

# CONCEPTOS Y CARACTERISTICAS DEL PROCESAMIENTO DE ARCHIVOS

POR: Nina Mamani Teodoro David  
Auxiliar de Docencia(2003)  
INF-131/LAB-131

[nmtdavid@hotmail.com](mailto:nmtdavid@hotmail.com)

[nmdavid@bolivia.com](mailto:nmdavid@bolivia.com)

## CONCEPTOS Y CARACTERISTICAS DEL PROCESAMIENTO DE ARCHIVOS

### Datos y la Información

En diversas ocasiones se usan términos tales como “datos” e “información” de manera indistinta, lo cual nos lleva a un manejo erróneo de dos conceptos diferentes es así que considero muy importante definir o revisar estos conceptos, que son de gran importancia.

#### ¿Qué son los datos?

Desde siempre el hombre ha generado, almacenado, manipulado y usado los datos, porque han tenido la necesidad de observar, de registrar o referir diversos sucesos. Por esto el concepto de dato no es nuevo ni mucho menos originado con la creación del computador.

Formalmente se puede decir que los datos vienen a ser los testimonios de la información que tenemos acerca de un hecho, es decir, son el reflejo de las condiciones en que se una situación dada.

Los datos previenen de diversos eventos tales como efectuar una venta, compra, el ingreso de un estudiante a una universidad, etc. Desde el nacimiento, nuestra vida diaria esta conformada por datos y aunque éstos parecen estar muy dispersos, son recopilados, procesados y consultados de muy diversas maneras o con sistemas diferentes que finalmente, utilizan datos.

### La información

En la Figura 1, aparecen ciertos datos que, si los sometemos a un proceso (combinación de operaciones), pueden generar información.

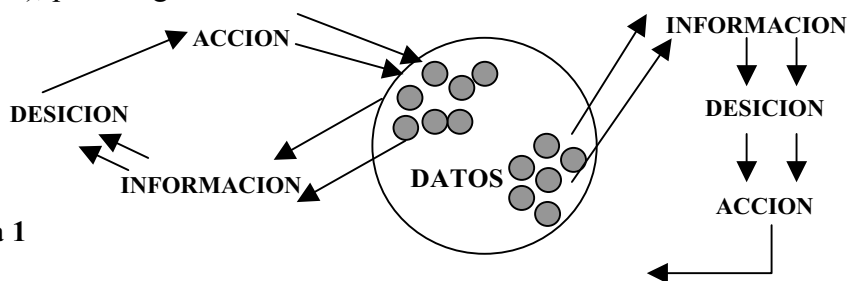


Figura 1

Al asignarle a la información algún modelo de organización, ésta puede acrecentar los conocimientos del usuario y su certidumbre al tomar decisiones, evitando así un sin numero de tentativas lo que, a su vez, permitirá la realización de acciones concretas y seguras que generarán nuevos datos o modificarlos.

#### Administración de los datos

Los datos son una especie de materia prima que, en ocasiones, se puede cosiderar costosas. Den ser administrados de tal manera que sean siempre correctos y estén disponibles para producir información necesaria. Las siguientes operaciones ilustran algunos aspectos del manejo de datos

*Captación* Esta operación se refiere al registro de datos realizado a partir de un evento, en forma de facturas nóminas, modificaciones, y otros.

*Validación* Se refiere a la comprobación de los datos con el fin de mantener su integridad.

*Clasificación* Esta operación clasifica los elementos de los datos en categorías específicas que tienen un sentido para el usuario.

*Almacenamiento* Los datos se guardan en algún dispositivo donde están disponible y pueden ser usados cuando sea necesario.

*Recuperación* Búsqueda y acceso a datos específicos que se encuentran almacenados.

*Protección* esta operación nos permite lograr la seguridad de los datos y mantener su integridad.

Al realizar dichas operaciones podemos aspirar a tener un recurso de datos flexible y capaz de soportar los procesos de toma de decisiones en una organización

Guías generales para un sistema de administración de datos.

1. Los datos deberán ser *representados y almacenados* de tal manera que puedan ser accesibles posteriormente.
2. Los datos deberán ser *organizados* en tal forma que puedan ser selectiva y eficientemente accedidos.
3. Los datos deberán ser *procesados y presentados* en tal disposición que puedan ser un soporte eficiente en el medio ambiente del usuario.
4. Los datos deberán ser *protegidos y administrados* de modo que retengan su valor.

A partir de este momento tratare de que tu obtengas los recursos necesarios para poner en practica las anteriores guías y pueda implementar un sistema de administración de datos.

## LAS ESTRUCTURAS DE DATOS

Mi intención no es tratar este tema de una manera formal, sino simplemente presentar la categorización de las estructuras de datos, ubicar y ubicar dentro de ellas la entidad de interés de este texto guía: el archivo.

