Esquema de un programa en C

Esquema básico

Headers

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

#include "lo_que_sea.h"
```

Constants

```
#define PI 3.14

#define CONSTANT value
```

Variables globales

Las podemos usar en cualquier función del programa

```
int a, b;
char c;
int vector[CONSTANT];
char buff[CONSTANT];
```

Funcions

```
void func(void) {
    // ...
}

int main(int argc, char *argv[]) {
    // code
    return 0;
}
```

Parámetros

```
// Si ejecutamos ./program

int main(int argc, char *argv[]) {
    // argc -> 1
    // argv[0] -> "./program"
    return 0;
```

```
}
```

```
// Si ejecutamos ./program a b c

int main(int argc, char *argv[]) {
    // argc -> 4
    // argv[0] -> "./program"
    // argv[1] -> "a"
    // argv[2] -> "b"
    // argv[3] -> "c"
    return 0;
}
```

Variables

- Simples: `type variable_name;`
- Vectores: `type vector_name[vector_size];`

El tamaño de un vector puede ser un número, una constante definida anteriormente o se puede dejar en blanco así: `int vec[] = {1,2,3};` Adoptará el tamaño de aquello a lo que se asigne.

Tipos

- **int** (entero)
- **char** (carácter)
- **punteros**
  - int *
  - char *
  - void *
  - ...
- No existen **bool**, usamos ints. **1 -> true, 0 -> false**
- Los **string** son un **char*** terminado con el carácter invisible __'\0'__

Asignación

```
int x = 4;
char a = 'A';
int *p = &x;
int v[10];
v[0] = 1;
v[1] = 2;
v[9] = 19;
```

Comparaciones

- Igual (x **==** y)
- No igual (x **!=** y)
- Mayor (x **>** y)

- Mayor o igual (x >= y)
- ...

Strings

- Vectores de chars acabados en **'\0'**
- **'\n'** -> Salto de línea
- **strlen(s)** -> devuelve el tamaño de s (un string)
- **sprintf(s, "Hello World %d", 3)** -> Guarda el string "Hello world 3" en la variable s. Puede usar diversos formatos
  - **%d** -> int
  - **%c** -> char
  - **%s** -> string
  - ...
- **strcmp(s1, s2)** -> Compara las strings devuelve **0** si son iguales, **>0** si s2 es mayor, **<0** si s2 es menor.

int -> string

```
int x = 1000;
char buffer[64];
sprintf(buffer, "%d", x);
write(1, buffer, strlen(buff));

// OUT: 1000
```

string -> int

```
char *s = "314";
int x = atoi(s);

// x = 314;
```