Интерпретатор

1. Назначение и цель: определяет представление грамматики для заданного языка и интерпретатор предложений этого языка. Как правило, данный шаблон проектирования применяется для часто повторяющихся операций
2. UML-диаграмма



1. Области применения: следует использовать когда вам необходимо интерпретировать запись в другом языке и тд. Как один из примеров может служить перевод римских цифр в арабские
2. Особенности: большинство интерпретаторов действует так, что исполняет исходную программу последовательно, по мере ее поступления на вход интерпретатора, следовательно не может выполнять оптимизацию исходной программы. В остальном же структура интерпретатора будет мало отличаться от структуры аналогичного компилятора
3. Примеры реализации

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Zadanie1

{

class MainApp

{

static void Main()

{

string tape = "00110";

string code = ">>>>\*>";

//string code = "]>>>\*>";

Context context = new Context(tape,code);

List<Expression> tree = new List<Expression>

{

new Smallfuck(),

};

//Interpreter

foreach (Expression exp in tree)

{

exp.Interpret(context);

}

Console.WriteLine("Input tape: {0} --> Output tape: {1}", tape, context.Output);

// Wait for user

Console.ReadKey();

}

}

class Context

{

// Constructor

public Context(string input, string code)

{

Input = input;

Code = code;

Tape = new char[input.Length];

for (int i=0; i<Tape.Length;i++)

{

Tape[i] = input[i];

}

}

public string Input { get; set; }

public string Code { get; set; }

public char[] Tape { get; set; }

public string Output { get; set; }

}

abstract class Expression

{

public void Interpret(Context context)

{

if (context.Input.Length == 0)

return;

for (int i=0, code=0;code<context.Code.Length;code++)

{

if (i == context.Tape.Length) break;

if (context.Code[code] =='>')

{

i++;

}

if (context.Code[code] == '<')

{

i--;

}

if (context.Code[code] == '\*')

{

if (context.Tape[i] == '1') context.Tape[i] = '0';

else context.Tape[i] = '1';

}

if ((context.Code[code] == '[')&&(context.Tape[i]=='0'))

{

for (int j=code+1;j<context.Code.Length;j++)

{

if (context.Code[j] == ']')

{

code = j;

j = context.Code.Length;

}

else if (j == context.Code.Length-1) Console.WriteLine("] is missing");

}

}

if ((context.Code[code] == ']') && (context.Tape[i] == '1'))

{

for (int j = code - 1; j <= 0; j--)

{

if (context.Code[j] == '[')

{

code = j;

j = 0;

}

else if (j == 0) Console.WriteLine("[ is missing");

}

}

}

for (int i = 0; i < context.Tape.Length; i++)

{

context.Output += context.Tape[i];

}

}

}

class Smallfuck : Expression

{

}

}