# Cadenas (strings) y Estructuras

Fundamentos de Programación
Fundamentos de Programación I

## 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;
```

**Necesita utilizar**
**#include <string>**

## Métodos para manipulación de string

- **Indica si la frase está vacia**
  - `vacia =palabra.empty()`
- **Devuelve la longitud de la cadena**
  - `i = palabra.length();`
- **Inserta palabra en la posición pos de frase**
  - `frase.insert(pos, 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 num caracteres de frase desde la posición pos**
  - `frase.erase (pos,num);`

## Métodos para manipulación de string

- **Sustituye (reemplaza) num caracteres de frase, empezando en la posición pos, por la cadena palabra**
  - `frase.replace (pos, num, palabra);`
- **busca palabra como una subcadena dentro de frase desde la posición pos, devuelve la posición donde la encuentra**
  - `i = frase.find(palabra,pos);`
- **devuelve la subcadena formado por num caracteres desde la posición pos de la frase**
  - `palabra = frase.substr(pos,num);`

## Manejo de cadenas en C++

Escriba un programa que lea el nombre de una persona en formato indicado:

```
Apellido, Nombre_de_pila, inicial_intermedia
```

y lo convierta al formato siguiente:

```
Lopez, Juan A.
```

Ejemplos: entradas: `Juan Antonio Lopez` … salida: `Lopez, Juan A.`
`Juan A. Lopez` … salida: `Lopez, Juan A.`
`Juan Lopez` … salida: `Lopez, Juan`

El programa deberá funcionar colocando un punto después de la inicial intermedia, aunque la entrada no contenga dicho punto. El programa deberá contemplar usuarios que no den un nombre intermedio o inicial. En tal caso, lógicamente la salida no contendrá una inicial intermedia (ver ejemplos) Deberá producir la salida:

```
//Programa que toma el nombre de una persona y lo cambia de formato
#include <iostream.h>
#include <string>

//Prototipos de funciones

void ProcesarNombreCompleto(string todo, string &primer,string &segundo, string &apellido);


//Función principal

int main()
{
   //Declaracion de variables
   string nom_comp;
   string npri,nseg,ape;
   string formatonuevo;

   //Leo el nombre de la entrada estandar
   cout << "Introduce tu nombre completo:\n";
   getline(cin,nom_comp);

   //Analizo la cadena introducida
   //La separo en 3 partes (1 nombre, 2 nombre y apellido)
   ProcesarNombreCompleto(nom_comp, npri,nseg, ape);

   //Reconstruyo el nombre con nuevo formato a partir de
   //las partes individuales
   formato = ape + ", " + npri + " " +  nseg;

   //Escribo el resultado
   cout << formato <<  endl;

   return 0;
}
```

```
//Funcion que procesa la cadena con el nombre completo

void ProcesarNombreCompleto(string todo, string & primer, string & segundo, string & apellido)
{
   //Declaracion de variables locales
   int posini, posfinal;
   int tamanyo;

   //Extraigo el primer nombre
   posini = 0;
   posfinal = todo.find(" ");
   primer = todo.substr(posini, posfinal - posini);

   //Ahora el segundo
   posini = posfinal + 1;
   posfinal=todo.find(" ", posini);

   //Si posfinal es -1 no hay segundo nombre
   if ( posfinal > 0)
   {
      //Cuando haya segundo nombre
      segundo=todo.substr(posini,posfinal-posini);
      posini=posfinal+1;

      //Modifico el segundo nombre
      tamanyo = segundo.length();
      segundo.erase(1, tamanyo - 1 );
      segundo = segundo + ".";
   }
   else
   {
            //Cuando no hay segundo nombre
       segundo = "";
   }
    //apellido
   posfinal = todo.length();
   apellido = todo.substr(posini,posfinal-posini);

   return ;

}
```

## Manejo de Cadenas: Cifrado de datos

Una técnica de cifrado elemental consiste en lo siguiente: se dispone de una frase clave que llamamos llave. A cada carácter leído del mensaje que queremos cifrar, le aplicamos la operación XOR bit a bit con un carácter de llave y el resultado es el carácter cifrado. El carácter de llave aplicado es el siguiente en número de orden siguiendo un ciclo. Es decir si llave tiene 4 caracteres, cifraremos el primer carácter leído con llave[0], el segundo con llave[1], ... el quinto con llave[0] etc.

Este método tiene la propiedad de que cifrando el texto ya cifrado con la misma llave vuelve a aparecer el texto original.

Implementa una función que reciba como argumento una llave, una cadena y la devuelva la cadena encriptada.

Realiza una función que pida al usuario una frase, la encripte, muestre la cadena encriptada y después la desencripte con la misma función y muestre el resultado por pantalla.

NOTA:

En C++ puedes aplicar la operación lógica XOR sobre dos variables de tipo carácter de la siguiente manera:

```
char a='A';
char b='B';
char resul;
resul = a ^ b;   // el signo '^' es el XOR en C++.
```

Esto realiza la operación XOR sobre cada uno de los bits de a y b

```
// Programa que encripta una frase introducida por teclado según una clave tambien introducida por
//teclado

#include <iostream.h>
#include <string>

//Prototipos de funciones
string encriptar(string frase, string clave);


int main()
{

   string frase;
   string clave;
   string res,res2;

   cout << "Este programa encripta una frase introducida
            por teclado \n segun una clave tambien
            introducida por teclado\n";
   cout << "Introduce una frase para encriptar\n";
   //Leo la frase
   getline(cin,frase);