Podemos definir lo que es una estructura de datos de dos maneras distintas, aunque complementarias:

- a. Es la forma en que están relacionados objetos de datos o datos complejos.
- b. Es una clase de datos que puede ser caracterizada por su organización y por las operaciones susceptibles de realizar con tales datos.

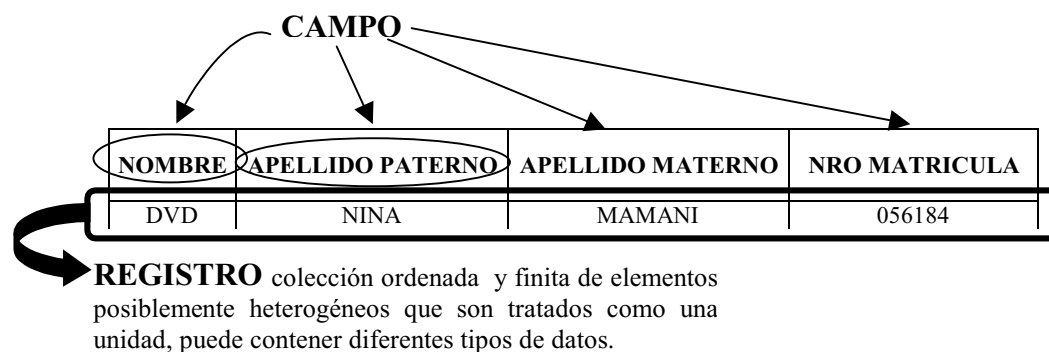
TIPO DE DATOS		ESTRUCTURA DE DATOS		ORGANIZACIÓN DE ARCHIVOS
<u>PRIMITIVO</u>	<u>COMPUESTO</u>	<u>SIMPLE</u>	<u>COMPUESTO</u>	
			<u>LINEAL</u>	<u>NO LINEAL</u>
ENTERO		ARREGLO	FILA	ARBOL BINARIO
BOOLEANO		REGISTRO	PILA	ARBOL BÚSQUEDA BINARIA
CARÁCTER	CADENA		LISTA	Arbol B
REAL				Arbol B* Arbol B+ Arbol Arbol General Grafo
				SECUENCIAL
				RELATIVO
				INDEXADO
				MULTILLAVE

En la tabla se categorizar las diferentes estructuras de datos. Algunas son primitivas, es decir, no se derivan de otras estructuras, y son llamadas comúnmente tipos de datos. Un tipo de dato compuesto está constituido de uno o más tipos de datos primitivos. Los lenguajes de programación proveen un soporte para estos tipos de datos.

Los tipos de datos pueden ser organizados en varias formas para establecer las estructuras de datos simples y estas estructuras pueden ser combinadas para formar estructuras más complejas, algunas veces llamadas estructuras de datos compuestas.

Pero las que nos interesa serán las *Lineales* y *No Lineales*. Los datos pueden ser representados, organizados y almacenados de la forma más eficiente posible, lo cual dependerá obviamente de la aplicación y de la selección de la estructura de datos más adecuada, misma que se transformara según la complejidad que se requiera.

## CAMPO, REGISTRO Y ARCHIVO



Con esto llego a la conclusión de que un conjunto de campos relacionados lógicamente constituye un registro, y un conjunto de registros relacionados lógicamente constituye un archivo.

Existen por lo menos tres buenas razones para estructurar una colección de datos como un archivo, obviamente éstas son diferentes del motivo principal para el uso de un archivo, que es la permanencia de los datos.

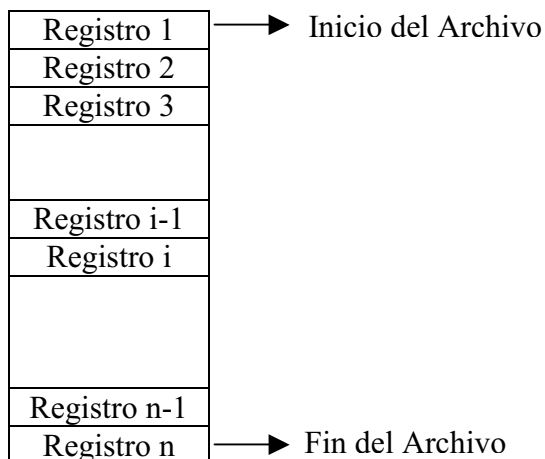
*Primera* Una colección de datos puede ser almacenado como un archivo porque puede ser tan grande para permanecer en memoria principal.

*Segunda* Puede ser almacenada como un archivo porque únicamente una pequeña porción de la colección es accesada por un programa en cualquier tiempo, haciendo incoherente almacenar la colección entera en memoria principal.

