МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**АДЫГЕЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

Инженерно-физический факультет
Кафедра автоматизированных систем обработки информации и управления

Отчет по практике
*Красно-черные деревъя*
2 курс, группа 2ИВТ2

Выполнил:
_____ И.И. Семякин
«___» _____ 2023г.
Руководитель:
_____ С.В. Теплоухов
«___» _____ 2023г.

Майкоп, 2023 г.

# Введение

1. Задание
2. Ход работы
   (a) Пример кода
   (b) Теория задачи
   (c) Изображение решения
3. Литература

## 1.Задание
Красно-черные деревья.

## 2.Ход работы
### а. Пример кода

```
 #include <iostream>
#include<iomanip>//для setw
#define RE "\x1B[31m"//для вывода цвета узлов в консоль
#define GRN "\x1B[32m"
#define RESET "\x1B[0m"


using namespace std;

//цвет узла
enum Color { RED, BLACK };


 //структура узла
struct Node {
    int key;
    Color color;
    Node* left;
    Node* right;
    Node* parent;

    Node(int k) : key(k), color(RED), left(nullptr), right(nullptr), parent(nullptr) {}
};
//левый поворот
void rotateLeft(Node*& root, Node* x) {
    Node* y = x->right;

    x->right = y->left;

    if (y->left != nullptr)
    {
        y->left->parent = x;
    }

    y->parent = x->parent;

    if (x->parent == nullptr)
    {
        root = y;
    }
    else if (x == x->parent->left)
    {
        x->parent->left = y;
    }
    else
```

```
    {
        x->parent->right = y;
    }

    y->left = x;
    x->parent = y;
}
//правый поворот
void rotateRight(Node*& root, Node* y) {
    Node* x = y->left;

    y->left = x->right;

    if (x->right != nullptr)
    {
        x->right->parent = y;
    }

    x->parent = y->parent;

    if (y->parent == nullptr)
    {
        root = x;
    }
    else if (y == y->parent->left)
    {
        y->parent->left = x;
    }
    else
    {
        y->parent->right = x;
    }

    x->right = y;
    y->parent = x;
}
//балансировка
void fixViolation(Node*& root, Node* z) {
    while (z->parent != nullptr && z->parent->color == RED) {
        if (z->parent == z->parent->parent->left) {
            Node* y = z->parent->parent->right;

            if (y != nullptr && y->color == RED) {
                z->parent->color = BLACK;
                y->color = BLACK;
                z->parent->parent->color = RED;
                z = z->parent->parent;
            }
            else {
                if (z == z->parent->right) {
```

```
                z = z->parent;
                rotateLeft(root, z);
            }

            z->parent->color = BLACK;
            z->parent->parent->color = RED;
            rotateRight(root, z->parent->parent);
        }
    }
    else {
        Node* y = z->parent->parent->left;

        if (y != nullptr && y->color == RED) {
            z->parent->color = BLACK;
            y->color = BLACK;
            z->parent->parent->color = RED;
            z = z->parent->parent;
        }
        else {
            if (z == z->parent->left) {
                z = z->parent;
                rotateRight(root, z);
            }

            z->parent->color = BLACK;
            z->parent->parent->color = RED;
            rotateLeft(root, z->parent->parent);
        }
    }
}

    root->color = BLACK;
}
//вставка числа
void insert(Node*& root, int key) {
    Node* z = new Node(key);//новый узел
    Node* y = nullptr;
    Node* x = root;

    while (x != nullptr) {
        y = x;

        if (z->key < x->key)
        {
            x = x->left;
        }
        else
        {
            x = x->right;
        }
```

```
    }

    z->parent = y;

    if (y == nullptr)
    {
        root = z;
    }
    else if (z->key < y->key)
    {
        y->left = z;
    }
    else
    {
        y->right = z;
    }

    fixViolation(root, z); // в конце вставки вызываем балансировку
}

//вывод красивого дерева
void postorder(Node* p, int indent)
{
    if (p != NULL) {
        if (p->right) {
            postorder(p->right, indent + 4);
        }
        if (indent) {
            cout << setw(indent) << ' ';
        }
        if (p->right) cout << " /\n" << setw(indent) << ' ';
        if (p->color==RED) {
            cout << RE << p->key << RESET << "\n ";
        }
        else {
            cout << GRN << p->key << RESET << "\n ";
        }

        if (p->left) {
            cout << setw(indent) << ' ' << " \\\n";
            postorder(p->left, indent + 4);
        }
    }
}

int main() {
    Node* root = nullptr;//корень
    setlocale(LC_ALL, "Russian");//вывод нормального русского языка
    int sum = 0,value;//колличество элементов и сам элемент
    while (1) {
```

```
        cout << "Введите число:" << endl;
        cin >> value;
        insert(root, value);
        sum++;
        system("cls");
        postorder(root, sum);
    }
    return 0;
}
```

