Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología
Sede: Oro Verde

FUNDAMENTOS DE PROGRAMACIÓN

RESUMEN DE CONTENIDOS N° 12

ARCHIVOS

RESUMEN DE CONTENIDOS N° 12

Archivos

Introducción

Se introducirá una nueva estructura de datos llamada **variable compuesta** o **registro**, la cual a diferencia de la estructura arreglo, permitirá agrupar datos de tipos diferentes.
Es muy común encontrar en los sistemas que nos rodean, entidades compuestas de varios elementos, los cuales poseen características diferentes. Los registros apuntan a este propósito.
Conceptualmente, un registro es una asociación de variables elementales, que pueden almacenar datos de distinta naturaleza.

Toda esta primer parte, es teoría "genérica", es decir, independiente del lenguaje de programación empleado. Más adelante, en este mismo apunte, haremos exclusiva referencia a C++.

Registros

Un registro es una estructura de datos que consiste en un número fijo de componentes que pueden ser de distintos tipos.

De acuerdo a la definición, el registro es otro tipo de estructura de datos, y por lo tanto, agrupa datos formando una unidad bajo un mismo nombre colectivo, con la característica de que dichos datos pueden ser de **distinto tipo**. Así por ejemplo un registro puede contener información de un empleado y los datos de ese registro pueden ser: el legajo del empleado, su nombre, su dirección, fecha de ingreso al empleo, estado civil, y otros. Cada uno de estos datos se denominan **campos** del registro. Un campo es la mínima unidad de información de un registro y representa un valor único que puede ser numérico, carácter o lógico.
Todo registro tiene un **nombre** y un **valor**. El nombre seguirá las reglas dadas anteriormente para el nombre de variables simples. El valor del registro está dado por el conjunto de valores de cada uno de sus componentes o campos.

Ejemplo: Supongamos que se tienen datos de N alumnos que han rendido exámenes de distintas asignaturas. Por cada alumno se ingresa:

- ⇨ Nombre y Apellido
- ⇨ Asignatura rendida
- ⇨ Nota obtenida

Una forma de representar los datos de un alumno es identificar cada uno de ellos por medio de variables simples. Así podemos definir un ambiente de la siguiente manera:

VARIABLE	NUM.	CAR.	LOG.	CLASE	SIGNIFICADO
Nombre		X		Simple	Nombre del alumno
Asignatura		X		Simple	Asignatura rendida
Nota	X			Simple	Nota obtenida

Pero en algunos casos, será conveniente organizar los datos como una unidad bajo la estructura de registro. Para ello agruparemos a los distintos datos, componentes del registro, bajo un único nombre, por ejemplo **Examen** y cada uno de esos datos será un campo del registro. Gráficamente:

EXAMEN		
Nombre	**Asignatura**	**Nota**

Donde queda definido el **Nombre del registro EXAMEN** y sus **Campos**
Nombre: caracter
Asignatura: caracter
Nota: numérico

Suponiendo que se le ha asignado valores a cada campo, se tendrá el siguiente registro **EXAMEN**:

EXAMEN		
Nombre	**Asignatura**	**Nota**
'Álvarez José'	'Inglés'	8

Debe destacarse que al indicar el registro **EXAMEN**, se está haciendo referencia al conjunto formado por los tres campos que los forman. Pero también puede ser necesario acceder a cada uno de esos datos en forma independiente y ello se hacea través del nombre de cada campo o variable simple: Nombre, Asignatura o Nota.
El **valor** del registro corresponde al conjunto formado por el valor de cada uno de sus campos. En nuestro ejemplo, el valor del registro será:

'Álvarez José' 'Inglés' 8

donde:

'Álvarez José' es el valor del campo Nombre,
'Inglés' es el valor del campo Asignatura,
8 es el valor del campo Nota.