*Tercera* Si la colección de datos es muy pequeña, puede ser deseable retener la colección de datos independientemente de la ejecución de cualquier programa particular.

## ARCHIVO SECUENCIAL

La manera más sencilla de organizar una colección de registros que forman un archivo es mediante la organización secuencial. Como se muestra en la Figura 2, los registros son escritos consecutivamente cuando el archivo es creado en su orden lógico secuencial y podrán ser accedidos consecutivamente cuando el archivo sea usado posteriormente como entrada.



**Figura 2**

La característica más importante de este tipo de archivo es que sólo permite el acceso secuencial, esto es, para poder acceder al registro **i** se tiene que recorrer los **i-1** registros que le anteceden en el archivo, una ventaja es que se puede acceder al siguiente registro rápidamente.

Si deseas bajar una implementación realizada en C solo visita

URL: <http://www.geocities.com/exesoftbolivia/prg.html>

Ahí encontraras código fuente y ...

## ARCHIVO Y FLUJO

C++ ve a cada archivo simplemente como una secuencia de bytes. Todo archivo termina ya sea con un *marcador de fin de archivo* o con un número de byte específico que está registrado en una estructura de datos administrativa mantenida por el sistema. Cuando se abre un archivo se crea un objeto y se asocia un flujo con ese objeto. Existen objetos que se crean automáticamente para nosotros, **cin**, **cout**, **cerr** y **clog**. Los flujos asociados con ellos proporcionan canales de comunicación entre un programa y un archivo o dispositivo particular.

**Cin** (objeto de flujo estándar) permite que un programa reciba datos desde el teclado.

**Cout** (objeto de flujo de salida estándar) permite que un programa envíe datos a la pantalla.

**Cerr** y **clog** (objetos de flujo de error estándar) permiten que un programa envíe mensajes de error a la pantalla.

**ifstream** para entrada desde un archivo.

**ofstream** para salida hacia un archivo.

**fstream** para entrada y salida de un archivo.

Los archivos se abren mediante la creación de objetos de esas clases de flujo, las cuales se derivan de (es decir, heredan la funcionalidad de) las clases **istream**, **ostream** e **iostream**, respectivamente.

//Programa: Creación de un archivo secuencial mediante flujo en C++

```
//

|    |
|----|
| CI |
|----|



|        |
|--------|
| NOMBRE |
|--------|



|              |
|--------------|
| APE. PATERNO |
|--------------|



|              |
|--------------|
| APE. MATERNO |
|--------------|


