



ugr

Universidad
de Granada

PRÁCTICA 1

ENTRADA/SALIDA UTILIZANDO INTERRUPCIONES CON LENGUAJE C

Periféricos y Dispositivos de Interfaz Humana

Autores:
Clara Sola Ruiz
Mario Casas Pérez

1. Introducción	3
2. Funciones obligatorias	4
a. gotoxy()	4
b. setcursortype()	5
c. setvideomode()	7
d. getvideomode()	8
e. textcolor()	9
f. textbackground()	10
g. clrscr()	11
h. cputchar()	12
i. mi_getche()	13
j. pixel()	14
3. Requisitos ampliados	16
a. dibujaRecuadro()	16
b. dibujosGrafico()	17
c. ascii_art()	20
4. Funciones adicionales	23
a. dibujaCirculo()	23
b. dibujaPerro()	24
c. escribirConColor()	26
d. escribirConFondo()	27
e. posiciónActual()	29
f. imprimirCadena()	30
g. dibujaSerpiente()	31
5. Bibliografía	32

1. Introducción

En esta práctica, implementaremos funciones de entrada y salida mediante interrupciones en lenguaje C. Utilizaremos las rutinas de servicio de interrupción de la BIOS para teclado y vídeo desde MS-DOS. Para ello, emplearemos la función no estándar `int86()`, definida en el archivo `dos.h`.

```
#include <dos.h>
int int86(int intno, union REGS *inregs, union REGS *outregs).
```

El parámetro de entrada **intno** nos muestra el número de interrupciones que se desea ejecutar. En **inregs** se indican los valores de los registros antes de la llamada, y en **outregs** se obtienen los valores de los registros tras ser ejecutada la rutina que corresponde. Tanto **inregs** como **outregs** se declaran como una unión de tipo `REGS` que está definida como:

```
union REGS{
    struct WORDREGS x;
    struct BYTEREGS h;
};
```

En consecuencia, podemos acceder a los registros internos ya sea como registros de 16 bits o de 8 bits. La estructura **WORDREGS** y **BYTEREGS** se definen en structs de la siguiente forma:

```
struct WORDREGS{
    unsigned int ax, bx, cx, dx;
    unsigned int si, di, cflag, flags;
};
```

```
struct BYTEREGS{
    unsigned char al, ah, bl, bh;
    unsigned char cl, ch, dl, dh;
};
```

2. Funciones obligatorias

En este apartado se desarrollan las 10 funciones que son requeridas para obtener los 7 puntos como máximo:

a. gotoxy()

Esta función mueve el cursor a una posición en concreto en la pantalla y se toman dos parámetros, x e y, los cuales representan la fila y la columna donde se va a poner el cursor.

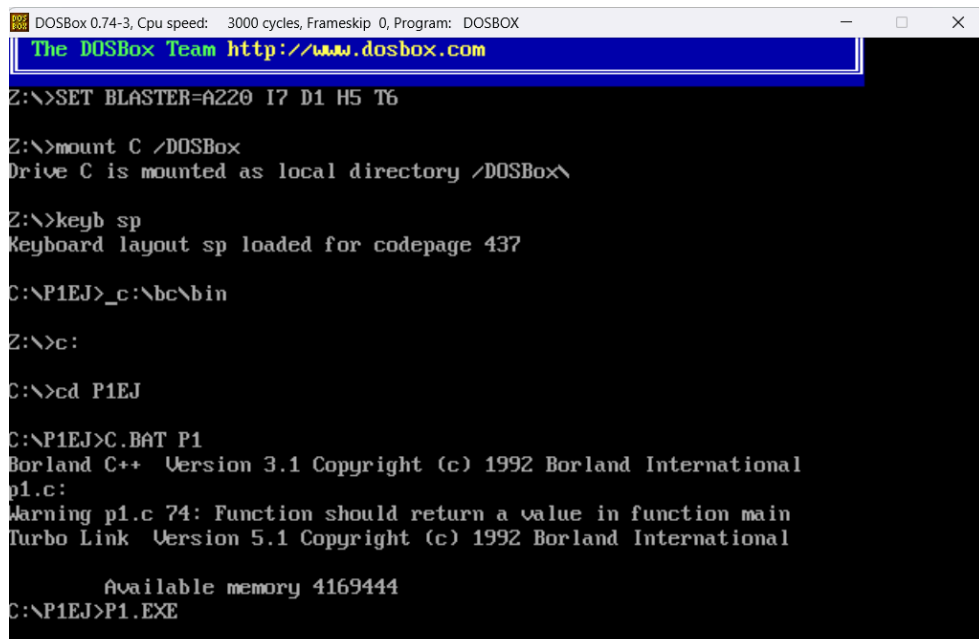
```
// Colocamos el cursor en una posición determinada
void gotoxy(int x, int y)
{
    union REGS inregs, outregs;

    inregs.h.ah = 0x02; // número de la función
    inregs.h.dh = x;    // número de fila
    inregs.h.dl = y;    // número de columna
    inregs.h.bh = 0x00; // Recibe el primer plano de la pantalla

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

int main()
{
    gotoxy(9, 8)
}
```

Los parámetros descrito es que se posicione el cursor en la fila 9, columna 8



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6
Z:\>mount C /DOSBox
Drive C is mounted as local directory /DOSBox\
Z:\>keyb sp
Keyboard layout sp loaded for codepage 437
C:\P1EJ>_c:\bc\bin
Z:\>c:
C:\>cd P1EJ
C:\P1EJ>C.BAT P1
Borland C++ Version 3.1 Copyright (c) 1992 Borland International
p1.c:
Warning p1.c 74: Function should return a value in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International
Available memory 4169444
C:\P1EJ>P1.EXE
```

b. setcursortype()

Esta función ajusta el tipo de cursor en la pantalla. Toma un parámetro (llamado tipo_cursor) el cual es el que determina si el cursor será invisible, normal o grueso. Además, el usuario escribirá un carácter para luego mostrarlo por pantalla previamente a mostrar los diferentes tipos de cursores.

Se ha realizado un switch(tipo_cursor) para que dependiendo del número que se le pase por parámetro a tipo_cursor, ponga este en invisible (0), normal (1) o grueso (2).

```
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 = 010;
            break;
        case 2: // grueso
            inregs.h.ch = 000;
            inregs.h.cl = 010;
            break;
    }
    int86(0x10, &inregs, &outregs);
}

int main()
{
    int tmp;

    printf("\nPulsa una tecla... ");
    tmp = mi_getchar();

    printf("\nHas pulsado: ");
    mi_putchar((char)tmp);

    printf("\nCursor invisible: ");
    setcursortype(0);
    mi_pausa();
    printf("\nCursor grueso: ");
    setcursortype(2);
    mi_pausa();
    printf("\nCursor normal: ");
    setcursortype(1);
    mi_pausa();

    return 0;
}
```

La salida respectiva es la siguiente:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: P1
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>C.BAT P1
Borland C++ Version 3.1 Copyright (c) 1992 Borland International
p1.c:
Warning p1.c 76: Call to function 'mi_getchar' with no prototype in function mai
Warning p1.c 83: Call to function 'mi_pausa' with no prototype in function main
Warning p1.c 86: Call to function 'mi_pausa' with no prototype in function main
Warning p1.c 89: Call to function 'mi_pausa' with no prototype in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

Available memory 4169444
C:\P1EJ>P1.EXE

Pulsa una tecla... M
Has pulsado: M
Cursor invisible:
```

