Estudio del lenguaje PSeInt

Carlos Javier Hellín Asensio

Se analizará usando el programa PSeInt http://prdownloads.sourceforge.net/pseint/pseint-w32-20200501.exe?download para hacer pruebas y también se estudiará el código fuente para conocer su funcionamiento real: http://prdownloads.sourceforge.net/pseint/pseint-src-20200501.tgz?download en concreto los ficheros SynCheck.cpp, utils.cpp y utls.h de la carpeta pseint son los ideales, aunque se pueden ir agregando más según se vea.

Características

El lenguaje es case-insensistive, es decir no distingue entre mayúsculas y minúsculas. Por ejemplo. Algoritmo, ALGORITMO o AlGoRiTMo es la misma palabra reservada. Lo mismo ocurre con las variables: se puede definir como nombre, pero luego más tarde se puede usar como Nombre o nOmBrE.
Dos soluciones:

- En Java: antes de pasarlo por el lexer poner todo el código de entrada en minúsculas o maýusculas (esto último lo hace el PSeInt si se observa en la función SynCheckAux1 de SynCheck.cpp):
  ```
  // pasar todo a mayusculas, reemplazar tabs, comillas, word_operators, corchetes, y quita espacios extras
  static string SynCheckAux1(string &cadena) {
  …
  …
  ```
- Usar expresiones regulares en el lexer como por ejemplo para Algoritmo sería: [Aa][Ll][Gg][Oo][Rr][Ii][Tt][Mm][Oo]

El lenguaje tiene muchos sinónimos. Se podría usar Algoritmo o Proceso para iniciar un programa. Al igual que para terminarlo como FinAlgoritmo o FinProceso.

El lenguaje hay que observar detenidamente cada palabra reservada porque puede aceptar que haya espacios. FinSi y Fin Si son lo mismo. Lo mejor es tener en cuenta los espacios con ‘ ‘?

Dentro del programa (Algoritmo o Proceso) habrá una o más instrucciones y/o estructuras de control. Todas ellas pueden terminar en punto y coma ‘;’ o no.

Se pueden hacer comentarios con // pero NO comentarios multilínea como /** **/ o /* */, en este tipo de casos se da error y se muestra el árbol hasta donde se puede.

El lenguaje es fuertemente tipado, es decir, si se define una variable del tipo Caracter luego no se puede asignar un número.

Palabras reservadas

Algoritmo <identificador> | Proceso <identificador>
solo puede haber uno en todo el programa

FinAlgoritmo | FinProceso | Fin Algoritmo | Fin Proceso
obligatorio cerrar con alguno de estos

Definir <var1>, <var2>, …, <varN> Como [Real|Entero|Logico|Caracter]
Se puede declarar explícitamente el tipo de una variable con esta palabra reservada o dejar que el intérprete intente deducir a partir de los datos que se guardan en la misma.
Existen sinónimos para los tipos de datos como Real, Numero, Numerico, etc.. y el uso de tildes: MIRAR APARTADO TIPOS DE DATOS.

Reales | Numero | Numeros | Numerica |, Numerico | Numericas | Numericos | Real | Número | Números | Numérica | Numérico | Numéricas | Numéricos
Tipo de dato simple numérico.

Enteros | Enteras | Entera | Entero
Tipo de dato simple solo enteros

Logicos | Logicas | Logica | Logico | Lógicos | Lógicas | Lógica | Lógico
Tipo de datos simple lógico

Verdadero
Valor del tipo lógico

Falso
Valor del tipo lógico

Caracter | Caracteres | Texto | Textos | Cadena | Cadenas | Carácter
Tipo de datos simple carácter

Leer <var1>, <var2>, …, <varN>
Si una variable donde se debe guardar el valor leído no existe, se crea durante la lectura. Si la variable existe se pierde su valor anterior ya que tomará el valor nuevo (destructiva).

(Escribir | Imprimir | Mostrar | Informar) <expr1> , <expr2> , ... , <exprN>
Si en algún punto de la línea se encuentran las palabras clave (SinSaltar | Sin Saltar | SinBajar | Sin Bajar) los valores se muestran en la pantalla, pero no se avanza a la línea siguiente. En caso contrario, se añade un salto de línea.
Ejemplos:
Escribir Sin Saltar <expr1>, …, <exprN>
Escribir <expr1>, Sin Saltar …, <exprN>
Escribir Sin Saltar <expr1>, …, <exprN> Sin Saltar

Borrar Pantalla | BorrarPantalla | Limpiar Pantalla | LimpiarPantalla
Borra la pantalla y coloca el cursor en la esquina superior izquierda

Esperar Tecla | EsperarTecla | Esperar Una Tecla

Esperar <numerico> (Segundo | Segundos | Milisegundo | Milisegundos)
Pausa el algoritmo durante un intervalo de tiempo predefinido, indicando a continuación de la palabra la clave la longitud y unidad de dicho intervalo.

