TEMA 11

ORGANIZACIÓN LÓGICA DE LOS DATOS. ESTRUCTURAS ESTÁTICAS.

ÍNDICE

1. INTRODUCCIÓN

Se denomina **dato** a cualquier objeto manipulable por el ordenador. Un dato puede ser un carácter leído de un teclado, información almacenada en un disco, un número que se encuentra en memoria principal, etc.

Los tipos de datos más frecuentes utilizados en los diferentes lenguajes de programación son los **datos simples** (lógico, carácter, entero, real) y los **datos estructurados o compuestos**. Los tipos de datos simples o primitivos no están compuestos de otras estructuras de datos; son los más frecuentes y utilizados por casi todos los lenguajes. Los tipos de datos compuestos se construyen a partir de otros tipos de datos; un ejemplo es la cadena de caracteres.

Una **estructura de datos** es una colección de datos organizada de un modo particular y sobre la que se definen ciertas *operaciones*. Será **homogénea** cuando todos los datos elementales que la forman son del mismo tipo o **heterogénea**, en caso contrario.

Las estructuras de datos pueden ser de dos tipos:

- **Estáticas**

  Son aquellas en las que el tamaño ocupado en memoria se define antes de que el programa se ejecute y no puede modificarse durante la ejecución del programa. Estas estructuras están implementadas en casi todos los lenguajes: **registros**, **ficheros** y **tablas** (o **arrays**).

- **Dinámicas**

  Son aquellas cuya ocupación en memoria puede aumentar o disminuir en tiempo de ejecución.

La elección del tipo de estructura de datos idónea a cada aplicación dependerá esencialmente del tipo de aplicación y en menor medida del lenguaje, ya que si no tiene implementada una estructura, deberá ser simulada con el algoritmo adecuado.

2. NOCIÓN DE REGISTRO

Un **registro lógico** es una estructura de datos formada por elementos que tienen información relativa a un mismo ente. A los elementos que componen el registro se les denomina **campos**. Cada campo es de un tipo determinado. Los campos dentro del registro aparecen en un orden determinado, y se identifican por un nombre. Para definir un registro es necesario especificar el nombre y tipo de cada campo. Los campos pueden ser de un tipo estructurado.

Registro físico será el conjunto de información que, de acuerdo con las posibilidades físicas de la máquina, se graba o se lee de una sola vez. También se le denomina **bloque**.

En general, se puede decir que el registro lógico está enfocado al problema (**software**) y el registro físico a la máquina (**hardware**).

Existirá una relación entre registro lógico y registro físico, en base a los respectivos tamaños, es decir, la cantidad de registros lógicos que estarán contenidos en un registro físico o viceversa. Sabiendo que el registro lógico es lo que se lee o escribe, cuantos más registros lógicos estén contenidos en un registro físico, el número de accesos que se deba realizar al soporte serán menores. Esta relación de tamaños recibe el nombre de **factor de bloqueo**, definiéndose como el número de registros lógicos contenidos en un registro físico y que designaremos como **FB**. El factor de bloqueo puede ser mayor que uno, igual o menor que uno.

3. DISEÑO DE REGISTROS

Independientemente del lugar en que se almacenen, los datos han de constituir unidades homogéneas de información, formadas a su vez por campos o unidades elementales. En otras palabras, los datos tienen que estar constituidos por registros.

Si los registros están contenidos en soportes externos de almacenamiento y queremos que sean procesados, lo primero que se hace es pasar los datos a la memoria interna del ordenador. El paso de información del soporte externo a memoria interna (lectura) se hace utilizando al registro físico como transporte de información. Una vez pasada la información a memoria, ésta es procesada en forma de registros lógicos.

Supongamos que queremos almacenar todos los datos de los empleados pertenecientes a una determinada empresa. El conjunto de todos los datos de todos los empleados formará el fichero. Los datos de cada empleado formarán un registro lógico, y los datos desglosados de cada empleado, tales como el nombre, DNI, etc., formarán los campos.

