Universidad de Los Andes
Escuela de Ingeniería de Sistemas
Departamento de Computación

PROGRAMACIÓN 2

Clase 12

Junior Altamiranda
altamira@ula.ve

AGENDA

- Datos en el computador
- Archivos en C++
- Archivos secuenciales
- Funciones para acceso secuencial

Los datos

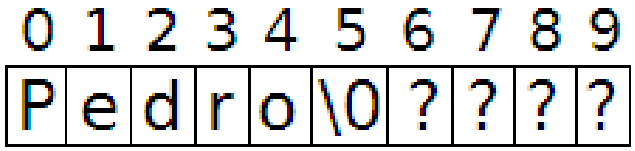
- Un computador es capaz de procesar únicamente voltajes altos y bajos: 1s y 0s
- Un 1 ó un 0 se denominan **bit**
- Para facilitarnos la vida, hemos codificado en bits los números, letras y símbolos: Formando juegos de caracteres
- El juego o código de caracteres más empleado es el ASCII, que utiliza 7 bits para su codificación
- Los humanos usamos estos caracteres para, al realizar nuestras actividades, conformar **Campos** (Grupos de caracteres con significado) o Propiedades

Los datos

- Los campos son utilizados para definir estructuras más complejas: structs o clases, que conforman **registros** o fichas de información
- Un registro es un conjunto de datos relacionados
- Un archivo de trabajo, suele estar conformado por un grupo de registros relacionados
- A un grupo de archivos relacionados, se le suele denominar **Base de Datos**
- A los sistemas encargados de gestionar bases de datos se les denomina **DBMS** (Sistemas Manejadores de Bases de Datos)

Cadenas en C++

```
char nombre[10] = “Pedro”
```



El caracter nulo ‘**\0**’ es lo que diferencia una cadena en C de un vector de caracteres

```
char c1[] = “Hola”;
char c1[5] = “Hola”;
```

es equivalente a

```
char c2[] = “Adios”;
char c2[] = {‘A’, ‘d’, ‘i’, ‘o’, ‘s’};
```

no es equivalente a

Si le quitamos a una Cadena en C el ‘\0’: ¿Cómo sabemos (o printf o cout) dónde termina?

Manipulación de cadenas

```
strcpy(c1,"algo");
```
Copia

```
strcmp(c1,c2);
```
Comparación

Como nueva línea, podemos usar '\n' y endl;

```
cout << texto
```
Escritura por pantalla

```
cin >> variable
```
Lectura desde teclado

```
getline(c1,80);
```
Lectura de cadena con espacios

```
cin.get(c1);
```
Lectura de un caracter

```
cin.put('d');
```
Escritura de un carácter

¿Son iguales '\n' y "\n"?

```
cin.ignore();
```
Para eliminar los \n

Otras funciones (<cctype>)

```
toupper(Char_Exp)
tolower(Char_Exp)
isupper(Char_Exp)
islower(Char_Exp)
isalpha(Char_Exp)
isdigit(Char_Exp)
isalnum(Char_Exp)
isspace(Char_Exp)
ispunct(Char_Exp)
```