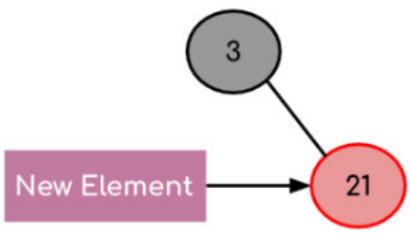
**b. Теория к задачи**

Красно-черное дерево - это бинарное дерево с следующими свойствами:

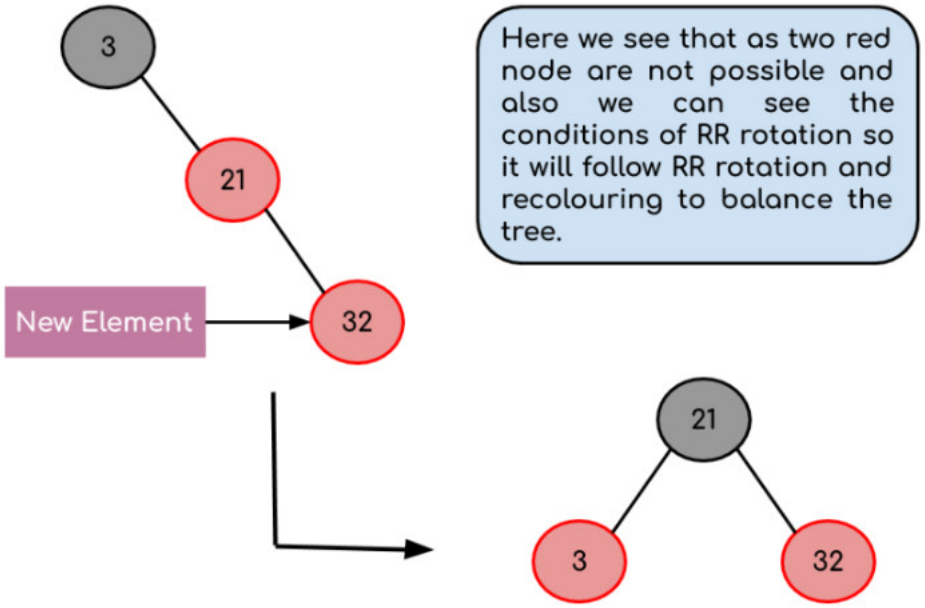
Каждый узел покрашен либо в черный, либо в красный цвет. Листьями объявляются NIL-узлы (т.е. "виртуальные"узлы, наследники узлов, которые обычно называют листьями; на них "указывают"NULL указатели). Листья покрашены в черный цвет. Если узел красный, то оба его потомка черны. На всех ветвях дерева, ведущих от его корня к листьям, число черных узлов одинаково. Количество черных узлов на ветви от корня до листа называется черной высотой дерева. Перечисленные свойства гарантируют, что самая длинная ветвь от корня к листу не более чем вдвое длиннее любой другой ветви от корня к листу. Чтобы понять, почему это так, рассмотрим дерево с черной высотой 2. Кратчайшее возможное расстояние от корня до листа равно двум - когда оба узла черные. Длиннейшее расстояние от корня до листа равно четырем - узлы при этом покрашены (от корня к листу) так: красный, черный, красный, черный. Сюда нельзя добавить черные узлы, поскольку при этом нарушится свойство 4, из которого вытекает корректность понятия черной высоты. Поскольку согласно свойству 3 у красных узлов непременно черные наследники, в подобной последовательности недопустимы и два красных узла подряд. Таким образом, длиннейший путь, который мы можем сконструировать, состоит из чередования красных и черных узлов, что и приводит нас к удвоенной длине пути, проходящего только через черные узлы. Все операции над деревом должны уметь работать с перечисленными свойствами. В частности, при вставке и удалении эти свойства должны сохраниться.