Cada registro tiene que tener un campo que lo diferencie de los demás. Este campo, denominado clave, nos tiene que servir para discriminar unos registros de otros dentro del fichero.

En algunos registros se pueden distinguir varios niveles de agrupamiento de la información. Estos niveles son los que determinan la estructura lógica de un registro y, en general, esta estructura se compone de una rama principal de la cual penden unas ramas secundarias con la información.

Además de la estructura lógica de los registros, éstos pueden tener distintos tratamientos dependiendo de su longitud. Los registros pueden ser de longitud fija, variable o indefinida. A continuación hablaremos de los distintos tipos de registros atendiendo a su longitud.

3.1. Registros de longitud fija

Reciben este nombre aquellos registros en los que la suma de las longitudes de cada campo (en caracteres), es siempre la misma.

Este tipo de registro se puede diseñar de tres formas:

a) Con igual número de campos e idéntica longitud de cada campo en cada registro

En este caso, la longitud del registro lógico es la misma para todos los registros del fichero. Además, cada registro tiene el mismo número de campos y la longitud de cada uno de estos campos es la misma en todos los registros.

b) Con igual número de campos y distinta longitud de cada campo en cada registro

En este caso, aunque la longitud de cada campo puede variar, el número total de ellos sigue siendo el mismo y la longitud total del registro también.

c) Con distinto número de campos y distinta longitud de cada campo en cada registro

Es evidente que si la longitud final del registro tiene que ser fija y el número de campos que lo forman es variable, la longitud de estos tiene que ser distinta. En este caso, el diseño de registros no sigue ningún rigor a excepción de la longitud final, que siempre será fija.

3.2. Registros de longitud variable

En realidad no responden realmente a lo que su nombre indica, ya que la ocupación real del soporte y de memoria interna siempre será la misma. La variabilidad de la longitud del registro radicará en la mayor o menor ocupación del espacio total dedicado a almacenar el registro. La ocupación total siempre oscilará entre un máximo y un mínimo.

Los registros de longitud variable están formados por una parte de longitud fija (**identificativo**), de unos campos de igual longitud comunes a todos los registros y de una serie de campos de igual longitud pero en número variable. El carácter de variabilidad viene especificado por la ocupación de más o menos campos de este tipo.

Este tipo de organización de registros no es de lo más adecuado, ya que, se desaprovecha espacio, tanto en el soporte como en memoria interna. Esta pérdida de espacio es debida a que no se ocupa totalmente el espacio reservado para almacenar la información de cada registro.

De todas formas, el manejo de este tipo de registros es bastante sencillo, ya que, prácticamente se gestionan igual que los registros de longitud fija.

3.3. Registros de longitud indefinida

Se diseñan generalmente para optimar la ocupación de los soportes de almacenamiento. Este tipo de registros se caracteriza porque su longitud y estructura son totalmente variables.

Necesitarán un espacio auxiliar para almacenar caracteres de control que indiquen los inicios y finales de campo y los de registro.

Se pueden diseñar de tres formas:

a) Por separadores de campos

Se coloca al final de cada campo un carácter especial que delimita su ubicación y se utilizan campos auxiliares para delimitar unos campos de otros.
Para indicar el final lógico del registro se utiliza otro carácter especial. Ninguno de los caracteres especiales formará parte de ningún campo de datos del registro.

b) Mediante indicadores de longitud

Consiste en incluir unos indicadores en los que se especifique cual es la longitud de cada campo componente del registro. La suma de estas longitudes indicará la longitud total del registro.
La longitud total del registro solo aumenta en una pequeña cantidad, sin aumentar la cantidad de información contenida.

c) Mediante máscaras

Se considera **máscara** a un campo fijo, generalmente ubicado en la primera parte del registro y que indica la ausencia o presencia de campos en el mismo. Cada posición de esta máscara puede tener un 1 o un 0 indicando la presencia o ausencia de campos. La máscara estará separada de los datos reales por algún carácter de control.

