Lenguaje de Diseño en PSeInt
Estructuración de datos

RESOLUCIÓN DE PROBLEMAS
FUNDAMENTOS DE LA PROGRAMACIÓN

Ingeniería en Informática
Ingeniería en Computación
Ingeniería en Minas
Profesorado en Computación

UNIVERSIDAD NACIONAL DE SAN LUIS
DEPARTAMENTO DE INFORMÁTICA

Índice

1. Introducción **2**

2. Arreglos en PSeInt **3**

2.1. Definición de Arreglo 3

2.2. Operaciones con arreglos: asignación y recuperación de valores 10

Estructuración de Datos

1. Introducción

En el manual de **Expresiones de Tipo** se introdujo el concepto de variables de datos que, como se recordará, podía ser de tipo numérica (entero o real), caracter o lógica. Ahora, extenderemos ese concepto a un conjunto o grupo de ellas.

Al resolver un problema particular, es fundamental tratar con la organización de sus datos en forma estructurada. El seleccionar los datos adecuadamente, es un paso necesario y fundamental al definir y, posteriormente, resolver el problema. Todas las formas posibles en que los datos primitivos se relacionan lógicamente, definen estructuras de datos.

Una **estructura de datos** es un conjunto de datos reunidos bajo un único nombre colectivo.

Las diferentes estructuras se diferencian por la forma en que sus componentes están relacionadas y por el tipo de los mismos.

Características:

1. Todos los datos estructurados deben, en última instancia, ser construidos a partir de los datos primitivos
2. El conjunto de datos se identifica con un único nombre
3. Una estructura se diferencia de otra por la forma en que sus componentes están relacionadas y el tipo de datos de las mismas.
4. Se encuentran tanto en memoria principal como en memoria secundaria. En esta oportunidad se hará referencia a las estructuras de memoria principal.

Tipos de Estructuras

Desde el punto de vista de cómo están almacenados los datos, es posible distinguir entre:

Estructuras Enlazadas: los datos no se encuentran en posiciones adyacentes de memoria.

Estructuras Contiguas o Físicas: los datos se encuentran en posiciones adyacentes de memoria.

Por ejemplo, una estructura conocida y muy simple es el número complejo, que toma la forma de un par ordenado de números reales, donde los números reales son de tipo primitivo. Otra estructura un poco más complicada, pero muy común, es el **arreglo lineal**.

Donde

- Todos los elementos del arreglo son del mismo tipo primitivo de datos, por lo tanto es una **estructura homogénea**.
- Es una **estructura estática**, es decir su tamaño (cantidad y tipo de elementos del arreglo) se define en tiempo de compilación a partir de la declaración y no cambia durante la ejecución del programa. Es posible definir arreglos cuya dimensión se modifique en tiempos de ejecución **(estructuras dinámicas)** pero este concepto no será abordado en este curso).
- Es una **estructura lineal de acceso directo**, es decir se accede a un dato en forma directa con sólo indicar la posición o subíndice. El número que indica la posición (subíndice) es un número natural.
- La **dimensión** es la cantidad de elementos de la estructura.

2. Arreglos en PSeInt

Los arreglos son estructuras homogéneas. A partir de la configuación de PSeInt que se utiliza en este curso, la creación de un arreglo se realiza en dos pasos: en primer lugar se **Define** el tipo de dato asociado a todos los elementos del arreglo y luego se establece la **Dimension** (cantidad de elementos) de la estructura.
En PSeInt el subíndice (posición) del arreglo puede comenzar con diferentes valores. El perfil de configuración que se utiliza en este curso establece que el subíndice comienza en 1.

2.1. Definición de Arreglo

Al igual que las variables, el nombre asignado al arreglo debe constar sólo de letras, números y/o guión bajo "_", comenzando siempre con una letra.
Se pueden declarar mas de un arreglo en una misma instrucción separándolos con el símbolo coma.

PSeInt
Definir <identificador > **Como** <tipo > **Dimension** <identificador >**[**<max1 >**]**
Ejemplo: Definir A Como Real Dimension A [10]

Otra forma de declarar el tipo de datos de cada elemento del arreglo, consiste en definir una variable de tipo específico y luego utilizarla para asignarle valores, al asignarle un valor directamente (por ejemplo por medio de la instrucción Leer), y el arreglo adopta el tipo del valor ingresado.

