ლაბ 10: ძებნის ორობითი ხის კლასი

განსახილველი საკითხები:

- კლასის აღწერადა იმპელემენტაცია
- პროგრამა დრაივერი >>>
- სავარჯიშოები >>>

კლასის აღწერა და იმპლემენტაცია. ძებნის ორობითი ხის კლასის შექმნა და იმპლემენტაცია ალგორითმების თვალსაზრისით არაა რთული. ამ თვალსაზრისით, ჩვენი ამოცანა ბევრად ადვილია ვიდრე დაბალანსებული ხის კლასის შექმნა. სამაგიეროდ, იმპლემენტაციის თვალსაზრისით სირთულეები ამ და უფრო რთული ხის სტურტურებისთვის თითქმის ერთნაირია: ტემპლიტიანი კლასის შექმნა, იტერატორების კლასების აწყობა.

ჩვენ განვიხილავთ ძალიან მარტივ შემთხვევას: არ ვცდილობთ ტემპლიტიანი კლასის გაკეთებას და იტერატორის ნაცვლად ვაკეთებთ ხის წვეროს მისამართის ტიპს. დაახლოებით ასეთი მიდგომა გამოიყენებოდა C ენაში, თუმცა C-ში ტემპლიტიანი სტრუქტურის ნაცვლად გამოიყენებოდა ე.წ. `void` ტიპის პოინტერები, რომლითაც ძალიან ბევრი რამის გაკეთება შეიძლება ზოგადობის თვლასაზრისით. სამწუხაროდ, ამ ტექნიკით ასევე ბევრი რამის გაფუჭება შეიძლება, ამიტომ გახდა საჭირო სხვა გზების ძიება (მათ შორისაა ტემპლიტების გამოყენება).

სიმარტივისთვის, კლასის აღწერას და იმპლემენტაციას ვათავსებთ ერთ ფაილში სახელით “BST.h”:

```
#include<memory>
typedef struct tree_node* link;
struct tree_node
{
	link left;
	link right;
	link p;
	int data;
	tree_node() {}
	tree_node(int k)
	{
		data = k;
		left = right = p = nullptr;
	}
};

class BinarySearchTree
{
private:
	link root;
	int size;
	void inorder(link, void vf(link)) noexcept;
public:
	BinarySearchTree(): root(nullptr), size (0)  {}
	~BinarySearchTree();
	bool isEmpty() const noexcept { return root == nullptr; }
	void print_inorder() noexcept;
	link search(int k) noexcept;
	void insert(int) noexcept;
};

void BinarySearchTree::insert(int d) noexcept
{
	++size;
	//create new node with key = d
	link t = new tree_node(d);
```

```
	link y = nullptr;
	link x = root;
	// Find the Node's parent
	while (nullptr != x)
	{
		y = x;
		if (t->data > x->data) x = x->right;
		else x = x->left;
	}
	t->p = y;
	if (nullptr == y)
	{
		root = t;
		return;
	}
	if (d < y->data)
		y->left = t;
	else
		y->right = t;
}

void BinarySearchTree::print_inorder() noexcept
{
	inorder(root, [](tree_node* p) {std::cout << " " << p->data << " "; });
}
BinarySearchTree::~BinarySearchTree()
{
	inorder(root, [](link p) { delete p; });
	root = nullptr;
	size = 0;
}
void BinarySearchTree::inorder(link p, void vf(link)) noexcept
{
	if (p != nullptr)
	{
		inorder(p->left, vf);
		tree_node* tmp(p->right);
		vf(p);
		inorder(tmp, vf);
	}
}
link BinarySearchTree::search(int k) noexcept
{
	link x = root;
	while (nullptr != x && x->data != k)
	{
		if (k < x->data)
			x = x->left;
		else
			x = x->right;
	}
	return x;
}
```

<<< პროგრამა დრაივერი

```
#include<iostream>
#include"BST.h"

using namespace std;
```

```
int main()
{
	BinarySearchTree b;
	int ch, tmp, tmp1;
	link node_address;
	while (1)
	{
		cout << endl << endl;
		cout << " Binary Search Tree Operations " << endl;
		cout << " ---------------------------- " << endl;
		cout << " 1. Insertion/Creation " << endl;
		cout << " 2. In-Order Traversal " << endl;
		cout << " 3. search " << endl;
		cout << " 4. Exit " << endl;
		cout << " Enter your choice : ";
		cin >> ch;
		switch (ch)
		{
		case 1:
			cout << " Enter Number to be inserted : ";
			cin >> tmp;
			b.insert(tmp);
			break;
		case 2:
			cout << endl;
			cout << " In-Order Traversal " << endl;
			cout << " --------------------" << endl;
			b.print_inorder();
			break;
		case 3:
			cout << " Enter Number to be searched : ";
			cin >> tmp;
			node_address = b.search(tmp);
			if ( node_address != NULL )
				cout << "Found node with data " << tmp << endl;
			else
				cout << "Not found..." << endl;
			break;
		case 4:
			return 0;
			break;
		}
	}
}
```

<<< სავარჯიშოები

1. კლასის აღწერაში და იმპლემენტაციაში დაამატეთ ფუნქციები ხეში მინიმალური და მაქსიმალური გასაღებების შემცველი წვეროების მისამართის მოძებნისთვის. პროგრამა დრაივერის მენიუში დაამატეთ შესაბამისი ოპერაციების გათვალსაჩინოების შესაძლებლობა.
2. კლასის აღწერაში და იმპლემენტაციაში დაამატეთ ხეში მოცემული კვანძის წინა და მომდევნო (გასაღებების სიდიდის თვალსაზრისით) წვეროების მისამართის განსაზღვრის ფუნქციები, ხოლო პროგრამა დრაივერის მენიუში დაამატეთ შესაბამისი ოპერაციების გათვალსაჩინოების შესაძლებლობა.