En general, la utilización de registros de longitud indefinida no es demasiado compleja, aunque sí implica un diseño previo. Los tres diseños anteriores no se suelen utilizar solos, se pueden diseñar registros con máscaras y, además, añadir a cada campo de datos un campo adicional que indique la longitud del mismo.

4. TABLAS O ARRAYS

Un **array** es una estructura de datos formada por una cantidad fija de datos de un mismo tipo, cada uno de los cuales tiene asociado uno o más índices que determinan de forma unívoca la posición de cada dato o elemento en el array. Podemos imaginar un array como una estructura de celdas donde se pueden almacenar valores. Un array viene determinado por:

Componentes: son cada uno de los elementos del array.
Índice: es la posición de cada componente dentro del array y viene determinado por una o varias de dichas posiciones. A cada componente se puede acceder de forma directa indicando su(s) índice(s) y el nombre del array.
Dimensión: es el número de índices que utiliza el array.
Longitud: es el número de componentes que contiene un array.
Tipo: es el tipo de los componentes del array.

Los arrays se clasifican según su dimensión en:

Unidimensionales.
Bidimensionales.
Multidimensionales.

5. ARRAYS UNIDIMENSIONALES: VECTORES

El tipo más simple de array es el array unidimensional o vector. Un vector es un tipo de dato estructurado, con tamaño fijo y elementos homogéneos (del mismo tipo).

Los elementos del array se almacenan en posiciones contiguas de memoria, a cada una de las cuales se puede acceder directamente.

Supongamos que se desean conservar las puntuaciones de 50 estudiantes de un examen de informática. Para almacenar éstas puntuaciones o calificaciones se necesita reservar 50 posiciones en memoria, dar un nombre al array, y a cada uno de los 50 estudiantes asignarles su calificación correspondiente, es decir, dar el índice o subíndice del array.

El índice de un elemento (1,2, .., n) designa su posición en la ordenación del vector. Cada elemento de un vector se puede procesar como si fuese una variable simple al ocupar una posición de memoria. Esto indica que cada elemento de un vector es accesible directamente.

Un vector de una dimensión denominado notas que consta de n elementos se puede representar de la siguiente forma:

Nota(1)	Nota(2)	Nota(3)	...	Nota(n-1)	Nota(n)

6. OPERACIONES SOBRE ARRAYS UNIDIMENSIONALES

Como ya sabemos, un vector es una secuencia ordenada de elementos, cuyo límite inferior es el elemento de posición uno y el último el de posición n. En algunos casos el límite inferior del vector puede ser considerado como de posición cero.

Las operaciones que se pueden realizar con vectores durante el proceso de resolución de un problema son las siguientes:

- Asignación
- Lectura/escritura
- Recorrido (acceso secuencial)
- Actualización (añadir, borrar, insertar)
- Ordenación
- Búsqueda

En general, las operaciones con vectores implicarán el proceso o tratamiento de los elementos individuales del vector.

6.1. Asignación

La asignación de valores a un elemento del vector se realiza mediante una instrucción de asignación del tipo A(10) = 7, ejemplo que asigna el valor 7 al elemento 10 del vector A.

Si se desea asignar valores a todos los elementos de un vector, se deberá recurrir a estructuras repetitivas de programación estructurada (mientras, para, repetir, ..., etc.).

6.2. Lectura/escritura de datos

La lectura/escritura de datos en arrays u operaciones de entrada/salida normalmente se realiza con estructuras repetitivas, aunque también puede hacerse con estructuras selectivas. Las instrucciones simples de escritura solamente indican que estamos introduciendo datos en determinada posición de un array, así como las operaciones de lectura indican que estamos examinando su contenido.

No hay que confundir las operaciones de lectura/escritura sobre arrays con las que podemos realizar en discos o soportes de almacenamiento, ya que, éstas últimas implican forzosamente un posicionamiento sobre el soporte y en el caso de arrays, al estar en memoria, no hay que hacer nada mecánico.

