# Práctica 7: Estructuras Estáticas Avanzadas

# Arrays multidimensionales, cadenas de caracteres, enumerados y registros.

## Recordatorio de la Teoría.

## Arrays Multidimensionales

A veces, es útil y necesario poder declarar arrays de más de una dimensión. Esto se corresponde con estructuras de datos muy utilizadas como matrices, tablas, cubos, etc. Formalmente, pueden ser declarados como tipos y luego usarlo para declarar variables:

```
typedef <tipo> <nomTipo> [<valor_cte>]{[<valor_cte>]};
```

Ejemplo:

```
const int DIM1=3;
const int DIM2=4;


typedef int TMatriz[DIM1][DIM2];
```

y posteriormente

```
TMatriz m;
```

El acceso a las componentes de una **variable** *m* de tipo T*Matriz* se realiza indicando sus índices entre corchetes. Por ejemplo, `m[i][j]` o bien `m[1][2]`

- **C++ SIEMPRE PASA LOS PARÁMETROS DE TIPO ARRAY POR REFERENCIA, AUNQUE POR LEGIBILIDAD COLOCAREMOS EL SIMBOLO & CUANDO QUERAMOS EXPRESAR QUE EL PARÁMETRO SE VA A MODIFICAR.**
- **C++ NO PERMITE IMPLEMENTAR FUNCIONES QUE DEVUELVAN ARRAYS**

## Cadenas de Caracteres (Strings).

Una cadena de caracteres no es más que un array de caracteres.

**EN C++ TODAS LAS CADENAS DE CARACTERES DEBEN LLEVAR EL CARÁCTER 0 COMO TERMINADOR.**

Veamos a continuación una posible definición de cadenas de caracteres:

```
// Zona de Declaración de Constantes
const char  FINCAD = char(0)[1];
const int   MAXCAD = 20;

// Zona de Declaración de Tipos
typedef char TCadena[MAXCAD+1]; // MAXCAD caracteres + FINCAD
```

## Entrada/Salida de Cadenas de Caracteres.

[1] También es válido el uso de '\0' para indicar el carácter constante 0

**Uso de `cin`**

El uso de `cin` de manera estándar tiene 2 problemas:

- La cadena es leída hasta que se encuentre `ENTER`, espacio o tabulador. Esto hace que se puedan perder datos, por ejemplo, si una persona tiene dos nombres, al usar `cin >> nombre`, se estaría perdiendo el segundo de ellos.
- Lo que no se ha leído, se queda almacenado en el buffer de teclado, por lo que si no queremos que se mezcle con otras lecturas debemos ignorarlo. Ese es el motivo de la sentencia `cin.ignore(MAXCAD,ENTER),` que lo que hace es ignorar lo que haya en el buffer de teclado hasta un máximo de caracteres o un separador (`ENTER` en nuestro caso).

**Uso de `cin.getline(TCadena &cadena, int max_caracteres, char separador)`**

Esta función será la que utilicemos normalmente (salvo que se diga explícitamente que no se puede usar) para leer cadenas de caracteres, ya que, aunque debemos pasarle como parámetros tanto el número máximo de caracteres como el separador de cadenas, la llamada `cin.getline(s, MAXCAD, ENTER)` tiene exactamente el mismo funcionamiento que la función LEER definida para el pseudolenguaje.

**Uso de la Función LeeCadena**

El uso de esta función es equivalente a la anterior, lo que se hace es leer carácter a carácter. La podremos usar siempre, aunque normalmente, la utilizaremos cuando se pida explícitamente, es decir, cuando se diga que no se puede usar el `cin.getline`.

# Tipos Enumerados.

El programador pueda definir sus propios tipos de manera que especifique literalmente los valores que una variable de ese tipo puede tomar. Estos son los llamados tipos enumerados. Así, por ejemplo, si quisiéramos definir una variable para que represente una situación de un problema en el que hay cinco posibles colores, (rojo, amarillo, verde, azul y naranja)

```
typedef enum
{
  negro, azul, rojo, verde, amarillo, azul, rosa, negro
} TColor;
```

La declaración de un tipo de esta índole consiste en asociar a un identificador una enumeración de los posibles valores que una variable de ese tipo puede tomar, por lo que una misma constante no puede aparecer en dos definiciones de tipo. Formalmente, la definición de un enumerados en formato BNF sería:

<TipoEnumerado>::= **typedef enum** '{ '[2] < literal > {, <literal > } '}' <nomTipo>;

El orden de los valores de estos nuevos tipos declarados por el programador será aquel en que aparecen dentro de la lista de enumeración de los elementos del tipo. El tipo enumerado es también un tipo escalar y ordinal, por lo que permite calcular su ordinal y valor.

**NOTA. Recuerde que normalmente hay que implementar las operaciones SUCESOR y/o PREDECESOR, y que NO PUEDEN SER LEIDOS/ESCRITOS DIRECTAMENTE.**

[2] '{' simboliza el carácter llave, para distinguirla del operador de iteración.