Когда вставляется первый элемент, он вставляется как корневой узел, а поскольку корневой узел имеет черный цвет, то он приобретает черный цвет.

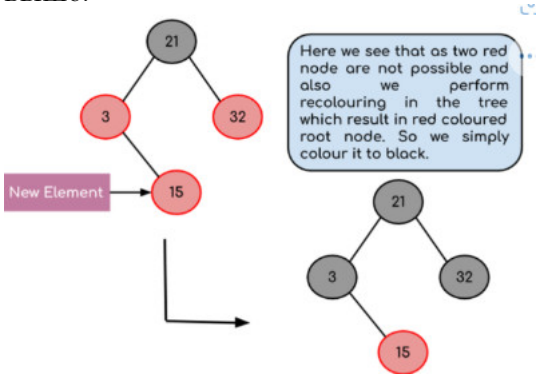

Новый элемент всегда вставляется красным цветом и как $21 > 3$, так что он становится частью правого поддерева корневого узла.



Теперь, когда мы вставляем 32, мы видим, что есть красная пара отец-потомок, которая нарушает правило красно-черного дерева, поэтому мы должны повернуть ее. Более того, мы видим условия поворота RR (рассматривая нулевой узел корневого узла как черный), поэтому после поворота корневой узел не может быть красным, поэтому мы должны выполнить перекраску в дереве, в результате чего получится дерево, показанное

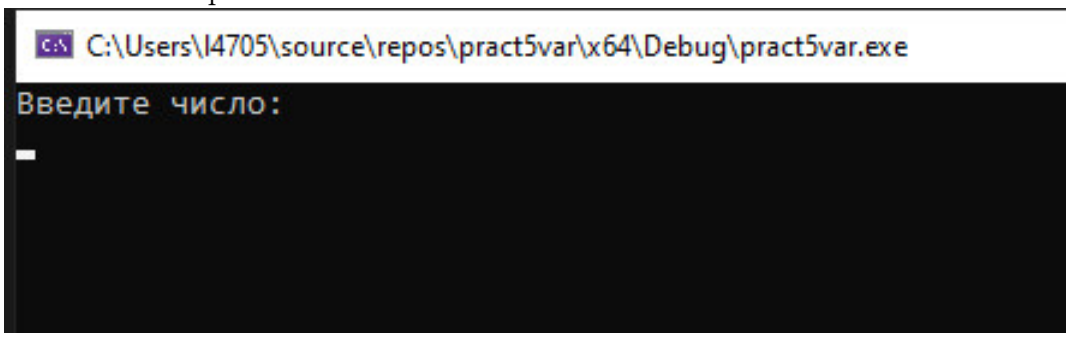



выше.

