Periféricos y dispositivos de interfaz humana

Práctica 1

UNIVERSIDAD DE GRANADA

Jordi Pereira Gil
DNI: 35674006V

Programa 1 - p1.c

Para este programa hemos creado un archivo llamado *p1.c* en el que hemos creado un método main con el flujo explicado a continuación.

Primero lo que hacemos es esperar a que el usuario pulse cualquier botón para poner el cursor en una posición determinada y escribir un caracter para hacerlo más visual, en este proceso hemos usado 3 métodos:

- **mi_pausa():** este método detiene la ejecución del programa hasta que se detecte una entrada por teclado, sea cual sea la tecla.

```
void mi_pausa(){
    union REGS inregs, outregs;
     inregs.h.ah = 8;
     int86(0x21, &inregs, &outregs);
}
```

- **gotoxy(int x, int y):** esta función nos permite poner el cursor en la posición deseada introduciendo dos 2 enteros en el registro *dx*, *y* en los bits más significativos(*dh*) y *x* en los bits menos significativos(*dl*). Además a diferencia de la función anterior, que llamaba a las interrupciones del sistema operativo (21h), este provoca una interrupción de la BIOS (10h) las cuales son más rápidas, pero menos versátiles.

```
void gotoxy(int x, int y){
    union REGS inregs, outregs;
    inregs.h.ah = 0x02;
    inregs.h.bh = 0x00;
    inregs.h.dh = y;
    inregs.h.dl = x;
    int86(0x10,&inregs,&outregs);
    return;
}
```

- **clrscr():** este método nos permite limpiar la pantalla, para ello lo que hacemos es una impresión de 25 saltos de línea, y después usamos el método *gotoxy()* para llevar el cursor a la posición de arriba a la izquierda.

```
void clrscr(){
   printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");

    gotoxy(0,0);
}
```

- **mi_putchar(char c):** la funcionalidad de este método es bastante clara, imprimir por pantalla un caracter *c* mediante una interrupción del sistema operativo.

```
void mi_putchar(char c){
    union REGS inregs, outregs;

    inregs.h.ah = 2;
    inregs.h.dl = c;
    int86(0x21, &inregs, &outregs);
}
```

Después de esto lo que haremos será cambiar el tipo del cursor observado en la pantalla, para ello haremos uso de **setcursortype(int x)** en el que tenemos un switch en el que en función del entero enviado pues se podrá poner en invisible, normal o grueso.

```
void setcursortype(int tipo_cursor){
    union REGS inregs, outregs;
    inregs.h.ah = 0x01;
    switch(tipo_cursor){
        case 0: //invisible
            inregs.h.ch = 010;
            inregs.h.cl = 000;
            break;
        case 1: //normal
            inregs.h.ch = 010;
            inregs.h.cl = 011;
            break;
        case 2: //grueso
            inregs.h.ch = 000;
            inregs.h.cl = 010;
            break;
    }
    int86(0x10, &inregs, &outregs);
}
```

Primero lo ponemos invisible, después grueso y por último normal, para que el usuario pueda verlo con claridad hemos utilizado *mi_pausa()* antes de pasar al siguiente tipo para esperar a que el usuario pueda detenerse a ver las diferencias del cursor.

El siguiente paso en el flujo del programa es cambiar el modo de vídeo, es decir la resolución de la pantalla del *dosbox* y sus propiedades. Para ello hemos creado el método **setvideomode(int x)** el cual mediante una interrupción de la BIOS cambia el modo de vídeo a uno de los posibles, esto nos permite por ejemplo pasar de modo texto a modo gráfico.

```
void setvideomode(BYTE modo){
   union REGS inregs, outregs;
   inregs.h.al = modo;
   inregs.h.ah = 0x00;
   int86(0x10, &inregs, &outregs);
}
```

Una vez visto como se cambia el modo de vídeo vamos a obtener el modo de video en el que estamos actualmente, para ello utilizamos **getvideomode().**

```
BYTE getvideomode(){
   union REGS inregs, outregs;
   inregs.h.al = 0x00;
   inregs.h.ah = 0x0F;
   int86(0x10, &inregs, &outregs);
   return outregs.h.al;
}
```

