на корень

ДЕМОНСТРАЦИОННЫЙ ПРИМЕР

Пусть имеется бинарное дерево $\{8, 3, 5, 6, 1\}$. Его графическое изображение приведено на рисунке 2. Необходимо добавить в него узел $\{12\}$ и вывести полученное дерево. Поскольку новое значение больше корневого $\{8\}$, необходимо двигаться вправо. Справа от корня значений нет, значит необходимо произвести вставку на пустующее место. Если бы, например, нужно было вставить элемент меньше корневого, то необходимо было бы повторять приведённые рассуждения для левого поддерева до тех пор, пока не нашлось бы подходящее место (или пока не появится возможность создать его из терминального узла – листа).

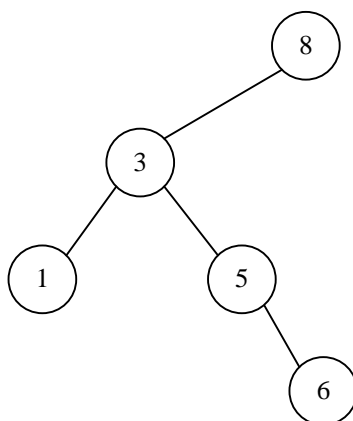


Рисунок 2 – Данное двоичное дерево

После вставки дерево имеет вид, приведённый на рисунке 3:

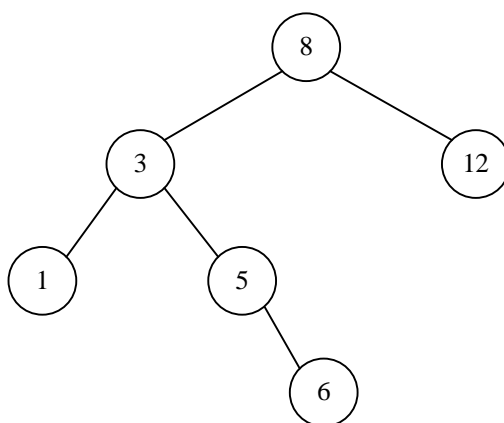
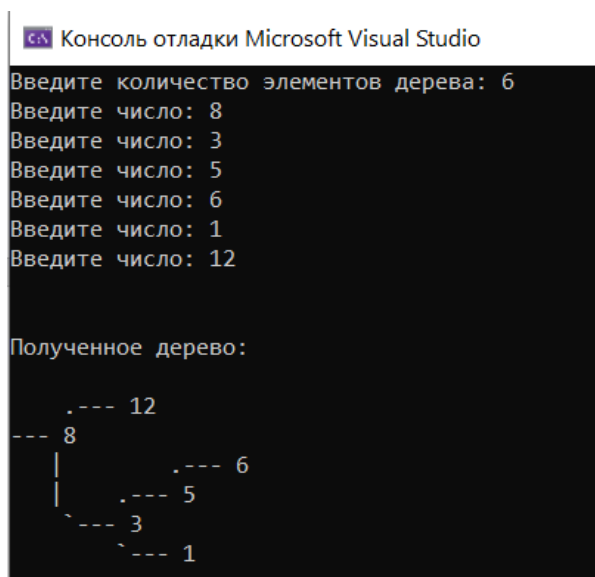


Рисунок 3 – Итоговое двоичное дерево

Результат работы программы (рисунок 4) для данного набора входных данных соответствует полученному теоретически.



```
Консоль отладки Microsoft Visual Studio
Введите количество элементов дерева: 6
Введите число: 8
Введите число: 3
Введите число: 5
Введите число: 6
Введите число: 1
Введите число: 12

Полученное дерево:

      .--- 12
     --- 8
    |
    | .--- 6
    | .--- 5
    | ^--- 3
    | ^--- 1
```

Рисунок 4 – Результат работы программы

ВЫВОДЫ

В ходе данной лабораторной работы был изучен принцип работы с древовидными структурами, в частности, с двоичными деревьями. Двоичные деревья – это структуры данных, состоящие из узлов, которые хранят значение, а также ссылку на свою левую и правую ветвь. Каждая ветвь, в свою очередь, является деревом. Для демонстрации полученных знаний была написана программа, позволяющая ввести дерево (добавляющая в дерево новые узлы), а также выводящая полученное дерево на экран. По результатам проверки программы можно сделать вывод о том, что она работает корректно.