

Periféricos y Dispositivos de Interfaz Humana



**UNIVERSIDAD
DE GRANADA**

CURSO 2024 - 2025

Práctica 1. Entrada/Salida utilizando interrupciones con lenguaje C

**GABRIEL VICO ARBOLEDAS
RAÚL RODRÍGUEZ RODRÍGUEZ**

Índice

1. Ejercicios obligatorios.....	3
1.1 gotoxy(int x, int y).....	3
1.2 setcursortype(int tipo).....	4
1.3 setvideomode(BYTE modo).....	5
1.4 getvideomode().....	6
1.5 textcolor(unsigned char color).....	7
1.6 textbackgorund(unsigned char color).....	7
1.7 clrscr(int lineas, int x, int y).....	7
1.8 cputchar(char c).....	9
1.9 getche().....	9
1.10 pixel(int x, int y, unsigned char color).....	10
Ejercicio 1 extra: Implementar una función que permita dibujar un recuadro en la pantalla en modo texto.....	12
Ejercicio 2 extra: Implementar en lenguaje C un programa que establezca modo gráfico CGA para crear dibujos sencillos en pantalla.....	13
Ejercicio 3 extra: Implementar un programa sencillo que realice un dibujo sencillo de tipo “ascii art”.....	15

1.Ejercicios obligatorios

1.1 gotoxy(int x, int y)

La función debe colocar el cursor en una posición indicada por las coordenadas x e y desde el main.

Para ello, se prepara una función donde:

- **ah** tiene el nº de la función (en este caso la 2).
- **bh** vale 0 para indicar la página en la que moverá el cursor.
- **dh** tiene el valor del parámetro y.
- **dl** tiene el valor del parámetro x.
- Y finalmente la interrupción 10h.

```
void xy(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;
}
```

Esta es la llamada a la función en el main, donde primero se inicializan una variables para el funcionamiento de las diferentes funciones. Después se indica por pantalla que se va a probar la función, un limpiado de pantalla para desplazar todo el contenido que había en pantalla, la función en las coordenadas 15, 7, la impresión del carácter '*' y una pausa para continuar con el main con otra limpieza de pantalla.

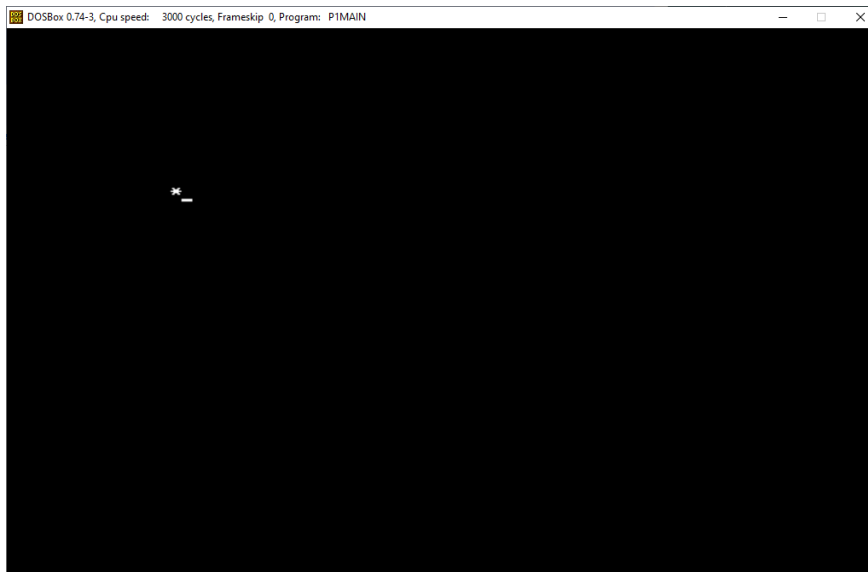
```
int tmp, x, y, color;

printf("\nDemostracion de la funcion gotoxy(x,y)");
mi_pausa();

clrscr(0, 24, 79);
xy(15,7);
printf("*");
mi_pausa();
clrscr(0, 24, 79);

xy(0,24);
```

Por último, la ejecución del mismo:



1.2 setcursortype(int tipo)

La función debe cambiar el tipo de cursor con el parámetro 'tipo' indicado desde el main.

