รายงาน

โครงการ สร้างคอมไพล์เลอร์

โดย

นายคณุตม์ บุญเรืองขาว

รหัสนิสิต 5870390321

เพื่อนำเสนอ

ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา

รายงานนี้เป็นส่วนหนึ่งของวิชา 2110714 Digital System

ประจำภาคเรียนที่ 1 ปีการศึกษา 2559

Project: Instruction

I define the following language which is easy to compile.

(do s1 s2 ...)	sequential s1 s2 ...
(= a b)	assignment a = b
(== a b)	equality test a == b
(ifelse cond T-action F-action)	if..then..else
(def fun-name formal body)	function definition
(print ex)	print integer to screen

with this simple language I can write a program like this:

```
(def fac n
   (ifelse (== n 0)
      1
      (* n (fac (- n 1)))))
(def main ()
   (print (fac 6)))
```

Write a compiler for this language. Your work is in three parts:

1) specification: Write regular expression for words in this language. Write CFG of this language.
2) parser: Write scanner and parser using Recursive Descent method.
3) code generation: Write a code generator to translate Parse Tree into S-code.

You can extend or change this minimal language to include interesting construct (such as for..loop, or array). You can define your own machine code (you don't have to use S-code).

Write your compiler and your report of the work. The report should describe all three parts with adequate explanation so that I can follow how you design your compiler. You should also give some example of input/output of your system. Do not include the source code of your compiler in the report. Expect the report to be read by an undergraduate student in computer engineering. Email your code to me so I can inspect it. I expect you to spend in average 20-30 man-hours for this project.

Part 1) <u>Specification :</u>

<u>Context-free Grammar</u>

L = (def fun-name formal-body)L | ƛ

def = def

fun-name = name param

formal-body = (if-else cond what) | command | ƛ

<u>Regular expression</u>

name = [a-z]

param = a|b|c|n|()

if-else = ifelse

cond = (sym1 param num)

what = num command | ƛ

command = (print func2)| assignment1 | assignment2 | ƛ

assignment2 = (sym2 param func2) | (sym2 param (fun-name2 assignment1))

func2 = (fun-name2(num))

sym1 = >=|<=|==

sym2 = +|-|*|/

fun-name2 = name

num = [1-10]

assignment1 = (sym2 param num)

Part 2) <u>Lexical Analysis (Scanner) &Parser:</u>

การสร้าง Scanner เพื่อแยกตัวอักษรออกจากแฟ้มมาประกอบเป็น token ของภาษา

1.สร้าง ArrayList เพื่อใช้ในการเก็บค่าที่ได้จากแฟ้มทั้งหมดแล้วนำมาเก็บลงใน ArrayList

```
(def fac n
    (ifelse (== n 1)
       1
       (* n (fac (+ n 5)))))
(def test a
    (+ a 3))
(def tet b
    (- b (test (+ b 1))))
(def tex c
    (/ c (test (10))))
(def main ()
    (print (fac (6))))
```

รูปที่ 2.1 แสดงข้อมูลในแฟ้ม

เมื่อทำคำสั่งอ่านข้อมูลจากไฟล์แล้วทั้งหมดจะถูกนำมาเก็บใน ArrayList โดยใช้เครื่องหมาย "[{};]" เป็นตัวอ้างอิงในการตัดคำ จะได้ข้อมูลใน ArrayList ดังข้อมูลด้านล่าง

```
(def, fac, n, , , , , (ifelse, (==, n, 1), , , , , , , , 1, , , , , , , , (*, n,
(fac, (+, n, 5))))), (def, test, a, , , , , (+, a, 3)), (def, tet, b, , (-, b,
(test, (+, b, 1)))), (def, tex, c, , (/, c, (test, (10)))), (def, main, (), , , ,
, (print, (fac, (6))))
```

ข้อมูลเหล่านี้จะสังเกตว่ายังไม่สามารถทำเป็นโทเค็นได้เนื่องจากประกอบด้วยช่องว่างมากมายทำให้จำเป็นต้องมีการกรองตัวที่ยังผิดกฎการเป็นโทเค็นออกไป