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: P1
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>C.BAT P1
Borland C++ Version 3.1 Copyright (c) 1992 Borland International
p1.c:
Warning p1.c 76: Call to function 'mi_getchar' with no prototype in function main
Warning p1.c 83: Call to function 'mi_pausa' with no prototype in function main
Warning p1.c 86: Call to function 'mi_pausa' with no prototype in function main
Warning p1.c 89: Call to function 'mi_pausa' with no prototype in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

Available memory 4169444
C:\P1EJ>P1.EXE

Pulsa una tecla... M
Has pulsado: M
Cursor invisible:
Cursor grueso: █
```

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: P1
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>C.BAT P1
Borland C++ Version 3.1 Copyright (c) 1992 Borland International
p1.c:
Warning p1.c 76: Call to function 'mi_getchar' with no prototype in function main
Warning p1.c 83: Call to function 'mi_pausa' with no prototype in function main
Warning p1.c 86: Call to function 'mi_pausa' with no prototype in function main
Warning p1.c 89: Call to function 'mi_pausa' with no prototype in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

Available memory 4169444
C:\P1EJ>P1.EXE

Pulsa una tecla... M
Has pulsado: M
Cursor invisible:
Cursor grueso:
Cursor normal: _
```

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>C.BAT P1
Borland C++ Version 3.1 Copyright (c) 1992 Borland International
p1.c:
Warning p1.c 76: Call to function 'mi_getchar' with no prototype in function main
Warning p1.c 83: Call to function 'mi_pausa' with no prototype in function main
Warning p1.c 86: Call to function 'mi_pausa' with no prototype in function main
Warning p1.c 89: Call to function 'mi_pausa' with no prototype in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

Available memory 4169444
C:\P1EJ>P1.EXE

Pulsa una tecla... M
Has pulsado: M
Cursor invisible:
Cursor grueso:
Cursor normal:
C:\P1EJ>_
```

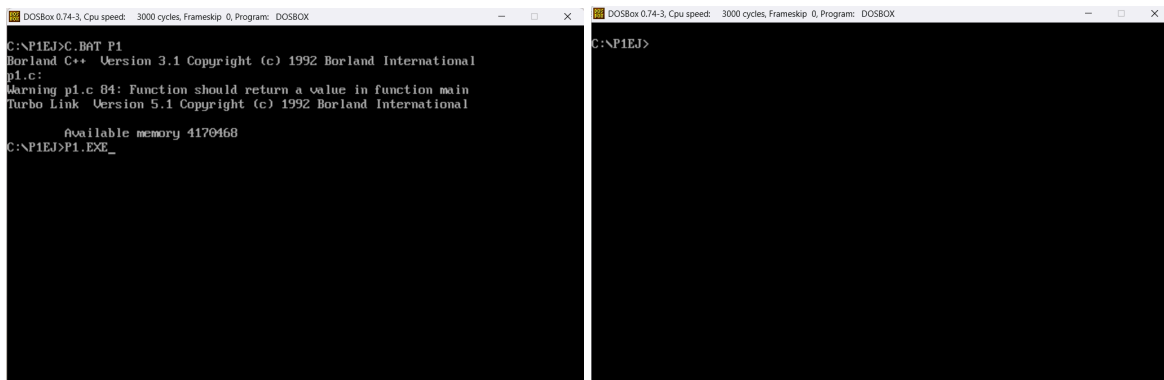
c. setvideomode()

El objetivo de esta función es cambiar el modo de video de la pantalla. La función toma un parámetro (modo), que determinará el nuevo modo de video que vamos a establecer. Usamos modo = 3 para texto y modo = 4 para gráfico.

- modo = 3

```
// Fijamos el modo de video que queramos, 3 para texto y 4 para gráfico
void setvideomode(unsigned char modo)
{
    union REGS inregs, outregs;
    inregs.h.al = modo;
    inregs.h.ah = 0x00;
    int86(0x10, &inregs, &outregs);
}

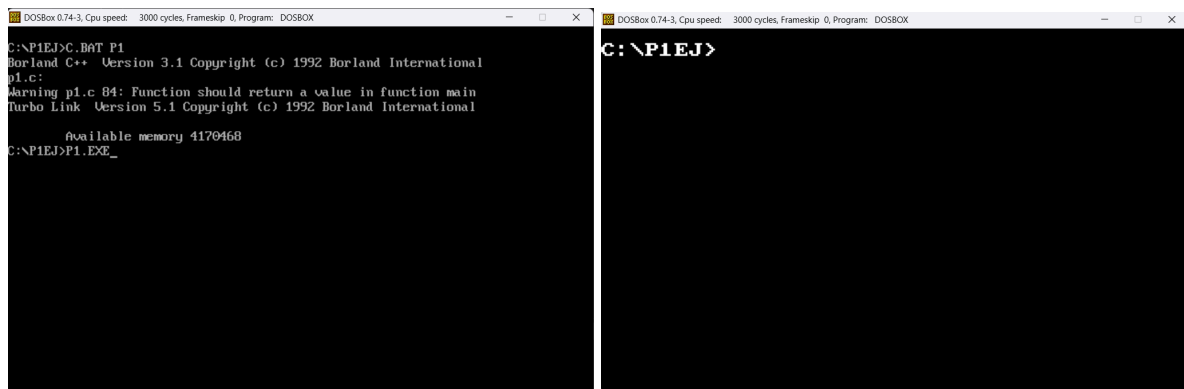
int main()
{
    setvideomode(3);
}
```



- modo = 4

```
// Fijamos el modo de video que queramos, 3 para texto y 4 para gráfico
void setvideomode(unsigned char modo)
{
    union REGS inregs, outregs;
    inregs.h.al = modo;
    inregs.h.ah = 0x00;
    int86(0x10, &inregs, &outregs);
}

int main()
{
    setvideomode(4);
}
```



d. getvideomode()

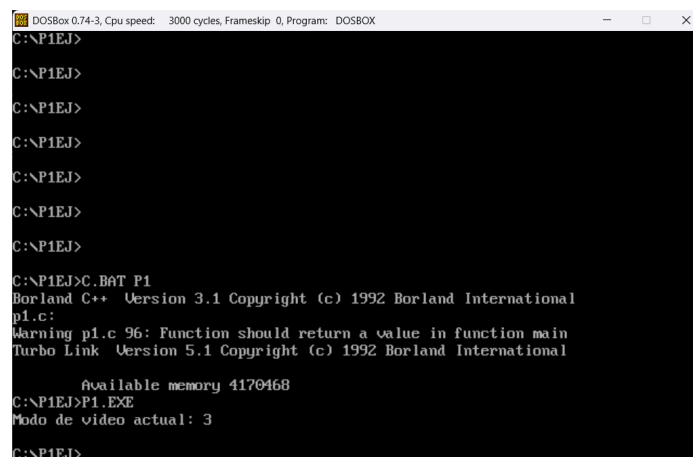
Esta función permite obtener el modo de video actual de la pantalla. Llama a la BIOS para conseguir el modo de video que se está usando y devuelve el valor del modo de video como un entero sin signo.