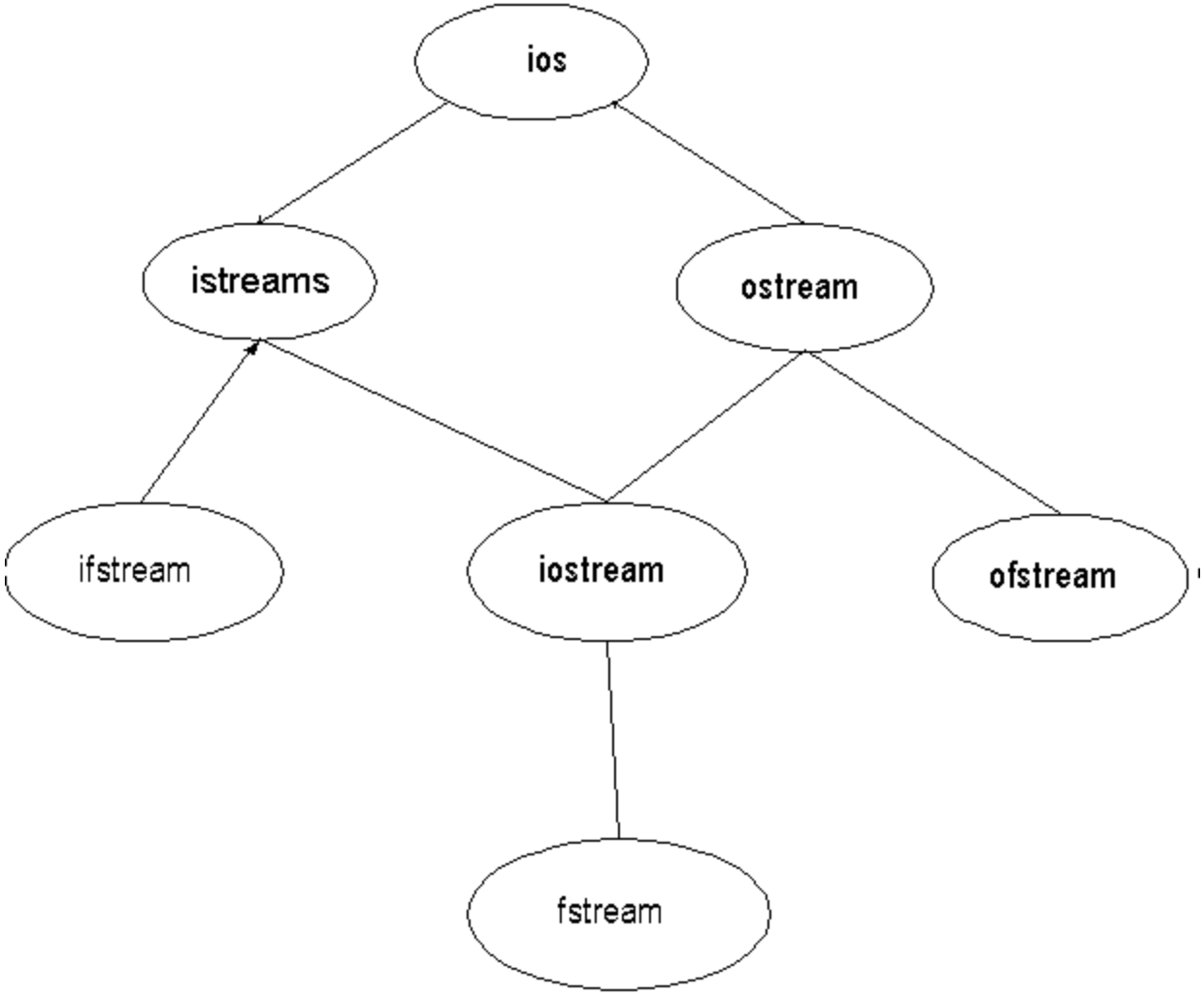
Archivos en C++

- C++ no obliga a que, dentro de un archivo, los registros tengan estar almacenados en cierto orden.
- C++ no sabe qué son registros. El programador es quien estructura el archivo.
- Algo común, es guardar los registros dentro del archivo según el orden que indique el campo clave de los registros. A este tipo de archivos, los denominamos **Archivo secuencial**
- C++ ve a los archivos como secuencias de bytes (streams)

Archivos en C++

- El archivo termina con el carácter fin de archivo (EOF)
- Para poder utilizar un archivo, se debe “Abrir”; lo cual crea un objeto **stream** asociado.
- Este stream permite leer y escribir datos al archivo
- C++ crea automáticamente 4 objetos:
  - cin, cout (Objetos de flujo de entrada y salida)
  - cerr y clog (Objetos de flujo de error)

Jerarquía de clases de E/S de flujo



Acceso secuencial

(Lectura)

```
#include <fstream>

int main() {
    ifstream <variableFile>("archivo.ext",modo);  // ios::in
    while (<variableFile> >> var1 >> var2) {
        // procesamiento
    }
    <variableFile>.close();   // Recomendable
}
```

Otra forma: (¿La usamos cuándo?)

```
fstream <variableFile>;
<variableFile>.open("archivo.ext",modo);
```

Validar la apertura del archivo

```
if(!<variableFile>) {  // retorna != 0 si error
   cerr << “Error al abrir el archivo” << endl;
   return;
}

// Otra forma:
if(<variableFile>.fail()) {
   cerr << “Error al abrir el archivo” << endl;
   return;
}
```

Fin de archivo

Se debe chequear siempre si se ha alcanzado el fin del archivo, para ello, utilice la función eof():

```
<ClaseArchivo> <variable>;
// Se debe haber leído algo antes de probar eof()
while(!<variable>.eof()) {
   . . .
}
```

Acceso secuencial
(Escritura)

```
#include <fstream.h>

int main() {
    ofstream <variableFile>("archivo.ext",modo); // ios:out
    // Utilización del archivo
    <variableFile> << var1 << ' ' << var2 << '\n';
    <variableFile>.close();   // Recomendable
}
```

Otra forma: (¿La usamos cuándo?)

```
fstream <variableFile>;
<variableFile>.open("archivo.ext",modo);
```

Modos de apertura

ios::app	Agrega todo contenido al final del archivo. La 1ra y las veces siguientes.
ios::ate	Al abrir el archivo, coloca el punto de escritura como el fin del archivo.
ios::in	Abre el archivo para leer datos desde este
ios::out	Abre el archivo para escribir datos en este
ios::trunc	Garantiza un archivo limpio. Igual que ios::out
ios::nocreate	Falla si el archivo no existe
ios::noreplace	Abre el archivo para escribir datos en este