

Tema 13

Estructuras No Acotadas: Ficheros Y Variables Dinámicas

Estructuras de datos no acotadas

Todas las estructuras de datos vistas hasta ahora tienen una característica en común: la capacidad total (número de elementos componentes) se determina explícitamente al definir las.

Las estructuras de datos que no tiene un tamaño fijado de antemano y que pueden ir creciendo o reduciendo su tamaño en función de las necesidades particulares del programa se denominan *estructuras dinámicas*.

La estructura secuencia

Puede definirse como un esquema de datos del tipo iteración, pero con un número variable de componentes. La estructura secuencia resulta parecida a una formación con número variable de elementos.

Se pueden realizar dos tipos de operaciones sobre las secuencias:

- **Operaciones de construcción:**
 - Añadir o retirar componentes al principio de la secuencia.
 - Añadir o retirar componentes al final de la secuencia.
 - Añadir o retirar componentes en posiciones intermedias de la secuencia.
- **Operaciones de acceso:**
 - Acceso secuencial: Las componentes deben tratarse una por una, en el orden en que aparecen en la secuencia.
 - Acceso directo: Se puede acceder a cualquier componente directamente indicando su posición, como en una formación o vector.

Variables dinámicas

Una variable dinámica no se declara como tal, sino que se crea en el momento necesario, y se destruye cuando ya no se necesita. Las variables dinámicas no tienen nombre, sino que se designan mediante otras variables llamadas punteros o referencias.

- **Punteros:** Los punteros o *referencias* son variables simples cuyo contenido es precisamente una referencia a otra variable. En la definición de punteros se ha de indicar el tipo de variable a la que hace referencia.

```
TYPE TpoPuntero =  
    POINTER TO TipoDeVariable
```

Una vez declarado el tipo, se pueden declarar variables puntero de dicho tipo. Una variable puntero se puede usar para designar la variable apuntada en la forma:

```
puntero^
```

Para poder detectar si un puntero señala realmente o no a otra variable, se ha previsto el valor especial NIL, que es compatible con cualquier tipo de puntero, e indica que el puntero no señala a ninguna parte.

- **Uso de variables dinámicas:** Las variables dinámicas no tiene ese espacio de memoria reservado de antemano, sino que se crean a partir de punteros en el momento en que se indique. Se crean mediante la operación:

```
NEW( puntero )
```

Que equivale a:

```
ALLOCATE( puntero, SIZE(TpoDato) )
```

Este procedimiento crea una variable dinámica del tipo al que hace referencia el puntero, y asigna como valor del puntero la referencia a la variable recién creada.

Cuando una variable dinámica ya no se necesita, se puede liberar el espacio que ocupa, dejándolo disponible para crear nuevas variables dinámicas. Se eliminan mediante la operación:

```
DISPOSE( puntero )
```

Que equivale a:

```
DEALLOCATE( puntero, SIZE(TpoDato) )
```

Para poder usar estos procedimientos se ha de realizar la siguiente importación:

```
FROM Storage  
IMPORT ALLOCATE, DEALLOCATE;
```

- **Realización de secuencias mediante punteros:** Se pueden realizar secuencias utilizando la siguiente definición:

```

TYPE TpoPuntero =
    POINTER TO TpoSecuencia

TYPE TpoSecuencia
    RECORD
        nombre: tipo;
        ...
        siguiente: TpoPuntero;
    END;
    
```

- **Operaciones con secuencias enlazadas:** Estas son algunas de las operaciones con secuencias enlazadas:
 - **Recorrido:** Se consigue mediante un bucle de acceso a elementos y avance del cursor. Puesto que la secuencia tiene un número indefinido de elementos, no se usará un bucle con contador.
 - **Búsqueda:** Ha de hacerse en forma secuencial. La búsqueda es parecida al recorrido, pero la condición de terminación será doble: que se localice el elemento y/o que se agote la secuencia.
 - **Inserción:** De un nuevo elemento se consigue creando una variable dinámica para contenerlo, y modificando los punteros para enlazar dicha variable en la mitad de la secuencia.

Datos persistentes

Se denominan datos persistentes aquellos que conservan su valor entre ejecuciones sucesivas de los programas que operan con ellos.

Ficheros

Los ficheros son estructuras de datos no acotadas en memoria externa, que pueden ser manipulados desde un programa, estableciendo una conexión entre dichos ficheros en memoria externa y las variables fichero definidas por el programa en memoria interna. Esta conexión se controla mediante las siguientes operaciones:

- **Abrir:** Establece la asociación entre una variable fichero interna y el fichero externo, designado mediante un nombre.
- **Cerrar:** Elimina la conexión anterior, garantizando que los datos almacenados en la memoria externa quedan debidamente actualizados.

Un *sistema de ficheros* permite tener almacenados en un mismo soporte diferentes ficheros.

- **Ficheros secuenciales:** Tienen las siguientes características:

- El acceso a los elementos ha de ser secuencial.
- El fichero ha de ser usado en uno de sus modos: lectura o escritura.
- El modo lectura se usa la información contenida en el fichero, sin modificarlo.
- En modo escritura el fichero se graba desde el comienzo. La información anterior, si la había, se pierde.

En DOS y UNIX la forma de especificar el fichero sobre el que trabajar es:

```

OrdenAEjecutar < FicheroDeEntrada
                > FicheroDeSalida
    
```

- **Ficheros de texto:** Un *fichero de texto* es en realidad un fichero secuencial de caracteres. La única diferencia está en el modelo lógico de esa secuencia de caracteres, que se considera dividida en *líneas de texto*, separadas unas de otras por caracteres especiales de *fin de línea*. La organización del texto en líneas permite estructurar los datos de entrada y salida, aprovechando los separadores de líneas para marcar la separación de grupos de datos.
- **Lectura y escritura con conversión:** Los procedimientos de lectura estándar ReadInt, ReadCard y ReadReal tienen generalmente el siguiente comportamiento:
 - Al comenzar la lectura, se ignoran los caracteres de espacio en blanco que se vayan leyendo inicialmente.
 - A continuación se leen tantos caracteres como sea posible, hasta encontrar algún espacio en blanco o salto de línea. Este carácter terminal queda ya leído, de manera que las operaciones de lectura que se ejecuten a continuación no lo volverán a leer. Los caracteres leídos se convierten a valor numérico.
 - Si los primeros caracteres no blancos que se encuentren no constituyen una representación válida de un valor numérico, el valor devuelto es arbitrario, y se asigna un valor falso a la variable Done exportada por el módulo InOut (la lectura de valores reales puede no producir este efecto, ya que se define en otro módulo distinto).

Nota: Es mejor no utilizarlos ya que son muy uniformes entre los distintos compiladores.

- **Ficheros de acceso directo:** Son aquellos en que se puede acceder directamente a un elemento de información contenido en el fichero, sin necesidad de acceder uno tras otro a los anteriores.

Existen varios modelos lógicos:

- **Ficheros con organización relativa:** Se especifica la componente a que se accede mediante su posición relativa en el fichero.
- **Ficheros indexados:** Se especifica la componente a utilizar mediante el valor de un campo clave contenido en ella.

Las características de las operaciones de acceso son:

- Se puede acceder directamente a una componente del fichero conociendo su posición relativa.
- El fichero puede ser usado simultáneamente en lectura y escritura.
- Una operación de lectura obtiene la información contenida en una componente de la secuencia, sin modificarla.
- Una operación de escritura permite grabar un nuevo valor en una componente de la secuencia, o añadir una nueva componente al final.
- Se puede truncar el fichero, eliminando las componentes finales, a partir de una cierta posición.