**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЯ ОБРАЗОВАНИЯ**

**“ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет информационных технологий

Кафедра технологий программирования

**Алгоритмы: теория и практика. Структуры данных**

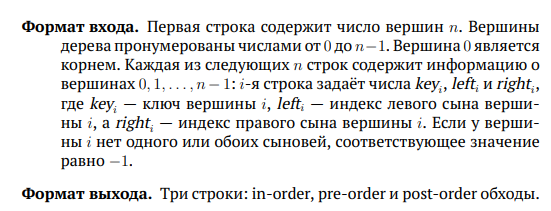
**Модуль №4**

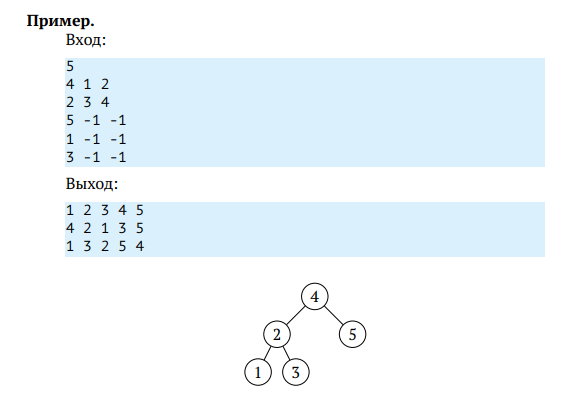
Выполнила студент 2 курса, группа 21-ИТ-1 Катушёнок И.В

Проверила Виноградова А.Д

Полоцк, 2022 г.

Обход двоичного дерева





Решение

# include <iostream>

# include <vector>

struct Node

{

int val;

int left;

int right;

};

class Tree

{

public:

Tree(int n)

{

\_tree.resize(n);

int val, left, right;

for (int i = 0; i < n; ++i)

{

std::cin >> val >> left >> right;

\_tree[i].val = val;

\_tree[i].left = left;

\_tree[i].right = right;

}

}

~Tree() { }

void printInOrder(int v)

{

if (v == -1)

return;

printInOrder(\_tree[v].left);

std::cout << \_tree[v].val << " ";

printInOrder(\_tree[v].right);

}

void printPreOrder(int v)

{

if (v == -1)

return;

std::cout << \_tree[v].val << " ";

printPreOrder(\_tree[v].left);

printPreOrder(\_tree[v].right);

}

void printPostOrder(int v)

{

if (v == -1)

return;

printPostOrder(\_tree[v].left);

printPostOrder(\_tree[v].right);

std::cout << \_tree[v].val << " ";

}

private:

std::vector<Node> \_tree;

};

int main(void)

{

int n;

std::cin >> n;

Tree tree(n);

tree.printInOrder(0);

std::cout << std::endl;

tree.printPreOrder(0);

std::cout << std::endl;

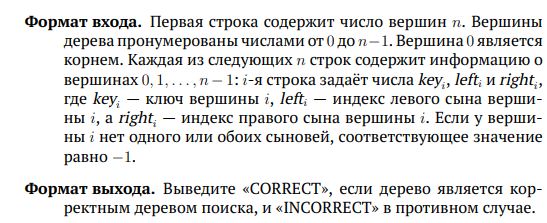
tree.printPostOrder(0);

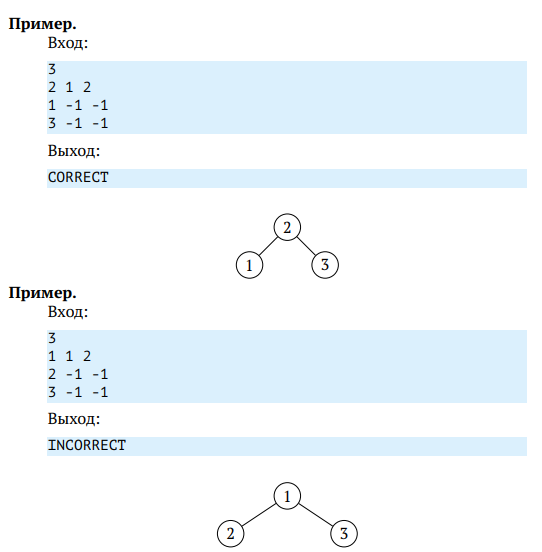
std::cout << std::endl;

return 0;

}

Проверка свойства дерева поиска





Решение

# include <iostream>

# include <vector>

struct Node

{

int val;

int left;

int right;

};

class Tree

{

public:

Tree(int n)

{

\_tree.resize(n);

int val, left, right;

for (int i = 0; i < n; ++i)

{

std::cin >> val >> left >> right;

\_tree[i].val = val;

\_tree[i].left = left;

\_tree[i].right = right;

}

}

~Tree() { }

void InOrder(int v)

{

if (v == -1)

return;

InOrder(\_tree[v].left);

\_output.push\_back(\_tree[v].val);

InOrder(\_tree[v].right);

}

bool check()

{

InOrder(0);

int prev = \_output[0];

for (int i = 1; i < \_output.size(); ++i)

{

int current = \_output[i];

if (!(current > prev))

return false;

prev = current;

}

return true;

}

private:

std::vector<Node> \_tree;

std::vector<int> \_output;

};

int main(void)

{

int n;

std::string res;

std::cin >> n;

if (!n)

{

std::cout << "CORRECT" << std::endl;

return 0;

}

Tree tree(n);

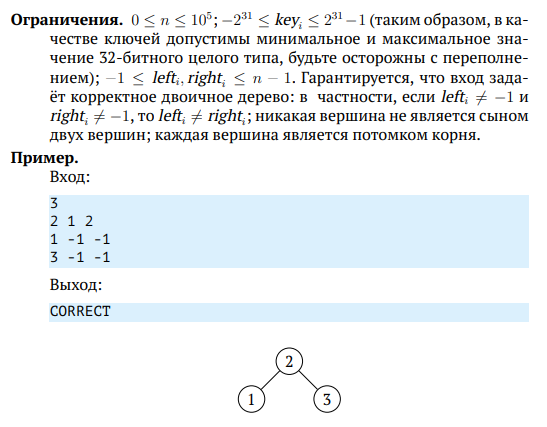
res = tree.check() ? "CORRECT" : "INCORRECT";

std::cout << res << std::endl;

return 0;

}

Проверка более общего свойства дерева поиска



Решение

#include <iostream>

#include <vector>

#include <climits>

struct Vertex

{

long key = -1;

int left = -1;

int right = -1;

};

typedef std::vector<Vertex> TreeV;

bool is\_bst(TreeV& T, int i = 0, long min = LONG\_MIN, long max = LONG\_MAX)

{

if (i == -1) return true;

if (T[i].key >= min && T[i].key < max)

return is\_bst(T, T[i].left, min, T[i].key)

&& is\_bst(T, T[i].right, T[i].key, max);

return false;

}

int main()

{

int n;

std::cin >> n;

TreeV T(n);

while (n)

{

std::cin >> T[T.size() - n].key

>> T[T.size() - n].left

>> T[T.size() - n].right;

n--;

}

if (!T.size()) std::cout << "CORRECT\n";

else std::cout << (is\_bst(T) ? "CORRECT" : "INCORRECT") << "\n";

return 0;

}