6.3. Acceso secuencial al vector (recorrido)

Se puede acceder a los elementos de un vector para intorducir datos (escribir) en él o bien para visualizar su contenido (leer). A la operación de efectuar una acción general sobre todos los elementos de un vector se le llama recorrido del vector. Estas operaciones se realizan utilizando estructuras repetitivas, cuyas variables de control se utilizan como subíndices del vector. El incremento del contador del bucle producirá el tratamiento sucesivo de los elementos del vector.

Un ejemplo de recorrido de un vector de 10 posiciones sería:

```
desde I = 1 hasta 10 hacer
        leer A(I)
fin-desde
```

6.4. Actualización de un vector*

Al actualizar un vector, podemos añadir elementos, insertarlos o borrarlos.

Se denomina **añadir** datos a un vector a la operación de añadir un nuevo elemento al final del vector. La única condición necesaria para esta operación consistirá en comprobar el espacio de memoria suficiente para el nuevo vector.

La operación de **insertar** un elemento consiste en introducir dicho elemento en el interior del vector. En este caso se necesita un desplazamiento previo hacia abajo para colocar el nuevo elemento en su posición relativa.

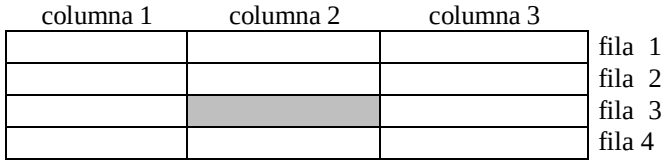
La operación de **borrado** de un elemento en el final del vector no presenta ninguna dificultad adicional; ahora bien, el borrado de un elemento intermedio implica que se tendrá que realizar un desplazamiento de los elementos superiores al borrado, una posición hacia abajo, para reorganizar el vector.

7. ARRAYS BIDIMENSIONALES: MATRICES

El array bidimensional se puede considerar como un vector de vectores. Es un conjunto de elementos, todos del mismo tipo, en el cual el orden de los componentes es significativo y en el que se necesitan especificar dos subíndices para poder identificar a cada elemento del array.

Puede verse como una matriz formada por filas y columnas. Entonces, para localizar o almacenar un valor, especificaremos dos posiciones, una para la fila y otra para la columna. En notación estándar, normalmente el primer subíndice se refiere a la fila del array, mientras que el segundo subíndice se refiere a la columna del mismo.

El siguiente diagrama representa una tabla o matriz A de 12 elementos (4 filas y 3 columnas):



El elemento sombreado se referencia con A(3, 2).

Un ejemplo de definición del array de dos dimensiones puede ser:

```
Type
    A=array[1...4, 1...3] of real;
```

8. ARRAYS MULTIDIMENSIONALES

Un array puede tener tres, cuatro, o más dimensiones. Los conceptos de rango de subíndices y número de elementos se pueden ampliar directamente desde los arrays de una y dos dimensiones a los arrays de orden superior.

En general, un array de n dimensiones requiere que se especifiquen los valores de los n subíndices a fin de identificar un elemento individual. Si cada componente de un array tiene n subíndices, se dice que el array es de n dimensiones.

El número de elementos en un array tridimensional será igual al producto de cada una de las dimensiones.

9. ALMACENAMIENTO DE ARRAYS EN MEMORIA

Al ser el array una estructura de datos estática, cuando se define especificamos el número de elementos que la constituyen. Este dato lo utiliza el compilador para reservar el espacio necesario para almacenarlo. Los arrays se almacenan en memoria ocupando un área contigua. Cada elemento ocupa el

mismo número de palabras, que será el que corresponda al tipo de éstos. Los elementos se colocan en la memoria según un orden prefijado de los índices.

9.1. Almacenamiento de un vector

Se realiza en celdas o posiciones secuenciales.

