Phụ lục I

1. Code tạo bộ luật phân tích và gọi tới các hàm khác để thực thi thuật toán CYK, đồng thời hiện thị các bước ra màn hình( cụ thể add vào các button)

PTCYK parser = new PTCYK();

List<Rule> rules = new List<Rule>();

// nạp bộ luật vào bộ phân tích

parser.AddRule('S', "AB");

parser.AddRule('S', "XB");

parser.AddRule('T', "AB");

parser.AddRule('T', "XB");

parser.AddRule('X', "AT");

parser.AddRule('A', "a");

parser.AddRule('B', "b");

rules = parser.PrintAllRules();

//gán bộ luật vào lbllist

string a = "";

foreach (Rule r in rules)

{

a += r.ToFineString() + "\n";

}

lbl\_list.Text = a;

//

//thực hiện gán giá trị vào button và show ra pnl\_Main

parser.Process(txt\_input.Text.ToString());

table = parser.PrintResult();

timer.Start();

//kiểm tra nếu đã chạy hết thì dừng timer, xóa text nhập và enable btnRefresh

if (kk == txt\_input.Text.Length+1)

{

timer.Stop();

button1.Enabled = true;

MessageBox.Show("CYK done!","Infomation");

txt\_input.Text = "";

kk = 0;

}

1. Class thực thi thuật toán CYK

class PTCYK

{

public List<Rule> rules = new List<Rule>();

string[,] table = null;

string w = null;

int n = 0;

// thêm luật left --> right vào tập luật

public void AddRule(char left, string right)

{

rules.Add(new Rule(left, right));

}

public List<Rule> PrintAllRules()

{

return rules;

//Console.WriteLine("<bo luat van pham>");

//foreach (Rule r in rules)

// Console.WriteLine(" " + r.ToFineString());

//Console.WriteLine();

}

/// <summary>

///

/// </summary>

/// <param name="x"></param>

void InitData(string x)

{

w = x;

n = x.Length;

table = new string[n + 1, n + 1];

for (int i = 1; i <= n; i++)

for (int j = 1; j <= n; j++) table[i, j] = "";

}

/// <summary>

///

/// </summary>

void FirstProcess()

{

for (int i = 1; i <= n; i++)

foreach (Rule r in rules)

if (r.right.Length == 1)

if (w[i - 1] == r.right[0]) table[i, 1] += r.left;

}

/// <summary>

///

/// </summary>

/// <param name="x"></param>

/// <param name="w"></param>

/// <returns></returns>

bool IsIn(char x, string w)

{

return w.IndexOf(x) != -1;

}

/// <summary>

///

/// </summary>

/// <param name="j"></param>

/// <param name="i"></param>

/// <param name="r"></param>

/// <returns></returns>

bool CanGen(int j, int i, Rule r)

{

for (int k = 1; k <= j - 1; k++)

if (IsIn(r.right[0], table[i, k]))

if (IsIn(r.right[1], table[i + k, j - k]))

return true;

return false;

}

void GenTable()

{

for (int j = 2; j <= n; j++)

for (int i = 1; i <= n - j + 1; i++)

foreach (Rule r in rules)

if (r.right.Length == 2)

if (!IsIn(r.left, table[i, j]))

if (CanGen(j, i, r))

{

table[i, j] += r.left;

// Createbutton(90, 90, w, table);

}

}

/// <summary>

/// run CYK :3

/// </summary>

/// <param name="x"></param>

public void Process(string x)

{

InitData(x);

FirstProcess();

GenTable();

}

//hiện thị ra ký tự test các bước

public string[,] PrintResult()

{

for (int i = 1; i <= n; i++)

{

table[0, i] = i.ToString();

table[i, 0] = w[i - 1].ToString();

}

return table;

}

}