```
// Vamos a obtener el modo de video actual
unsigned int getvideomode(void)
{
    union REGS inregs, outregs;
    inregs.h.ah = 0x0F;           // cogemos el valor 0x0F que es la función de BIOS que obtiene el modo de video actual
    int86(0x10, &inregs, &outregs); // número de interrupción correspondiente

    return outregs.h.al;
}

int main()
{
    unsigned int modoVideo = getvideomode();
    printf("Modo de video actual: %u\n", modoVideo);
}
```

- **modo texto** (deberá de devolver el valor de 3)



- **modo gráfico** (deberá de devolver el valor de 4)

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\P1EJ>C.BAT P1
Borland C++ Version 3.1 Copyright (c) 1992 Borland International
p1.c:
Warning p1.c 97: Function should return
a value in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International
Available memory 4170468
C:\P1EJ>P1.EXE
Modo de video actual: 4
C:\P1EJ>

```

e. textcolor()

La función textcolor() cambia el color del texto en pantalla sin imprimir ningún carácter, solo configurando el color de acuerdo con los parámetros especificados mediante una interrupción BIOS.

```

void textcolor(int color) {
    union REGS inregs, outregs;

    ctexto = color; // Se guarda el nuevo color del texto

    inregs.h.ah = 0x09; // Función 09h de la interrupción 10h
    inregs.h.al = ' '; // Aquí solo configuramos el color, por lo tanto ponemos ' '
    inregs.h.bl = (cfondo << 4) | ctexto; // Combinamos color de fondo y texto
    inregs.h.bh = 0x00;
    inregs.x.cx = 1;

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

```

```

int main() {
    clrscr();

    gotoxy(10, 5);
    textcolor(RED);
    cputchar('X');

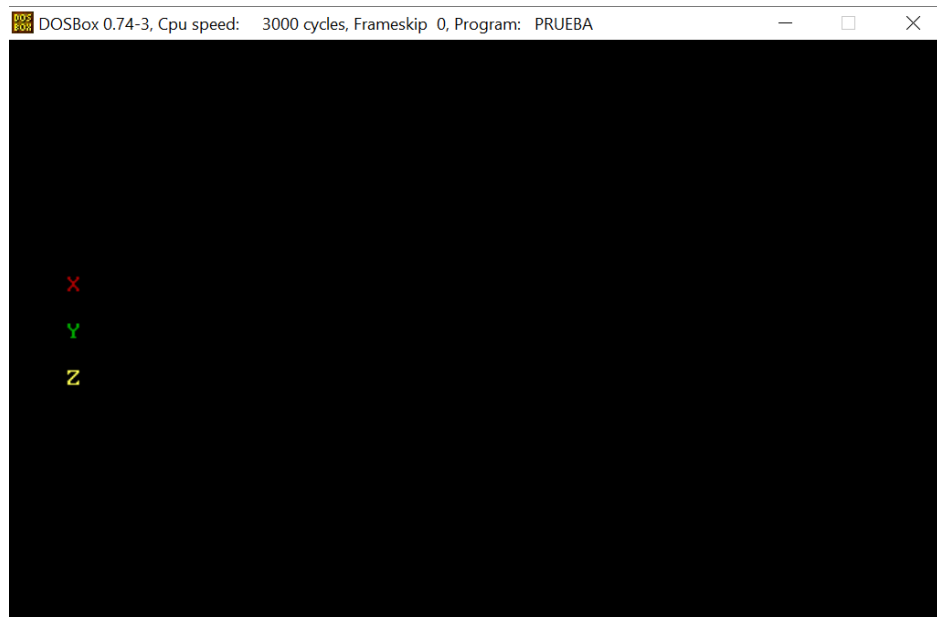
    gotoxy(12, 5);
    textcolor(GREEN);
    cputchar('Y');

    gotoxy(14, 5);
    textcolor(YELLOW);
    cputchar('Z');

    mi_pausa();
    return 0;
}

```

En este ejemplo podemos hacer uso de la función gotoxy() (que se implementará posteriormente) y podemos ver cómo se imprimen las letras de distintos colores.



f. textbackground()

Esta función configura el color de fondo que se usará para escribir texto en la pantalla, pero no escribe nada por sí misma. El color de fondo se aplicará a los caracteres posteriores que se escriban.

```
void textbackground(int color) {
    union REGS inregs, outregs;

    cfondo = color; // Se guarda el nuevo color de fondo

    inregs.h.ah = 0x09;
    inregs.h.al = ' ';
    inregs.h.bl = (cfondo << 4) | ctexto; // Combinamos color de fondo y texto
    inregs.h.bh = 0x00;
    inregs.x.cx = 1;

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

```
int main() {
    clrscr();

    gotoxy(10, 5);
    textbackground(RED);
    textcolor(YELLOW);
    cputchar('X');

    gotoxy(12, 5);
    textbackground(GREEN);
    textcolor(MAGENTA);
    cputchar('Y');

    gotoxy(14, 5);
    textbackground(YELLOW);
    textcolor(BLUE);
    cputchar('Z');

    mi_pausa();
    return 0;
}
```

En este ejemplo también hacemos uso de la función gotoxy(), y cambiamos tanto el fondo de ese píxel como el de la letra, con la anterior función.

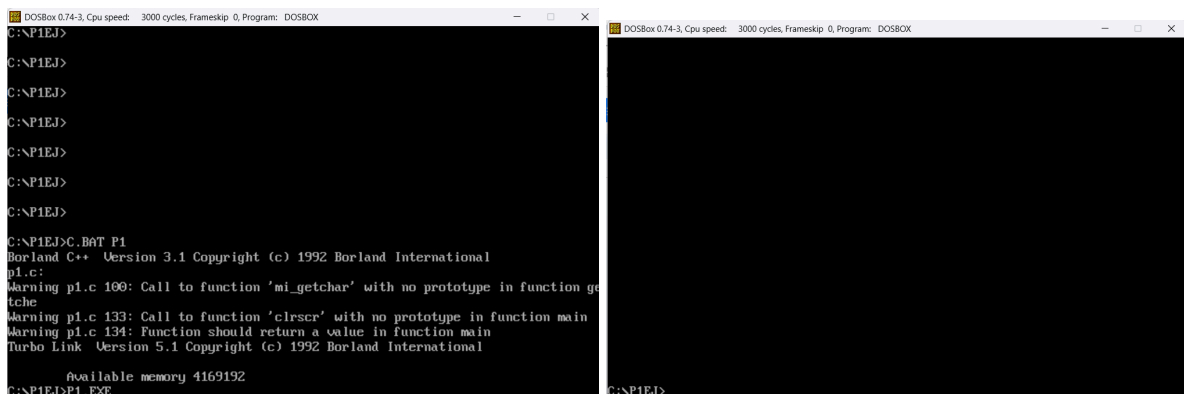


g. clrscr()

Esta función se encarga de limpiar la pantalla, colocando el cursor en la esquina superior izquierda y borrando todo el contenido que pueda haber en la pantalla, usando una llamada a la BIOS.

```
// borra toda la pantalla
void clrscr()
{
    union REGS inregs, outregs;
    inregs.h.ah = 0x06; // número de la función
    inregs.h.al = 0;    // número de líneas a desplazar, en donde 0 indica toda la pantalla
    inregs.h.dh = 25;   // fila inferior derecha de la parte a desplazar
    inregs.h.dl = 80;   // columna inferior derecha de la parte a desplazar
    inregs.h.bh = 0x00; // color
    inregs.x.cx = 0x00; // fila y columna superior izquierda de la parte a desplazar
    int86(0x10, &inregs, &outregs);
    return;
}

