Objetivo de la práctica:
- Aprender a utilizar el tipo cadena (o **string)** y las funciones definidas sobre él.
- Declarar y utilizar el tipo de dato estructurado registro (estructura o **struct**)

Conceptos básicos: tipo string

Las cadenas de caracteres (strings) permiten la manipulación de textos. Un string se puede considerar un vector de caracteres con unas operaciones añadidas. En C++, las cadenas de caracteres se representan mediante el tipo *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 con vectores):
```
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;**):

```
// indica si la frase es vacía
vacía = frase.empty();

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

Conceptos básicos: Registro

Una estructura es una colección de elementos de información, denominados 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 vectores). Ejemplo:

```
// información de interés sobre un empleado de una empresa
struct empleado
{
   string nombre;
   long int salario; // salario anual en euros
   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. Ejemplo:

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

También es posible definir *arrays* de estructuras. Ejemplo:

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

PROBLEMAS

1. Escribe una función que transforme una cadena de caracteres de minúsculas a mayúsculas.

2. (*)Se dice que la palabra α es un anagrama de la palabra β si es posible obtener α cambiando el orden de las letras de β. Por ejemplo, MORA y ROMA son anagramas de RAMO. Realizar una función que compruebe si dos palabras son anagramas entre si. Las palabras pueden contener letras mayúsculas y minúsculas (Utiliza la función escrita anteriormente).

3. Escribe un programa que cuente el número de palabras que aparecen en una sola línea de entrada. Se entiende por palabra una secuencia no vacía de letras mayúsculas o minúsculas delimitada por el principio de la línea, el final de ella o cualquier carácter no alfabético.

   Por ejemplo, si al ejecutar el programa el usuario introduce la línea que reproduce las primeras frases de La Regenta,

   La heroica ciudad dormia la siesta. El viento sur, caliente y <retorno>

   el programa debe contabilizar 11 palabras.

4. (*)Escribe una función en C++ que simule la operación reemplazar de cualquier editor de texto. Esta función, dada una cadena de caractéres y una palabra, deberá sustituir cada aparición de esta por una nueva pasada también como parámetro. Un prototipo adecuado podría ser:

```
int BuscarReemplazar(string &frase, string viejacadena, string nuevacadena);
```

   La función devolverá el número de apariciones de '`viejacadena`' en `frase`.

5. (*)Escriba un programa para jugar al ahorcado. El programa debe elegir una palabra (que pueda codificarse directamente en el programa) y mostrar lo siguiente:

   Adivine la palabra: xxxxx

   Cada `X` representa a una letra. Si el usuario adivina correctamente, el programa debe mostrar:

   ¡Felicidades! Usted pudo adivinar mi palabra. Desea jugar otra vez? si/no

   Debe introducir la respuesta si o no apropiada. Si el usuario no adivina correctamente, debe mostrarse la parte del cuerpo apropiada. Después de 7 intentos fallidos, el usuario debe ser colgado. La pantalla debe verse así:

6.

   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',...)

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',....).

7. (*)Hacer un programa para almacenar la información de una tabla de clasificaciones de un equipo de fútbol de un campeonato. Se utilizará un registro para almacenar el nombre del equipo, los puntos obtenidos, los partidos ganados, empatados y perdidos, los goles a favor y los goles en contra. Se utilizará un vector para almacenar la información de los 10 equipos del campeonato. Se generarán datos aleatorios para rellenar los valores de cada equipo para hacer pruebas. El programa indicará, después de los cálculos aleaotorios, cuales son los equipos primero y último, mostrando la información completa. También podrá mostrar la información completa sobre la clasificación.

8. Crear un programa para realizar cálculos sobre triángulos en un espacio 2-D. El programa debe permitir la introducción de las coordenadas (x, y) para los tres vértices del triángulo, que deberán ser almacenados en un registro. Posteriormente, se deberá poder calcular el área del triángulo, el perímetro del triángulo y el punto medio del triángulo.

```
Area = base * altura / 2;

Area = ½ * ((x3.x – x1.x) * (x1.y- x2.y) + (x3.y - x1.y)*(x2.x - x1.x))

Perimetro = distancia(x1,x2) + distancia (x2,x3) + distancia(x3,x1)

Punto medio = ( (x1.x + x2.x + x3.x)  / 3, (x1.y + x2.y  + x3.y) / 3 )
```