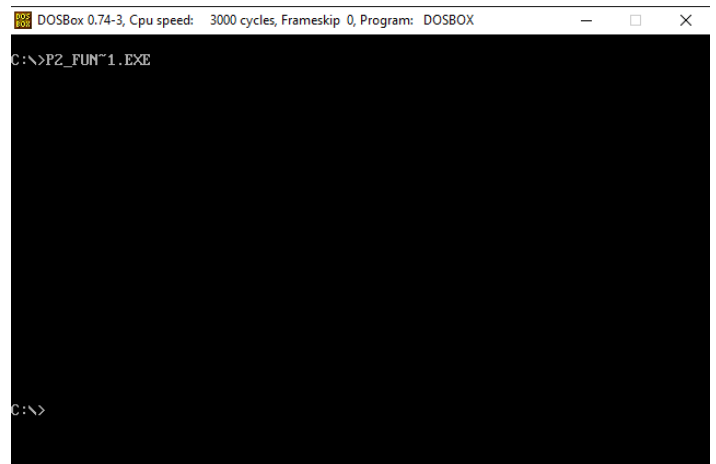


- **Ejercicio:** colocar el cursor en una posición determinada

Tras lanzar la función para la posición 20 la terminal nos quedaría como se muestra a continuación.

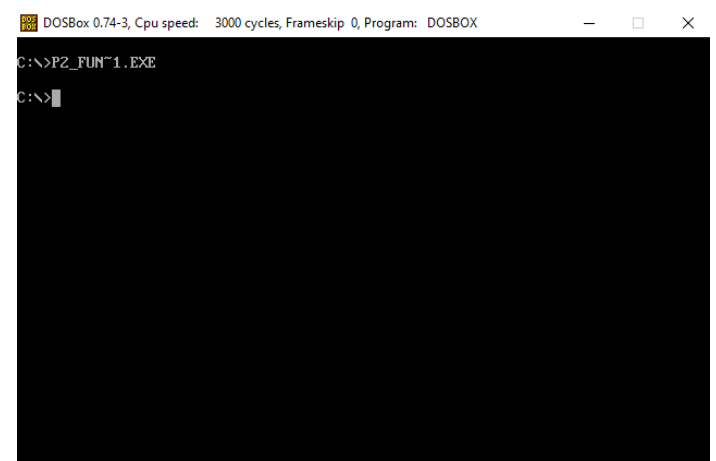
```
////////////////////////////////////  
void gotoXY(int f, int c){  
    union REGS inregs, outregs;  
    inregs.h.ah = 2;  
    inregs.h.dh = f; // Numero de fila  
    inregs.h.dl = c; // Numero de columna  
    inregs.h.bh = 0; // Numero de pagina  
  
    int86(0x10, &inregs, &outregs);  
}
```



- **Ejercicio:** fijar aspecto del cursor para 3 modos diferentes

Tras lanzar la función para el modo 'Grueso' el cursor quedaría como se muestra a continuación.

```
////////////////////////////////////  
void setTipoCursor(int tipo){  
    union REGS inregs, outregs;  
    inregs.h.ah = 1;  
  
    switch(tipo){  
        case 0: // Invisible  
            inregs.h.ch = 010;  
            inregs.h.cl = 000;  
            break;  
        case 1: // Normal  
            inregs.h.ch = 010;  
            inregs.h.cl = 010;  
            break;  
        case 2: // Grueso  
            inregs.h.ch = 000;  
            inregs.h.cl = 010;  
            break;  
    }  
  
    int86(0x10, &inregs, &outregs);  
}
```



• Ejercicio: fijar el modo de video deseado

Tras lanzar la función para el case '1' (320x200 - 4 colores) la terminal queda como se muestra a continuación.