Para ello, se prepara una función donde:

- **ah** tiene el nº de la función (en este caso la 1).
- **ch** nº de línea inicial.
- **cl** nº de línea final.
- Y finalmente la interrupción 10h.

```
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;
            printf("\nSe ha activado el tipo invisible");
            break;
        case 1: //normal
            inregs.h.ch = 010;
            inregs.h.cl = 010;
            printf("\nSe ha activado el tipo normal");
            break;
        case 2: //grueso
            inregs.h.ch = 000;
            inregs.h.cl = 010;
            printf("\nSe ha activado el tipo grueso");
            break;
    }
    int86(0x10, &inregs, &outregs);
}
```

La llamada a la función desde el main, donde se comprueba que sea correcto el valor:

```
printf("Demostracion de la funcion setcursortype(tipo)");
printf("\nSelecione una de las opciones: Invisible (0), Normal (1) o Grueso (2)\n");
do{
    tmp = mi_getchar() - '0'; //se le resta 0 para que reste el valor ASCII al valor que queremos realmente
}while (tmp != 0 && tmp != 1 && tmp != 2);

setcursortype(tmp);
mi_pausa();
```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: P1MAIN

```
Demostracion de la funcion setcursortype(tipo)
Selecione una de las opciones: Invisible (0), Normal (1) o Grueso (2)
2
Se ha activado el tipo grueso

Demostracion de la funcion setvideomode(modos)
Selecione una de las opciones: Texto (3) o Video (4)
```

1.3 setvideomode(BYTE modo)

La función debe cambiar el tipo de modo a texto o vídeo, con el parámetro modo indicado de main.

Para ello, se prepara una función donde:

- **ah** tiene el nº de la función (en este caso la 0).
- **al** tiene el valor del parámetro 'modo'.
- Y finalmente la interrupción 10h.

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

```
printf("\n\nDemostracion de la funcion setvideomode(modos)");
printf("\nSeleccione una de las opciones: Texto (3) o Video (4)\n");
do{
    tmp = mi_getchar() - '0';
}while (tmp != 3 && tmp != 4);
setvideomode(tmp);
mi_pausa();
```

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: P1MAIN
Demostracion de la funcion getvideomode(
)
Modo de video actual: 0x4
Numero de columnas (solo texto): 4
```

1.4 getvideomode()

La función debe obtener y decir el tipo de modo de video actual del sistema.

Para ello, se prepara una función donde:

- **ah** tiene el nº de la función (en este caso la Fh).
- La interrupción 10h.
- **al** tiene el valor del modo.
- **ah** tiene el nº de columnas en el modo texto

```
void getvideomode(){
    union REGS inregs, outregs;
    inregs.h.ah = 0x0F;
    int86(0x10, &inregs, &outregs);

    printf("\nModo de video actual: 0x%X", outregs.h.al);
    printf("\nNumero de columnas (solo texto): %d", outregs.h.ah);
}
```

```
printf("\n\nDemostracion de la funcion getvideomode()");
getvideomode();
```

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: P1MAIN
Demostracion de la funcion getvideomode(
)
Modo de video actual: 0x4
Numero de columnas (solo texto): 4
```

1.5 textcolor(unsigned char color)

Cambia el color del texto. Modifica el valor de ctexto, una variable global inicializada al inicio del programa.

```
void textcolor(unsigned char color){  
|   ctexto = color;  
}
```

```
printf("\n\nDemostracion de la funcion textcolor(color)");  
printf("\nElige un color para el color del texto de forma numerica\n");  
tmp = mi_getchar();  
tmp = tmp - '0';  
textcolor(tmp);  
printf("\nEsto es un texto de ejemplo, donde no cambia el color");
```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: P1MAIN

```
Demostracion de la funcion textcolor(color)  
Elige un color para el color del texto de forma numerica  
3  
Esto es un texto de ejemplo, donde no cambia el color
```

