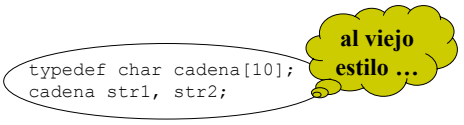
strings i estructures de *dades*

• **Els tipus de dades estructurades o tipus composts són agrupacions d'altres tipus de dades.**

Els més comunes són:

vectors i matrius (array), cadenes de caràcters (string), registres/estructures i unions.




```
typedef char cadena[10];
cadena str1, str2;
```

```
string s;
string s2 = "Hola";
```

I al contrari que amb els vectors, si que es poden realitzar assignacions sense cap problema

```
s = s2;
s = "Adios";
s2 = 'a';
```

Es pot accedir a les components del string mitjançant indexació, exactament igual que en qualsevol vector.

```
s = "Hola";
s[1] = 'a';
cout << s;
          ->     "Hala"
```

La longitud del string es consulta amb length:

```
s = "Hola";
cout << s.length();        ->     4
```

Ojo!! només llegeix paraules, quan arriba a un separador (espai, salt de línia, etc...) s'acaba el string

```
cin >> s;
```

Si per exemple, introduïm "Hola Pepe", el valor que prendrà s serà "Hola".

Per a llegir una línia sencera (fins al caràcter fi de linea) es pot usar getline:

```
getline(cin, s);
```


Ja defineix un tipus no cal utilitzar typedef.

```
struct complejo
{
    float re;
    float im;
}

complejo c;
c.re = 0;
c.im = 0
```

•**Asignació de estructures:** Les estructures sí es poden asignar, al igual que els strings o un altre tipus simple. Una asignació de estructures es equivalent a una asignació de cadascun dels components.

```
Complejo c1, c2;
c1 = c2;
```

Paràmetres … igual que la resta de tipus …

```
// Suma de complejos
Complejo SumaC(Complejo c1, Complejo c2)
{
    Complejo cres;

      cres.re = c1.re + c2.re;
      cres.im = c1.im + c2.im;

    return cres;
}
```


Fer un programa que canvie de format els noms intrduits pel teclat :

`Nom1, Nom2, Cognom` → **Representa a una *persona***
(definició de l'estructura corresponent)

nou format: `Cognom, Nom1 Inicial(Nom2).`

Exemples: entrades:

`Juan Antonio Lopez`	…	**eixida:** `Lopez, Juan A.`
`Juan A. Lopez`	…	**eixida :** `Lopez, Juan A.`
`Juan Lopez`	…	**eixida :** `Lopez, Juan`

```
#include <iostream.h>
#include <string>
using namespace std;

//Declaracion del tipo Persona
struct Persona{
   string nom1, nom2, apellido;
   string formato;
};

bool analizar(Persona &p, string todo);

int main()
{
    Persona p;
    string entrada;
    bool conseguido;

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

   //Analizo la cadena introducida
   //La separo en 3 partes (1 nombre, 2 nombre y apellido)
   conseguido = analizar(p, entrada);

   //Escribo el resultado
   if(conseguido)
     cout << p.formato <<  endl;
   else
     cout << "Error: No ha sido posible analizar: " << entrada << endl;


   system("pause");

   return 0;
}
```



```
bool analizar(Persona &p, string todo)
{
   //Declaracion de variables locales
   int posini, posfinal;
   int tamanyo;

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

   if(posfinal > 0)
     p.nom1 = todo.substr(posini, posfinal - posini);
   else
     return false;

   //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
      p.nom2=todo.substr(posini,posfinal-posini);
      posini=posfinal+1;

      //Modifico el segundo nombre
      tamanyo = p.nom2.length();
      p.nom2.erase(1, tamanyo - 1 );
      p.nom2 = p.nom2 + ".";
   }
   else //Cuando no hay segundo nombre
       p.nom2 = "";

   // Apellido
   posfinal = todo.length();
   p.apellido = todo.substr(posini,posfinal-posini);

   //Reconstruyo el nombre con nuevo formato a partir de
   //las partes individuales
   p.formato = p.apellido + ", " + p.nom1 + " " +  p.nom2;
   return true;
}
```

// Funcion que procesa la cadena
// y rellena p (persona)

Encriptació de dades

Una tècnica d'encriptació elemental consisteix en aplicar la operació XOR entre la cadena font i la paraula secreta o *clau*.

Ex: Si volem encriptar la cadena A i clau te 4 caracters, encriptarem el primer carácter de A, es a dir A[0], amb clau[0], el segond amb clau[1], ... el quint amb clau[0] etc (mitjançant el mòdul, %).

Amb aquest metod desencriptar contsisteix en encriptar per segona volta la cadena encriptada, i apareix el text original.

Fes un programa que mostre la cadena encriptada i la desencripte amb la mateixa funció.

NOTA: En C++ pots aplicar la operació XOR sobre dos caracters :

```
char a = 'A';
char b = 'B';
char resul;

   resul = a ^ b;   // el signo '^' es el XOR en C++ (sobre cadascun dels bits de a i b)
```


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

#include <iostream.h>
#include <string>
using namespace std;

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


int main()
{

   string frase;
   string clave;
   string res,res2;

   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
//segun la clave tambien pasada como parametro
//devuelve la frase encriptada

string encriptar(string frase, string clave)
{
   string res;
   char 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 =  (frase[i]);

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

      //Operación xor
      c = a ^ b;

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

   }


   return res;

}
```


Calculo de qüalificacions escolars

Escriu un programa per calcular la nota final d'un grup escolar (100 alumnes) amb la següent politica de calificació:

a) 2 qüestionaris (sobre 10 punts) i 2 examens (parcial i final, sobre 100 puntos.)
c) La qüalificació final s'obté de la següent manera:
L'examen final correspon al 50% de la nota, el parcial al 25% i els 2 qüestionaris completaran el 25% restant.

Objetius:
1) **Define una estructura per a guardar la informació de cada estudiant.**
2) El programa deu fer:
2.1) Demanar el nom de l'alumne i les seues notes parcials.
2.2) Calcular la nota final de cada alumne, d'acord a les dades anteriors.
2.3) Mostrar les dades introduides i les qüalificacions finals calculades.

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;

}
```