Eric García de Ceca
Ignacio Terriza

Parte 1: Especificación de la sintaxis del lenguaje

Las características del lenguaje serán las siguientes:

- Identificadores y ámbito:

El ámbito de las variables será estático y local para cada bloque, y serán declaradas al inicio de este. Habrá una sección especial para las variables de entrada y de salida de las funciones, que se pasarán por valor. Para evitar problemas con valores no inicializados, se forzará la inicialización de todas las variables en su declaración. Sin embargo, para facilitar la inicialización de los arrays se proporcionará el operador "**all** N **to** value", que inicializa los N valores del array al valor *value*. El operador "**all…to**" se puede anidar para inicializar arrays de arrays.

Dado que el cuerpo de las instrucciones están delimitadas por palabras reservadas, se permitirá el anidamiento de bloques entre dichas palabras reservadas, sin necesidad de llaves u otros delimitadores extras.

- Tipos:

Habrá 3 tipos básicos: enteros (**int**), booleanos (**bool**), y arrays (**array of**), donde los elementos del array pueden ser enteros, booleanos u otros arrays. Los operadores serán los comunes de la mayoría de lenguajes de programación, teniendo que destacar dos: el operador **all…to** explicado anteriormente, y el operador **@**, que permitirá acceder a los elementos de un array. Los tipos serán comprobados estáticamente.

- Instrucciones:

Habrá 6 instrucciones básicas: asignación (**=**), if de una rama, (**if…then…done**), if de 2 ramas (**if…then…else…done**), clausula case (**considering…choose…chosen**), bucle (**while…do…done**) y llamada a funciones (**call…with…receiving…**). En las instrucciones de asignación y en las condiciones de **if** y **while** se permitirán expresiones con los operadores usuales de manejo de enteros y booleanos, acceso a elementos de vectores (**@**) y el operador **all…to**. En las ramas de **if**, en las alternativas del **choose** y en el cuerpo de **while** se permitirán bloques de declaraciones e instrucciones, pudiendo anidar bloques. Las variables de entrada y de salida de las funciones podrán ser de cualquier tipo, pero la variable de alternativa del **choose** deberá ser siempre entera.

- Errores:

Se realizará detección de errores, informando al usuario de la fila del error encontrado, y mostrando la línea del fichero, junto con la anterior y la siguiente. Se mostrarán todos los errores que sea posible dentro de una misma fase de la compilación (análisis léxico-sintáctico, resolución de identificadores y comprobación de tipos).

Gramática:

Código → FunList Main
FunList → Funcion FunList | Ɛ
Main → **main start** Programa **end;**
Función → **method** id
takes input Entrada
declares output Declaracion DeclList
makes Programa **returns output;**
Entrada → Tipo id**;** Entrada | Tipo id**;**

Programa → DeclList SentList
DeclList → Declaracion DeclList | Ɛ
SentList → Sentencia SentList | Sentencia
Declaración → Tipo id **=** E_0**;**
Tipo → **int** | **bool** | **array of** num Tipo
Sentencia → var **=** E_0**;**
| **if** E_0 **then** Programa **done;**
| **if** E_0 **then** Programa **else** Programa **done;**
| **considering** var **choose** Casos **chosen;**
| **while** E_0 **do** Programa **done;**
| **call** id **with** Vars **receiving** Vars;
Casos → **value** num **do** Programa **done;** Casos
| **value** num **do** Programa **done;**

Vars → var**,** Vars | var
var → id
| var **@ (**E_0**)**
| var **@** id
| var **@** num

array → elem**,** array | elem
elem → num | **TRUE** | **FALSE** | **{**array**}**

E_0 → **all** num **to** E_0 | E_1 | **{**array**}**
E_1 → E_1 **OR** E_2 | E_2
E_2 → E_2 **AND** E_3 | E_3
E_3 → **NOT** E_3 | E_4
E_4 → **TRUE** | **FALSE**
| E_5 **==** E_5 | E_5 **!=** E_5
| E_5 **<** E_5 | E_5 **>** E_5 | E_5 **<=** E_5 | E_5 **>=** E_5
| E_5
E_5 → E_5 **+** E_6 | E_5 **-** E_6 | E_6
E_6 → E_6 **/** E_7 | E_7
E_7 → E_7 ***** E_8 | E_8
E_8 → num | var | **(**E_0**)**

Ejemplos

Programa de ejemplo 1: Encuentra el elemento máximo de un vector:

```
method maximo
takes input array of 10 int a;
declares output int max = 0;
makes
int i = 0;
max = a @ 0;
while i < 10 do
      if max < a @ i then
            max = a @ i;
      done;
      i = i + 1;
done;
returns output;

main start
array of 10 int a = {5,4,8,10,0,3,9,1,2,3};
      int max = 0;
      call maximo with a receiving max;
end;
```

Programa de ejemplo 2: Algoritmo de inserción (array de 10 enteros):

```
method ordenar
takes input array of 10 int lista;
declares output array of 10 int listabis = all 10 to 0;
makes
      int temp = 0;
      int i = 1;
      int j = 0;
      while i < 10 do
            temp = lista @ i;
            j = i - 1;
            while lista @ j > temp AND j >= 0 do
                  lista @ (j + 1) = lista @ j;
                  j = j - 1;
            done;
            lista @ (j + 1) = temp;
            i = i + 1;
      done;
      listabis = lista;
returns output;

main start
array of 10 int lista = {5,4,8,10,0,3,9,1,2,3};
      call ordenar with lista receiving lista;
end;
```

Programa de ejemplo 3: Multiplicación de matrices (matrices de 10x10 enteros):

```
method mult
takes input
	array of 10 array of 10 int a;
	array of 10 array of 10 int b;
declares output
	array of 10 array of 10 int c = all 10 to all 10 to 0;
makes
	int i = 0;
	while i < 10 do
		int j = 0;
		while j < 10 do
			int k =0;
			while k < 10 do
				c @ i @ j = c@i@j + a@i@k * b@k@j;
				k = k+1;
			done;
			j = j+1;
		done;
		i = i + 1;
	done;
returns output;

main start
	array of 10 array of 10 int a = {
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,89,45,2,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,4,8,333,6,9},
		{5,4,8,10,0,3,8,5,6,9}
	};
	array of 10 array of 10 int b = {
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,89,45,2,5,6,9},
		{5,4,8,10,0,4,8,333,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9},
		{5,4,8,10,0,3,8,5,6,9}
	};
	array of 10 array of 10 int c = all 10 to all 10 to 0;

	call mult with a,b receiving c;
end;
```