1.6 textbackground(unsigned char color)

Cambia el color del fondo de la letra. Modifica el valor de cfondo, una variable global.

```
void textbackground(unsigned char color){  
|   cfondo = color;  
}
```

```
printf("\n\nDemostracion de la funcion textbackground(color)");  
printf("\nElige un color para el fondo en forma numerica\n");  
tmp = mi_getchar();  
tmp = tmp - '0';  
textbackground(tmp);  
printf("\nEsto es otro texto de ejemplo, donde no cambia el fondo");
```

```
Demostracion de la funcion textbackground(color)  
Elige un color para el fondo en forma numerica  
5  
Esto es otro texto de ejemplo, donde no cambia el fondo
```

1.7 clrscr(int lineas, int x, int y)

La función debe limpiar la pantalla imprimiendo líneas en blanco con el nº de líneas y las coordenadas a las que debe afectar la función, indicado desde el main.

Para ello, se prepara una función donde:

- **ah** tiene el nº de la función (en este caso la 6).
- **al** tiene el nº de líneas que debe imprimir.

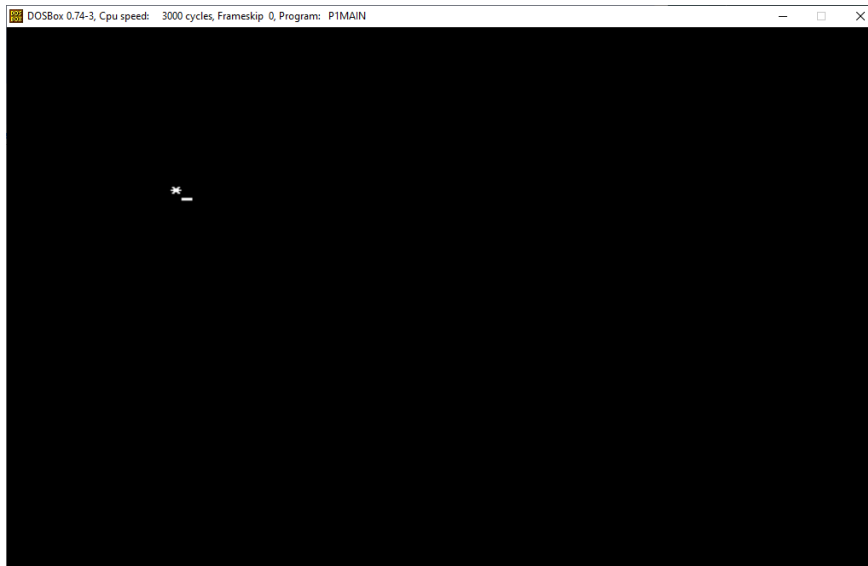
- **bh** tiene el color de los espacios en blanco.
- **ch** la línea de la esquina superior izquierda.
- **cl** la columna de la esquina superior izquierda.
- **dh** tiene el valor del parámetro x (línea de la esquina inferior derecha).
- **dl** tiene el valor del parámetro y (columna de la esquina inferior derecha) .
- Y finalmente la interrupción 10h.

```
void clrscr(int lineas, int x, int y){
    union REGS inregs, outregs;
    inregs.h.ah = 0x06;
    inregs.h.al = lineas;
    inregs.h.bh = 0x0F;
    inregs.h.ch = 0x00;
    inregs.h.cl = 0x00;
    inregs.h.dh = x;
    inregs.h.dl = y;
    int86(0x10, &inregs, &outregs);
    return;
}
```

La llamada a la función desde main, con :

```
clrscr(0, 24, 79);
```

Y la demostración:



1.8 cputchar(char c)

