2011 | 2012

Especificación del Lenguaje nADA

Compiladores

Alberto Manuel Mireles Suárez
David Guillermo Morales Sáez
Eduardo Quesada Díaz
María del Carmen Sánchez Medrano

Universidad de Las Palmas de Gran Canaria

Tabla de contenido

Introducción

A lo largo de este documento se definen las características del lenguaje de programación nADA, un subconjunto del lenguaje ADA.
Hemos decidido emplear un lenguaje basado en ADA puesto que ya hemos trabajado con él y estamos familiarizados con el mismo. A su vez tiene la parte de declaraciones bien definida y es un lenguaje fuertemente tipado, lo cual garantiza que se trabajará con tipos compatibles. Finalmente, podemos añadir que la sintaxis del lenguaje ADA es muy similar a Pascal, que es un lenguaje apropiado para escribir un compilador.

Características del lenguaje

Identificadores

Los identificadores estarán formados por los caracteres alfanuméricos del alfabeto inglés, empezando siempre por una letra y de tamaño máximo 20 caracteres.

Literales

Admitimos tres tipos de literales: números enteros, en punto flotante y caracteres. Los rangos de los valores son los siguientes:

Tipo de dato	Valor mínimo	Valor máximo
Entero	$-2^{15} + 1$	$2^{15} - 1$
Punto flotante		
Caracteres	0	126

Operadores

Además de los operadores matemáticos básicos (+, -, *, /), de comprobaciones (=, <, <=, >, >=) y de asignación (:=), se incluyen los operadores binarios and, or y not.

Datos construidos

Los datos construidos que utilizaremos en nADA son tres:

- Ristras
- Vectores
- Enumerados

Las ristras son un conjunto de caracteres encapsulados por el símbolo ' " '. Tendrán como máximo un tamaño de 255 y un último carácter que represente el fin de ristra ('\0').

Los vectores son un conjunto de elementos ordenados. Pueden ser de cualquier tipo, pero dentro de un vector, todos los elementos han de ser del mismo. Se pude acceder a cada elemento a través de un índice, suministrado en su declaración. Su declaración se lleva a cabo con la palabra reservada "array", seguido del índice encapsulado por paréntesis. El rango se define con el elemento inicial seguido de dos puntos y el elemento final.

Los tipos enumerados son aquellos que establecen un conjunto definido de elementos que puede tomar una variable. Son utilizados principalmente por las estructuras de selección para facilitar su manejo.

Comentarios

Se permitirán los comentarios de línea que vendrán precedidos por "--". Por tanto, no permitiremos los comentarios de bloque.

Procedimientos

Los procedimientos tienen una estructura que comienza con la palabra reservada 'procedure', seguido del nombre del procedimiento, seguido de sus parámetros (tanto de entrada como de salida) con su tipo, encapsulados por paréntesis. Seguidamente, encontraremos la palabra reservada 'is', que marca el inicio de la declaración de las variables internas del proceso, así como las declaraciones de los subprogramas que vayan a ser llamados.

Tras la declaración, se debe encontrar la palabra reservada 'begin', con la cual comienza el código de ejecución del procedimiento. Este mismo terminará cuando se encuentre la palabra reservada 'end', seguido del nombre del procedimiento y finalizando con un ';'. Un ejemplo de procedimiento simple es el siguiente:

```
procedure programa(declaración var. In out) is
    definición variables y/o subprocedimientos;
begin
    instrucciones;
end programa;
```

Preprocesador

Para la inclusión de librerías, utilizaremos las directivas 'with' y 'use', que nos permitirán incluir distintas librerías en nuestro programa y utilizar aquellas que necesitemos. Se distinguen las dos directivas ya que ciertas librerías pueden requerir otras librerías, por lo que serán declaradas con el 'with', pero sólo utilizaremos de forma directa aquellas que estén declaradas en la directiva 'use'.

Ficheros

Para la utilización de ficheros, utilizaremos una estructura llamada File_Type. Distinguiremos la creación y la apertura de ficheros, es decir, las funciones serán distintas. Para la creación de un fichero, utilizaremos la función Create, cuyos parámetros serán una variable de tipo File_Type y el nombre del mismo. Por otro lado, la apertura de un fichero se llevará a cabo con la función Open, cuyos parámetros serán una variable de tipo File_Type, el modo de apertura y la ruta del fichero. Permitiremos abrir el fichero de tres modos distintos:

- Modo lectura
- Modo escritura
- Modo escritura al final del fichero

Para la lectura de datos de un fichero, se utilizará la familia de funciones get() y, para la escritura, se utilizará la familia de funciones put(), además de permitir la comprobación de fin de línea y fin de fichero.

Estructuras de control

Este lenguaje dispondrá de las siguientes estructuras de control:

- Estructura condicional
- Estructura de selección
- Estructura de repetición

Como estructura condicional utilizaremos el bloque *if – then – else - end if*, estableciendo una condición y ejecutándose las instrucciones pertinentes en función de la misma. Por ejemplo:

```
if(condición) then
instrucciones;
else
instrucciones;
end if;
```

Para la estructura de selección, utilizaremos la instrucción *case – is – when - end case*, estableciendo la variable a evaluar y escogiendo la entrada correspondiente. Por ejemplo:

```
case variable is
     when alternativa => sentencias;
     when others => sentencias;
end case;
```

Para la estructura de repetición, se empleará la instrucción *loop – exit when – end loop*, definiendo la condición de salida en el lugar que sea necesario. Por ejemplo:

```
loop
     instrucciones;
     exit when condición;
     instrucciones;
end loop;
```