Diferencias entre Registros y Arreglos

Los Arreglos y Registros, son ambos estructuras de datos, que permiten organizar en cada caso la información de distinta manera y presentan en su manejo algunas diferencias:

- ⇨ Los componentes de un registro pueden ser de distinto tipo, mientras que un arreglo está formado por componentes de igual tipo.
- ⇨ Cada dato que forma un registro es accedido por el nombre de su campo, mientras que en un arreglo, la referencia a un dato se hace a través del nombre del arreglo y de un índice. En nuestro ejemplo inicial, para acceder a la nota obtenida por un alumno lo haremos a través de la componente **Nota** del registro **Examen**.

Componentes de un registro

Hemos visto que un registro puede estar formado por variables simples. Pero, en general, los componentes de un registro pueden ser variables simples y/o registros.
Suponiendo que se dispone de los siguientes datos de varias escuelas de la zona: nombre del establecimiento, director, dirección y ciudad. Podrá definirse al registro **Establecimiento** como sigue:

Establecimiento				
Nombre	**Director**	**Domicilio**		
		Calle	**Nro**	**Ciudad**

véase que está formado por 3 campos: **Nombre, Director, Domicilio**. Pero a su vez, el campo

Domicilio es otro registro formado por 3 variables simples: **Calle, Nro, Ciudad**.

Notación

Los nombres de los campos deben ser únicos dentro de un registro. Pero puede usarse el mismo nombre para indicar componentes de distintos registros.
Esto da lugar a ambigüedades y hace necesario buscar una manera de identificar un dato sin inconvenientes dentro del ambiente.
Si los registros vistos anteriormente: **EXAMEN** y **ESTABLECIMIENTO** formaran parte del mismo ambiente, se debería distinguir los campos que llevan igual nombre. Así, al refrirse al campo **Nombre**, no se sabrá si corresponde a **EXAMEN** o a **ESTABLECIMIENTO**. Para evitar esta dualidad, se designa a una componente, anteponiendo a su nombe el nombre del registro al cual pertenece. Por ejemplo:
Examen.Nombre para referirse al campo **Nombre** del alumno del registro **Examen**.
Establecimiento.Nombre para referirse al campo **Nombre** de la escuela del registro **Establecimiento**.
Sin embargo, no estaría permitido un registro como el siguiente:

Domicilio		
Domicilio	**Dirección**	**Ciudad**

por emplearse el identificador **Domicilio** como nombre del registro y de uno de sus campos.

Operaciones con registros

Los registros pueden intervenir en las siguientes operaciones:

- ⇨ Asignación
- ⇨ Lectura
- ⇨ Escritura

Asignación

El valor de un registro puede ser asignado a otro registro siempre que ambos tengan la misma estructura.
Observación: Un registro puede ser pasado a un procedimiento o función como parámetro, pero el resultado de una función no puede ser un registro.

Lectura y escritura

Las variables registros serán usadas para ***leer*** y ***escribir*** en **ARCHIVOS**, por lo tanto las acciones de lectura y escritura las veremos más adelante.

Importante: Los registros no pueden usarse como operandos de expresiones aritméticas, relacionales o lógicas. Pero sí pueden usarse en expresiones los campos que forman un registro.

ARCHIVOS

Los datos tratados por un programa, como hemos visto hasta ahora, tienen dos limitaciones importantes: por un lado, la cantidad de datos que puede almacenar la memoria principal del procesador (RAM) ya que ésta es limitada; y por otro, la existencia de los datos esta condicionada al tiempo que dure la ejecución del programa, es decir, cuando éste termina todos sus datos desaparecen de la memoria central del procesador y se pierden.
Se utilizan entonces las estructuras de datos externas denominadas **FICHEROS** o **ARCHIVOS** para

manipulación y almacenamiento de datos para usos futuros.

Los archivos no están contenidos en la memoria central del procesador, sino que residen en las denominadas **"memorias auxiliares"** o **"secundarias"**, tales como dispositivos magnéticos (discos duros, cintas), discos ópticos (CDROM, DVD), memorias permanentes de estado sólido (memorias flash USB), etc.

Estas "memorias" permiten guardar datos o disponer de ellos en el momento que sea necesario mediante una escritura o lectura respectivamente.

