**Описание типа. Спецификация подпрограмм. Тесты**

Тестирование проводилось методом белого ящика.

class Key

{

public string Field1;

public string Field2;

public string Field3;

}

Тип данных, описывающий код направления. Состоит из трех строк: Field1, Field2, Field3 – по два элемента, каждый из которых – число от 0 до 9.

class Node

{

public int Data;

public bool Color; //red - false; black - true

public Node Left;

public Node Right;

public Node Parent;

public int Count;

}

Один узел дерева Tree. Состоит из полей: Data – целочисленное представление номера подгруппы, Color – цвет узла (true – черный, false – красный), Left – ссылка на левого потомка узла, Right – ссылка на правого потомка узла, Parent – ссылка на родителя узла, Count – количество добавленных узлов со значением Data.

class Tree

{

private Node \_nil;

private Node \_root;

}

Красно-черное дерево, состоящее из узлов Node. \_root – указатель на корень дерева, \_nil – узел, не содержащий кода направления в поле Data и указывающий сам на себя (в красно-черном дереве используется вместо ссылок на null)

public Node Root

Передает ссылку на корень \_root дерева Tree.

Входные данные: нет. Выходные данные: \_root.

public void Clean()

Удаляет все узлы дерева.

Входные данные: Tree. Выходные данные: \_nil.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Отчистка дерева |  | - |
| Отчистка пустого дерева | - | - |

public void Insert(string z)  
Входные данные: z, Tree. Выходные данные: Tree.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Добавление элемента в пустое дерево | -  00.00.01 |  |
| Добавление существующего числа | 01.00.05 |  |
| Дядя добавляемого элемента и его родитель красные | 00.00.05 |  |
| Дядя добавляемого элемента черный, удаляемый элемент является правым потомком красного узла | 01.00.04 |  |
| Дядя добавляемого элемента черный, удаляемый элемент является левым потомком красного узла | 00.02.03 |  |

public void Delete(string z)

Удаляет узел со значением z из дерева если или уменьшает значение count в этом узле, если оно больше 1. При необходимости заменяет удаляемый узел на минимальный справа.

Входные данные: z, Tree. Выходные данные: Tree, \_nil.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Удаление из пустого дерева | -  00.00.00 | - |
| Удаление единственного элемента | 00.00.01 | - |
| Удаление элемента с count больше 1 | 01.00.05 |  |
| Удаление корня дерева | 04.00.05 |  |
| Брат удаляемого узла красный | 09.00.05 |  |
| Брат удаляемого узла и его дети черные | 06.00.05 |  |
| Брат дочернего узла черный, его левый потомок красный, а правый - черный | 07.00.05 |  |
| Брат удаляемого элемента черный, его правый потомок красный | 02.00.05 |  |

public void Find(string k)

Находит узел со значением k и выводит в консоль False, если узел не найден и True, если найден.

Входные данные: k, Tree. Выходные данные: True, False.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Поиск узла | 02.00.05 | True |
| Поиск несуществующего узла | 00.00.00 | False |

public void PrintMinimum()

Находит узел min с минимальным значением в дереве Tree и выводит его в консоль.

Входные данные: Tree. Выходные данные: min.Data.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Поиск минимума |  |  |
| Поиск минимума в пустом дереве | - |  |

public void LRPrint()

Выводит узлы дерева Tree в порядке прямого обхода.

Входные данные: Tree. Выходные данные: Tree.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Обход дерева | - |  |
| Обход пустого дерева | - | - |

public void RLPrint()

Выводит узлы дерева Tree в порядке симметричного обхода.

Входные данные: Tree. Выходные данные: Tree.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Обход дерева | - |  |
| Обход пустого дерева | - | - |

public void NLRPrint()

Выводит узлы дерева Tree в порядке обратного обхода.

Входные данные: Tree. Выходные данные: Tree.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Обход дерева | - |  |
| Обход пустого дерева | - | - |

public void Draw()

Выводит дерево Tree в консоль.

Входные данные: Tree. Выходные данные: Tree, \_nil.

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Входные данные** | **Выходные данные** |
| Вывод дерева | - |  |
| Вывод пустого дерева | - | - |