Лабораторна робота 5(1)

Варіант 18

1)

#include <iostream>

using namespace std;

class Triad {

private:

int a, b, c;

public:

Triad(int a, int b, int c) {

this->a = a;

this->b = b;

this->c = c;

}

int compare(Triad other) {

if (this->a == other.a && this->b == other.b && this->c == other.c) {

return 0;

} else if (this->a > other.a || (this->a == other.a && this->b > other.b) || (this->a == other.a && this->b == other.b && this->c > other.c)) {

return 1;

} else {

return -1;

}

}

};

class Date : public Triad {

public:

Date(int year, int month, int day) : Triad(year, month, day) {}

int compare(Date other) {

return Triad::compare(other);

}

};

int main() {

Date d1(2022, 2, 24);

Date d2(2022, 2, 23);

int result = d1.compare(d2);

if (result == 0) {

cout << "d1 is equal to d2" << endl;

} else if (result > 0) {

cout << "d1 is greater than d2" << endl;

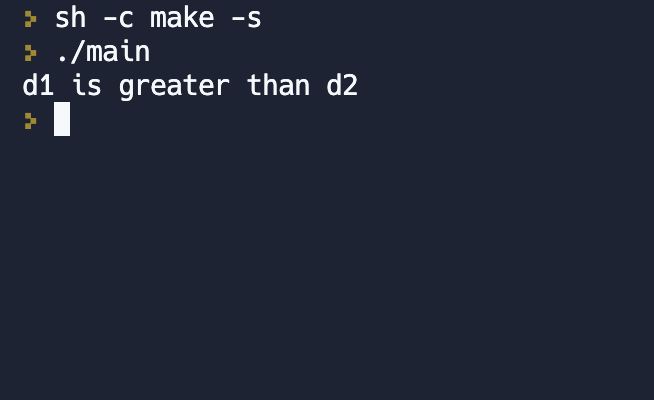
} else {

cout << "d1 is less than d2" << endl;

}

return 0;

}



2)

#include <iostream>

using namespace std;

class Node {

public:

int value;

Node\* left;

Node\* right;

Node(int value) {

this->value = value;

this->left = NULL;

this->right = NULL;

}

};

class BinaryTree {

public:

Node\* root;

BinaryTree() {

this->root = NULL;

}

void insert(int value) {

Node\* newNode = new Node(value);

if (this->root == NULL) {

this->root = newNode;

} else {

Node\* current = this->root;

while (true) {

if (value == current->value) {

cout << "Value already exists" << endl;

return;

} else if (value < current->value) {

if (current->left == NULL) {

current->left = newNode;

return;

} else {

current = current->left;

}

} else {

if (current->right == NULL) {

current->right = newNode;

return;

} else {

current = current->right;

}

}

}

}

}

void inorderTraversal(Node\* node) {

if (node == NULL) {

return;

}

inorderTraversal(node->left);

cout << node->value << " ";

inorderTraversal(node->right);

}

void preorderTraversal(Node\* node) {

if (node == NULL) {

return;

}

cout << node->value << " ";

preorderTraversal(node->left);

preorderTraversal(node->right);

}

void postorderTraversal(Node\* node) {

if (node == NULL) {

return;

}

postorderTraversal(node->left);

postorderTraversal(node->right);

cout << node->value << " ";

}

};

int main() {

BinaryTree tree;

tree.insert(5);

tree.insert(2);

tree.insert(8);

tree.insert(1);

tree.insert(3);

tree.insert(7);

tree.insert(9);

cout << "Inorder traversal: ";

tree.inorderTraversal(tree.root);

cout << endl;

cout << "Preorder traversal: ";

tree.preorderTraversal(tree.root);

cout << endl;

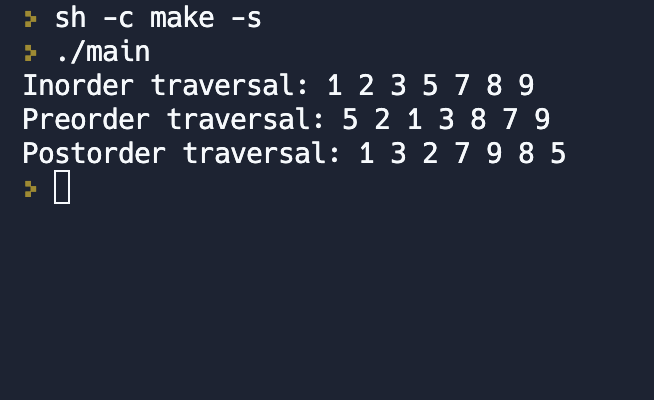
cout << "Postorder traversal: ";

tree.postorderTraversal(tree.root);

cout << endl;

return 0;

}



3)

#include <iostream>

#include <cstring>

using namespace std;

class String {

protected:

char\* data;

int length;

public:

String() {

length = 0;

data = new char[length + 1];

data[0] = '\0';

}

String(const char\* str) {

length = strlen(str);

data = new char[length + 1];

strcpy(data, str);

}

String(const String& str) {

length = str.length;

data = new char[length + 1];

strcpy(data, str.data);

}

virtual ~String() {

delete[] data;

}

virtual void print() {

cout << data << endl;

}

friend istream& operator>>(istream& is, String& str) {

char buffer[1000];

is >> buffer;

str.length = strlen(buffer);

str.data = new char[str.length + 1];

strcpy(str.data, buffer);

return is;

}

friend ostream& operator<<(ostream& os, String& str) {

os << str.data;

return os;

}

String& operator=(const String& str) {

if (this != &str) {

delete[] data;

length = str.length;

data = new char[length + 1];

strcpy(data, str.data);

}

return \*this;

}

};

class NumericString : public String {

public:

NumericString() : String() {}

NumericString(const char\* str) : String(str) {}

NumericString(const NumericString& str) : String(str) {}

virtual ~NumericString() {}

virtual void print() {

cout << data << " (numeric)" << endl;

}

friend istream& operator>>(istream& is, NumericString& str) {

char buffer[1000];

is >> buffer;

str.length = strlen(buffer);

str.data = new char[str.length + 1];

strcpy(str.data, buffer);

for (int i = 0; i < str.length; i++) {

if (!isdigit(str.data[i])) {

cerr << "Invalid input: " << buffer << " (not a numeric string)" << endl;

exit(1);

}

}

return is;

}

};

int main() {

String str1("Hello world");

String str2 = "Goodbye";

String str3;

str3 = str2;

String str4 = str3;

NumericString numstr1("1234");

NumericString numstr2;

cout << "Enter a numeric string: ";

cin >> numstr2;

cout << endl;

str1.print();

str2.print();

str3.print();

str4.print();

numstr1.print();

numstr2.print();

return 0;

}

