Національний технічний університет України «КПІ ім. І. Сікорського»

Факультет інформатики і обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

**Лабораторна робота №4**

**з дисципліни**

**«ОРПЗ .NET»**

на тему

# «Структурні шаблони»

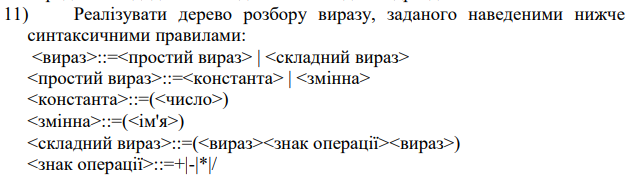
Виконав

студент групи ІС-03

Ніколенко Гліб

Київ – 2022

Варіант 11:



Я використав шаблон Компонувальник, оскільки він ідеально підходить під мою задачу – створити дерево.

Код програми:

using System;

using System.Collections.Generic;

namespace lab4

{

class Program

{

public static Component BuildTree(string exp)

{

var root = new Expression();

Component currentNode = root;

var currentToken = GetToken(exp);

while (currentToken != null)

{

exp = exp.Substring(currentToken.Length);

switch (currentToken)

{

case "(":

currentNode.Left = new Expression();

currentNode.Left.Parent = currentNode;

currentNode = currentNode.Left;

break;

case "+":

case "-":

case "\*":

case "/":

if (!string.IsNullOrEmpty(currentNode.Exp))

{

root = new Expression();

root.Left = currentNode;

currentNode.Parent = root;

currentNode = root;

}

currentNode.Exp = currentToken;

currentNode.Right = new Expression();

currentNode.Right.Parent = currentNode;

currentNode = currentNode.Right;

break;

case ")":

if (currentNode.Parent == null && exp.Length != 0)

{

root = new Expression();

root.Left = currentNode;

currentNode.Parent = root;

}

currentNode = currentNode.Parent;

break;

default:

currentNode.Exp = currentToken;

currentNode = currentNode.Parent;

break;

}

currentToken = GetToken(exp);

}

return root;

}

public static void PrintTree(Component root)

{

root.Print();

if (root.Left != null)

{

Console.Write("Лівий: ");

PrintTree(root.Left);

}

if (root.Right != null)

{

Console.Write("Правий: ");

PrintTree(root.Right);

}

}

public static string GetToken(string exp)

{

if (string.IsNullOrEmpty(exp))

return null;

string token = string.Empty;

foreach (var c in exp)

{

if (c == '(' || c == ')' || c == '+' || c == '-' || c == '\*' || c == '/')

{

if (token.Length != 0)

break;

token += c;

break;

}

token += c;

}

return token;

}

static void Main(string[] args)

{

Console.WriteLine("Write down the expression");

string exp = Console.ReadLine();

var root = BuildTree(exp);

PrintTree(root);

}

}

abstract class Component

{

public string Exp { get; set; }

public Component Parent { get; set; }

public Component Left { get; set; }

public Component Right { get; set; }

public Component()

{

}

public virtual void Print()

{

Console.WriteLine("Вузол " + Exp);

if (Left == null && Right == null)

Console.WriteLine(" Підвузли відсутні");

}

}

class Expression : Component

{

public Expression() : base()

{

}

}

}

Програма працює наступним чином:

1. Користувач вводить якийсь вираз.
2. Програма проходить по кожному символу данного виразу і створює дерева, за такими правилами:

1.Якщо символ : (, то сворюється новий вузол зліва від вузла, в якому стоїть програма зараз

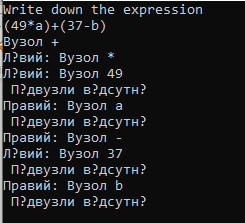
2.Якщо символи +/\*-, то створюється вузол справа, а в нинішній вузол записується знак операції

3.Якщо символ ), то перевіряємо чи є родовий вузол в того, в якому ми зараз стоїмо, та чи нема інших символи після «)». В позитивному випадку ми обзиваємо цей вузол коренем.

4. У випадку, якщо символ – це будь-яка цифра, або змінна, то просто записуємо значення у вузол, в якому зараз знаходимось і пересуваємось в попередній вузол, який є родовим для данного.