Ejemplo:

Definir A como Entero
Definir B como Caracter
Dimension A[5]
Dimension B[5]
Definir x como Caracter
Leer A[1]; // Suponer que se ingresa un 8
Escribir A[1]; // Muestra por pantalla 8
x <- 'p'
B[1] <- x //B[1] tiene el caracter 'p'
B[2] <- 58 // ERROR DE TIPO porque tenía asignado un valor de tipo caracter y 58 es un valor entero

Para acceder a un elemento del arreglo se utiliza el siguiente comando:

<identificador >[posición]

Ejemplo:

Escribir A[3] //Muestra por pantalla el contenido del arreglo A en la posición 3
i <- 3
Escribir A[i] //Muestra por pantalla el contenido del arreglo A en la posición 3

Consideremos el siguiente problema:

Enunciado: Una empresa recibe mensualmente información sobre las ventas de cada una de sus tres sucursales y, desea obtener un listado de aquellas, cuyas ventas superan el promedio de las mismas.
Para dicho problema podemos encontrar distintos algoritmos que lo solucionen.

Solución 1:

Una forma de resolver el enunciado planteado con lo visto hasta el momento es definiendo tres variables reales, cuyos valores correspondan a las ventas de cada una de las sucursales.

Ambiente del algoritmo:

VARIABLES	**DESCRIPCION**
VENT1	Variable de entrada, real, cuyo valor representa las ventas de la sucursal 1 y de salida en caso que corresponda.
VENT2	Variable de entrada, real, cuyo valor representa las ventas de la sucursal 2 y de salida en caso que corresponda.
VENT3	Variable de entrada, real, cuyo valor representa las ventas de la sucursal 3 y de salida en caso que corresponda.
PROMEDIO	Variable auxiliar, real, en la cual se calcula el promedio de las ventas.

Algoritmo:

Algoritmo "Ventas"
Definir VENT1, VENT2, VENT3, PROMEDIO como real
Leer VENT1, VENT2, VENT3
PROMEDIO $\longleftarrow$ (VENT1 + VENT2 + VENT3) / 3
Si VENT1 >= PROMEDIO
Entonces

```
            Escribir "1-", VENT1
        FinSi
        Si VENT2 >= PROMEDIO
            Entonces
            Escribir "2-", VENT2
        FinSi
        Si VENT3 >= PROMEDIO
            Entonces
            Escribir "3-", VENT3
        FinSi
    FinAlgoritmo
```

Se observa que este algoritmo es válido, sólo, para tres sucursales. ¿Cómo se plantearía el algoritmo, por ejemplo, para 100 sucursales?.

Si se utiliza el mismo enfoque se debería definir 100 variables, una para cada sucursal: VENT1, VENT2, . . . , VENT100 y el algoritmo anterior quedaría:

Algoritmo “Ventas”
Definir VENT1, VENT2, ..., VENT100, PROMEDIO como real
Leer VENT1, VENT2, ..., VENT100
PROMEDIO $\longleftarrow$ (VENT1 + VENT2 + ... + VENT100) / 100
Si VENT1 >= PROMEDIO
Entonces
Escribir “1−”, VENT1
FinSi
Si VENT2 >= PROMEDIO
Entonces
Escribir “2−”, VENT2
FinSi
.
.
.
Si VENT100 >= PROMEDIO
Entonces
Escribir “100−”, VENT100
FinSi
FinAlgoritmo

Como se puede observar se han utilizado los símbolos:

... y
.
.
.

Al primero, se los ha utilizado para abreviar la escritura de la declaración, lectura y la suma de las 100 variables y, al segundo, para indicar la comprobación de las ventas desde la sucursal 3 hasta la 99.

Estos dos símbolos no son comprendidos por ningún procesador y la solución sería escribir, explícitamente, las 100 declaraciones, lecturas, suma más los 100 condicionales.

Solución 2:

Una forma **más eficiente de resolver este problema** consiste en reunir las ventas de las 100 sucursales bajo un único nombre VENTA.

VENTA podría estar representada, esquemáticamente, por la siguiente tabla:

120	578	625		1230
1	2	3		100

Luego, para designar uno de los 100 valores utilizamos su ubicación en la tabla: VENTA[3] representa el tercer valor (625) de la variable VENTA e índice 3 indica la tercera sucursal.

De esta manera definimos un **arreglo lineal** o **vector**, de nombre VENTA, cuyos 100 elementos son VENTA [1], VENTA[2], VENTA[3], . . ., VENTA[100].

Se define entonces a una variable con estructura de arreglo como:

Un conjunto de variables del mismo tipo.