int main()
{
    clrscr();
}
```



h. cputchar()

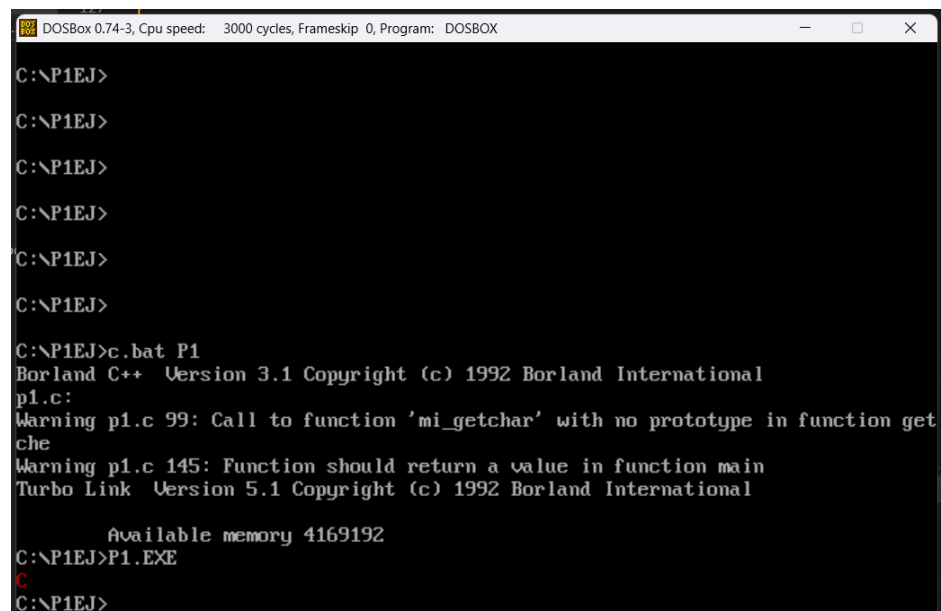
Este método se encarga de imprimir un carácter en la pantalla con un determinado color de fondo y de texto usando interrupciones del BIOS. Para ello, se han definido los siguientes atributos:

- **unsigned char cfondo = 0** (indica que el fondo sea negro)
- **unsigned char ctexto = 4** (indica que el texto sea de color rojo).

Vamos a escribir el caracter C en rojo:

```
// se escribe un caracter por pantalla indicando el color
void cputchar(unsigned char character)
{
    union REGS inregs, outregs;
    inregs.h.ah = 0x09;
    inregs.h.al = character;
    inregs.h.bl = cfondo << 4 | ctexto;
    inregs.h.bh = 0x00;
    inregs.x.cx = 1;
    int86(0x10, &inregs, &outregs);
    return;
}

int main()
{
    cputchar('C');
}
```



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

C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>
C:\P1EJ>c.bat P1
Borland C++ Version 3.1 Copyright (c) 1992 Borland International
p1.c:
Warning p1.c 99: Call to function 'mi_getchar' with no prototype in function get
che
Warning p1.c 145: Function should return a value in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

    Available memory 4169192
C:\P1EJ>P1.EXE
C
C:\P1EJ>
```

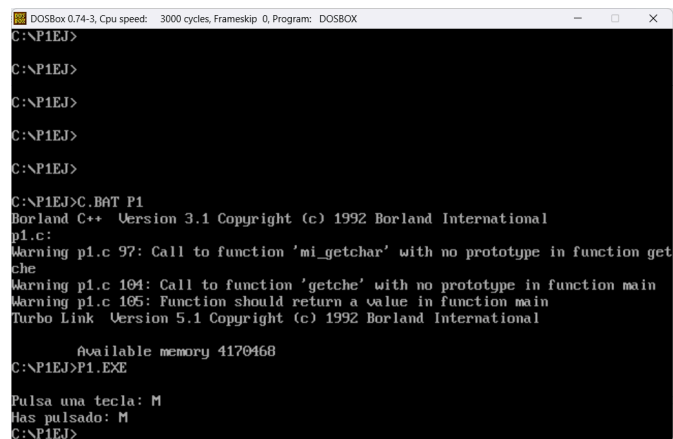
i. mi_getche()

Esta función obtiene un carácter del teclado y lo muestra por pantalla. Para ello hemos usado las funciones `mi_getchar()` y `mi_putchar(char c)` dadas en el guión de prácticas para, obtener el carácter escrito por el usuario y posteriormente mostrarlo por pantalla, respectivamente.

```
// Obtenemos el caracter del teclado y lo mostramos por la pantalla
void mi_getche()
{
    int tmp;
    printf("\nPulsa una tecla: ");
    tmp = mi_getchar();
    printf("\nHas pulsado: ");
    mi_putchar((char)tmp);
}

int main() {
    mi_getche();
}
```

- Pulsando tecla M:



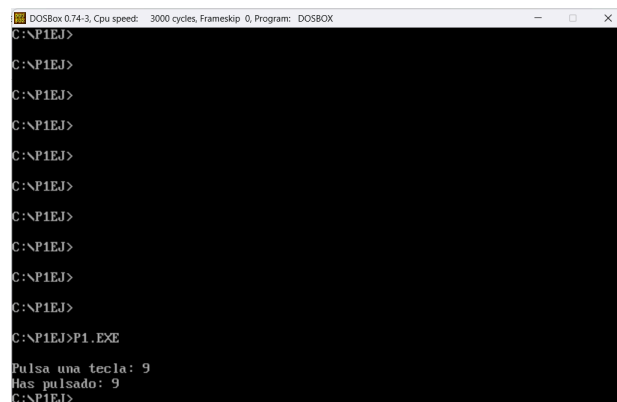
The screenshot shows a DOSBox window with the following text:

```
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>C.BAT P1
Borland C++ Version 3.1 Copyright (c) 1992 Borland International
p1.c:
Warning p1.c 97: Call to function 'mi_getchar' with no prototype in function get
che
Warning p1.c 104: Call to function 'getche' with no prototype in function main
Warning p1.c 105: Function should return a value in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

    Available memory 4170468
C:\NPIEJ>P1.EXE

Pulsa una tecla: M
Has pulsado: M
C:\NPIEJ>
```

- Pulsando tecla 9:

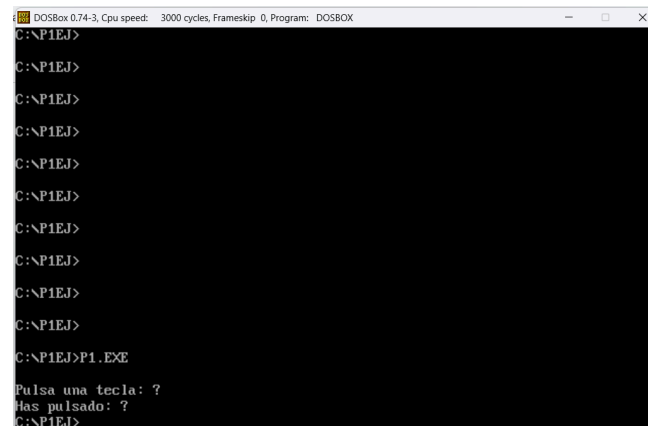


The screenshot shows a DOSBox window with the following text:

```
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>
C:\NPIEJ>P1.EXE

Pulsa una tecla: 9
Has pulsado: 9
C:\NPIEJ>
```

- Pulsando tecla ?:



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>
C:\PIEJ>P1.EXE
Pulsa una tecla: ?
Has pulsado: ?
C:\PIEJ>
```

j. pixel()

