Práctica 1. Entrada/Salida utilizando interrupciones con lenguaje C

Realizado por Rubén Martín Jáimez

Comprobación y explicación funciones requeridas:

- gotoxy(): Para esta función, he convertido del código ensamblador ,proporcionado en el propio guión, al lenguaje c.

  Información proporcionada por el guión:

  Colocar el cursor en una posición determinada
  Número de interrupción: 10h
  Número de función: 2

  Entrada: AH = 2
  DH = número de fila (00h indica arriba del todo)
  DL = número de columna (00h indica
  izquierda del todo)
  BH = 0
  Salida: No tiene

  Conversión a código c:

```c
void gotoxy(int x, int y){
    union REGS inregs, outregs;

    inregs.h.ah = 2;
    inregs.h.dh = x;
    inregs.h.dl = y;
    inregs.h.bh = 0;

    int86(0x10, &inregs, &outregs);
}
```

Resultado:

- setcursortype(): Para este caso, he utilizado la propia función proporcionada por el profesor.

Código:

```
void setcursortype(int tipo_cursor){
    union REGS inregs, outregs;

    inregs.h.ah = 0x01;
    switch(tipo_cursor){
        case 0:
            inregs.h.ch = 010;
            inregs.h.cl = 000;
            break;

        case 1:
            inregs.h.ch = 010;
            inregs.h.cl = 010;
            break;

        case 2:
            inregs.h.ch = 000;
            inregs.h.cl = 010;
            break;
    }

    int86(0x10, &inregs, &outregs);

}
```

Resultado:

- setvideomode(): Para este caso, he realizado los mismos pasos que con la primera función.

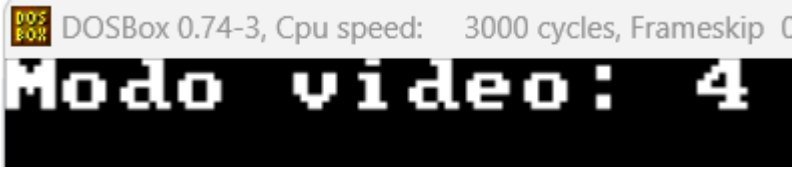
  Información proporcionada por el guión:

  Seleccionar el modo de vídeo
  Número de interrupción: 10h
  Número de función: 0
  Entrada: AH = 0
  AL = modo
  Salida: No tiene

  Código:

```
void setvideomode(BYTE modo){
    union REGS inregs, outregs;

    inregs.h.al = modo;
    inregs.h.ah = 0x00;

    int86(0x10, &inregs, &outregs);
}
```

  Resultado:



- getvideomode(): Mismos pasos que la función anterior.

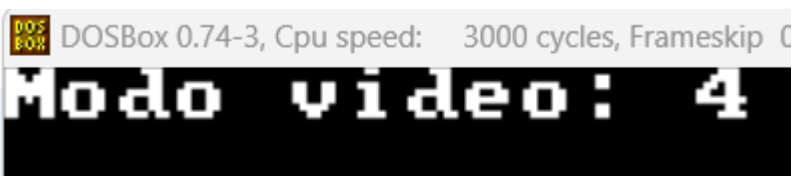
  Información proporcionada por el guión:

  Averiguar el modo de vídeo actual
  Número de interrupción: 10h
  Número de función: Fh
  Entrada: AH = Fh
  Salida: AL = modo actual
  AH = número de columnas (sólo en los modos de texto)

  Código:

```
int getvideomode(){
    union REGS inregs, outregs;
    int modo;

    inregs.h.ah = 0x0F;

    int86(0x10, &inregs, &outregs);

    modo = outregs.h.al;
  return modo;

}
```

  Resultado:



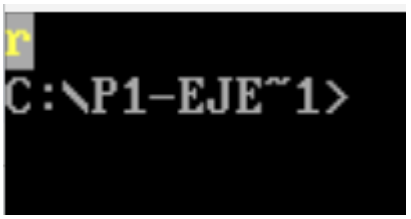
* He utilizado la misma captura que en la función anterior, ya que la forma de comprobar que obtiene de forma correcta el tipo de modo de vídeo es obteniendo el byte que establece el tipo de modo de vídeo deseado, en mi caso es 4 para modo gráfico y 3 para modo de texto.