Según la configuración del perfil que se utiliza en este curso, todo arreglo debe ser declarado previo a su utilización. La declaración del arreglo se realiza de la siguiente manera:

```
Definir VENTA como real
```

Donde:

- Se indica el **nombre** de la variable arreglo; VENTA
- El **tipo de datos de los elementos de la variable**, en este caso **reales**.

A continuación se especifica la dimensión del arreglo. Se puede declarar más de un arreglo en una misma instrucción, separándolos con el símbolo coma.

```
Dimension VENTA [100]
```

Donde:

- La instrucción **Dimensión** permite declarar la cantidad de elementos de la estructura que debe ser un número entero positivo. En este ejemplo es 100.

Para designar un elemento utilizamos el nombre del arreglo, seguido de una expresión encerrada entre corchetes. El valor de la expresión, da la ubicación del elemento, dentro del arreglo.

Ejemplos:

- VENTA[15] donde 15 es una constante, designa al décimo quinto elemento del arreglo.
- VENTA[I] donde I es una variable y su valor i designa al i-ésimo elemento del arreglo y puede tomar valores entre 1 y 100.
- VENTA[K + 5], si el valor de la variable K es k, entonces la expresión denota al (k + 5)-ésimo elemento del arreglo.

Cada elemento de un arreglo puede ser de cualquiera de los tipos primitivos: entero, real, caracter o lógico, **pero todos los elementos son del mismo tipo**. Mientras que la expresión que indica la posición de cada elemento es un número natural, denominado **subíndice**. Por consiguiente, si el subíndice es real, 0 o negativo, la evaluación de la expresión da error.

Cada elemento de un arreglo se denota explícitamente, y es accedida directamente, mencionando el nombre del arreglo seguido de una expresión encerrada entre corchetes, a la que se llama **subíndice** del arreglo.
Es posible ahora reescribir el algoritmo "Ventas" usando este nuevo concepto.
Ambiente del algoritmo:

VARIABLES	**DESCRIPCION**
VENTA	Arreglo, de entrada y de salida, de 100 elementos, cada elemento de tipo real y representando las ventas de cada sucursal.
I	Variable de tipo entero, que se usará como índice del arreglo y que tomará los valores 1,. . . , 100
PROMEDIO	Variable auxiliar, de tipo real donde se calcula el promedio de las ventas.

Algoritmo "Ventas"
Definir VENTA como real
Dimension VENTA[100]
PROMEDIO como real
I como entero
PROMEDIO $\longleftarrow$ 0
Para $I <-1$ **Hasta** 100 **Con Paso** 1 **Hacer**
Leer VENTA[I]

```
        PROMEDIO ⟵ PROMEDIO + VENTA[I]
    FinPara
    PROMEDIO ⟵ PROMEDIO / 100
    Para I <- 1 Hasta 100 Con Paso 1 Hacer
        SI VENTA[I] >= PROMEDIO Entonces
            ESCRIBIR I, VENTA[I]
        FinSi
    FinPara
FinAlgoritmo
```

A continuación de la declaración de las variables, se asigna a la variable PROMEDIO el valor inicial 0 para luego continuar con una iteración que repite 100 veces la acción LEER que permite ingresar las ventas de cada sucursal e ir acumulándola en la variable PROMEDIO. En cada ejecución de la iteración se ingresa un dato; por ejemplo, cuando I vale 1 en VENTA[1] se ingresa el valor de las ventas de la sucursal 1 y así siguiendo. Luego, en secuencia, sigue el cálculo del promedio y, por último, una repetición que comprueba, para cada sucursal, si las ventas de la misma, superan o no, el valor del promedio calculado y, en caso afirmativo, imprime el número de la sucursal y el valor de la venta de la misma.

2.2. Operaciones con arreglos: asignación y recuperación de valores

Al igual que sucede con cualquier variable simple es posible **asignar un valor** a un elemento de un arreglo y **recuperar su valor**.

Por ejemplo, sea V declarado como un arreglo de 50 elementos enteros, entonces la operación de asignación:

$$V[10] \longleftarrow 15$$

indica que al décimo elemento de V se le asigna el valor 15, mientras que en:

$$J \longleftarrow 13 * J + V[I]$$

Primero, se obtiene el valor de la variable J que se multiplica por la constante 13 y, al resultado, se le suma el valor del i-ésimo elemento del arreglo V y todo el resultado de la expresión se almacena en la variable J.

Otro ejemplo:

Enunciado: Escribir un algoritmo que ordene de menor a mayor los elementos de un arreglo de 7 elementos enteros e imprima el arreglo ordenado.