Nota: revisar las demás secciones que puede haber más palabras reservadas no puestas otra vez aquí.

Variables

La asignación en principio puede ser una de estas:
<variable> <- <expresión>
<variable> := <expresión>

El tipo de variable y el de la expresión deben coincidir (fuertemente tipado)
Si la variable de la izquierda no existía previamente a la asignación, se crea. Si la variable existía se pierde su valor anterior y toma el valor nuevo (acción destructiva). Los contenidos de las variables que intervienen en la expresión de la derecha no se modifican.

Una variable se hace referencia mediante un identificador (el nombre de la variable). Un identificador puede empezar por guión bajo '_' o por una letra incluyendo á, é, í, ó, ú, ñ y ü. Puede contener solo letras (incluyendo las que tienen tilde como antes), números y el guión bajo '_'.
No puede contener ni espacios ni operadores, no coincidir con una palabra reservada o función del lenguaje.

Ejemplo: nombre, _nombre, ánombre, _no1, nómbre9

Tipos de datos

Existen tres tipos de datos básicos:

- Númerico: números , tanto enteros como reales. Para separar decimales se utiliza el punto. Ejemplos: 12 23 0 -2.3 3.14
  Al usar Definir se usa Reales, Numero, Numeros, Numerica, Numerico, Numericas, Numericos y Real. También ojo a las tildes como Número, Números, Numérica, Numérico, Numéricas, Numéricos.
  Si se usa Entero sólo se permite almacenar valores enteros; cualquier valor no entero que se lea o asigne en una variable este tipo será truncado (da error en tiempo de ejecución y no de compilación) Tiene como sinónimos: Enteros, Enteras, Entera, Entero
- Lógico: sólo puede tomar dos valores. Verdadero o Falso. En Definir se usa Logicos, Logicas, Logica, Logico, Lógicos, Lógicas, Lógica, Lógico.
- Carácter: caracteres o cadenas de caracteres encerrados entre comillas (pueden ser dobles o simples). Al usar Definir se usa Caracter, Caracteres, Texto, Textos, Cadena, Cadenas, Carácter..
  Estas pueden contener, cero, o uno o más caracteres arbitrarios y no tienen una longitud máxima. Si se declara una variable de este tipo y en una lectura el usuario ingresa un número o un valor lógico, se asignará una cadena que contiene el texto ingresado (ejemplo: "1", "Verdadero", etc.)
  No existen caracteres de escape en cadenas, del tipo "\n" o '\"', etc...

Existe una estructura de datos: los arreglos.

Los arreglos son estructuras de datos homogéneas (todos sus datos son del mismo tipo) que permiten almacenar un determinado número de datos bajo un mismo identificador, para

luego referirse a los mismos utilizando uno o más subíndices. Los arreglos pueden pensarse como vectores, matrices, etc. Para poder utilizar un arreglo, primero es obligatorio su dimensionamiento; es decir, declarar los rangos de sus subíndices, lo cual determina cuántos elementos se almacenarán y cómo se accederá a los mismos.

Dimension | Dimensión <identificador> (<maxI>,...,<maxN>)

Se pueden declarar más de un arreglo en una misma instrucción, separándolos con una coma (,).

Dimension | Dimensión <ident1> (<max11>,...,<max1N>),..., <identM> (<maxM1>,...,<maxMN>)

Expresiones

Operadores

Relacionales / Expresiones Coloquiales

> | Es Mayor A | Es Mayor Que
3>2

< | Es Menor A | Es Menor Que
‘ABC’<’abc’

= | == | Es Igual A | Es Igual Que
4=3

<= | (Es Igual ‘|’ Menor A) | (Es Igual ‘|’ Menor Que) | (Es Menor ‘|’ Igual A) | (Es Menor ‘|’ Igual Que)
‘a’<=’b’

>= | (Es Igual ‘|’ Mayor A) | (Es Igual ‘|’ Mayor Que) | (Es Mayor ‘|’ Igual A) | (Es Mayor ‘|’ Igual Que)
4>=5

<> | != | Es Distinto A | Es Distinto De | Es Distinta A | Es Distinta De
‘a’<>’b’

Es Par
%2=0

Es Impar
%2=1

Es Positivo | Es Positiva
>0

Es Negativo | Es Negativa
<0

Es Cero
=0

Es Divisible Por | Es Multiplo De
X % Y= 0

Es
X = Y

Nota: El '|' no es un OR, es el carácter de la barra | tal cual

Lógicos

& | Y
(7>4)

| | O
(1=1 | 2=1)