A modo de ejemplo, se supone que una empresa tiene los siguientes datos por cada empleado:

- ⇨ Nombre y apellido
- ⇨ Documento de identidad
- ⇨ Sueldo

Con ellos, se forma un registro al que se llamará **Empleado**.

Empleado		
Nom	**DNI**	**Suel**

Los campos corresponden a la información del nombre, documento de identidad y sueldo de la entidad empleado.
Si ahora se quisiera operar o manipular los datos de todos los empleados de la empresa, se debería disponer de los registros de todos ellos.
Ese conjunto de registros, es posible reunirlos en una estructura de almacenamiento permanente, denominando a ese conjunto **ARCHIVO**. Cada **archivo se individualizará por su nombre**. Dicho nombre seguirá las mismas reglas que se usa para nombre de variables. En el ejemplo propuesto, podría llamarse a ese archivo **EMPLE**. Por lo tanto, el archivo **EMPLE** estará formado por un conjunto de registros cuya estructura corresponde a la variable registro **Empleado**.

Definimos a un archivo de la siguiente manera:

Un archivo es un conjunto de registros con una estructura común y organizados para un propósito particular.

Un conjunto de archivos relacionados a su vez, constituye una **BASE DE DATOS**.

Organización de los datos de un archivo

Los archivos se organizan para su almacenamiento y acceso, según las necesidades de los procesos que los van a usar.
Existen en general 3 tipos de organizaciones:

1. *Secuencial*
2. *Directa*
3. *Secuencial con índice*

1. SECUENCIAL: cada registro, salvo el primero, tiene otro que le antecede y todos, excepto el último, uno que le sigue.
El orden en que aparecen, es el orden en que han sido escritos.
En este tipo de organización, por lo tanto, el orden físico coincide con el orden lógico.
Para acceder al registro **n** es necesario leer los **n-1** registros anteriores.

2. DIRECTO: no necesariamente el orden físico de los registros se corresponden con un determinado orden lógico que nos propongamos.
Podemos, en este tipo de organización, acceder directamente a un registro, sin necesidad de leer los precedentes.
La información se coloca y se accede aleatoriamente mediante su posición, es decir indicando el lugar relativo que ocupa dentro del conjunto de posiciones posibles.

3. SECUENCIAL CON ÍNDICE: habrá una tabla con índices, que permitirá indicar la localización de un grupo de registros que se encuentran almacenados en forma secuencial. Por ejemplo si la ubicación de los registros de aquellos empleados que empiezan con A se encuentra en la posición X, aquellos que empiezan con B en la posición Y, etc.; para acceder a ALVAREZ primero nos ubicamos en X (posición inicial) y de allí al apellido buscado.

Incluyendo archivos en un programa.

Existen acciones que, independientemente del lenguaje de programación escogido, deben realizarse para el óptimo funcionamiento de los archivos.

Dichas acciones a realizar son:
a- *Crear una variable lógica que maneje el archivo físico en el disco:* Es decir, se tendrá almacenado (o se creará por primera vez) en la memoria secundaria, el archivo, el cual tendrá un nombre y una extensión, y estará almacenado en algún lugar, con lo cual tendrá además una ruta de acceso (root). Este archivo, será la *variable física*, y se deberá conocer su nombre y ruta exactos para que cualquier programa pueda, sobre todo, usar la información almacenada en el mismo.
Pero, en el programa, cuando se procede a codificar y se desea usar un archivo, se utiliza lo que se denomina, una *variable lógica*. Es decir, se da un nombre (siguiendo las pautas para cualquier nombre de variable) a una variable que hará referencia a ese archivo.
Dependiendo de cada lenguaje, variará la forma de vincular ambas variables, pero el concepto no varía.
Con esto, cada persona que programe, podrá utilizar el nombre de variable lógica que desee para hacer referencia al mismo archivo físico.
Ej:

ARCHIVO FÍSICO	ARCHIVO LÓGICO
`C:/PROGRAMAS/proyecto1/empleados.txt`	Programador uno, lo llamará: `ARCHIVO`
	Programador dos, lo llamará: `ARCHI`
	Programador tres, lo llamará: `fichero_a1`

b- *Abrir el archivo:* El término *"Abrir",* se utiliza para dos conceptos: CREAR y OBTENER. Es decir, la acción ABRIR, permite decirle al complilador, qué es lo que se quiere hacer, si crear por primera vez el archivo, o abrirlo para obtener sus datos.
En resumen, se deberá especificar cómo será utilizado el archivo:
- Entrada de datos; o sea, recuperar desde memoria externa la información, mediante la lectura.
- Salida de información; o sea escribir en memoria externa los datos que deseamos almacenar.