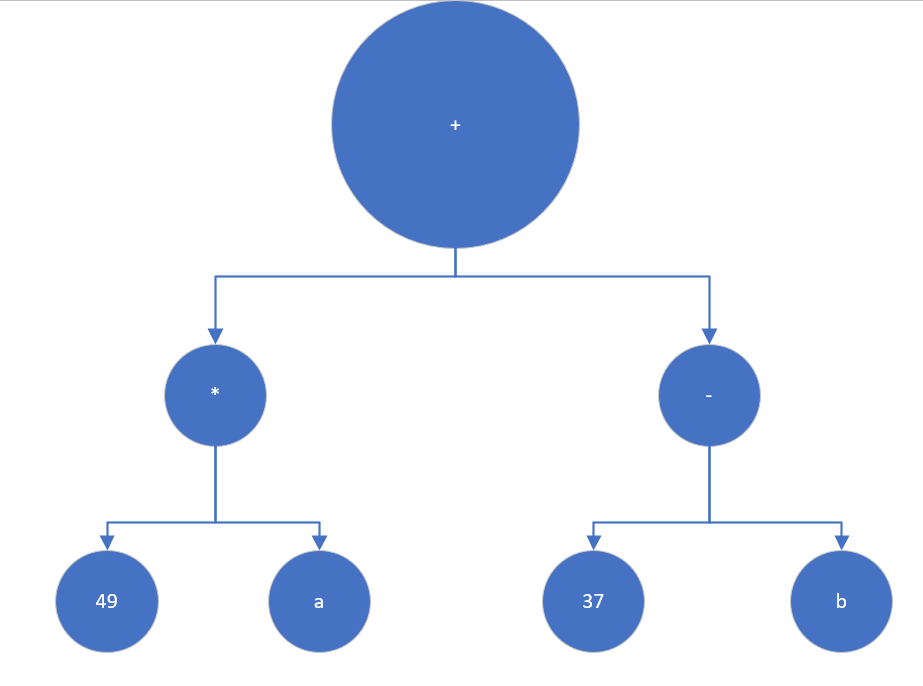
3)Виводимо дерева в консоль. Вивід працює таким чином, що спочатку показує корінь, а потім всі вузли, починаючя з самого лівого.

Приклад:

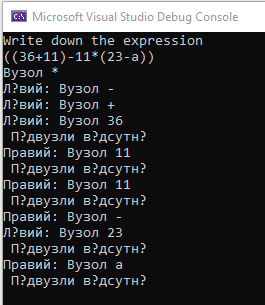


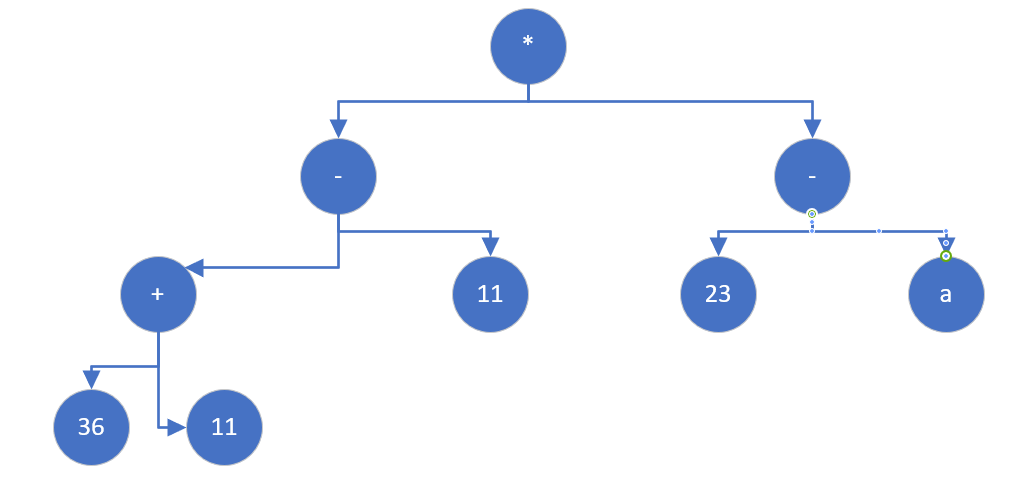
Результат виводу дерева для виразу: (49\*a)+(37-b).

З даних, які ми отримали, можна графічно зобразити відповідь так:



Результат виводу дерева для виразу: ((36+11)-11\*(23-a)).





Висновок: Після виконання цієї лабораторної роботи, я ознайомився зі структурними паттернами і на практиці дізнався що таке шаблон «Компонувальник».

Посилання на github: