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*)

Conceptos básicos: tipo *string*

Las cadenas de caracteres (strings) permiten la manipulación de textos. Generalmente, se considera que un string no es más que un array de caracteres. Por esa razón, se suele relacionar el concepto de string con el de array. En C++, existe el tipo (clase, en la nomenclatura de los lenguajes orientados a objetos) *string*. Para su uso es preciso utilizar **`#include <string>`**, no **`<string.h>`**.

Operaciones básicas definidas para *string*:

- Creación de variables:

```
string palabra, frase;
```

- Asignación:

```
frase = palabra;
frase = "hola";
```

- Acceso a los caracteres (como arrays):

```
palabra[0]
```

- Comparación lexicográfica (**`==`**, **`!=`**, **`<`**, **`>`**):

```
frase == palabra
frase > palabra
```

- Lectura/escritura:

```
cin >> palabra;
getline (cin, frase);    //lee frase de cin hasta encontrar el fin de línea
cout << frase << endl;
```

- Manipulación de textos (suponer **`unsigned int i;`**):

```
// nº de caracteres de palabra
i = palabra.length();

// inserta palabra en la posición 3 de frase
frase.insert(3, palabra);

// concatena (une) palabra y "hola" y almacena el  resultado en frase
frase = palabra + "hola";

// concatena (añade al final) palabra a frase
frase += palabra;

// borra 7 caracteres de frase desde la posición 3
frase.erase (3,7);

// sustituye (reemplaza) 6 caracteres de frase, empezando en la posición 1,
// por la cadena palabra
frase.replace (1, 6, palabra);

//busca palabra como una subcadena dentro de frase, devuelve la posición
//donde la encuentra
i = frase.find(palabra);

//devuelve la subcadena formado por 3 caracteres desde la posición 3 de
```

```
//frase
palabra = frase.substr(5,3);
```

Conceptos básicos: Registros

Una estructura es una colección de elementos de información denominados miembros (o campos) que representan información relevante sobre una entidad. Cada uno de estos campos puede pertenecer a un tipo de dato diferente y el acceso se realiza mediante el nombre del campo (y no mediante un índice como en los *arrays*). Ejemplo:

```
// información de interes sobre un empleado de una empresa
struct empleado
{
   string nombre;
   long int salario;    // salario anual en pesetas
   string num_telefono;
};
```

Operaciones básicas

- Creación de una variable estructurada.

```
empleado jose, antonio;
```

- Asignación de estructuras.

```
jose = antonio;
```

- Acceso a los datos contenidos en la estructura (operador 'punto'.)

```
jose.nombre = "Jose";
```

- La lectura y escritura de la información de la estructura se debe hacer campo a campo.

```
cin >> jose.nombre;
cin >> jose.salario;
cin >> jose.telefono;
cout << antonio.nombre << antonio.telefono << endl;
```

Una estructura puede tener como campos otras estructuras:

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

También es posible definir *arrays* de estructuras:

```
typedef empleado plantilla [30];
```

Una función puede devolver una estructura:

```
empleado PedirDatos (void);
```

Utiliza los conocimientos aprendidos en las prácticas anteriores especialmente utilizando funciones.

EJERCICIOS

(*)1. Escribir un programa que convierta a mayúsculas todas las letras de una frase dada por teclado y muestre por pantalla la conversión.

(*)2. Realizar un programa en C++ que pida una frase y nos indique si es o no palíndroma (se lee igual de izquierda a derecha que de derecha a izquierda). Por ejemplo:

```
Entrada → dabale arroz a la zorra el abad
Salida  → cierto
```

(*)3. Escribir un programa en C++ que dada una cadena de caracteres permita sustituir todas las ocurrencias de una subcadena. Ejemplo:

```
Cadena: Este problema es mas interesante que el problema anterior.
Vieja Subcadena: problema
Nueva Subcadena: programa
Cadena Resultado: Este programa es más interesante que el programa anterior.
```

4.a. Escribir un programa en C++ que calcule el número de vocales que contiene una frase introducida por teclado y nos diga cuantas vocales de cada clase hay (cuantas 'a', cuantas 'e', ...)

4.b. Escribir un programa en C++ que calcule el número de letras de cada clase que contiene una frase introducida por teclado y nos diga cuantas hay (cuantas 'a', cuantas 'b', ...)

(*)**5.** Un `punto` es un elemento compuesto por tres coordenadas `x`, `y` y `z`. Realizar un programa que pida dos `puntos`, los guarde en dos variables de tipo `punto` y calcule y muestre, sin menús, la suma y la distancia entre las dos variables.

(*)**6.** Realizar un programa en C++ que permita, a través de un pequeño menú realizar operaciones (suma, resta, multiplicación y división) con números complejos. Los números complejos serán guardados en registros. Antes de aparecer el menú se pedirán dos números complejos, y tras cada operación se mostrará el resultado de la misma.

(*)**7.** Una manera de definirse matrices generales es mediante la utilización de una matriz 'suficientemente grande' y dos índices que nos indiquen cuántas de las filas y columnas de la matriz están siendo utilizadas realmente. Con ello una matriz podría definirse mediante el siguiente registro:

```
const int MAX = .??.;

typedef tipo array [MAX. MAX];

struct matriz
{
   array mat;
   int fil, col;
};
```

Aprovechando esta característica y el ejercicio 4 de la práctica 6, realiza un programa en C++ que lea matrices generales, de un máximo de 10x10 elementos), las multiplique, si es posible, y muestre el resultado.

8. Aprovechando los ejercicios 2 y 3 de esta práctica, realiza un programa en C++ que sea capaz de multiplicar matrices generales de números complejos.

(*)**9.a.** Un itinerario de un viajante se puede aproximar por una línea poligonal, en la que cada segmento se define con un par de valores:
- distancia recorrida (d).
- ángulo en radianes respecto del Norte (a).

  - Escribir un programa en C++ que pida pares de valores y los guarde en un vector de estructuras. El programa parará de pedir valores cuando introduzcamos un valor de distancia negativo.
  - Escribir una función a la que le pasaremos el vector y el número de pares introducido y nos devuelva la distancia total recorrida por el viajante.

9.b. Realizar un subprograma en el ejercicio anterior que nos diga en qué punto está el viajante si salió del punto (0,0).

10. Realizar un programa en C++ que guarde en un vector las fichas de los libros de una biblioteca. Las fichas contendrán el nombre del libro, el autor y un número de identificación, relacionado con el tema del libro (Teatro: 001, Narrativa: 002, ...)

Realizar un subprograma para dar de alta nuevos libros y otros subprogramas para buscar libros por el nombre del autor, el titulo del libro o el tema.