Para ello, básicamente se debe hacer: **ABRIR <VAR_LOGICA>** O **ABRIR <VAR_LOGICA> PARA S.**
La indicación *para **E***, indica al procesador que se usará el archivo para la entrada de datos; *para **S***, le indica al procesador la salida de información; la acción ABRIR debe figurar en el programa antes de realizar cualquier operación relacionada con ese archivo.

c- *Leer y Escribir en el archivo:* El término *"Leer",* se utiliza para ingresar los datos del archivo al programa. El término *"Escribir",* se utiliza para grabar los datos procesados en el archivo.
La sintaxis de estas acciones, varía mucho entre los diversos lenguajes de programación, con lo cual, sólo se dejará plasmada la idea central de dichas acciones.

d- *Cierre del archivo:* Luego de haber concluido con el procesamiento de un determinado archivo, se debe proceder a cerrarlo, o sea, la acción contraria a cuando se necesitaba comenzar a usarlo.
Básicamente la acción es: **Cerrar <VAR_LOGICA>**

Operaciones sobre archivos

Vamos entonces a resumir las operaciones más comunes que se pueden realizar sobre archivos secuenciales:

- **Creación**: Escritura de todos sus registros
- **Consulta**: Lectura de algunos de sus registros
- **Apareamiento**: Enfrentamiento de dos archivos para la obtención de un listado, otro archivo, etc.
- **Actualización**: Es un caso especial de apareamiento, donde tenemos dos archivos de entrada y uno de salida con igual estructura de registros que uno de los de entrada.

Para profundizar en las dos últimas operaciones, debemos definir ciertos elementos que en ellas intervienen:

- **Archivo Maestro**: Es un archivo que contiene información permanente de ciertas entidades. Por ejemplo, el archivo de empleados de una empresa, o el de artículos de un almacén, contienen datos de cada entidad existente en dicho lugar.
- **Archivo de Novedades o Movimientos**: Es el archivo que contiene información para actualizar el archivo maestro. Puede o no tener igual estructura de registros que el maestro.
- **Archivo Maestro Actualizado**: Es el archivo obtenido como resultado del proceso de actualización. Contiene información del maestro, actualizada con la del de movimientos. Será, en la siguiente actualización, el archivo maestro de entradas.

Proceso de Apareamiento de archivos

Es el proceso de enfrentamiento de dos archivos para la obtención de un resultado (informe u otro archivo). Ambos archivos deben tener campos que referencien a sus registros, con el mismo tipo de información, y deben estar ordenados por dichos campos. Se puede dar que cualquiera de los dos archivos puede tener registros que no existan en el otro.

Proceso de Actualización de archivos

Dijimos antes que la actualización de archivos es un caso especial de apareamiento, donde intervienen como entrada un archivo maestro y un archivo de novedades, y un maestro actualizado, como salida. El archivo maestro y el maestro actualizado tienen igual estructura de registro.

El proceso consiste en tomar el archivo maestro y actualizarlo con el archivo de novedades, que puede tener registros nuevos, o actualizar los ya existentes en el maestro, generando el archivo maestro actualizado.

ARCHIVOS (FICHEROS) EN C++

Los archivos se pueden clasificar atendiendo a diferentes criterios. En función de la codificación o formato en el que almacenan la información, los archivos se pueden clasificar en: archivos de Texto y Archivos Binarios.