```
// ponemos un pixel en la coordenada X,Y de color C
void pixel(int x, int y, unsigned char 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);
}

int main()
{
    setvideomode(MODOGRAFICO); // Tiene el valor 4 que es el modo gráfico

    pixel(250, 300, 1);

    mi_pausa();

    setvideomode(MODOTEXTO); // Tiene el valor 3 que es el modo texto
}
```

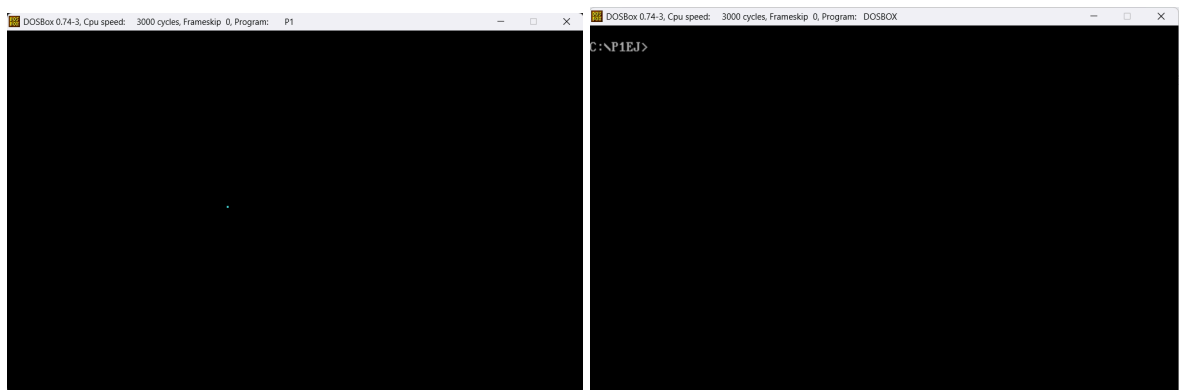
Esta función coloca un píxel en una coordenada concreta x,y que se le pasa por parámetro. Además, se le indicará el color del píxel a poner (valor C).

Hemos definido una variable *MODOTEXTO* y *MODOGRAFICO* para que se establezcan los valores para fijar

el modo de video deseado (3 y 4, respectivamente) . Lo que hace el programa es entrar en modo gráfico, pintar un píxel, y luego cuando queramos volver al modo texto pulsaremos una tecla.

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\P1EJ>C.BAT P1
Borland C++ Version 3.1 Copyright (c) 1992 Borland International
p1.c:
Warning p1.c 100: Call to function 'mi_getchar' with no prototype in function ge
tche
Warning p1.c 123: Call to function 'mi_pausa' with no prototype in function main
Warning p1.c 126: Function should return a value in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

Available memory 4170468
C:\P1EJ>P1.EXE_
```



3. Requisitos ampliados

En este apartado se definen las funciones para conseguir 3 puntos más en la práctica.

a. `dibujaRecuadro()`

Esta función dibuja un recuadro en la pantalla en modo texto. Este método recibe como parámetros las coordenadas superior izquierda e inferior derecha del recuadro, el color del primer plano y el color de fondo. Lo primero que se hace es cambiar los colores según los parámetros descritos, para luego dibujar la línea superior, los lados y la línea inferior del recuadro.

Para realizar correctamente la parte de cambiar el color del texto y del fondo, hemos creado 2 nuevas funciones las cuales modifican el color que tenían los parámetros `ctexto` y `cfondo`.

```
// cambiamos el color del texto
void mi_textcolor(int color)
{
    ctexto = color;
}

// cambiamos el color de fondo
void mi_textbackground(int color)
{
    cfondo = color;
}
```

Luego, la función para dibujar el recuadro en pantalla en modo texto es el siguiente:

```
// dibujo de un recuadro en modo texto
void dibujaRecuadro(int supIzq_x, int supIzq_y, int infDer_x, int infDer_y, unsigned char ctexto, unsigned char cfondo)
{
    int i, j, k;
    char espacio = ' ';

    // cambiamos el color por el que se ha especificado por parámetro
    mi_textcolor(ctexto);
    mi_textbackground(cfondo);

    // línea superior
    for (i = supIzq_x; i < infDer_x - 1; i++)
    {
        gotoxy(i, supIzq_y);
        cputchar(espacio);
    }

    // lados
    for (k = supIzq_y; k < infDer_y; k++)
    {
        gotoxy(infDer_x - 1, k);
        cputchar(espacio);
        gotoxy(supIzq_x, k);
        cputchar(espacio);
    }

    // línea inferior
    for (j = supIzq_x; j < infDer_x - 1; j++)
    {
        gotoxy(j, infDer_y - 1);
        cputchar(espacio);
    }
}
```

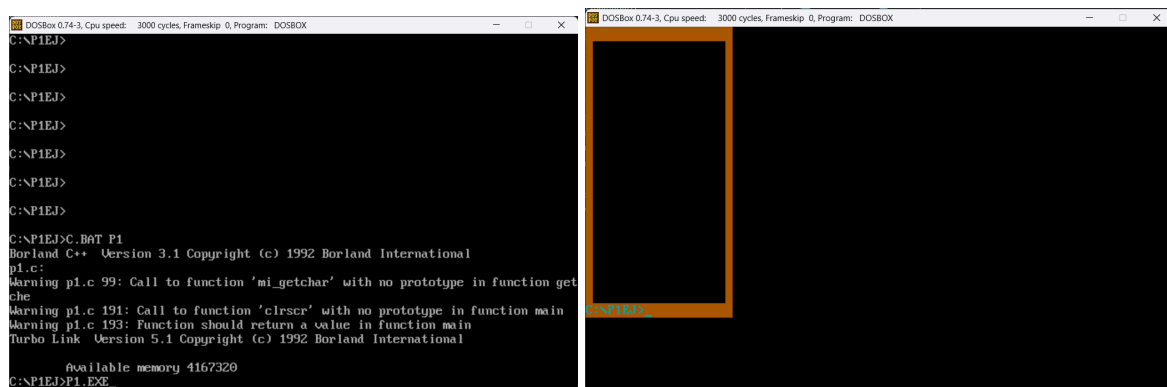

Vemos que hacemos uso de dos funciones previamente realizadas:

- **gotoxy():** para poner el cursor en la posición deseada para dibujar el recuadro.
- **cputchar():** para dibujar los espacios con el color correspondiente.

Para llamar la función desde el main, primeramente limpiamos la pantalla y luego dibujamos el recuadro.

```
int main()
{
    clrscr();
    dibujaRecuadro(0, 0, 20, 20, 3, 6);
}
```

La salida es la siguiente:



b. dibujosGrafico()

En esta función se ha decidido hacer dos dibujos en modo gráfico (modo = 4), uno que sea una casa y otro un monigote. Para la realización de estos dos dibujos se ha realizado lo siguiente:

- **dibujaCasa():**

Este método se encarga de dibujar una casa simple mediante la representación gráfica basada en píxeles (hemos usado para ello el método *píxel* realizado previamente). El dibujo lo hemos dividido por secciones:

- **Base de la casa:** dibuja un cuadrado rosa que representará la base de la casa. Para ello, se ha usado un rango de píxeles en las coordenadas x e y.

```
// base de la casa como rectángulo
for (x = 50; x < 150; x++)
{
    for (y = 100; y < 180; y++)
    {
        pixel(x, y, 2); // color 2 (Rosa)
    }
}
```

- **Techo de la casa:** dibuja un techo triangular azul cian en donde se va ampliando la base conforme se avanza hacia abajo en las coordenadas y. Además, hemos ajustado para cada línea horizontal el rango de coordenadas x donde la variable ancho incrementa con cada iteración del bucle de la coordenada y.

```
// dibujamos el techo
ancho = 0;
for (y = 60; y < 100; y++)
{
    for (x = 75 - ancho; x < 125 + ancho; x++)
    {
        pixel(x, y, 1); // color 1 (Azul)
    }
    ancho++;
}
```