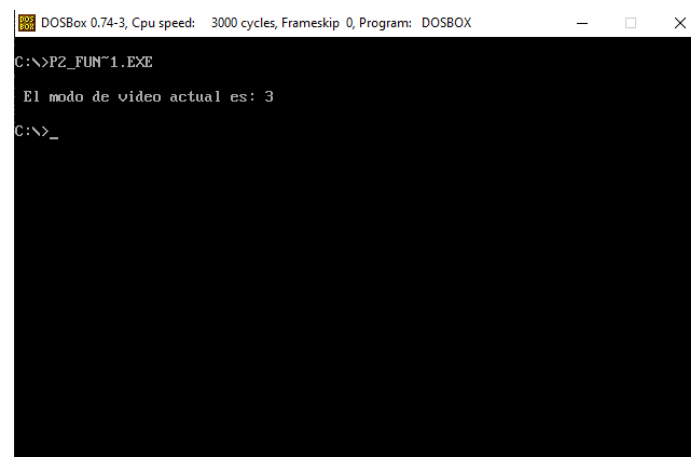
```
////////////////////////////////////  
void setModoGrafico(int modo){  
    union REGS inregs, outregs;  
    inregs.h.ah = 0;  
  
    switch(modo){  
        case 1: // 320x200 - 4 colores  
            inregs.h.al = 0x04; break;  
        case 2: // 320x200 - 4 colores  
            inregs.h.al = 0x05; break;  
        case 3: // 640x200 - 2 colores  
            inregs.h.al = 0x06; break;  
        case 4: // 320x200 - 16 colores  
            inregs.h.al = 0x0D; break;  
        case 5: // 640x200 - 16 colores  
            inregs.h.al = 0x0E; break;  
        case 6: // 640x350 - 2 colores  
            inregs.h.al = 0x0F; break;  
        case 7: // 640x480 - 16 colores  
            inregs.h.al = 0x12; break;  
        case 8: // 320x200 - 256 colores  
            inregs.h.al = 0x13; break;  
    }  
  
    int86(0x10, &inregs, &outregs);  
}
```



• Ejercicio: obtener el modo de video actual

Tras lanzar la función se indica por terminal el modo de video que hay corriendo actualmente.

```
////////////////////////////////////  
void getModoVideoActual(){  
    union REGS inregs, outregs;  
  
    inregs.h.ah = 0xF;  
    int86(0x10, &inregs, &outregs);  
  
    printf("\n El modo de video actual es: %x\n", outregs.h.al);  
}
```



- **Ejercicio:** incluye los modificadores de color de 1º y 2º plano y la escritura en pantalla de un carácter en el color indicado

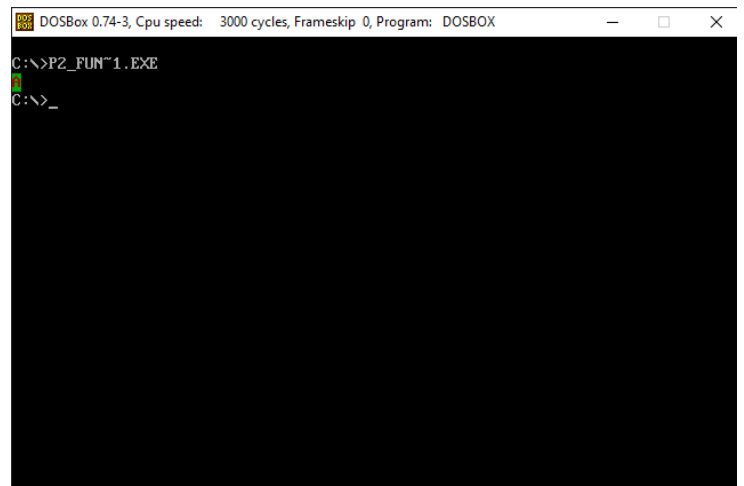
Tras lanzar la función para el carácter 'A' y los colores rojo y verde el resultado es el siguiente.

```
////////////////////////////////////
//  VAR. GLOBALES
int front_color, back_color;
////////////////////////////////////
void setColor(int fc, int bc){
    front_color = fc;
    back_color = bc;
}
////////////////////////////////////
void caracterEnColor(const char c){
    union REGS inregs, outregs;

    int color = back_color << 4 | front_color;

    inregs.h.ah = 0x09;
    inregs.h.al = c;
    inregs.h.bl = color;
    inregs.h.bh = 0x00;
    inregs.x.cx = 1;

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



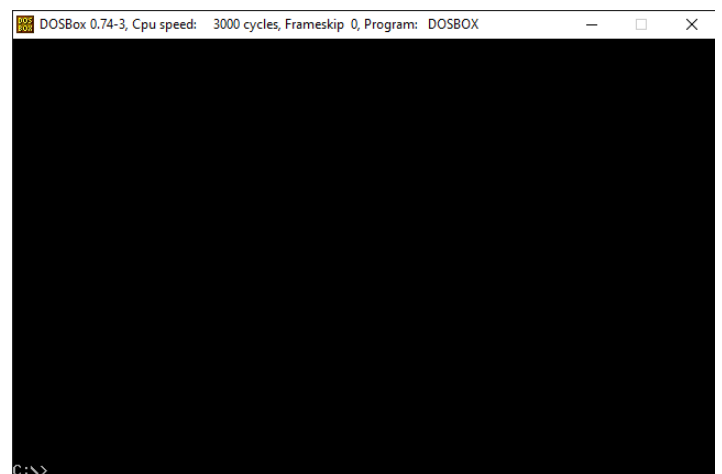
- **Ejercicio:** borrar toda la pantalla

Tras lanzar la función la terminal queda como se muestra a continuación. Básicamente lo que hace es lanzar 30 retornos de carro.