En los archivos de texto, la información se almacena como una secuencia de caracteres y cada carácter se almacena utilizando una codificación estándar (usualmente basada en la codificación ASCII, UTF-8, etc). Al tratarse de un formato estandarizado, otros programas diferentes de aquel que creó el archivo podrán entender y procesar su contenido. Por ejemplo, un programa podría generar un archivo de texto con los datos de las personas de una agenda y posteriormente dicho archivo podría ser entendido y procesado por otros programas. Por ejemplo, podría ser visualizado y editado mediante programas de edición de textos de propósito general, tales como gedit, kate, gvim, emacs, etc. en Linux, textedit en MacOS-X y notepad en Windows, entre otros.

Flujos de Entrada y Salida Asociados a Ficheros

Un programa codificado en C++ realiza la entrada y salida de información a través de flujos (stream en inglés) de entrada y salida respectivamente. En unidades anteriores, se vió cómo realizar la entrada y salida de datos a través de los flujos estándares de entrada y salida (cin y cout, respectivamente), o flujos de E/S o I/O, usualmente conectados con el teclado y la pantalla de la consola. El mismo concepto explicado para E/S de los flujos estándares o la E/S de cadenas de caracteres también es aplicable a los flujos de E/S vinculados a archivos.

En el caso de entrada y salida a y desde archvios, C++ posee mecanismos para asociar y vincular estos flujos con ficheros almacenados en memoria secundaria en el sistema de archivos. De esta manera, toda la entrada y salida de información se realiza a través de estos flujos vinculados a archivos, denominados manejadores de archivos. De este modo, una vez que un programa vincula un archivo con un determinado flujo de entrada o salida, las operaciones de lectura o escritura funcionan como los flujos estándar cin y cout.

Cuando un programa quiere realizar una entrada o salida de datos con un determinado archivo, debe realizar las siguientes acciones:

1. Incluir la biblioteca **`<fstream>`**, que contiene los elementos necesarios para procesar el archivo.
2. Usar el espacio de nombres std. (**`using namespace std;`**)
3. Declarar las variables que actuarán como manejadores de archivos (Variables lógicas).
4. Abrir el flujo de datos, vinculando la variable correspondiente con el archivo especificado. Esta operación establece un vínculo entre la variable (manejador de archivo) definida en el programa y el archivo gestionado por el sistema operativo. De esta forma, toda transferencia de

información entre el programa y un archivo se realizará a través de la variable manejador que ha sido vinculada con dicho archivo.
5. Comprobar que la apertura del archivo del paso previo se realizó correctamente. Si la vinculación con el archivo especificado no pudo realizarse por algún motivo (por ejemplo, si se quiso hacer entrada de datos de un archivo que no existe, o si es imposible crear un archivo en el que escribir datos), entonces la operación de apertura fallaría.

6. Realizar la transferencia de información (de entrada o de salida) con el archivo a través de la variable de flujo vinculada al mismo. En el caso de salida, los datos deberán escribirse siguiendo un formato adecuado que permita su posterior lectura. Por ejemplo, escribiendo separadores adecuados entre los diferentes valores almacenados.
Normalmente, tanto la entrada como la salida de datos se realizan mediante un proceso iterativo. En el caso de entrada dicho proceso suele requerir la lectura de todo el contenido del archivo.
7. Comprobar que el procesamiento del archivo del paso previo se realizó correctamente. En el caso de procesamiento para entrada ello suele consistir en comprobar si el estado de la variable manejador indica que se ha alcanzado el final del fichero (EOF). En el procesamiento para salida suele consistir en comprobar si el estado de la variable manejador indica que se ha producido un error de escritura.
8. Finalmente, cerrar el flujo para liberar la variable manejador de su vinculación con el fichero. Encaso de no cerrar el flujo, éste será cerrado automáticamente cuando termine el ámbito de vida de la variable manejador del fichero.

Nota: es importante tener en cuenta que cuando un flujo pasa al estado erróneo (fail()), entonces cualquier operación de entrada o salida que se realice sobre él también fallará.

Las variables de tipo flujo pueden ser usadas como parámetros de subprogramas. En este caso hay que tener en cuenta que es necesario que, tanto si el archivo se quiere pasar como un parámetro de entrada, salida o entrada/salida, dicho paso de parámetro debe hacerse por referencia (no constante).