- **Puerta de la casa:** hemos dibujado un rectángulo azul cian pequeño dentro del cuadrado que simula la base de la casa. Además, está centrada en dicha base.

```
// dibujamos la puerta como un rectángulo más pequeño
for (x = 90; x < 110; x++)
{
    for (y = 140; y < 180; y++)
    {
        pixel(x, y, 1); // color 1 (Azul)
    }
}
```

- **Ventana de la casa:** lo hemos dibujado como un cuadrado blanco a la izquierda de la casa.

```
// dibujamos la ventana siendo un cuadrado
for (x = 60; x < 80; x++)
{
    for (y = 120; y < 140; y++)
    {
        pixel(x, y, 3); // color 3 (Blanco)
    }
}
```

- **dibujaMonigote():**

Este método se encarga de dibujar un monigote justo al lado derecho de la casa mediante la representación gráfica basada en píxeles (hemos usado para ello el método *pixel* realizado previamente). El dibujo, al igual que la casa, lo hemos dividido por secciones:

- **Cabeza del monigote:** la hemos dibujado como un cuadrado blanco.

```
// dibujamos la cabeza con un cuadrado
for (x = 170; x < 185; x++)
{
    for (y = 90; y < 105; y++)
    {
        pixel(x, y, 3); // color 3 (Blanco)
    }
}
```

- **Cuerpo del monigote:** lo hemos dibujado como un rectángulo rosa vertical estrecho.

```
// dibujamos el cuerpo como un rectángulo más corto
for (x = 175; x < 180; x++)
{
    for (y = 105; y < 125; y++)
    {
        pixel(x, y, 2); // color 2 (Rosa)
    }
}
```

- **Brazos del monigote:** se han dibujado como un rectángulo azul cian estrecho y horizontal.

```
// dibujamos los brazos como un rectángulo pequeño
for (x = 165; x < 190; x++)
{
    for (y = 110; y < 113; y++)
    {
        pixel(x, y, 1); // color 1 (Azul)
    }
}
```

- **Piernas del monigote:** las piernas las hemos dibujado como dos rectángulos azules cian finos (más que el cuerpo) y en paralelo, los cuales se extienden hacia abajo desde la parte inferior del cuerpo.

```
// dibujamos las piernas los cuales son dos rectángulos finos
for (x = 175; x < 177; x++)
{
    for (y = 125; y < 140; y++)
    {
        pixel(x, y, 1); // color 1 (Azul)
    }
}
for (x = 178; x < 180; x++)
{
    for (y = 125; y < 140; y++)
    {
        pixel(x, y, 1); // color 1 (Azul)
    }
}
```

Para que sea más fácil la llamada al método desde el main, hemos creado la función *dibujosGrafico* la cual se encarga de establecer el modo gráfico, crear los dibujos de la casa y del monigote, hacer una pausa y luego volver al modo texto.

```
void dibujosGrafico()
{
    setvideomode(MODOGRAFICO);

    dibujaCasa();
    dibujaMonigote();

    mi_pausa();

    setvideomode(MODOTEXTO);
}

int main()
{
    dibujosGrafico();
}
```

La salidas correspondientes son las siguientes:



c. `ascii_art()`

La función `ascii_art()` se encarga de dibujar una figura de arte ASCII en la consola utilizando caracteres específicos. En este caso hemos realizado la primera figura propuesta:

```
(\(\
( -.-)
o_(")(")
```

Para ello hemos implementado el siguiente código utilizando las funciones previamente implementadas gotoxy() y cputchar():

```
void ascii_art() {  
    clrscr(); // Borra la pantalla antes de dibujar  
  
    // Orejas  
    gotoxy(10, 5);  
    cputchar('(');  
  
    gotoxy(10, 6);  
    cputchar('\\');  
  
    // Orejas  
    gotoxy(10, 7);  
    cputchar('(');  
  
    gotoxy(10, 8);  
    cputchar('\\');
```

```
    // Cara  
    gotoxy(11, 5);  
    cputchar('(');  
  
    gotoxy(11, 6);  
    cputchar('-');  
  
    gotoxy(11, 7);  
    cputchar('.');  
  
    gotoxy(11, 8);  
    cputchar('-');  
  
    gotoxy(11, 9);  
    cputchar('');
```

```
    // Piernas  
  
    gotoxy(12, 5);  
    cputchar('o');  
  
    gotoxy(12, 6);  
    cputchar('_');  
  
    gotoxy(12, 7);  
    cputchar('(');  
  
    gotoxy(12, 8);  
    cputchar('');  
  
    gotoxy(12, 9);  
    cputchar(')');  
  
    gotoxy(12, 10);  
    cputchar('(');  
  
    gotoxy(12, 11);  
    cputchar('');  
  
    gotoxy(12, 12);  
    cputchar(')');  
  
    mi_pausa();
```

Con esto conseguimos representar a nuestra figura:



4. Funciones adicionales

Aquí se definen funciones extras a parte de las implementadas.

a. `dibujaCirculo()`

Esta función se encarga de crear un círculo en la pantalla en modo texto. Para ello, los parámetros necesarios serán:

- **xc**: coordenada x del centro del círculo.
- **yc**: coordenada y del centro del círculo.
- **radio**: radio del círculo.
- **ctexto**: color de texto que se usará para el círculo.
- **cfondo**: color de fondo del texto donde se dibuja.

Para crear la función para dibujar el círculo nos hemos basado en el algoritmo de Bresenham, el cual es un método rápido para el trazado de líneas en dispositivos gráficos, donde se realizan cálculos con números enteros.

Primeramente se calculan los puntos simétricos del círculo para no hacer cálculos innecesarios. Luego usamos las funciones creadas previamente `gotoxy(x,y)` y `cputchar('O')` para indicar la posición donde dibujar y poner el carácter 'O' en la pantalla. La posición xc y yc determinan la posición del círculo en la pantalla, mientras que el radio determina el tamaño del círculo y ctexto y cfondo son los que ajustan los colores del dibujo.

La función es la siguiente:

```
// dibuja un círculo en modo texto
void dibujaCirculo(int xc, int yc, int radio, unsigned char ctexto, unsigned char cfondo)
{
    int x = 0, y = radio;
    int d = 3 - 2 * radio;

    // Configuramos colores
    mi_textcolor(ctexto);
    mi_textbackground(cfondo);

    while (y >= x)
    {
        // primeramente dibujamos los 8 puntos simétricos
        gotoxy(xc + x, yc + y);
        cputchar('O');
        gotoxy(xc - x, yc + y);
        cputchar('O');
        gotoxy(xc + x, yc - y);
        cputchar('O');
        gotoxy(xc - x, yc - y);
        cputchar('O');
        gotoxy(xc + y, yc + x);
        cputchar('O');
        gotoxy(xc - y, yc + x);
        cputchar('O');
        gotoxy(xc + y, yc - x);
        cputchar('O');
        gotoxy(xc - y, yc - x);
        cputchar('O');

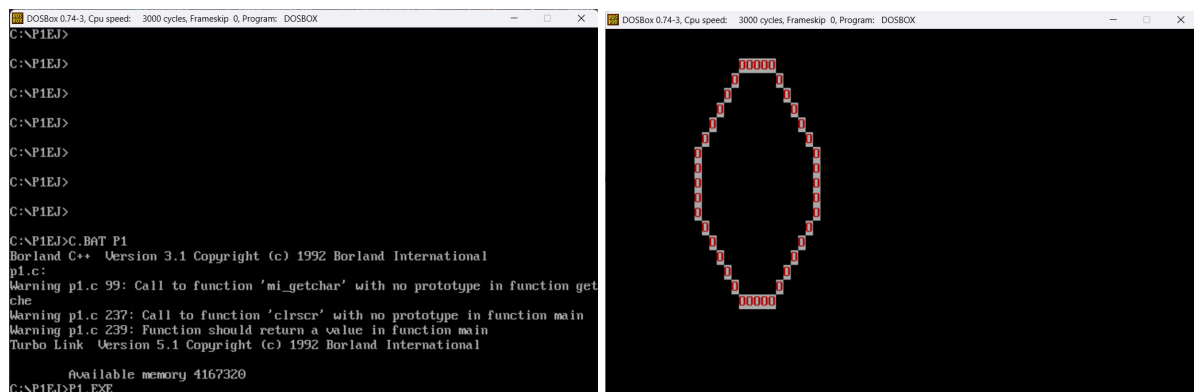
        x++;

        // nos basamos en el algoritmo de Bresenham para calcular d
        if (d > 0)
        {
            y--;
            d = d + 4 * (x - y) + 10;
        }
        else
        {
            d = d + 4 * x + 6;
        }
    }
}
```