   //Leo la clave
   cout << "clave:";
   cin >> clave;
   res = frase;
   res2 = frase;

   res = encriptar(frase, clave);

   cout << " La frase encriptada es:\n";
   cout << res;

   cout << "Proceso inverso (desencripto)\n";
   res2 = encriptar(res,clave),
   cout << "La frase desencriptada es:\n");
   cout << res2;
   return 0;
}
```

```
//Funcion que encriptar una frase pasada como parametro
/7segun la clave tambien pasada como parametro
//devuelve la frase encriptada

string encriptar(string frase, string clave)
{
   string res;
   int a,b,c;
   int i,j;
   int longi;

   res = frase;
   //Recorro la cadena para obtener la frase encriptada
   for(i = 0; i < frase.length() ; i++)
   {
      a = int (frase[i]);

      //Calculo el indice de la clave
      j = i % clave.length();
      b = int (clave[j]);

      //Operación xor
      c = a ^ b;

      //Guardo el caracter encriptado
      res[i] = char( c );

   }


   return res;

}
```

## Calculo de calificaciones escolares

Escribe un programa para calcular la **nota final** de un grupo escolar que siga la siguiente política de calificación:

a) Hay 2 cuestionarios (calificados sobre 10 puntos) y 2 exámenes (parcial y final, sobre 100 puntos.)
c) Calificación final se obtiene de la siguiente forma:
El examen final supone el 50% de la nota, el parcial el 25% y los 2 cuestionarios completarán el 25% restante.

**Objetivos:**
**1)** Define una estructura para almacenar la información de cada estudiante.
2) El programa debe:
2.1) Pedir el nombre del alumno y sus notas parciales.
2.2) Calcular la nota final de cada alumno, de acuerdo a los datos anteriores.
2.3) Mostrar los datos introducidos y las calificaciones finales calculadas.

**Nota: Tabla de Calificaciones**
Sobresaliente = nota entre [90,100].
Notable = nota entre [70,90].
Aprobado = … [50,70].
Suspendido < 50.

```
//Programa que calcula calificaciones de alumnos
#include <iostream.h>
#include <stdlib.h>
#include <string>

const int MAXIMO = 100 ;
//Definición de estructuras
struct Alumno
{
  string n_alumno;
  int test1,test2;
  int parcial, final;
  string notaglobal;
};
//Definicion de tipos
typedef Alumno Curso[MAXIMO];

//Prototipos de funciones
void LeerRegistro(Alumno & alu);
void MostrarRegistro(Alumno alu);
void CalcularNotaRegistro(Alumno & alu);
```

```
int main()
{
   //Declaro un vector para almacenar la información de
   //los alumnos
    Curso micurso;
    int numero;
    bool seguir;
    int i, elementos; //Numero de alumnos introducidos

    seguir = true;
    i = 0;
    do{
         //Lee datos
         LeerRegistro(micurso[i]);

         //Calcula nota global
         CalcularNotaRegistro(micurso[i]);

         //Muestro el resultado
         MostrarRegistro(micurso[i]);

       //Compruebo condicion de salida
         if (i > MAXIMO)
            seguir = false;
         cout << "Para terminar pulse 0. Para continuar
                 1\n";
         cin >> numero;
         if (numero == 0)
            seguir = false;
         i++;

    }while (seguir);

    elementos = i;

    cout << "Notas almacenadas en esta sesion:" << endl;

    for (i = 0; i < elementos ; i++)
        MostrarRegistro(micurso[i]);

    return 0;
}
```

```
void LeerRegistro(Alumno &alu)
{
    //Leo cada elemento de la estructura de forma
    //independiente

    cout << "Nombre: " ;
    getline(cin,alu.n_alumno);
    cout << endl;
    cout << "Introduce las calificaciones\n";
    cout << "Test 1: ";
    cin >> alu.test1;
    cout << endl;
    cout << "Test 2: " ;
    cin >> alu.test2;
    cout << endl;
    cout << "Parcial: ";
    cin >> alu.parcial;
    cout << endl;
    cout << "Final: ";
    cin >> alu.final;

    return;

}
//Escribo los datos almacenados en el registro por pantalla

void MostrarRegistro(Alumno alu)
{
     cout << "Nombre: ";
     cout << alu.n_alumno << endl;

     cout << "\t\tCalificaciones: ";
     cout << "\t" << alu.test1 << " " << alu.test2 <<" " << alu.parcial << " " << alu.final << endl;

     cout << "\t\tCalificacion global: ";
     cout << "\t" << alu.notaglobal<< endl;

     return
}
```

```
//Calculo la nota final

void CalcularNotaRegistro(Alumno & alu)
{
     float notanumerica;

     notanumerica = (alu.test1 + alu.test2) * 25 / 20 + (alu.final / 2 )  + (alu.parcial / 4);

     if (notanumerica > 90.0)
        alu.notaglobal= "Sobresaliente";

     else if (notanumerica > 70.0)
        alu.notaglobal= "Notable";
        else if (notanumerica > 50.0 )
                alu.notaglobal="Aprobado";
                else
                        alu.notaglobal="Suspendido";

     return;

}
```