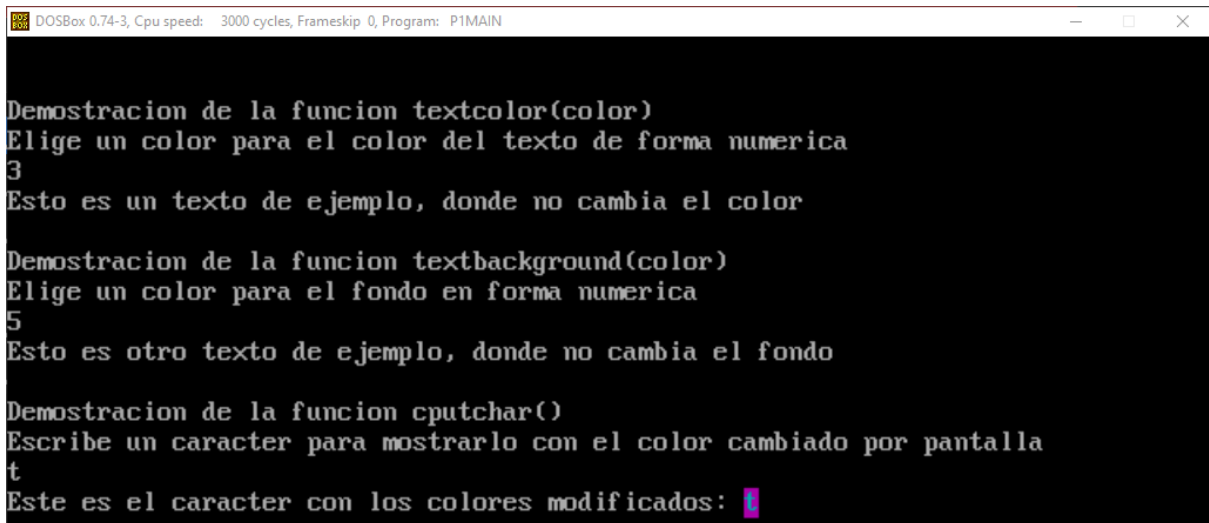
La función debe escribir un carácter 'c' de main en pantalla con el color que tiene indicado actualmente.

Para ello, se prepara una función donde:

- **ah** tiene el nº de la función (en este caso la 9).
- **al** tiene el carácter pasado por parámetro 'c'.
- **bl** tiene el color de ambos, texto y fondo.
- **bh** vale 0, que es la página activa.
- **cx** el nº de veces que repite la acción.
- Y finalmente la interrupción 10h.

```
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;
}
```

```
printf("\n\nDemostracion de la funcion cputchar()");
printf("\nEscribe un caracter para mostrarlo con el color cambiado por pantalla\n");
tmp = mi_getchar();
printf("\nEste es el caracter con los colores modificados: ");
cputchar(tmp);
```



1.9 getche()

La función obtiene un carácter de teclado y lo muestra por pantalla.

Para ello, se prepara una función donde:

- **ah** tiene el nº de la función (en este caso la 0).

- La interrupción 16h.
- **al** tiene el carácter pulsado por teclado.

```
char getche(){
    union REGS inregs, outregs;
    inregs.h.ah = 0x00;
    int86(0x16, &inregs, &outregs);
    return outregs.h.al;
}
```

```
printf("\n\nDemostracion de la funcion getche()");
printf("\nPresiona una tecla: ");
tmp = getche();
printf("\nHas pulsado: %c\n", tmp);
```

```
Demostracion de la funcion getche()
Presiona una tecla:
Has pulsado: w
```

1.10 pixel(int x, int y, unsigned char color)

La función debe pintar un pixel en modo gráfico según los parámetros de coordenadas x e y y el color del punto con color.

Para ello, se prepara una función donde:

- **ah** tiene el nº de la función (en este caso la C).
- **al** tiene el valor del parámetro color.
- **cx** tiene el valor del parámetro x.
- **dx** tiene el valor del parámetro y.
- **bh** vale 0 para la pantalla activa.
- Y por último la interrupción 10h.