A continuación vamos a imprimir un símbolo con un color propio y de fondo personalizado, para ello haremos uso de 3 métodos diferentes:

- **textcolor(BYTE x):** aquí únicamente modificamos la variable global *ctexto* que usaremos más adelante.

```
void textcolor(BYTE c) {
    ctexto = c;
}
```

- **textbackground(BYTE x):** al igual que el anterior este modifica la variable global *cfondo*.

```
void textbackground(BYTE c) {
    cfondo = c;
}
```

- **cputchar(char c):** este método es el encargado de imprimir un caracter *c* con el color de texto y de fondo que hayamos seleccionado con anterioridad, a diferencia de *mi_putchar()* éste utiliza una interrupción de la BIOS. Para la configuración de los colores se usa el registro *bl* en el que los 4 bits más significativos son para el color de fondo y los otros 4 para el color del texto.

```
void cputchar(char c){
    union REGS inregs, outregs;
    inregs.h.ah = 0x09;
    inregs.h.al = c;
    inregs.h.bl = cfondo << 4 | ctexto;
    inregs.h.bh = 0x00;
    inregs.x.cx = 1;
    int86(0x10,&inregs,&outregs);
    return;
}
```

Siguiendo con la ejecución del programa ahora limpiaremos la pantalla con el método *clrscr()* y pediremos al usuario que escriba un número o letra, para esto utilizamos la función **getche()**.

```
char getche(){
    union REGS inregs, outregs;
    int caracter;

    inregs.h.ah = 1;
    int86(0x21, &inregs, &outregs);

    caracter = outregs.h.al;
    return caracter;
}
```

Para ver la ejecución del programa accede al siguiente enlace:
https://drive.google.com/file/d/1DWE8yncqKqCXCSzAUKgEXXu-QrfBnRAh/view?usp=sharing

Programa 2 - grafico.c

En este programa lo que se hace es dibujar una serie de rectángulos en la pantalla. Para esto primero que hemos hecho ha sido cambiar el modo de video con *setvideomode()* y después hemos creado un método **pixel(int x, int y, BYTE c)** que pinta un píxel específico de un color determinado.

```
void pixel(int x, int y, BYTE C){
   union REGS inregs, outregs;
   inregs.x.cx = x;
   inregs.x.dx = y;
   inregs.h.al = C;
   inregs.h.ah = 0x0C;
   int86(0x10, &inregs, &outregs);
}
```

En el main encontramos varias secciones con dos bucles anidados cada uno en la que pintamos los rectángulos:

```
for (i = 4; i < 40; i++)
{
    for ( j = 20; j < 50; j++)
    {
        pixel(i, j, c);
    }

}
```

Una vez pintados los rectángulos la ejecución se pausa con *mi_pausa()* hasta que el usuario le da a algún botón finalizando así el programa dejando el modo de video predeterminado.

La ejecución del programa se puede ver en el siguiente enlace:
https://drive.google.com/file/d/1z7xLmqRXhcGlTudARxyWM0SH4BjzJQLf/view?usp=sharing

Programa 3 - ascii.c

En este programa hemos dibujado en ascii art un conejo, para ello lo que hacemos es ir desplazando el cursor por distintas posiciones para obtener el dibujo deseado, para dicho desplazamiento usamos el método creado en el ejercicio 1 *gotoxy()*, poniendo en cada línea los caracteres correspondientes.

```c
int main(){
    clrscr();
    gotoxy(20,10);
    printf("(\\(\\");
    gotoxy(20,11);
    printf("(-.-)");
    gotoxy(20,12);
    printf("o_(')(')");
    gotoxy(0,25);
    printf(" pulsa una tecla para salir");
    mi_pausa();
    clrscr();
    return 0;
}
```

La ejecución del programa se puede ver en
https://drive.google.com/file/d/11GoEVXcPPmp_OkuTXD-iRKGOrtcAENxO/view?usp=sharing