Método: El método para efectuar el ordenamiento que utilizaremos consiste en:

1. Encontrar el menor elemento, entre los n, del arreglo.
2. Intercambiar el elemento encontrado con el primero del arreglo.
3. Repetir estas operaciones con los $n - 1$ elementos restantes, obteniendo, el segundo menor elemento del arreglo; proseguir con los $n - 2$ elementos restantes, hasta que quede solamente el mayor valor.

En el ejemplo siguiente, mostramos cómo se intercambian en cada caso los valores:

estado inicial: 21 35 17 8 14 42 2

2	35	17	8	14	42	21	1 intercambio
↑						↑	
2	8	17	35	14	42	21	2 intercambio
	↑		↑				
2	8	14	35	17	42	21	3 intercambio
		↑		↑			
2	8	14	17	35	42	21	4 intercambio
			↑	↑			
2	8	14	17	21	42	35	5 intercambio
				↑		↑	
2	8	14	17	21	35	42	6 intercambio
					↑	↑	

Ambiente del algoritmo "Ordenar":

VARIABLES	DESCRIPCION
V	Variable de entrada-salida que es el arreglo de enteros a ordenar y donde queda el arreglo ordenado.
I, J	Variables auxiliares enteras que se usan como índices del arreglo
MIN	Variable auxiliar de tipo entero que representa al valor del índice donde se encuentra el elemento mínimo de V
VAL_MI	Variable auxiliar, entera, usada para el intercambio de dos valores en el arreglo.

Versión 1:

T_1 Declarar V como arreglo de tipo entero y las variables auxiliares I, J, MIN y VAL_MI de tipo entero

T_2 Ingresar los 7 elementos del arreglo a ordenar

T_3 Ordenar el arreglo de 7 números de menor a mayor

T_4 Imprimir el arreglo ya ordenado

Versión final:

Algoritmo "Ordenar"
Definir V Como entero
Dimension $V[7]$
Definir I, J, MIN,VAL_MI Como entero
Para $I <- 1$ **Hasta** 7 **Con Paso** 1 **Hacer**
Leer $V[I]$
FinPara
Para $I <- 1$ **Hasta** 7 - 1 **Con Paso** 1 **Hacer**
MIN $\longleftarrow I$
Para $J <- I + 1$ **Hasta** 7 **Con Paso** 1 **Hacer**
SI $V[J] < V$[MIN]
Entonces
MIN $\longleftarrow J$
FinSi
FinPara
VAL_MI $\longleftarrow V$[MIN]

V[MIN] $\longleftarrow V[I]$
$V[I] \longleftarrow$ VAL_MI
FinPara
Para $I <- 1$ **Hasta** 7 **Con Paso** 1 **Hacer**
Escribir $V[I]$
FinPara
FinAlgoritmo

Una ejecución parcial del algoritmo usando como valores de entrada los dados como ejemplo para explicar el funcionamiento del método de ordenamiento a usar es:

después de la lectura	$V[1]$	$V[2]$	$V[3]$	$V[4]$	$V[5]$	$V[6]$	$V[7]$	I	MIN	J	VAL_MI
	21	**35**	**17**	**8**	**14**	**42**	**2**				
ingreso al **para** externo								1^6			
1er iteración sobre I								**1**	**1**		
ingreso al **para** interno										2^7	
1er iteración sobre J										**2**	
2da iteración sobre J									**3**	**3**	
3er iteración sobre J									**4**	**4**	
4ta iteración sobre J										**5**	
5ta iteración sobre J										**6**	
6ta iteración sobre J									**7**	**7**	
salida del **para** interno	**2**	35	17	8	14	42	**21**				**2**
2da iteración sobre I								**2**	**2**		
ingreso al **para** interno										3^7	
1er iteración sobre J									**3**	**3**	
2da iteración sobre J									**4**	**4**	
3er iteración sobre J										**5**	
4ta iteración sobre J										**6**	
5ta iteración sobre J										**7**	
salida del **para** interno	2	**8**	17	**35**	14	42	21				**8**

Queda como ejercicio para el alumno completar la tabla de ejecución del algoritmo.

En la tabla anterior:

- Los valores en negrita representan los valores de las variables que se modifican durante la ejecución que se está analizando.

- Para las variables de control de la repetición, por ejemplo I, al ingresar al PARA externo, usar la notación 1^6 se interpreta: el 1 como valor inicial de dicha variable y el superíndice 6 como el valor final de la misma.