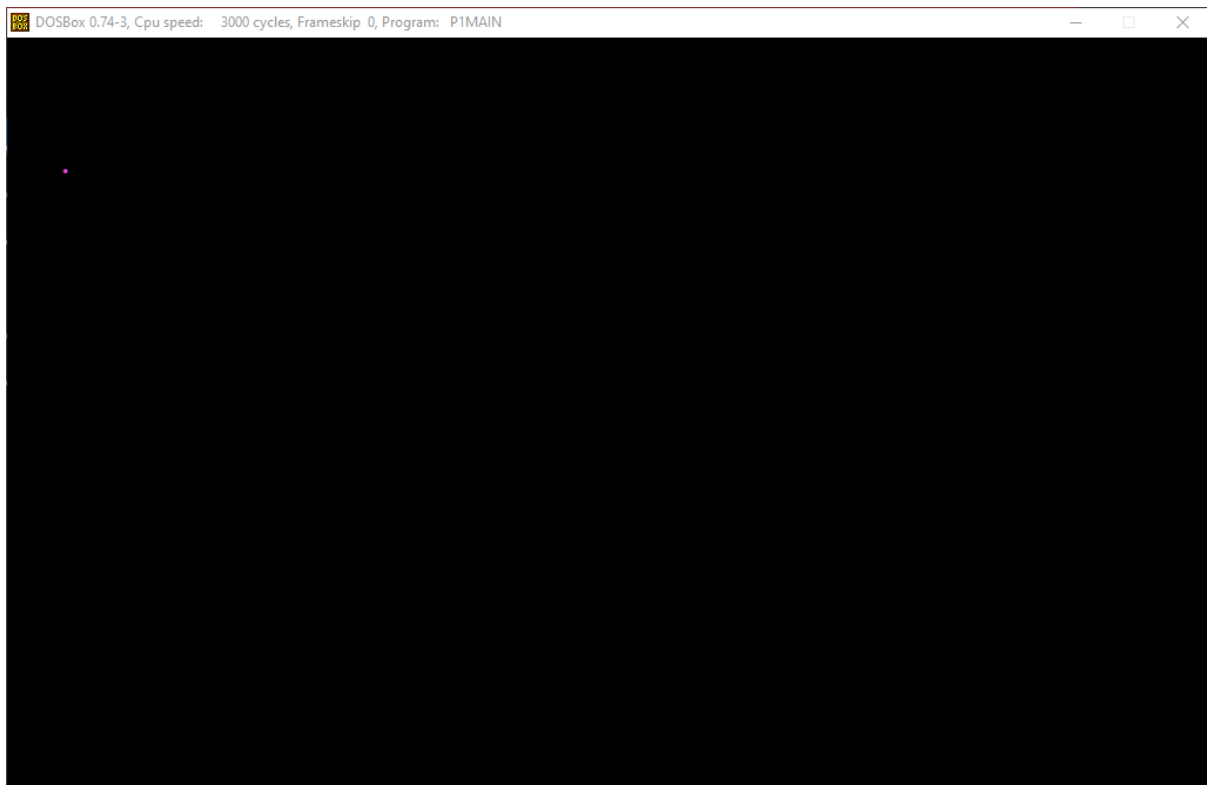
```
void pixel(int x, int y, unsigned char color){
    union REGS inregs, outregs;
    inregs.h.ah = 0x0C;
    inregs.h.al = color;
    inregs.x.cx = x;
    inregs.x.dx = y;
    inregs.h.bh = 0x00;
    int86(0x10, &inregs, &outregs);
}
```

```
printf("\n\nDemostracion de la funcion pixel(x,y,color)");
printf("\nIndica la coordenada x: ");
scanf("%d", &x);
printf("\nIndica la coordenada y: ");
scanf("%d", &y);
printf("\nIndica el color con el que pintar el pixel: ");
color = mi_getchar();
color = color - '0';
setvideomode(4);
pixel(x, y, color);
mi_pausa();
```

```
Demostracion de la funcion pixel(x,y,color)
Indica la coordenada x: 15

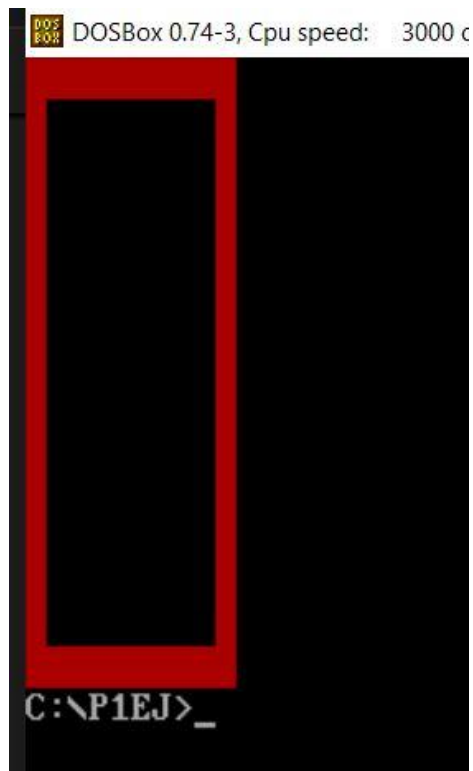
Indica la coordenada y: 35

Indica el color con el que pintar el pixel:
```