Para llamarlo en el main, es muy similar al de *dibujaRecuadro()*, en donde primero limpiamos la pantalla y luego dibujamos:

```
int main()
{
    clrscr();
    dibujaCirculo(10, 20, 8, 4, 7);
}
```

La salida por pantalla es:



b. dibujaPerro()

La función *dibujaPerro()* se encarga de dibujar una figura de un perro mediante el arte ASCII en la consola usando caracteres específicos. Hemos realizado el siguiente dibujo:



Para ello lo hemos implementado con una estructura similar al dibujo del conejo, usando las funciones previamente implementadas gotoxy() y cputchar():

```
void dibujaPerro()
{
    clrscr(); // Borra la pantalla antes de dibujar

    // Orejas
    gotoxy(10, 5);
    cputchar('/');
    gotoxy(10, 7);
    cputchar('\');

    // Cabeza
    gotoxy(10, 8);
    cputchar('_');
    gotoxy(10, 9);
    cputchar('_');
    gotoxy(11, 10);
    cputchar('\');

    // Oculo
    gotoxy(11, 11);
    cputchar('_');
    gotoxy(11, 12);
    cputchar('_');
    gotoxy(11, 13);
    cputchar('_');
    gotoxy(11, 14);
    cputchar('_');
    gotoxy(12, 14);
    cputchar('o');

    // Boca
    gotoxy(13, 13);
    cputchar('/');
    gotoxy(13, 12);
    cputchar('_');
    gotoxy(13, 11);
    cputchar('_');
    gotoxy(13, 10);
    cputchar('_');
    gotoxy(13, 9);
    cputchar('_');
    gotoxy(13, 8);
    cputchar('_');
    gotoxy(13, 7);
    cputchar('c');
    gotoxy(14, 12);
    cputchar('u');

    // Cuello
    gotoxy(14, 8);
    cputchar('/');
    gotoxy(14, 7);
    cputchar('_');
    gotoxy(14, 6);
    cputchar('_');
    gotoxy(14, 5);
    cputchar('_');
    gotoxy(14, 4);
    cputchar('_');
    gotoxy(14, 3);
    cputchar('_');
    gotoxy(14, 2);
    cputchar('_');

    // Espalda
    gotoxy(13, 1);
    cputchar('/');
    gotoxy(12, 3);
    cputchar('/');

    // Oreja
    gotoxy(11, 4);
    cputchar('c');

    // Ojo
    gotoxy(11, 9);
    cputchar('@');

    // Posicion inicial del cursor
    gotoxy(0, 0);

    mi_pausa(); // Detener la ejecucion para observar el dibujo
}
```

Con esta implementación conseguimos representar la figura correspondiente:



c. escribirConColor()

En este método se ha implementado un menú para que el usuario pueda interactuar con el programa. Para ello, hemos creado una función la cual le pide al usuario que introduzca un carácter y un número del 0 al 9 (los diferentes colores que tenemos definidos).

```
#define BLACK 0
#define BLUE 1
#define GREEN 2
#define CYAN 3
#define RED 4
#define MAGENTA 5
#define BROWN 6
#define LIGHTGRAY 7
#define DARKGRAY 8
#define LIGHTBLUE 9
```

Luego, la función se encarga de poner el color que ha introducido el usuario al carácter y, finalmente, mostrarlo por pantalla.

```
void escribirConColor()
{
    char caracter;
    int color;

    printf("Introduce un caracter: ");
    caracter = getchar();
    getchar(); // Debemos de coger el ENTER para evitar interferencias con la entrada del color

    printf("Ahora dime un numero de color (0-9): ");
    color = getchar() - '0'; // Esto lo que hace es convertir el número a carácter

    if (color < 0 || color > 9)
    {
        printf("Ese color no es valido. Se usara el blanco.\n");
        color = WHITE;
    }

    textcolor(color);
    putchar(caracter);
}

int main()
{
    escribirConColor();
}
```

La salida por pantalla es la siguiente:

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\P1EJ>P1.EXE
Introduce un caracter: P
Ahora dime un numero de color (0-9): 2
P
C:\P1EJ>P1.EXE
Introduce un caracter: Ñ
Ahora dime un numero de color (0-9): 1
Ñ
C:\P1EJ>P1.EXE
Introduce un caracter: L
Ahora dime un numero de color (0-9): 3
L
C:\P1EJ>P1.EXE
Introduce un caracter: T
Ahora dime un numero de color (0-9): 4
T
C:\P1EJ>P1.EXE
Introduce un caracter: C
Ahora dime un numero de color (0-9): 8
C
C:\P1EJ>P1.EXE
Introduce un caracter: M
Ahora dime un numero de color (0-9): 9
M
C:\P1EJ>
```

En el caso de que el usuario inserte un caracter en vez de los números puestos en la función para cambiar de color, se mostrará un mensaje indicando que se mostrará el caracter en blanco (el que hemos puesto por defecto).

```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
Warning p1.c 379: Call to function 'dibujaMonigote' with no prototype in function
n dibujosGrafico
Warning p1.c 381: Call to function 'mi_pausa' with no prototype in function dibuj
josGrafico
Warning p1.c 388: Call to function 'clrscr' with no prototype in function ascii_
art
Warning p1.c 446: Call to function 'mi_pausa' with no prototype in function ascii_
art
Warning p1.c 451: Call to function 'clrscr' with no prototype in function dibujo
Perro
Warning p1.c 529: Call to function 'mi_pausa' with no prototype in function dibujo
jaPerro
Warning p1.c 539: Code has no effect in function escribirConColor
Warning p1.c 556: Call to function 'escribirConColor' with no prototype in functi
on main
Warning p1.c 557: Function should return a value in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

Available memory 4151104
C:\P1EJ>P1.EXE
Introduce un caracter: U
Ahora dime un numero de color (0-9): R
Ese color no es valido. Se usara el blanco.
U
C:\P1EJ>
```

d. escribirConFondo()

Está función es muy similar a la anterior, en vez de cambiar el color del carácter, lo que hace es cambiar el fondo donde se mostrará el carácter. Se usan los mismos colores puestos en el método previo pero esta vez teniendo en cuenta que será el color de fondo.

```

#define BLACK 0
#define BLUE 1
#define GREEN 2
#define CYAN 3
#define RED 4
#define MAGENTA 5
#define BROWN 6
#define LIGHTGRAY 7
#define DARKGRAY 8
#define LIGHTBLUE 9

```

Nos encargamos de que a través de un menú el usuario introduzca el carácter que se le mostrará por pantalla. Luego, se le pedirá que introduzca el número del color de fondo que desea. Finalmente, se le muestra por pantalla.

