Sesión 4

Aprende C desde cero

- Conjuntos de datos
  - vectores y matrices
  - cadenas
- Funciones

Vectores y Matrices

Vectores

Un vector es una estructura que contiene elementos del mismo tipo

```
tipo_dato nombre[tamaño];
```

Donde:

- **`tipo_dato`**: indica el tipo que tendrán los elementos.
- **`nombre`**: indica el nombre del vector (al igual que hacemos con las variables)
- **`tamaño`**: indica el número de elementos que tendrá el vector

Ejemplos:

- `int muestras[1000];`
- `float temperaturas[24];`
- `int respuesta[5] = {2,4,5,6,3};`

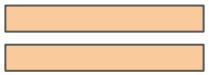
Modos de uso

Inicialización:

```
nombre[posición] = valor;
```

```c
#include <stdio.h>

int main(void)
{
    int vector[5];
    vector[0] = 0;
    vector[1] = 1;
    vector[2] = 2;
    vector[3] = 3;
    vector[4] = 4;
    return 0;
}
```



```c
#include <stdio.h>

int main(void)
{
    int vector[5] = {0,1,2,3,4};
    return 0;
}
```

¡La primera posición del vector es la 0!

Modos de uso

Inicialización:

```
nombre[posición] = valor;
```

```
#include <stdio.h>

int main(void)
{
     int i;
     int vector[5];
     for(i = 0; i < 5; i++)
     {
          vector[i] = i;
     }
     return 0;
}
```

Modos de uso

Acceso al valor:

```
nombre[posición];
```

```
#include <stdio.h>

int main(void)
{
     int i;
     int vector[5];
     int suma = 0;

     for(i = 0; i < 5; i++)
     {
          vector[i] = i;
     }

     for(i = 0; i < 5; i++)
     {
          suma = suma + vector[i];
     }
     return 0;
}
```

Modos de uso

Modificar el valor:

```
nombre[posición] = nuevo_valor;
```

```
#include <stdio.h>

int main(void)
{
     int i;
     int vector[5];
     int suma = 0;

     for(i = 0; i < 5; i++)
     {
          vector[i] = i;
     }

     for(i = 0; i < 5; i++)
     {
          vector[i] = vector[i] + i;
     }
     return 0;
}
```

Matrices

Una matriz es una estructura que contiene un vector de vectores

```
tipo_dato nombre[tamañoX][tamañoY]
```

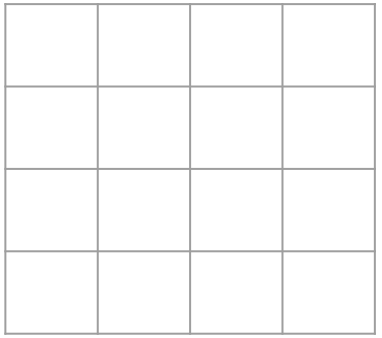
Donde:

- **`tipo_dato`**: indica el tipo que tendrán los elementos.
- **`nombre`**: indica el nombre de la matriz (al igual que hacemos con las variables).
- **`tamañoX`**: el número de elementos que tendrá la matriz en su coordenada X.
- **`tamañoY`**: el número de elementos que tendrá la matriz en su coordenada Y.

Ejemplos:

- `int muestras[1000][2];`
- `float temperaturas[7][24];`
- `int respuesta[2][5] = { {2,4,5,6,3}, {0,2,4,6,8} };`

Matriz[4][4]



Cadenas

Cadenas

Una cadena es un vector pero de caracteres

```
char nombre[tamaño];
```

Donde:

- **`nombre`**: indica el nombre del vector (al igual que hacemos con las variables)
- **`tamaño`**: indica el número de elementos que tendrá el vector

Ejemplos:

- `char curso[11] = "CdesdeCero";`
- `char sesion[5] = "";`
- `char universidad[4] = "UJI";`

¡El tamaño de la cadena debe de ser de un carácter más que la cadena para almacenar el carácter especial que indica el final de ésta: "\0"!

Cadenas

- Las cadenas tienen su propia marca de formato para la función printf que imprime por pantalla desde el inicio de la cadena hasta el primer caracter “\0” que encuentra.