Ejercicio 1 extra: Implementar una función que permita dibujar un recuadro en la pantalla en modo texto

```
1  #include <dos.h>
2  #include <stdio.h>
3  #include <conio.h>
4  #define BYTE unsigned char
5
6
7  void dibujar_recuadro(int x1, int y1, int x2, int y2, int color_fondo, int color_texto) {
8      int i, j;
9
10     // Establecer el color de fondo y texto
11     textbackground(color_fondo); // Color de fondo
12     textcolor(color_texto);      // Color del texto
13
14     // Dibujar los bordes horizontales
15     for (i = x1; i <= x2; i++) {
16         gotoxy(i, y1);
17         putch(205); // Carácter de línea horizontal
18         gotoxy(i, y2);
19         putch(205);
20     }
21
22     // Dibujar los bordes verticales
23     for (i = y1; i <= y2; i++) {
24         gotoxy(x1, i);
25         putch(186); // Carácter de línea vertical
26         gotoxy(x2, i);
27         putch(186);
28     }
29
30     // Dibujar las esquinas
31     gotoxy(x1, y1);
32     putch(201); // Esquina superior izquierda
33     gotoxy(x2, y1);
34     putch(187); // Esquina superior derecha
35     gotoxy(x1, y2);
36     putch(200); // Esquina inferior izquierda
37     gotoxy(x2, y2);
38     putch(188); // Esquina inferior derecha
39 }
40
41 int main() {
42     // Limpiar la pantalla antes de dibujar el recuadro
43     clrscr();
44
45     // Ejemplo de uso de la función dibujar_recuadro
46     dibujar_recuadro(1, 1, 10, 15, RED, RED);
47
48     return 0;
49 }
```