- textcolor(), cputchar() y textbackground(): Aquí he juntado tres funciones, ya que he utilizado la función que proporcionó el profesor, la cuál dibuja un caracter con un color de letra y fondo con colores distintos.

  Código:

```
void escribir_char_con_color(char caracter){
    union REGS inregs, outregs;

    inregs.h.ah = 0x09;
    inregs.h.al = caracter;
    inregs.h.bl = cfondo << 4 | ctexto;
    inregs.h.bh = 0x00;
    inregs.x.cx = 1;

    int86(0x10, &inregs, &outregs);
}
```

  Comprobación:

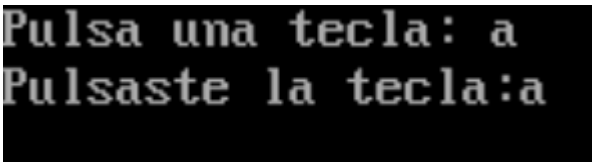


- clrscr(): En esta función, como el fin es limpiar toda la información que tenemos en pantalla, básicamente lo que hace es cambiar a modo gráfico y luego volver al modo texto. No pongo captura de comprobación ya que no sé como comprobarlo, pero en la parte final de mi código se puede ver que está implementada y hace su debida función.
- getche: Para este caso, he utilizado las funciones auxiliares *mi_getchar y mi_putchar* proporcionadas en el guión y en el seminario 2. Por lo que obtengo el carácter con la primera función y lo muestro con el segundo.

Código:

```
void getche(){
    int caracter;
    printf("Pulsa una tecla: ");
    caracter=mi_getchar();
    printf("\n");
    printf("Pulsaste la tecla:");
    mi_putchar(caracter);
}
```

Comprobación:



- Dibujar recuadro: Para realizar esta función, ha sido necesario implementar la función pixel proporcionada por el mismo guión. Básicamente, realizo un bucle anidado básico de recorrido de matriz pero usando las coordenadas dadas por el usuario, y dibujo un píxel en cada valor de la matriz.

  Información proporcionada por el guión:

```
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);
}
```

Código:

```
void dibujar_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);
}
```

Resultado:

- Dibujar dibujos sencillos: En mi caso, he dibujado una R, M y J, ya que son las iniciales de mi nombre y apellidos. Mi código se resume en ir recorriendo la pantalla con bucles e ir dibujando y dando forma.

Código:

```
// Esta función dibuja las iniciales de mi nombre y apellido
void dibujarfiguras(){
    int i;
    int j=0;

    setvideomode(MODOGRAFICO);

    //Dibujo de la letra R

    j=5;
    for(i=5; i<80; i++){
      dibujar_pixel(j,i, 1 );
      dibujar_pixel(i,j, 1 );
    }

    j=i;
    for(i=5; i<40; i++){
      dibujar_pixel(j,i, 1 );
    }

    j=i;
    for(i=40; i<80; i++){
      dibujar_pixel(i,j, 1 );
    }

    for(i=40; i<80; i++){
      dibujar_pixel(i,i, 1 );
    }

    // Dibujo de la letra M

    j=100;
    for(i=5; i<80; i++){
      dibujar_pixel(j,i, 2 );
    }

    j=5;
    for(i=100; i<125; i++,j++){
      dibujar_pixel(i,j, 2 );
    }

    j=30;
    for(i=125; j>=5; i++,j--){
      dibujar_pixel(i,j, 2 );
    }

    j=150;
    for(i=5; i<80; i++){
      dibujar_pixel(j,i, 2 );
    }
```

```
    //Dibujo de la letra J

    j=5;
    for(i=175; i<225; i++){
      dibujar_pixel(i,j, 3 );
    }

    j=200;
    for(i=5; i<80; i++){
      dibujar_pixel(j,i, 3 );
    }

    j=80;
    for(i=175; i<200; i++){
      dibujar_pixel(i,j, 3 );
    }

    pausa();
    setvideomode(MODOTEXTO);

}
```

Aclaraciones: Aunque en las capturas aparezcan errores en las variables de los registros, el programa compila y se ejecuta perfectamente. Debe ser que el IDE (en mi caso, VSCode) no reconoce las librerías. También aparecen algunos *warnings* durante la compilación, pero, de nuevo, todo funciona correctamente.