

**Objetivo de la práctica:**

- Aprender a utilizar el tipo `string` y las funciones definidas sobre él.
- Declarar y utilizar el tipo de dato estructurado registro (o `struct`).

NOTA 1: Durante la práctica todos los ejercicios deberán ser guardados **temporalmente** en el directorio `\tmp`. Una vez finalizada la misma y transferidos los ficheros a un disquete, se deberá eliminar dicho directorio.

**Conceptos básicos: tipo string.**

Las cadenas de caracteres (strings) permiten la manipulación de textos. Un string se puede considerar un vector de caracteres y en el que la manipulación de los mismos se realiza mediante unas operaciones añadidas. En C++, las cadenas de caracteres se representan mediante el tipo `string`. Para su uso es preciso utilizar `#include <string>`.

**Operaciones básicas definidas para string.**

- Declaración de variables de tipo `string`:
  - o `string palabra, frase;`
- Asignación:
  - o `frase = palabra;`
  - o `frase = "hola";`
- Comparación lexicográfica (`==`, `!=`, `<`, `>`) producen como resultado un `bool`
  - o `if ( frase == palabra ) ...;`
  - o `if ( frase > palabra ) ...;`
- Lectura/Escritura:
  - o `cin >> palabra;`
  - o `getline(cin, frase); //lee de cin hasta el fin de línea y`  
`//almacena en frase`
  - o `cout << frase << endl;`
- Manipulación de cadenas (asumimos que en los ejemplos `palabra` y `frase` son de tipo `string`, y que `i` es una variable de tipo `int`)
  - o Número de caracteres en `palabra`:
    - `i = palabra.length();`
  - o Insertar una cadena en otra cadena en una posición:
    - `frase.insert(i, palabra);`
  - o Concatenar una cadena con otra:
    - `frase = palabra + "hola"`
  - o Borrar un número de caracteres de una cadena desde una posición inicial:
    - `frase.erase(inicio, numero);`
  - o Sustituir un número de caracteres de una cadena desde una posición inicial por los de otra cadena:
    - `frase.replace(inicio, numero, palabra);`
  - o Búsqueda de una subcadena dentro de otra cadena obteniendo la posición en la que se encuentra o `-1` si no se encuentra:
    - `i = frase.find(palabra);`
  - o Obtención de una subcadena de un número de caracteres comenzando en una determinada posición:
    - `palabra = frase.substr(i, numero);`



## Conceptos básicos: Registros.

Una estructura es una colección de datos, denominados campos, que representan información relevante sobre una entidad. Cada uno de los campos puede pertenecer a un tipo de dato diferente. El acceso a los datos se realiza mediante el nombre del campo (y no mediante un índice como en los vectores).

Ejemplo de declaración de un registro:

```
// Agrupa la información de interés sobre un empleado de una empresa:
struct empleado
{
    string nombre;
    float salario;
    string num_telefono;
}
```

### Operaciones básicas:

- Creación de una variable estructurada:
  - o empleado emp1, emp2;
- Asignación de registros:
  - o emp1 = emp2;
- Acceso a un campo de la estructura (operador punto):
  - o emp1.nombre = "Jose";
- La lectura y escritura de la información de la estructura se debe hacer campo a campo:
  - o cin >> emp1.nombre;
  - o cin >> emp1.salario;
  - o cin >> emp1.telefono;
  - o cout << emp1.nombre << endl << emp1.telefono << endl;

- Una estructura puede tener como campos otras estructuras:

```
struct departamento
{
    string nombre;
    int numero_empleados;
    empleado jefe_departamento;
}
```

- Es posible definir vectores de estructuras:
  - o typedef empleado Plantilla[30];
  - o Plantilla plan; // plan es del tipo vector de empleado
- En este caso se puede acceder a un campo del registro correspondiente a una posición del vector del siguiente modo:
  - o cout << plan[posición].salario;



## SOBRE LA PRÁCTICA ANTERIOR:

Prueba el programa en el fichero `mastermind.cpp` (accesible en la página web de la asignatura); si te parece que no funciona correctamente modifícalo hasta que funcione como debería.

## EJERCICIOS:

*NOTA: Se valorará la correcta división en módulos de todos los ejercicios planteados.*

1) Escribe un programa que pida dos palabras y diga si riman o no. El criterio que vamos a utilizar es que las dos palabras riman si coinciden sus tres últimas letras, riman un poco si coinciden las dos últimas y no riman en caso contrario. Ejemplo:

```
Palabra: ojo
Palabra: viejo
Las palabras ojo y viejo riman un poco.
```

```
Palabra: cangrejo
Palabra: pellejo
Las palabras cangrejo y pellejo riman.
```

```
Palabra: farola
Palabra: calle
Las palabras farola y calle no riman.
```

2) Escribir un programa en C++ que indique cuántas veces se repite una determinada palabra en una frase. Ejemplo:

```
Palabra: el
Frase: el programa más visto el lunes fue el documental sobre el cambio climático.
Numero de repeticiones: 4
```

3) Escribir un programa que sustituya todas las apariciones de un grupo de caracteres en una frase por otro grupo. Ejemplo:

```
Frase: hola hondo hongo
Caracteres: ho
Nuevos caracteres: co
Frase resultado: cola condo congo
```

4) Escribir un programa que convierta un string a entero. El programa devolverá también un `bool` indicando si la conversión se ha podido realizar con éxito o no. Un string se puede convertir a un número entero si está formado exclusivamente por los caracteres '0', '1', ..., '9'. Para convertir un carácter a entero se le resta el código del carácter '0'. Utilizar este programa para sumar dos enteros leídos como `string`.

5) Hacer un programa para realizar operaciones sobre vectores en un espacio de dos dimensiones. Se asume que el origen es el punto (0,0), de forma que los vectores quedarán especificados por las coordenadas del extremo (x,y) almacenadas en un registro. El programa pedirá las coordenadas de dos vectores y calculará la distancia entre sus extremos, el vector suma, el vector resta y los mostrará por pantalla.

6) Hacer un programa para almacenar la información sobre jugadores de un equipo de baloncesto. Se utilizará un registro para almacenar la información sobre el nombre del jugador, los minutos que



ha jugado, las canastas de tres puntos que ha lanzado y las que ha encestado y lo mismo para las de dos puntos y los tiros libres.

Se utilizará un vector para almacenar la información de los 5 jugadores del equipo. El programa indicará, tras introducir datos, la información completa sobre el jugador con mejor porcentaje con tiros de tres puntos, la información completa sobre el jugador que más puntos ha introducido y la información completa sobre el jugador con más puntos por minuto jugado.