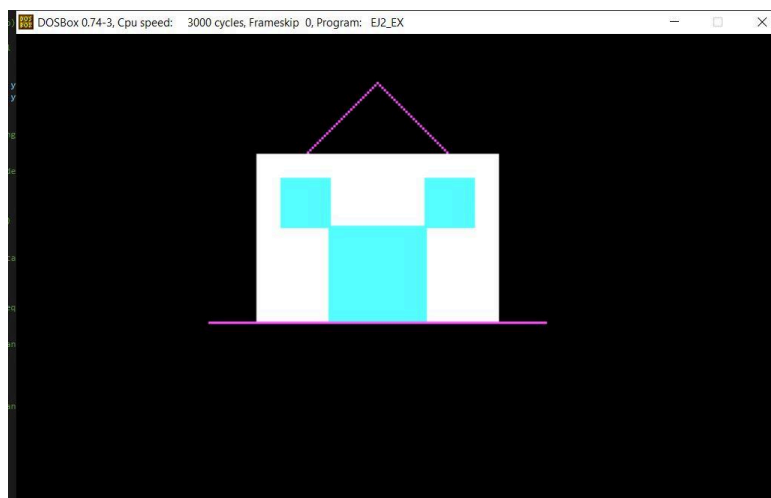
Ejercicio 2 extra: Implementar en lenguaje C un programa que establezca modo gráfico CGA para crear dibujos sencillos en pantalla.

```
1  #include <dos.h>
2  #define BYTE unsigned char
3
4
5  void video_mode (BYTE modo){
6      union REGS inregs, outregs;
7      inregs.h.ah = 0x00;
8      inregs.h.al = modo;
9      int86(0x10, &inregs, &outregs);
10     return;
11 }
12
13 // Función para dibujar un píxel en la pantalla
14 void dibujar_pixel(int x, int y, unsigned char color) {
15     union REGS inregs, outregs;
16     inregs.h.ah = 0x0C; // Función para dibujar un píxel
17     inregs.h.al = color; // Color del píxel
18     inregs.x.cx = x; // Coordenada X
19     inregs.x.dx = y; // Coordenada Y
20     int86(0x10, &inregs, &outregs); // Llamar a la interrupción 10h de la BIOS
21 }
22
23
24 void pause(){
25     union REGS inregs, outregs;
26     inregs.h.ah = 1;
27     int86(0x21, &inregs, &outregs);
28 }
29
```

```

30 int main() {
31     int x;
32     int y;
33     video_mode(4);
34
35     // Dibujar el techo de la casa (triángulo)
36     for (x = 100; x <= 200; x++) {
37         dibujar_pixel(x, 50, 2); // Base del techo
38     }
39     for (y = 50; y >= 20; y--) {
40         dibujar_pixel(100 + (50 - (y - 20)), y, 2); // Lado izquierdo del techo
41         dibujar_pixel(200 - (50 - (y - 20)), y, 2); // Lado derecho del techo
42     }
43
44     // Dibujar las paredes de la casa (rectángulo)
45     for (x = 100; x <= 200; x++) {
46         for (y = 50; y <= 120; y++) {
47             dibujar_pixel(x, y, 3); // Paredes (blanco)
48         }
49     }
50
51     // Dibujar la puerta (rectángulo pequeño)
52     for (x = 130; x <= 170; x++) {
53         for (y = 80; y <= 120; y++) {
54             dibujar_pixel(x, y, 1); // Puerta (cian)
55         }
56     }
57
58     // Dibujar las ventanas (dos cuadrados pequeños)
59     for (x = 110; x <= 130; x++) {
60         for (y = 60; y <= 80; y++) {
61             dibujar_pixel(x, y, 1); // Ventana izquierda (cian)
62         }
63     }
64     for (x = 170; x <= 190; x++) {
65         for (y = 60; y <= 80; y++) {
66             dibujar_pixel(x, y, 1); // Ventana derecha (cian)
67         }
68     }
69
70     // Dibujar el suelo (línea horizontal)
71     for (x = 80; x <= 220; x++) {
72         dibujar_pixel(x, 120, 2); // Suelo
73     }
74
75
76     pause();
77
78     video_mode(3);
79
80     return 0;
81 }

```



Ejercicio 3 extra: Implementar un programa sencillo que realice un dibujo sencillo de tipo “ascii art”

```
1  #include <stdio.h>
2
3  int main() {
4      // Dibujar un corazón en ASCII art
5      printf("  ****  ****\n");
6      printf(" ***** *\n");
7      printf("*****\n");
8      printf(" ***** *\n");
9      printf(" *****\n");
10     printf(" *****\n");
11     printf(" *****\n");
12     printf(" *****\n");
13     printf("  ***\n");
14     printf("   *\n");
15
16     return 0;
17 }
```

C:\P1EJ>ej3_ex.exe

```
  ****  ****
 ***** *
*****
 ***** *
 *****
 *****
 *****
 *****
  ***
   *
```

C:\P1EJ>