```

//Autor : David Nina <Auxiliar de Docencia Lab-131>

//Web site: <http://www.geocities.com/exesoftbolivia>

```
#include <iostream.h>
```

```
#include <fstream.h>
```

```
#include <stdlib.h>
```

```
#include <ctype.h>
```

```
#include <conio.h>
```

```
#include <string.h>
```

```
#define msg "Archivo Alumno no se puede abrir"
```

```
#define T_llena "22±±°° INTRODUSCA °°±±22"
```

```
#define T_veo "22±±°° DATOS DEL REGISTRO °°±±22"
```

```
/****** Prototipos de Funciones/Procedimientos *****/
```

```
void CreaArchivo_S();
```

```
void ImprimeArchivo_S();
```

```
void BuscaEnArchivo_S();
```

```
void menucito();
```

```
/****** Fin Prototipos de Funciones/Procedimientos *****/
```

```
int main()
```

```
{ int opcion=9;
```

```
clrscr();
```

```
while(opcion != 0)
```

```
{ menucito();
```

```
switch(opcion)
```

```
{
```

```
case 1: CreaArchivo_S();break;
```

```
case 2: ImprimeArchivo_S();break;
```

```
case 3: BuscaEnArchivo_S(); break;
```

```
default:break;
```

```
}
```

```
}
```

```
menucito();
```

```
cin >> opcion;
```

```
}
```

```
return 0; //El destructor de ofstream cierra el archivo
```

```
}
```

```

void menucito()
{ clrscr();
  cout << "22222222 MENU 222222" << endl
    << " 1. Crear Archivo" << endl
    << " 2. Imprimir Archivo" << endl
    << " 3. Buscar Alumno [por CI]" << endl
    << " 0. Salir" << endl
    << "\n\n Elija su opcion ";
}

void CreaArchivo_S()
{
  //El constructor ofstream abre un archivo Alumno.dat
  //Segun los modos de apertura de archivo
  //ios::app escribe toda la salida al final del archivo
  //ios::ate abre un archivo para salida y se mueve al final (= a app)
  //ios::out abre un archivo para salida
  //ios::in abre archivo para entrada
  //ios::trunc Elimina contenido del archivo (= a out)
  //ios::nocreate si el archivo no existe, la operacion apertura falla
  //ios::noreplace si el archivo existe, la operacion apertura falla
  ofstream ArchivoAlumnoSalida("Alumno.dat", ios::app);
  //verificamos si se abrio un archivo
  if (!ArchivoAlumnoSalida)//operador ! sobrecargado
  {
    //cerr mensaje de error definido por el usuario
    cerr << msg << endl;
    exit(1); //Termina el programa, parte de stdlib.h
  }
  unsigned long ci; //Tipo de dato en 32 bits. De 0 hasta 4,294,967,295
  char nombre[20], apellidoP[20], apellidoM[20], op='S';
  while(op != 'N')
  {
    clrscr(); //Limpia texto de pantalla, esta en conio.h
    cout << T_llena << endl;
    cout << "Su C.I.      : "; cin >> ci;
    //Enviando de la variable --ci-- al archivo logico ArchivoAlumnoSalida
    ArchivoAlumnoSalida << ci << " ";
    cout << "Su nombre      : "; cin >> nombre;
    ArchivoAlumnoSalida << nombre << " ";
    cout << "Su apellido Paterno : "; cin >> apellidoP;
    ArchivoAlumnoSalida << apellidoP << " ";
    cout << "Su Apellido Materno : "; cin >> apellidoM;
    ArchivoAlumnoSalida << apellidoM << "\n"; //\n Salto de linea
    cout << "Desea continuar adicionando registros [S/N]";
    //getche captura un caracter desde teclado, muestra el caracter en pantalla
    //toupper convierte a mayusculas, parte de ctype.h
    op=toupper(getche());
  } clrscr();
}

void ImprimeArchivo_S()
{
  //El constructor ofstream abre un archivo Alumno.dat
  //Segun los modos de apertura de archivo
  //ios::in abre archivo para entrada
  ifstream ArchivoAlumnoEntrada("Alumno.dat", ios::in);
  //verificamos si se abrio un archivo
  if (!ArchivoAlumnoEntrada)//operador ! sobrecargado
  {
    //cerr mensaje de error definido por el usuario
    cerr << msg << endl;
    exit(1); //Termina el programa, parte de stdlib.h
  }
  unsigned long ci; //Tipo de dato en 32 bits. De 0 hasta 4,294,967,295

```

```

char nombre[20], apellidoP[20], apellidoM[20];
int c=1;
clrscr(); //Limpia texto de pantalla, esta en conio.h
//Del archivo logico ArchivoAlumnoEntrada a las variable ci, ...
while(ArchivoAlumnoEntrada >> ci >> nombre >> apellidoP >> apellidoM)
{
    cout << c << T_veo << endl;
    cout << "Su C.I.      : " << ci << endl;
    cout << "Su nombre    : " << nombre << endl;
    cout << "Su apellido Paterno : " << apellidoP << endl;
    cout << "Su Apellido Materno : " << apellidoM << endl;
    c++;
    getch();
}

void BuscaEnArchivo_S()
{
    //El constructor ofstream abre un archivo Alumno.dat
    //Segun los modos de apertura de archivo
    //ios::in abre archivo para entrada
    ifstream ArchivoAlumnoEntrada("Alumno.dat", ios::in);
    //verificamos si se abrio el archivo
    if (!ArchivoAlumnoEntrada) //operador ! sobrecargado
    {
        //cerr mensaje de error definido por el usuario
        cerr << msg << endl;
        exit(1); //Termina el programa, parte de stdlib.h
    }
    unsigned long ci, ci_aux; //Tipo de dato en 32 bits. De 0 hasta 4,294,967,295
    char nombre[20], apellidoP[20], apellidoM[20];
    int sw=1;
    clrscr(); //Limpia texto de pantalla, esta en conio.h
    //Del archivo logico ArchivoAlumnoEntrada a las variable ci, ...
    cout << "Introduzca CI a buscar: "; cin >> ci_aux;
    ArchivoAlumnoEntrada >> ci >> nombre >> apellidoP >> apellidoM;
    while(!ArchivoAlumnoEntrada.eof())
    {
        if(ci == ci_aux)
        {
            sw = 0;
            cout << T_veo << endl;
            cout << "Su C.I.      : " << ci << endl;
            cout << "Su nombre    : " << nombre << endl;
            cout << "Su apellido Paterno : " << apellidoP << endl;
            cout << "Su Apellido Materno : " << apellidoM << endl;
        }
        ArchivoAlumnoEntrada >> ci >> nombre >> apellidoP >> apellidoM;
    }
    if(sw)
        cout << "CI no encontrado";
    getch();
}

```