­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­



Лабораторна робота

з курсу «**МЕТОДИ ПРОЕКТУВАННЯ**»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 11

Виконав студент гр. КНз-41

Чалий Михaйло

Перевірив

Каркульовський В.І.

­­

Львів 2016

## Мета роботи

Ознайомитись з особливостями реалізаціі методології об’єктно-орієнтованого проектування для вирішення прикладних задач в середовищі Windows на Visual C++.

## Теоретичні відомості

## Завдання

*ВАРІАНТ № 11*

Використовуючи методологію об’єктно-орієнтованого програмування, скласти програму на Visual С++ для вирішення задачі роботи з бінарним деревом (формування дерева, виведення елементів дерева, видалення заданих вершин з дерева).

## Реалізація

Лістінг 1. l.rs

struct Node {

key: i32,

left: Option<Box<Node>>,

right: Option<Box<Node>>

}

fn dump(node: &Node){

print!("{}", node.key);

print!("; left: ");

match node.left {

Some(ref left) => print!("{}", left.key),

None => print!("NONE")

}

print!("; right: ");

match node.right {

Some(ref right) => print!("{}", right.key),

None => print!("NONE")

}

println!("");

}

fn walk(node: &Node, cb: &Fn(&Node)){

cb(node);

match node.left {

Some(ref left) => {

walk(&left, cb);

},

None => ()

}

match node.right {

Some(ref right) => {

walk(&right, cb);

},

None => ()

}

}

fn insert(node: &mut Node, key: i32) {

if key < node.key {

match node.left {

Some(ref mut left) => insert(&mut \*left, key),

None => node.left = Some(Box::new(Node { key: key, left: None, right: None })),

}

} else {

match node.right {

Some(ref mut right) => insert(&mut \*right, key),

None => node.right = Some(Box::new(Node { key: key, left: None, right: None })),

}

}

}

fn walk\_mut(node: &mut Node, cb: &Fn(&Node, &Node) -> Option<Box<Node>>){

match node.left {

Some(ref mut left) => {

node.left = cb(node, &\*left);

walk\_mut(&mut \*left, cb);

},

None => ()

}

match node.right {

Some(ref mut right) => {

node.right = cb(node, &\*right);

walk\_mut(&mut \*right, cb);

},

None => ()

}

}

fn drop(root: &mut Node, key: i32) {

// walk\_mut(root, &|parent: &Node, node: &Node| -> Option<Box<Node>> {

// if node.key == key {

// match (&node.left, &node.right) {

// (&None, &None) => {

// println!("LEAF: {}", node.key);

// return None;

// },

// \_ => {

// println!("TBD: {}", node.key);

// return Some(Box::new(\*node));

// }

// }

// } else {

// return Some(Box::new(\*node));

// }

// });

match node.left {

Some(ref mut left) => {

node.left = cb(node, &\*left);

walk\_mut(&mut \*left, cb);

},

None => ()

}

match node.right {

Some(ref mut right) => {

node.right = cb(node, &\*right);

walk\_mut(&mut \*right, cb);

},

None => ()

}

}

fn main() {

// It is funny, but I was not able to satify rust borrow

// to implement this stuff in classic recusive way

let mut root = Node { key: 100, left: None, right: None };

insert(&mut root, 60);

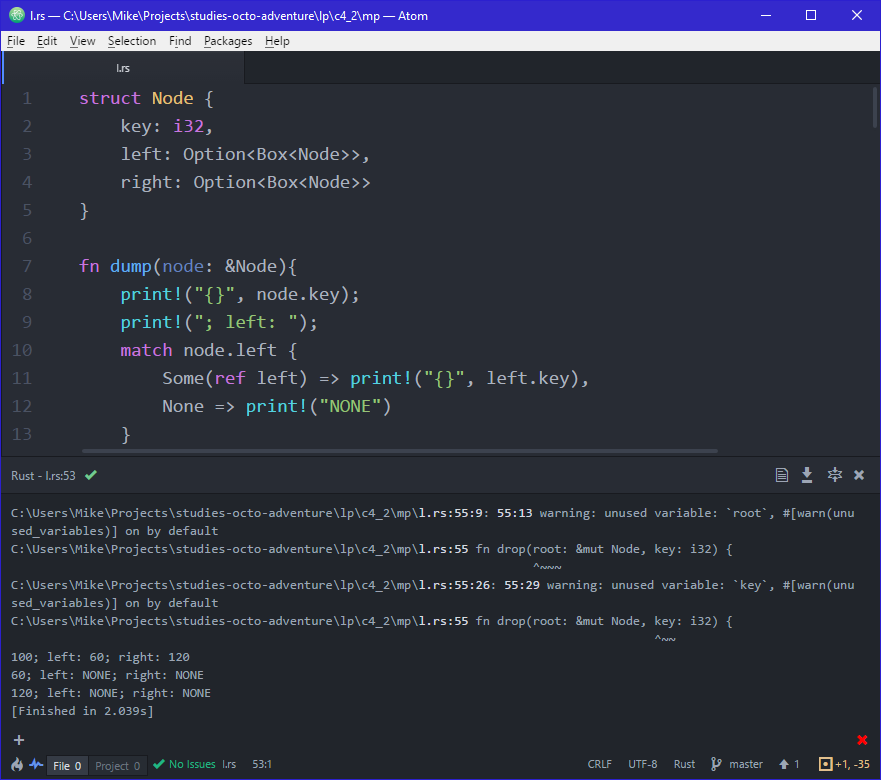
insert(&mut root, 120);

walk(&root, &dump);

drop(&mut root, 60);

}

## Результат



## Висновки

Ознайомився з особливостями реалізаціі методології об’єктно-орієнтованого проектування для вирішення прикладних задач в середовищі Windows на Visual C++.