# Registros

Un registro es una estructura cuyos componentes pueden ser de diferente tipo. Para declarar un tipo registro se debe incluir el nombre y el tipo de cada componente del mismo. En **C++** se usa la palabra reservada **struct** para indicar que el tipo que se está definiendo es un registro o estructura. Los campos de un registro pueden ser de cualquier tipo, incluyendo arrays y registros y son llamados campos o variables miembro que pueden ser accedidas individualmente mediante notación '.' Formalmente, podemos definir un registro de la siguiente manera:

```
<TipoRegistro>::= struct <nombreTipo>
                  '{'
                       { <tipo> <nomVar> {, <tipo> <nomVar>}; }
                  '}';
```

Ejemplo:

```
struct TFecha
{
  int dia, mes, anho;
};
```

Las declaraciones de variables se realizan normalmente, es decir:

```
TFecha f1;
```

Los nombres de los campos de un registro son *locales* a él, por lo que no hay conflicto con otros nombres usados en el módulo. Una referencia a un campo de un registro consiste en el nombre de la variable registro y el nombre del campo, separados por un punto, por ejemplo:

```
f1.dia = 30;
```

**Ejercicio 1.** Implementa un procedimiento que concatene 2 cadenas de caracteres, de tal manera que la segunda quede "pegada" detrás de la primera. Posteriormente, escriba un programa que lea 2 cadenas y muestre su concatenación por pantalla. Luego escribe un programa que lea 2 cadenas, las concatene y muestre el resultado en pantalla.

```
void append(TCadena &s1, TCadena s2)
```

**Ejercicio 2.** Escriba un programa que lea un NIF y compruebe que su letra es correcta, es decir, la letra introducida por el usuario se corresponde con la calculada.
Algoritmo para el cálculo del NIF:

1. Calcule el Módulo 23 (resto de la división) del número del DNI.
2. La letra corresponde a la que haya para ese valor en la siguiente tabla:

| MODULO | LETRA | MODULO | LETRA |
|---|---|---|---|
| 0 | T | 12 | N |
| 1 | R | 13 | J |
| 2 | W | 14 | Z |
| 3 | A | 15 | S |
| 4 | G | 16 | Q |
| 5 | H | 17 | V |
| 6 | Y | 18 | H |
| 7 | F | 19 | L |
| 8 | P | 20 | C |
| 9 | D | 21 | K |
| 10 | X | 22 | E |
| 11 | B | | |

**Ejercicio 3.** Escriba un programa en que se declare un tipo fecha con un registro como se ha visto conanterioridad. EL programa deberá leer 2 variables de tipo fecha y decir cuál es la mayor de las dos.

**Ejercicio 4.** ***"Las 7 y Media".*** Este popular juego de cartas es para 2 jugadores (humano y máquina en nuestro caso). El juego consiste en acercarse lo más posible (sin pasarse) al valor de 7.5 sumando el valor de todas las cartas de un jugador. El juego se desarrolla con una baraja española ( valores del 1 al 7 más sota, caballo y rey con cuatro palo: oros, copas, espadas y bastos, en total 40 cartas). La puntutación de todas las cartas viene dada por su número, mientras que todas las figuras (sota, caballo y rey) valen 0.5 puntos. El funcionamiento del programa será el siguiente:

1. Se sacan 2 cartas de la baraja: Una para cada Jugador y se muestran por pantalla.
2. El jugador humano podrá pedir cartas sucesivamente hasta que decida plantarse o supero los 7.5 puntos, en cuyo caso ya habrá perdido.
3. El jugador máquina, que ya sabe la puntuación obtenida por el jugador humano, hace lo siguiente:
   - 3.1. Si el humano se ha pasado, se planta y gana.
   - 3.2. Si el humano no se ha pasado, saca cartas hasta superar o igualar la puntuación del humano o pasarse (en cuyo caso ha perdido la máquina).

Veamos un ejemplo de Ejecución:

```
Jugador : as de bastos
Maquina : sota de copa
Total de Puntos = 1
Otra Carta (S/N)? s
sota de espadas
Total de Puntos = 1.5
Otra Carta (S/N)? s
caballo de copas
Total de Puntos = 2
Otra Carta (S/N)? s
rey de espadas
Total de Puntos = 2.5
Otra Carta (S/N)? s
cinco de espadas
Total de Puntos = 7.5
Otra Carta (S/N)? n
dos de espadas
tres de espadas
tres de bastos
Yo tengo  8.5 puntos.
Ops .. me he pasado ..
Tu tienes 7.5 puntos.
Tu GANAS!!
Otra Partida (S/N)?
```

Nota. Para barajar las cartas intercambiarenos un número determinado o no de veces dos cartas de forma aleatoria y para ello deberemos usar estas dos funciones.

```
veces  aleatorio(número de cartas)

PARA i 1 HASTA veces  HACER
      x  aleatorio(número de cartas)
      y  aleatorio(número de cartas)

      Intercambiar la Carta que ocupa la
      posición x con la que ocupa la
      posición y

FINPARA
```

**Código Fuente 1 Algoritmo para Barajar Cartas**

```
/*-------------------------------------------------------------
|     Autor:                                                   |
|     Fecha:                          Versión: 1.0             |
|-------------------------------------------------------------|
```

```
/*  ------------------------------------------------------------
    |     Programa para Jugar a las 7 y media                  |
    |   Se han eliminados los acentos y caracteres españoles en la    |
    |     E/S para que se vea bien en una consola de MS-DOS            |
    |  -----------------------------------------------------------*/

#include <iostream.h> // E/S estándard
#include <stdlib.h>   // Librería estándard
#include <ctype.h>    // Necesario para la función toupper
#include <math.h>     // Necesario para las funciones rand y srand
#include <time.h>     // Necesario para la función time

// Zona de Declaración de Constantes

// Zona de Declaración de Tipos

// Zona de Cabeceras de Procedimientos y Funciones

void inicializa_aleatorios();
int aleatorio(int max);

// Programa Principal
int main()
{
  // Deeclaración de Variables del Programa principal

  inicializa_aleatorios();



  system("pause");
  return 0;
}

// Implementación de Procedimientos y Funciones

}

void inicializa_aleatorios()
{
  /* Seed the random-number generator with current time so that
     the numbers will be different every time we run.
   */
  srand( (unsigned)time( NULL ) );
}

int aleatorio(int max)
{
  int ale;

  ale = (rand())%max;
  return ale;
}
```