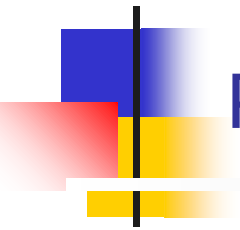
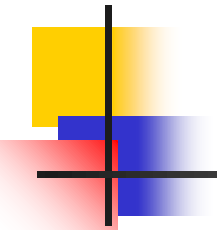


Práctica 1. Acceso a la BIOS

Consejos para realizar la práctica



Objetivos

- Implementar funciones de entrada-salida en C similares a las ofrecidas por `conio.h` usando las subrutinas del BIOS para el manejo del teclado y del video.

- Aprender a utilizar funciones y variables C de acceso a la BIOS.

Introducción

- La ROM-BIOS cuenta con diversas rutinas que pueden ser utilizadas por las aplicaciones mediante interrupciones software.

- Estas rutinas permiten la comunicación con diversos periféricos básicos, sin depender del SO:
  - El teclado usa el número de interrupción 16h
  - El video usa el número de interrupción 10h
  - Las unidades de disco usan el número de int. 13h

- Cada rutina aglutina a diversas subrutinas (subfunciones).

Servicios de interrupción del teclado

- El acceso al teclado se realiza mediante la interrupción 16h

- Esta interrupción permite las siguientes tareas:

  - Detectar la pulsación de una tecla
  - Captar una tecla desde el búfer
  - Averiguar el estado actual del teclado

Servicios de interrupción del video

- El acceso al video se realiza mediante la interrupción 10h

- Esta interrupción permite varias tareas:
  - Escribir un carácter en pantalla
  - Colocar el cursor en un punto determinado de pantalla
  - Averiguar la posición del cursor
  - Cambiar el aspecto del cursor
  - Realizar scroll vertical de pantalla
  - Escoger el modo de video
  - Averiguar el modo de video actual
  - Etc

Interrupciones software en C

#include <dos.h>

```
int86 (int intno, union REGS *inregs,
       union REGS *outregs)
```

1. Copia inregs en los registros internos (valor de los parámetros necesarios para la rutina).

2. Ejecuta la rutina de interrupción intno

3. Copia los registros internos en outregs (valores devueltos por la rutina).

Interrupciones software en C

La unión `REGS` está definida en `dos.h`. Permite acceder tanto a los registros de 8 bits como a los de 16.

```
union REGS {
  struct WORDREGS x;
  struct BYTEREGS h;
};
struct BYTEREGS {
  unsigned char al, ah, bl, bh;
  unsigned char cl, ch, dl, dh;
};
struct WORDREGS {
  unsigned int ax, bx, cx, dx;
  unsigned int si, di, cflag, flags;
};
```

Se puede hacer: `inregs.x.ax=0x0305;`

o bien:
```
inregs.h.ah=0x03;
inregs.h.al=0x05;
```

Interrupciones software en C

Ejemplo: función en C que permite modificar el modo de video

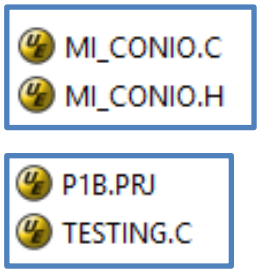
```
#define BYTE unsigned char

/* Selecciona el modo de video deseado */
void selecciona_modo_video(BYTE modo)
{
 union REGS inregs, outregs;

 inregs.h.ah = 0x00;
 inregs.h.al = modo;
 int86(0x10,&inregs,&outregs);
 return;
}
```

Objetivo de la práctica

A partir de las subrutinas que se indican en el guión se deben realizar una serie de funciones similares a las que implementa la biblioteca `conio.h` de Borland C:

- kbhit()
- gotoxy()
- wherex()
- wherey()
- clrscr()
- clreol()
- textcolor()
- textbackground()
- setcursortype()

...



Objetivo de la práctica

Requisitos básicos (hasta 7 puntos):

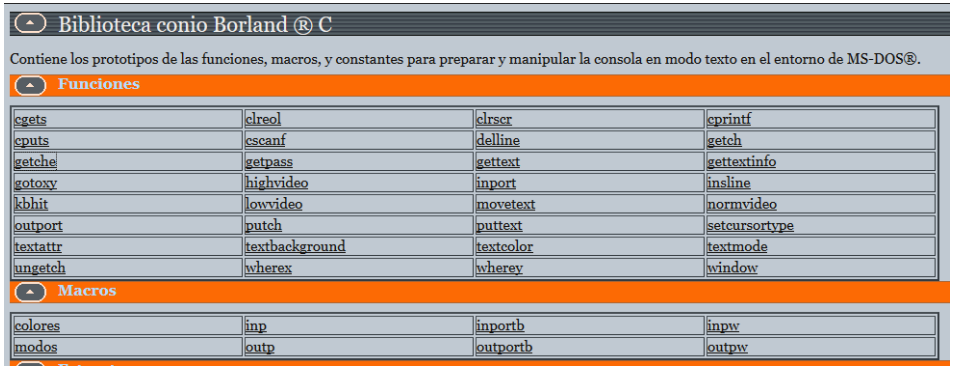
- implementar estas funciones como una biblioteca propia.
- realizar una programa principal (fichero aparte) que las use todas en modo secuencial

Requisitos ampliados (hasta 3 puntos):

- implementar una función que permita dibujar un recuadro en la pantalla.
- realizar un programa principal que demuestre el uso de las funciones de la biblioteca en modo de menú navegable

Consejo 1

- Implementar funciones con la misma funcionalidad e interfaz (iguales parámetros, iguales resultados) que las correspondientes a las bibliotecas del Borland C que se tratan de imitar.



http://c.conclase.net/borland/?borlandlib=conio#inicio

- Usar la ayuda de Borland C sobre las funciones a emular.
- Usar las funciones originales y sustituirlas progresivamente por las nuevas para comprobar que su funcionalidad es idéntica.

Consejo 2

- Para el manejo del color, se recomienda usar una/dos variables globales que almacenen el valor del color fijado en cada momento en vez de pasarlo continuamente como parámetro en las funciones de salida.

```
BYTE FG_COLOR=7;
BYTE BG_COLOR=0;
```

- Esta variable modificará su valor mediante las funciones `textcolor()` y `textbackground()`.

- Cuando use cputchar(), cputs() o clrscr(), la función tomará como valor del color el que indique esa variable en ese momento.

Consejo 3

- Ser estructurado, no mezclar funcionalidades. Es preferible tener varias funciones simples con pocos parámetros, que pocas funciones complejas con muchos parámetros. Por ejemplo, es mejor tener las funciones `wherex()` y `wherey()` para conocer la posición `x` e `y` del cursor, respectivamente, que tener una única `wherexy()` que devuelva `x` e `y` simultáneamente.

- Buscar la máxima independencia del contexto posible. La librería debería poder ser utilizada por futuros programas.

Consejo 4

- Por último, añadir comentarios claros al código del programa que expliquen claramente:
  - Los parámetros de entrada de cada función: su nombre y su utilidad.
  - Los valores devueltos por la función: tipo de dato, si es por valor o referencia y utilidad.
  - Utilidad de la función que ha implementado.
  - Comentar los detalles concretos que sean necesarios para entender el algoritmo que se ha utilizado y los detalles que se han tenido en cuenta.