APELLIDO: Santillan NOMBRE: Facundo DNI 39771280 Lenguajes y Compiladores – UNLAM TERCERA EVALUACION DE APRENDIZAJE	1/12/2020 POSICION

Evaluación de Aprendizaje Nº 3
Consignas

Sea un lenguaje sencillo que permite tres tipos de sentencias

Este lenguaje permite tres sentencias

1) WRITE
2) READ
3) POSICION

READ

Permite la lectura de una variable numérica

WRITE

Permite la escritura de una variable numérica y de una constante string

POSICION

La sentencia permite encontrar en que posición de una lista de constantes enteras positivas se encuentra un elemento pivot.

La lista de constantes puede ser cambiada alterando los valores y su cantidad en el programa *test.txt*.

El elemento pivot deberá ser mayor o igual a uno y será ingresado por el usuario, si no cumpliese con la validación deberá mostrar un mensaje “El valor debe ser >=1”

Si el elemento pivot estuviera varias veces en la lista, el resultado de la sentencia es la primera posición en la que aparece.

En caso de que no sea encontrado se emitirá el mensaje “Elemento no encontrado”

~~Se deberá verificar que el pivot no sea mayor a la longitud de la lista. Si este fuera el caso se deberá emitir un mensaje “La lista tiene menos elementos que el indicado.”~~

La lista de constantes podría ser vacía en cuyo caso se emitirá un mensaje “La lista está vacía”

Por ejemplo:

resul =posicion (4;[10,20,30,40,5,4]) Elemento Encontrado en posición 6

resul =posicion (5;[2,2,2,4]) Elemento no encontrado resul =posicion (51;[2,2,2,4]) Elemento no encontrado resul =posicion (1;[2,1,1,4])

APELLIDO: Santillan NOMBRE: Facundo DNI 39771280 Lenguajes y Compiladores – UNLAM TERCERA EVALUACION DE APRENDIZAJE	1/12/2020 POSICION

Elemento Encontrado en posición 2 resul =posicion (1;[]) La lista está vacía

Sea la gramática del lenguaje enunciado

Gramática **< {S, POSICION, LISTA, WRITE, PROG, SENT, READ, ASIG},**
{cte,id,asigna,para,parc,cte_s,write,posicion,pyc,ca,cc, coma, read **} , S , Reglas }**

Reglas:

0. S → PROG
1. PROG → SENT
2. PROG → PROG SENT
3. SENT → READ | WRITE | ASIG
4. READ → read id
5. ASIG → id asigna POSICION
6. POSICION → posicion para id pyc ca LISTA cc parc
7. POSICION → posicion para id pyc ca cc parc
8. LISTA → cte
9. LISTA → LISTA coma cte
10. WRITE → write cte_s
11. WRITE → write id

Se pide:

Ejercicio I Hacer un compilador completo que solo se base en la gramática dada y con los siguientes requisitos

1) Los elementos léxicos son los indicados como terminales en la definición de la gramática •
   CTE : secuencia de dígitos (Solo representa ctes enteras positivas)
   - ID: letra seguida de letras o dígitos o una letra sola.
   - WRITE, POSICION, READ : representan las palabras reservadas correspondientes
   - ASIGNA: =
   - PARA: (
   - PARC:)
   - CA: [
   - CC:]
   - COMA: ,
   - PYC: ;
   - CTE_S: texto de letras y símbolos únicamente, encerrados entre comillas.

APELLIDO: Santillan NOMBRE: Facundo DNI 39771280 Lenguajes y Compiladores – UNLAM TERCERA EVALUACION DE APRENDIZAJE	**1/12/2020** **POSICION**

2) El programa test.txt debe ser el siguiente

WRITE "Ingrese un valor pivot mayor o igual a 1: "

READ pivot

resul = posicion (pivot ; [$x_{1....}x_n$])

WRITE "Elemento encontrado en posición: "

WRITE resul

donde $x_{1.....}x_n$ son cada una de las constantes. La variable pivot es elegida por el usuario y tendrá un valor entero y positivo.

