

Sistemas Operativos

[Inicio](#) [Teoría](#) ▾ [Prácticas](#) [Notebooks](#) [Exámenes](#) ▾ [SO – Blog](#) [Logout](#) 🔍

[Inicio](#) » [PRACTICAS](#) » [C Prácticas](#) » Ordenación de un array

Ordenación de un array

[C Prácticas](#)



La función [qsort](#) de librería [stdlib](#), implementa el algoritmo de ordenación [Quick Sort](#) que podemos utilizar para ordenar nuestro array de apuestas. Os dejo un ejemplo de cómo utilizar esta función de librería.

La función `qsort` ordenará un array apoyándose en una función que determinará si un elemento es mayor, menor o igual que otro. De esta forma podemos comparar cualquier cosa, puesto que nosotros, vía esta función, determinaremos cuando algo es mayor, igual o menor que otro.

La función de librería tiene este prototipo:

```
void qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const void *)
```

donde:

base = es un puntero al primer elemento del array de elementos a ordenar

nitems = es el número de elementos del array

size = es el tamaño de cada uno de esos elementos

compar = es un puntero a la función que compara los elementos.

Recordar que un puntero a una función es simplemente el nombre de la función.

La función deberá recibir dos parámetros que son punteros a los elementos a comparar, y deberá decidir si el primero es mayor, menor o igual que el segundo. La función recibe punteros a **void**, nosotros dentro deberemos hacer un casting a puntero a **int** para poder compararlos. Recordar que cuando recibo en una función un puntero, para obtener su valor debo usar *****, así si el puntero es **a**, su valor es ***a**.

La función de comparación deberá devolver:

- un valor negativo si $a < b$
- un valor positivo si $a > b$
- cero si $a = b$

El siguiente ejemplo muestra cómo usar `qsort` para ordenar un array de enteros.

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

#define COUNT(x) ((int) (sizeof(x) / sizeof(int)))
```

```
int values[] = { 5, 15, 3, 1, 8, 12 };

int CompareFunction (const void * a, const void * b)
{
    /* integer comparison: returns negative if b > a and positive if a > b */
    return ( *(int*)a - *(int*)b );
}

int main()
{
    int n;
    int numElem = COUNT(values);
    printf("Valores del array sin ordenar: \n");
    for( n = 0 ; n < numElem; n++ ) {
        printf("%d ", values[n]);
    }

    //Llamamos a la función qsort
    qsort(values, 6, sizeof(int), CompareFunction);

    printf("\nValores del array tras la ordenación: \n");
    for( n = 0 ; n < numElem; n++ ) {
        printf("%d ", values[n]);
    }
    printf("\n");

    return(0);
}
```

[← Entrada anterior](#)

[Entrada siguiente →](#)

Deja un comentario

Conectado como alu28. [Edita tu perfil.](#) [¿Salir?](#) Los campos obligatorios están marcados con *

Escribe aquí...

Publicar comentario »

Copyright © 2025 Sistemas Operativos
Escuela Politécnica Superior de Elche
Universidad Miguel Hernández
Miguel Onofre Martínez Rach