**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР.**

**отчет**

**по лабораторной работе №4**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: «Двоичные деревья».**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1301 |  | Ищенко Д.О. |
| Преподаватель |  | Родионова Е. А. |

Санкт-Петербург

2022

Оглавление

[Формулировка задания. 3](#_Toc30565)

[Описание методов и оценка временной сложности. 3](#_Toc14052)

[Пример работы программы. 4](#_Toc31347)

[Текст программы. 6](#_Toc30347)

# Формулировка задания.

Реализовать структуру данных двоичное дерево поиска и следующие методы: нахождение минимума, нахождение максимума, прямой (preorder), центрированный (inorder) и обратный обход (postorder) по дереву, поиск элемента, нахождение следующего и предыдущего элемента, удаление элемента, обход в ширину.

Реализовать визуализацию дерева.

Указать теоретическую временную сложность для всех операций.

С помощью реализованной структуры данных написать программу, позволяющую преобразовать запись из префиксной/инфиксной/постфиксной нотации в префиксную/инфиксную/постфиксную нотацию.

# Описание методов и оценка временной сложности.

|  |  |  |  |
| --- | --- | --- | --- |
| Имя метода | Оценка сложности | | |
| Лучший случай | Средний случай | Худший |
| Нахождение минимума | O(1) | O(h), h - высота дерева | O(n) |
| Нахождение максимума | O(1) | O(h), h - высота дерева | O(n) |
| Прямой обход | O(n) | O(n) | O(n) |
| Центрированный обход | O(n) | O(n) | O(n) |
| Обратный обход | O(n) | O(n) | O(n) |
| Поиск элемента | O(1) | O(h) | O(h) |
| Нахождение следующего | O(1) | O(h) | O(h) |
| Нахождение предыдущего | O(1) | O(h) | O(h) |
| Удаление элемента | O(1) | O(h) | O(h) |
| Обход в ширину | O(n) | O(n) | O(n) |

# Пример работы программы.

**Использование api двоичного дерева поиска.**

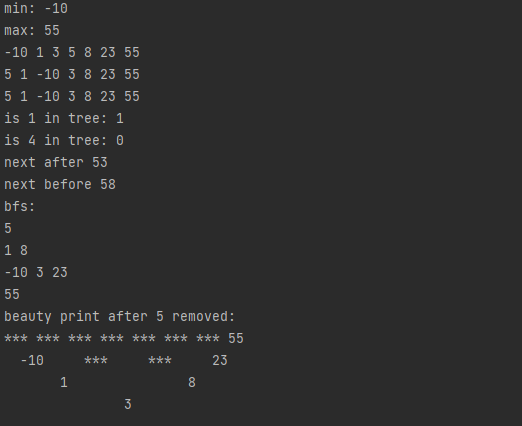
Код:

#include “MyBinaryTree.h”

#include <iostream>

int main() {  
 std::vector<int> v {5,1,8,3,-10,23,55};  
  
 BinarySearchTree<int> tree(v.begin(), v.end());  
 std::cout<<"min: "<<tree.getMin()<<"\nmax: "<<tree.getMax()<<"\n";  
  
 tree.inorderPrint();  
 tree.preorderPrint();  
 tree.preorderPrint();  
  
 std::cout << "is 1 in tree: " << bool(tree.contains(1)) <<"\n";  
 std::cout << "is 4 in tree: " << bool(tree.contains(4))<<"\n";  
  
 std::cout<<"next after 5"<<tree.findPreorderSuccessor(5)<<"\n";  
 std::cout<<"next before 5"<<tree.findInorderSuccessor(5)<<"\n";  
  
 std::cout<<"bfs:\n";  
 tree.bfsPrint();  
 tree.remove(5);  
 std::cout<<"beauty print after 5 removed:\n";  
 tree.beautyPrint();  
  
 return 0;  
}

Результат работы:



**Изменение нотации математического выражения.**

Код:

#include “MyBinaryTree.h”

#include <iostream>

Int main(){

MathExpression expr;

expr.parseExpr("-+/ABC\*DE", Prefix);

std::cout<<"Infix : ";

expr.printInfix();

std::cout<<"Prefix : ";

expr.printPrefix();

std::cout<<"Postfix: ";

expr.printPostfix();

std::cout<<"\ntree:\n";

expr.beautyPrint();

expr.parseExpr("KCDE\*\*AB---", Postfix);

std::cout<<"Infix : ";

expr.printInfix();

std::cout<<"Prefix : ";

expr.printPrefix();

std::cout<<"Postfix: ";

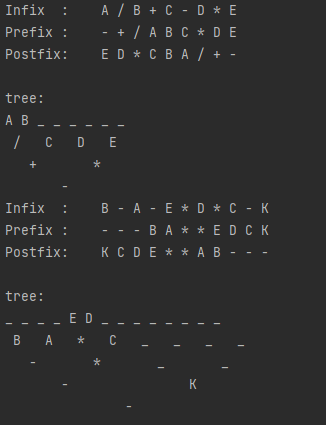
expr.printPostfix();

std::cout<<"\ntree:\n";

expr.beautyPrint();

}

Результат работы:



# Текст программы.

[Ссылка на github](https://github.com/Nekttuman/Algosee)