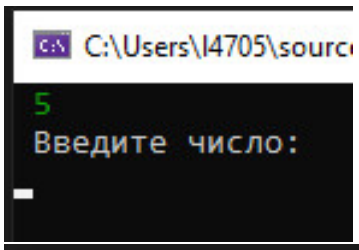


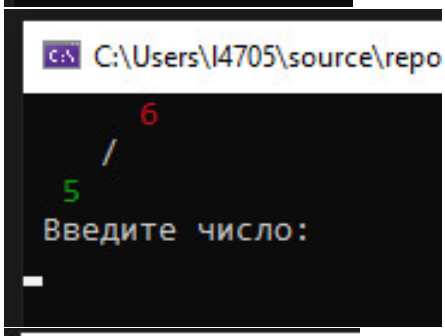

**c. Изображение решения**

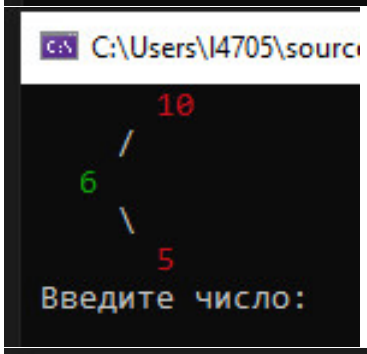
Начальный экран

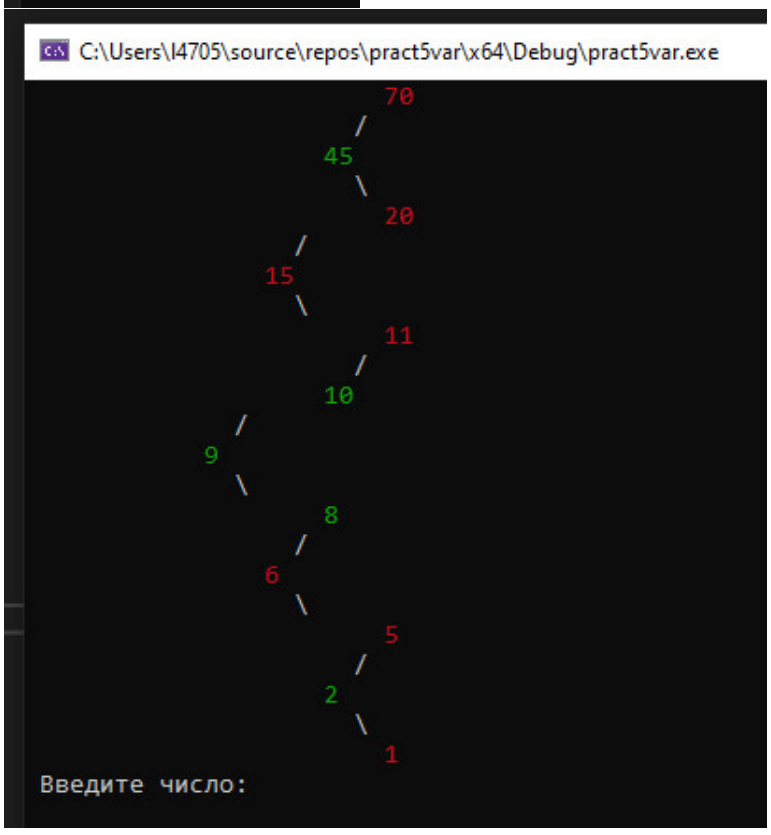



Ввод бесконечный, после каждого ввода происходит вывод дерева (зеленые числа-черные узлы, красные числа-красные узлы)


C:\Users\I4705\sourc
5
Введите число:


C:\Users\I4705\source\repo
6
/
5
Введите число:


C:\Users\I4705\sourc
10
/
6
\
5
Введите число:


C:\Users\I4705\source\repos\pract5var\x64\Debug\pract5var.exe
70
/
45
\
20
/
15
\
11
/
10
/
9
\
8
/
6
\
5
/
2
\
1
Введите число:

# Литература


1. https://habr.com/ru/articles/557328/
2. https://neerc.ifmo.ru/wiki/index.php?title=
3. https://algolist.manual.ru/ds/rbtree.php?ysclid=lifuizrex2220424657