~ | NO | !
~(2<5)

Algebraicos

+
total <- cant1 + cant2
Se permite concatenar variables de texto con el operador +. Por ejemplo
Nombre+""+Apellido

-
stock <- disp - venta

*
area <- base * altura

/
porc <- 100 * parte / total

^
sup <- 3.41 * radio ^ 2

% | MOD
resto <- num MOD div

La jerarquía de los operadores matemáticos es igual a la del álgebra, aunque puede alterarse mediante el uso de paréntesis. Para el caso de los operadores & y |, la evaluación se realiza en cortocircuito. Esto significa que si dos expresiones están unidas por el operador ∧ y la primera se evalúa como Falso, o están unidas por el operador | y la primera se evalúa como Verdadero, la segunda no se evalúa ya que no altera el resultado.

Nota: en el caso de las palabras serán reservadas y no se podrán utilizar como nombre de variables.

Funciones

pi
rc(x) | raiz(x)
abs(x)
ln(x)
exp(x)
sen(x)
cos(x)
tan(x)
asen(x)
acos(x)
atan(x)
azar(x)
aleatorio(a,b)
trunc(x)
redon(x)
convertiranumero(x) | convertiranúmero(x)
convertiratexto(s)
longitud(s)
subcadena(s,x,y)
mayusculas(s) | mayúsculas(s)
minusculas(s) | minúsculas(s)
concatenar(s1,s2)

Nota: todos los nombres de las funciones son palabras reservadas

Estructuras de Control

Condicionales

Si <condición> Entonces
    <instrucciones>
SiNo
    <instrucciones>
FinSi | Fin Si
Entonces puede estar o no. Al igual que SiNo por si no existe un “else” como tal

Segun | Según <variable> Hacer
    <número1>: <instrucciones>
    <número2>,<número3>: <instrucciones>
    <...>
    De Otro Modo: | De Otro Modo | Otro Modo: <instrucciones>
FinSegun | Fin Segun | FinSegún | Fin Según
Hacer puede estar o no.

Se puede utilizar también variables de tipo carácter en <número1>, <número2>, etc..

Repetitivas

Mientras <condición> Hacer
    <instrucciones>
FinMientras | Fin Mientras
Hacer puede estar o no.

Repetir
    <instrucciones>
Hasta Que | Mientras Que <condición>

Para <variable> <- <inicial> Hasta <final> Con Paso <paso> Hacer
    <instrucciones>
FinPara | Fin Para
Hacer y Con Paso <paso> puede estar o no.

La primera variante consiste en reemplazar el operador de asignación por la palabra clave *Desde*:
Para <variable> Desde <inicial> Hasta <final> Con Paso <paso> Hacer ...

La segunda variante solo sirve para recorrer arreglos de una o más dimensiones. Se introduce con la construcción *Para Cada* seguida de un identificador, la palabra clave *De* y otro identificador: Para Cada <elemento> De <Arreglo> Hacer ...
Hacer puede estar o no.

Funciones/Subprocesos

Funcion | Función | SubProceso | SubAlgoritmo variable_de_retorno <- nombre_de_la_funcion (argumento_1, argumento_2, ...)
acción 1;
acción 1;
.
.
.
acción n;
FinFuncion | Fin Funcion | Fin Función | FinFunción | FinSubProceso | Fin SubProceso | FinSubAlgoritmo | Fin SubAlgoritmo

Se pueden declarar nuevas funciones o subprocesos fuera de Algoritmo/FinAlgoritmo y sus sinónimos.

Existen variantes para esta estructura. Si la función no retorna ningún valor, pueden omitirse la variable de retorno y el signo de asignación. Es decir, se puede colocar directamente el nombre y los argumentos a continuación de la palabra clave *Funcion*. Si el subproceso no recibe ningún valor pueden colocarse los paréntesis vacíos u omitirse, finalizando la primera línea con el nombre del subproceso. Las reglas para los nombres de subprocesos, variables de retorno y argumentos son las mismas que para cualquier identificador.

Además, opcionalmente pueden agregarse las palabras claves *Por Valor* o *Por Referencia* para indicar el tipo de pasaje en cada argumento. Por ejemplo: (argumento_1 Por Valor, argumento_2 Por Referencia, …)

Para invocar a la función se debe utilizar su nombre y entre paréntesis los parámetros, que podrán ser expresiones sólo si el tipo de pasaje es por referencia. Una llamada puede ser en sí una instrucción, pero si la función retorna algún valor, también puede utilizarse como operando dentro de una expresión.