Si cada elemento del vector ocupa S Bytes (1 Byte = 8 bits) y B es la dirección inicial de la memoria central del ordenador, la dirección inicial del elemento I-ésimo sería:

$$B + (I - 1) * S$$

9.2. Almacenamiento de arrays multidimensionales

Los lenguajes de programación pueden almacenar los arrays en memoria de dos formas: orden de fila mayor y orden columna mayor. El orden de fila mayor es el más natural y utilizado en la mayoría de los ordenadores.

La siguiente figura muestra el almacenamiento de una matriz B(3, 2) (de tres filas y dos columnas), por filas (a) y por columnas (b):

B(1, 1)
B(1, 2)
B(1, 3)
B(1, 4)
B(2, 1)
B(2, 2)
B(2, 3)
B(2, 4)

(a)

B(1, 1)
B(2, 1)
B(1, 2)
B(2, 2)
B(1, 3)
B(2, 3)
B(1, 4)
B(2, 4)

(b)

10. ARCHIVOS

Tanto las estructuras de tablas como las de registros son de tipo internas. Esto lleva a plantear nuevas estructuras que amplíen la capacidad de memoria para los datos: **archivos**.

Algunas definiciones importantes son las siguientes:

Archivo: Es un conjunto de informaciones estructuradas en unidades de acceso denominadas registros, todos del mismo tipo.
Registro: Es una estructura de datos formada por uno o más elementos denominados, campos, que pueden ser de diferentes tipos y que, a su vez, pueden estar compuestos por subcampos.
Clave: Se denomina así a un campo especial del registro que sirve para identificarlo.
Bloque: Corresponde a la cantidad de información que se transfiere en cada operación de lectura o escritura sobre un archivo.
Factor de bloqueo: Es el número de registros que contiene cada bloque.

Las principales *características de esta estructura* son:

Independencia de las informaciones respecto de los programas.
Un archivo puede ser accedido por distintos programas en distintos momentos.
La información almacenada es permanente.
Gran capacidad de almacenamiento.

10.1. Clasificación y organizaciones

Los archivos *se clasifican según su uso* en tres grupos:

Permanentes o maestros: Contienen información que varía poco. En algunos casos es preciso actualizarlos periódicamente.
De movimientos: Se crean para actualizar los archivos maestros. Sus registros son de tres tipos:

altas, bajas y modificaciones.
De maniobra o trabajo: Tienen una vida limitada, normalmente menor que la duración de la ejecución de un programa. Se utilizan como auxiliares de los anteriores.

Los archivos se organizan para su almacenamiento y acceso, según las necesidades de las aplicaciones que los van a utilizar. Las **organizaciones** más utilizadas son:

Secuencial: El almacenamiento se hace en posiciones físicas contiguas.
Directa o **aleatoria**: El almacenamiento se hace a través de un identificativo o clave que indica la posición del registro dentro del archivo.
Secuencial indexada: Se utilizan tablas de índices para acceder a los registros.

10.2. Operaciones usuales

Las **operaciones generales** que se realizan sobre un archivo son:

Creación: Escritura de todos sus registros.
Consulta: Lectura de todos sus registros.
Actualización: Inserción, supresión o modificación de alguno de sus registros.
Clasificación: Reubicación de los registros de tal forma que queden ordenados según determinados criterios.
Borrado: Eliminación total del archivo, dejando libre el espacio del soporte que ocupaba.

Las *operaciones más usuales a nivel de registro* son:

Inserción: Añadir un nuevo registro al archivo.
Supresión: Quitar un registro del archivo.
Modificación: Alterar la información de un registro.
Consulta: Leer el contenido de un registro.

11. BIBLIOGRAFÍA

Alberto Prieto
Introducción a la Informática
Mc Graw-Hill, 2ª edición, 1997

Alfonso Ureña López
Fundamentos de Informática
Ra-ma, 1997

Pascual Laporta, G.
Estructura de la Información
Mc Graw-Hill, 1992