```
////////////////////////////////////
void limpiarPantalla(){
    union REGS inregs, outregs;
    char rc = '\n';
    int i = 0;

    inregs.h.ah = 2;
    inregs.h.dl = rc;

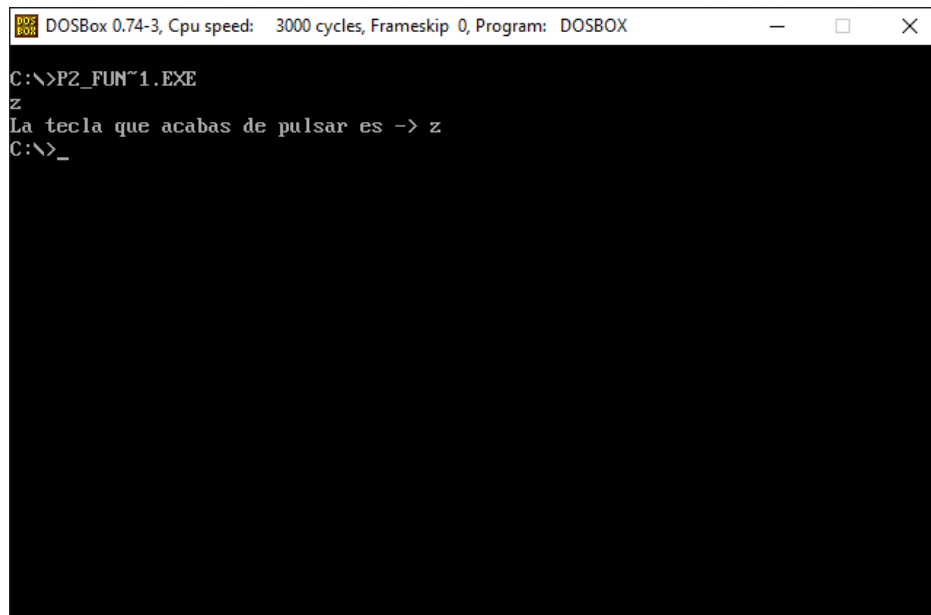
    while(i<=30){ // 30 retornos de carro
        int86(0x21, &inregs, &outregs);
        i++;
    }
}
////////////////////////////////////
```



- **Ejercicio:** obtener carácter de teclado y mostrarlo por pantalla

Tras lanzar la función y pulsar el carácter 'z' el resultado es el siguiente.

```
////////////////////////////////////  
void leerMostrarCaracter(){  
    union REGS inregs1, outregs1;  
    union REGS inregs2, outregs2;  
  
    // Leer caracter pulsado  
    inregs1.h.ah = 1;  
    int86(0x21, &inregs1, &outregs1);  
  
    // Mostrar caracter por pantalla  
    inregs2.h.ah = 2;  
    inregs2.h.dl = outregs1.h.al; // Le pasamos de 'outregs1' el cod. ASCII del caracter pulsado  
    printf("\nLa tecla que acabas de pulsar es -> ");  
    int86(0x21, &inregs2, &outregs2);  
}
```



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX  
C:\>P2_FUN~1.EXE  
z  
La tecla que acabas de pulsar es -> z  
C:\>_
```