TRABAJO PRÁCTICO N° 10

Deben entregarse por grupos en la entrega correspondiente vía Campus los archivos .c correspondientes a los ejercicios indicados.

1. La librería `<stdlib.h>` incluye una función `qsort()`[1] que permite aplicar el algoritmo Quicksort para ordenar un arreglo. El usuario puede ordenar cualquier tipo de elementos y debe proveer un puntero a una función que realiza la comparación entre dos elementos, utilizando el concepto de callback.

   Utilizar la función `qsort()` para ordenar un arreglo de números enteros.

2. Utilizar la función `qsort()` para ordenar un arreglo de strings en orden alfabético.

3. Crear un arreglo de alumnos, representados por una estructura que almacene nombre, apellido y nota promedio en tres campos distintos. Luego ordenar el arreglo usando la función `qsort()`, de manera que queden ordenados por nota en orden decreciente, si comparten la misma nota por apellido en orden alfabético, y si tienen la misma nota y el mismo apellido ordenados por nombre en orden alfabético.

4. **[ENTREGAR] Versión propia de printf()**

   La función `printf()` es muy útil para escribir texto a pantalla y mostrar el valor de distintas variables. Sin embargo y dependiendo el contexto en el cual se use, la misma puede ocupar demasiada memoria para poder ser utilizada (por ejemplo, microcontroladores con muy poca memoria de programa).

   Se pide escribir una función propia, llamada `mi_printf()`, que cumpla las siguientes condiciones:

   - El primer argumento debe ser un string que deberá mostrarse en pantalla.
   - El usuario podrá agregar argumentos adicionales con variables a imprimir en pantalla, que podrán ser enteros o caracteres.
   - El especificador "%d" en el string, indicará un entero signado a imprimir en decimal.
   - El especificador "%x" en el string, indicará un entero no signado a imprimir en hexadecimal.
   - El especificador "%c" en el string, indicará un carácter a imprimir.
   - No hace falta que los especificadores tengan sub-especificadores de ancho, precisión o modificadores, solo se pide una implementación básica.
   - Solo se podrá escribir a la pantalla usando la función `putchar()`.
   - No usar librerías para realizar la conversión de entero a string.

[1] http://www.cplusplus.com/reference/cstdlib/qsort/

5. [ENTREGAR] Emulación de un puerto

Cuando se realiza software para un microprocesador, no resulta conveniente probarlo en el mismo para cada modificación que se desea realizar. Es por esto que se suelen emular los dispositivos y puertos por software para la etapa de debuggeo.

Se pide escribir una librería que permita emular el funcionamiento de los puertos A, B y D. A y B son dos puertos de 8 bits, configurables tanto de entrada como de salida. D tiene 16 y es simplemente un alias para los puertos A y B juntos, siendo el B el menos significativo.

Crear las siguientes funciones o macros: `bitSet`, `bitClr`, `bitToggle`, `bitGet`, `maskOn`, `maskOff`, `maskToggle`, utilizables para todos los puertos definidos anteriormente.

- `bitSet`: Dado un puerto y un número de bit, debe cambiar su estado a 1.
- `bitClr`: Dado un puerto y un número de bit, debe cambiar su estado a 0.
- `bitGet`: Dado un puerto y un número de bit, debe devolver su valor.
- `bitToggle`: Dado un puerto y un número de bit, debe cambiar al estado opuesto en el que está (si está en 0 pasar a 1, y si está en 1 pasar a 0).
- `maskOn`: Dado un puerto y una máscara, debe prender todos aquellos bits que estén prendidos en la máscara, sin cambiar el estado de los restantes. Por ejemplo, dado el puerto A, que originalmente se encuentra en el estado 0x01, al aplicarle la máscara 0x0A, el resultado será 0xB.
- `maskOff`: Dado un puerto y una máscara, debe apagar todos aquellos bits que estén prendidos en la máscara, sin cambiar el estado de los restantes. Por ejemplo, dado el puerto A, que originalmente se encuentra en el estado 0x0A, al aplicarle la máscara 0x02, el resultado será 0x08.
- `maskToggle`: Dado un puerto y una máscara, debe cambiar el estado de todos aquellos bits que estén prendidos en la máscara al opuesto, sin cambiar el estado de los restantes. Por ejemplo, dado el puerto A, que originalmente se encuentra en el estado 0x02, al aplicarle la máscara 0x03, el resultado será 0x01.

El contenido de los puertos debe almacenarse en una variable estática dentro del archivo fuente de la librería. El usuario debe poder leer y escribir en los puertos solamente utilizando las funciones definidas.

6. [ENTREGAR] Simulación

Escribir un programa que utilice la librería escrita en el ejercicio 5 para simular que se tienen 8 *LEDs* conectados al puerto A. Se debe usar la librería para almacenar y acceder al estado del puerto, y mostrar el estado de los *LEDs* en pantalla.

- Por teclado, el usuario ingresará el número (del 0 al 7) del LED que se desea prender, en tiempo real.
- Con la letra '*t*', todos los *LEDs* deben cambiar al estado opuesto (si están encendidos apagarse y si están apagados encenderse).
- Con la letra *'c'*, se deberán apagar todos, y con *'s'*, prender.
- Con la letra *'q'*, el programa finalizará.