Toda la semántica deberá traducirse a notación intermedia

Ejercicio II

Se sabe que la fórmula de acceso que el compilador agrega al código ejecutable para acceder a un array de dos dimensiones ordenado por filas es la siguiente:

*$z(i,j):=dir[v(F_i,C_i)] + [((i - F_i) * (C_n - C_i + 1)) + (j - C_i)]$*tamaño del componente (tipo)*

Suponga que el compilador solo soporta limites fijos para sus vectores. Es decir, el límite inferior y superior de las filas (F_i y F_n) y de las columnas (C_i y C_n) son conocidos en tiempo de compilación.

Por ejemplo: para acceder al componente z(i,j) del vector int z(10..18,20..30)

z (i,j) = dir **[** z (10,20) **]** + **[** ((i–10) * (30-20+1)) + (j-20) **]** * 2 bytes

También se sabe que es posible hacer una optimización por reducción simple en esta fórmula al generar polaca inversa. Indique dónde se podría hacer. Explique y ejemplifique como lo haría.

Ejercicio III

Suponga un lenguaje que tiene las siguientes reglas de promoción numérica para sus tipos de datos

Int -> long int -> double
float -> double

APELLIDO: Santillan NOMBRE: Facundo DNI 39771280 Lenguajes y Compiladores – UNLAM TERCERA EVALUACION DE APRENDIZAJE	**1/12/2020** **POSICION**

Suponga también que en una versión del compilador se generó el siguiente conjunto de tercetos para la sentencia: w= b*c*d+ (2.0*a) con int d; double w; long a,c; float b

25 (*, b, c)
26 (*, [25], d)
27 (*, 2.0 , a)
28 (+, [26], [27])
29 (=, w, [28])

APELLIDO: ……

25/11/2020

NOMBRE: ……DNI……

POSICION

Lenguajes y Compiladores – UNLAM TERCERA EVALUACION DE APRENDIZAJE

En otra versión del compilador se pide que el mismo genere los tercetos con conversiones. Escribir como quedaría el conjunto de tercetos de la sentencia anterior con conversiones de tipo según la promoción numérica establecida

2) En este caso podría realizarse una reducción simple en la notación intermedia ya generada. Esto consiste en analizar en la polaca, y si es posible realizar un calculo en tiempo de ejecución. Para esto deberemos encontrar operaciones entre constantes. Estas serán realizadas y sus resultados guardados en la misma polaca inversa. Aquellas celdas que queden libres sufrirán un "baja lógica" (llenaremos las celdas con "X").

Iré apilando operandos, y al momento de encontrar un operador, tendré que analizar si los dos operandos anteriores son constantes, si lo son, realizo el cálculo y guardo el resultado.

@res	@dir	i	10	-	30	20	-	1	+	*	J	20	-	+	2	*	+	=

La primera etapa de la pila quedaría así:

i 10 "-" Al encontrar el operador "-" noto la presencia de una constante en un operando anterior, pero como el otro operando es un id no puedo realizar ningún calculo.

Segunda etapa de la pila:

30 20 "-" En este caso es distinto, al encontrar el "-" tengo apiladas dos constantes que pueden ser calculadas

Tercera etapa de la pila

10 1 + En este otro caso nos encontramos con otra constante, y como ya habíamos guardado en la pila el resultado de la operación anterior que también es constante puedo realizar el cálculo.

Al analizar el resto de la polaca me doy cuenta que no puedo realizar ningún otro calculo en tiempo de compilación, por lo cual el algoritmo terminara y la notación intermedia quedara de la siguiente forma:

@res	@dir	i	10	-	11	X	X	X	X	*	J	20	-	+	2	*	+	=

Esta optimización será útil para no tener que realizar cálculos extras en tiempo de ejecución.

3) Para la conversión utilizaremos 3 etiquetas según corresponda. Estas serán colocadas en los tercetos para indicar el tipo de conversión que sufrirán los datos:

CI-L : Conversion de Int a Long Int

CL-D: Conversión de Long int a Double

CF-D : Conversión de Float a Double

APELLIDO: ……	

Los tercetos quedarían así:

[25] (b, CF-D,)

[26] (c, CL-D,)

[27] (*, [25], [26])

[28] (d, CI-L,)

[29] ([28], CL-D,)

[30] (*, [27], [29])

[31] (2.0, CF-D,)

[32] (a, CL-D,)

[33] (*, [31], [32])

[34] (+, [30], [33])

[35] (=, w, [34])