2.กรองตัวที่เป็นช่องว่างและตัวที่เป็นวงเล็บออกเพื่อให้ตรงตาม CFG

จะได้ ArrayList ที่ผ่านการกรองแล้วดังรูปข้างล่าง

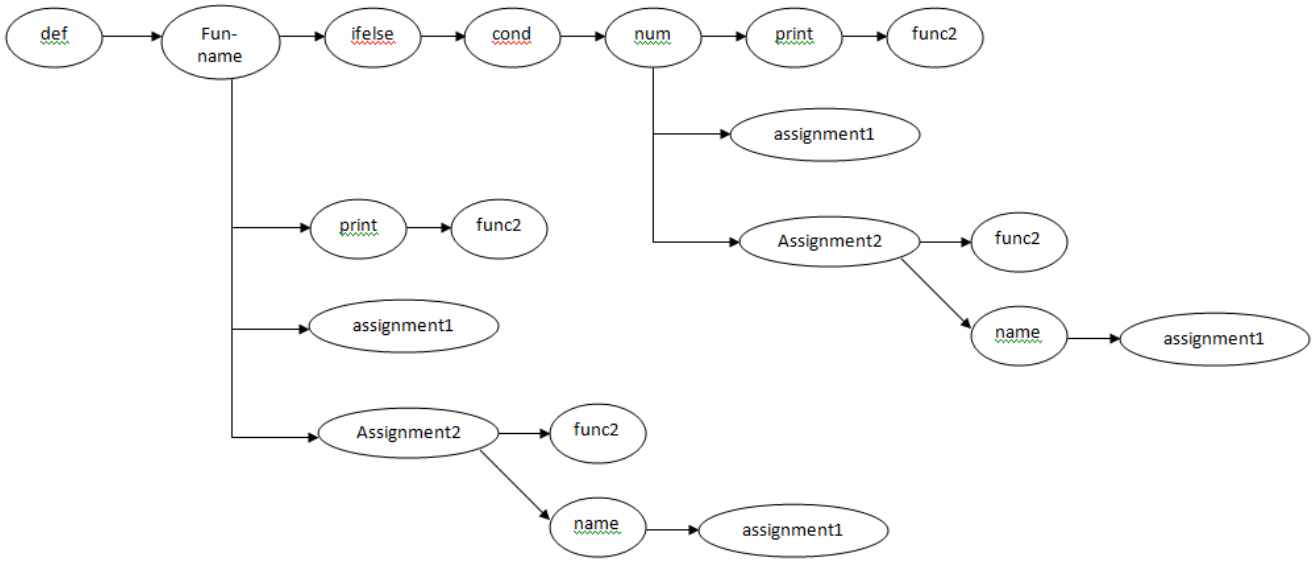
```
(, def, fac, n, (, ifelse, (, ==, n, 1, ), 1, (, *, n, (, fac, (, +, n, 5, ), ),
), ), ), (, def, test, a, (, +, a, 3, ), ), (, def, tet, b, (, -, b, (, test, (,
+, b, 1, ), ), ), ), (, def, tex, c, (, /, c, (, test, (, 10, ), ), ), ), (, def,
main, (, ), (, print, (, fac, (, 6, ), ), ), )
```

3.นำ ArrayList ที่ผ่านการกรองแล้วมาตรวจสอบว่าถูกต้องตาม CFG หรือไม่ โดยตรวจสอบตามกฎของ CFG โดยเมทอด lang เมื่อ ArrayList ถูกส่งเข้ามาจะถูกตรวจสอบว่าถูกต้องหรือไม่ และหากว่าถูกต้อง โปรแกรมจะแสดงข้อความว่า Matched และเก็บข้อมูลโดยแยกประเภทโดยจำแนกเป็นเบอร์ต่างๆตามตารางที่ 2.1 การเก็บโทเค็นตามประเภททำโดยการสร้างคลาส tokennum ที่เก็บตัวแปร String และ int เพื่อเก็บข้อมูลที่ประกอบด้วยข้อความและตัวเลข

ประเภท	เบอร์
def	10
funcname	20
ifelse	30
print	41
assignment1	51
assignment2.1	52
assignment2.2	53

ตารางที่ 2.1 แสดงการจำแนกประเภทของโทเค็น

3.1 Flow ของการตรวจ CFG



รูปที่ 2.2 แสดงผังการตรวจสอบ CFG

```
public void lang(ArrayList<String> text)
{
	if(count<text.size())
	{
		if(!text.get(count).equals(""))
		{
			match("(");
			match("def");
			queue1.add(data.change("def", 10));

			funcname(text.get(count),text.get(count+1));

			formalbody(text);
			match(")");

			lang(text);
		}
	}
}
```

รูปที่ 2.3 แสดง code ของ CFG

```
Token ( Matched
Token print Matched
Token ( Matched
Token fac Matched
Token ( Matched
Token 6 Matched
Token ) Matched
Token ) Matched
Token ) Matched
Token ) Matched
Correct grammar
```

รูปที่ 2.4 แสดงการแสดงผลเมื่อเขียนถูกต้องตาม CFG

Part 3) Code generator:

เมื่อจำแนกประเภทของโทเค็นแล้วจึงนำ tokennum ทั้งหมดไปใส่ไว้ในโครงสร้างประเภท queue `Queue<tokennum> queue1` เมื่อได้มาทั้งหมดแล้วจึงนำตัวแปรทั้งหมดมาแปลงเป็นภาษาจาวา โดยใช้ตัวเลขที่เก็บเป็นตัวอ้างอิงในการแปลง

เมื่อแปลงเรียบร้อยแล้วจึงนำไปเก็บใน `ArrayList<String> returntojava` หลังจานั้นก็ให้พิมพ์คำสั่งที่แปลงได้ออกมา

เบอร์	คำสั่ง
10	`"public static void " + num20[0] + "() {"` หรือ `"public static int " + num20[0] + "(" + "int " + num20[1] + ") {`
30	`"if(" + ifelse[1] + " " + ifelse[0] + " " + ifelse[2] + "){"`
41	`"System.out.println(" + x[0] + "(" +x[1] + ");}"`
51	`"return " + x[1] + " " + x[0] + " " + x[2] +";}"`
52	`"return (" + x[1] + x[0] + "(" + x[2] + "(" + x[4] + x[3] + x[5] +"))));}"`
53	`"return (" + x[1] + " " + x[0] + "(" + x[2] + "(" + x[3] +")));}"`

ตารางที่ 3.1 แสดงเงื่อนไขการแปลงคำสั่งให้เป็นภาษาจาวา

```
(def fac n
    (ifelse (== n 1)
       1
       (* n (fac (+ n 5)))))
(def test a
    (+ a 3))
(def tet b
     (- b (test (+ b 1))))
(def tex c
     (/ c (test (10))))
(def main ()
    (print (fac (6))))
```

```
public static int fac(int n)
{
if(n == 1){
return 1;}
else{
return (n*(fac(n+5)));}
public static int test(int a)
{
return a + 3;}
public static int tet(int b)
{
return (b-(test(b+1)));}
public static int tex(int c)
{
return (c /(test(10)));}
public static void main() {
   System.out.println(fac(6);}
```

รูปที่ 3.1 แสดง codeที่ถูกแปลงจากภาษา L ไปเป็น java