%s

- Las cadenas tienen una gran cantidad de funciones en la siguiente biblioteca:

```
#include <string.h>
```

Ejemplos:

- **`strlen`**: devuelve el número de caracteres.
- **`strcat`**: concatena dos cadenas.
- **`strcpy`**: copia el contenido de una cadena en otra.

Cadenas

```
#include <stdio.h>
#include <string.h>

int main(void)
{
     char hola[5] = "hola";
     char adios[6] = "adiós";
     char saludo[10];

     printf("%s ocupa %d caracteres y %s ocupa %d",
               hola, strlen(hola), adios, strlen(adios));
     strcpy(saludo,hola);
     strcat(saludo,adios);
     printf("%s\n", saludo);

     return 0;
}
```

Funciones

Funciones

Una función es una porción de código que se encuentra fuera del programa principal y que se invoca desde éste. Además una función, debe devolver un resultado.

```
tipo_dato nombre_funcion(parámetros);
```

Donde:

- **`tipo_dato`**: indica el tipo que tendrá el elemento retornado. Será el valor de salida.
- **`nombre_funcion`**: indica el nombre de la función. Este nombre se utilizará para invocarla.
- **`parámetros`**: es una lista de variables (argumentos) separadas por comas. Serán los valores de entrada.

Ejemplo:

- Esta podría ser la definición de una función que recibe dos argumentos (num1 y num2) de tipo entero y devuelve su suma.

```
int suma(int num1, int num2);
```

Funciones

Podemos utilizar funciones que pertenezcan a una biblioteca:

- **printf** de la biblioteca **stdio.h**
- **strcpy** de la biblioteca **string.h**

O bien podemos definirlas nosotros:

```
#include <stdio.h>
```

```
int suma(int num1, int num2)
{
    return num1+num2;
}
```

```
int main(void)
{
    int a=5, b=6;
    int result;

    res = suma(a,b);
    printf("La suma de a y b es %d\n",res);

    return 0;
}
```

Funciones

Definiendo funciones:

- La función debe definirse antes de la función **main.**
- El datos que se devuelve mediante return debe de ser del mismo dato que indica la definición.
- Los tipos de datos de entrada y salida no tienen porqué coincidir.

```
#include <stdio.h>

int maximo(float num1, float num2)
{
     if(num1 >= num2)
     {
          return 0;
     }
     else
     {
          return 1;
     }
}
```

```
int main(void)
{
     float a = 3.5;
     float b = 5.3;
     int res = maximo(a,b);
     if (res == 0)
     {
          printf("El maximo es a\n");
     }
     else
     {
          printf("El maximo es b\n");
     }
     return 0;
}
```

Ejercicio

Ejercicio: Calculadora 4.0

Implementa una calculadora que además de números opere con cadenas.
Dependiendo del modo de funcionamiento (números o cadenas), la calculadora llamará a unas funciones u otras (función suma, función resta...).

Para el modo “números” la salida será parecida a la imagen de la izquierda.
Ya que estamos en el tema de las funciones, cada operación debe implementarse en una función distinta.

Para el modo “cadenas” (imagen de la derecha), implementaremos utilizando la biblioteca <string.h>, una función que nos devuelva la suma de los caracteres de las cadenas.

```
Bienvenido/a a “Calcooladora v4.0”.

Modo “números”:
    Operandos: 3.00 y 4.00
    Operación: Suma

Resultado:   7.00

¡Adiós!
```

```
Bienvenido/a a “Calcooladora v4.0”.

Modo “cadenas”:
    Operandos: “cuarta” y “sesión”
    Operación: Cuenta caracteres

Resultado:   12.00

¡Adiós!
```

Recuerda que puedes:

Interactuar con otros alumnos en el foro del curso:
http://mooc.uji.es/mod/forum/view.php?id=1654

Acceder a todos los materiales en el repositorio:
https://siserte@bitbucket.org/siserte/repositoriocursoc.git

Contactar con los profesores:

- adcastel@uji.es
- siserte@uji.es