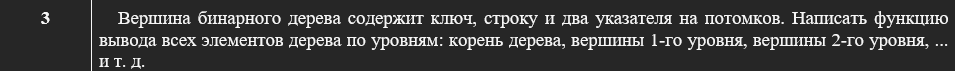
Лабораторная работа 11

По дисциплине «Основы алгоритмизации и программирования»

На тему «Бинарные деревья»

***Задание#1***



*Macros.h*

#include <iostream>

#include <vector>

#include <ctime>

#include <cmath>

#include <iomanip>

#include <stack>

#include <map>

#include <cstring>

#include <string>

#include <chrono>

#include <fstream>

#include <Windows.h>

#include <queue>

#define vi cout <<

#define vv cin >>

#define nl cout << "\n";

#define nw cout << "\t";

#define SCOCP1251 SetConsoleOutputCP(1251);

#define SCCP1251 SetConsoleCP(1251);

#define ret return

#define pause system("pause");

using namespace :: std;

*.cpp*

#include "Macros.h"

// Структура для представления узла бинарного дерева

struct Node {

int key; // Ключ типа int, данные типа string, указатели на левую и правую дочерние вершины

string data;

Node\* left;

Node\* right;

};

// Функция для создания нового узла дерева

Node\* createNode(int key, string data) { // Принимает ключ и данные

Node\* newNode = new Node(); // Инициализируем новый узел с этими значениями

newNode->key = key;

newNode->data = data;

newNode->left = nullptr;

newNode->right = nullptr;

ret newNode;

}

// Функция для вставки нового элемента в дерево

Node\* insert(Node\* root, int key, string data) { // Принимает указатель на корень дерева, ключ и данные

if (root == nullptr) {

ret createNode(key, data);

}

// Если корень пуст, то создается новый узел, иначе элемент вставляется в соответствии с значением ключа

if (key < root->key) {

root->left = insert(root->left, key, data);

}

else {

root->right = insert(root->right, key, data);

}

ret root;

}

// Функция вывода всех элементов дерева по уровням

void printLevelOrder(Node\* root) { // Принимает указатель на корень

if (root == nullptr) {

ret;

}

// Используем очередь для обхода узлов дерева по уровням и вывода их значений

queue<Node\*> q;

q.push(root);

while (!q.empty()) {

int size = q.size();

for (int i = 0; i < size; i++) {

Node\* curr = q.front();

q.pop();

vi curr->key << " " << curr->data << " ";

if (curr->left != nullptr) {

q.push(curr->left);

}

if (curr->right != nullptr) {

q.push(curr->right);

}

}

nl

}

}

int main() {

SCOCP1251

SCCP1251

Node\* root = nullptr; // Указатель на корень дерева

// Вставляем элементы в дерево

root = insert(root, 10, "A");

root = insert(root, 5, "B");

root = insert(root, 15, "C");

root = insert(root, 2, "D");

root = insert(root, 7, "E");

vi "Элементы дерева по уровням:"; nl

printLevelOrder(root); // Выводим элементы дерева по уровням

ret 0;

}