```

void escribirConFondo()
{
    char caracter;
    int fondo;

    printf("Introduce un caracter: ");
    caracter = getchar();
    getchar(); // Debemos de coger el ENTER para evitar interferencias con la entrada del fondo de color

    printf("Ahora dime un numero de color (0-9): ");
    fondo = getchar() - '0'; // Esto lo que hace es convertir el número a carácter

    if (fondo < 0 || fondo > 9)
    {
        printf("Ese color no es valido. Se usara el blanco.\n");
        fondo = WHITE;
    }

    textbackground(fondo);
    putchar(caracter);
}

int main()
{
    escribirConFondo();
}

```

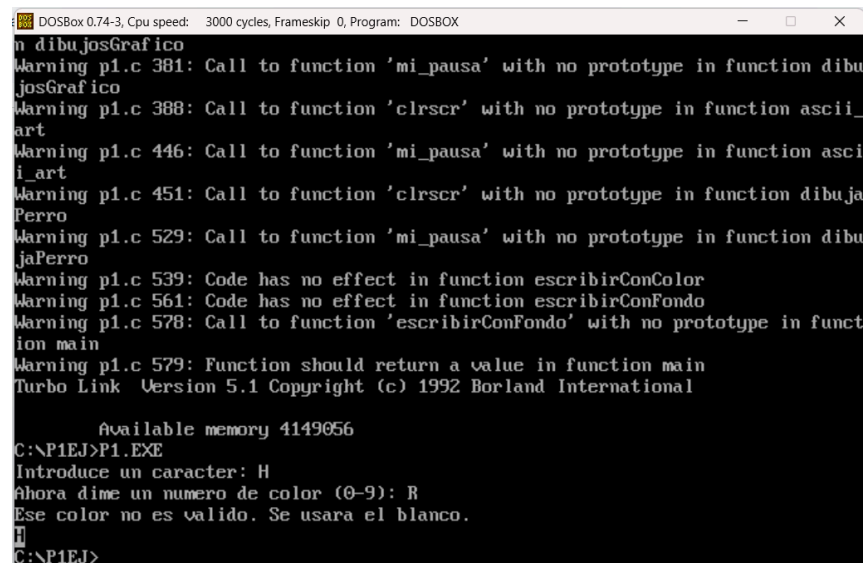
La salida por pantalla es la siguiente:

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
C:\P1EJ>P1.EXE
Introduce un caracter: R
Ahora dime un numero de color (0-9): 1
1
C:\P1EJ>P1.EXE
Introduce un caracter: 2
Ahora dime un numero de color (0-9): 2
2
C:\P1EJ>P1.EXE
Introduce un caracter: P
Ahora dime un numero de color (0-9): 3
3
C:\P1EJ>P1.EXE
Introduce un caracter: U
Ahora dime un numero de color (0-9): 5
5
C:\P1EJ>P1.EXE
Introduce un caracter: Y
Ahora dime un numero de color (0-9): 6
6
C:\P1EJ>P1.EXE
Introduce un caracter: Ñ
Ahora dime un numero de color (0-9): 7
7
C:\P1EJ>_

```

En el caso de que el usuario inserte un carácter en vez de los números puestos en la función para cambiar de fondo, se mostrará un mensaje indicando que se mostrará el carácter en blanco (el que hemos puesto por defecto).



```
DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
n dibujosGrafico
Warning p1.c 381: Call to function 'mi_pausa' with no prototype in function dibujosGrafico
Warning p1.c 388: Call to function 'clrscr' with no prototype in function ascii_art
Warning p1.c 446: Call to function 'mi_pausa' with no prototype in function ascii_art
Warning p1.c 451: Call to function 'clrscr' with no prototype in function dibujarPerro
Warning p1.c 529: Call to function 'mi_pausa' with no prototype in function dibujarPerro
Warning p1.c 539: Code has no effect in function escribirConColor
Warning p1.c 561: Code has no effect in function escribirConFondo
Warning p1.c 578: Call to function 'escribirConFondo' with no prototype in function main
Warning p1.c 579: Function should return a value in function main
Turbo Link Version 5.1 Copyright (c) 1992 Borland International

Available memory 4149056
C:\P1EJ>P1.EXE
Introduce un caracter: H
Ahora dime un numero de color (0-9): R
Ese color no es valido. Se usara el blanco.
H
C:\P1EJ>
```

e. posiciónActual()

La función obtener_posicion_cursor obtiene la posición actual del cursor en la pantalla en DOS utilizando una interrupción del BIOS (int 0x10).

inregs.h.ah = 0x03; → Indica al BIOS que queremos obtener la posición del cursor.

outregs.h.dh; → Devuelve la fila en la que se encuentra el cursor.

outregs.h.dl; → Devuelve la columna en la que se encuentra el cursor.

```
void obtener_posicion_cursor(int *fila, int *columna) {
    union REGS inregs, outregs;

    inregs.h.ah = 0x03; // Obtener posición del cursor
    inregs.h.bh = 0x00; // Página de pantalla

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

    // DL (columna) y DH (fila)
    *columna = outregs.h.dl;
    *fila = outregs.h.dh;
}

int main() {
    int fila, columna;

    clrscr();

    gotoxy(5, 10);
    obtener_posicion_cursor(&fila, &columna);
    printf("%d,%d\n", fila, columna);

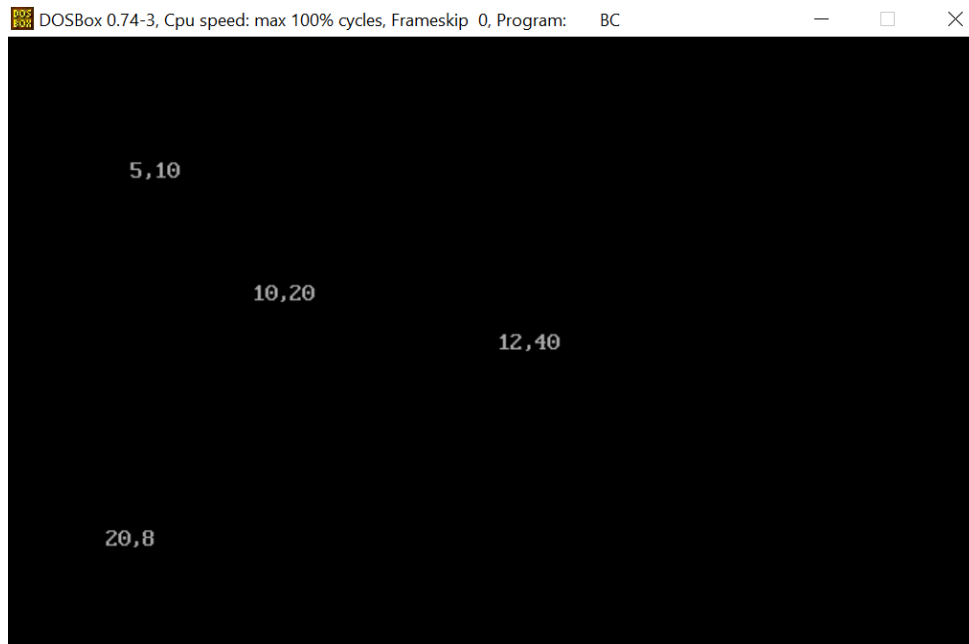
    gotoxy(10, 20);
    obtener_posicion_cursor(&fila, &columna);
    printf("%d,%d\n", fila, columna);

    gotoxy(12, 40);
    obtener_posicion_cursor(&fila, &columna);
    printf("%d,%d\n", fila, columna);

    gotoxy(20, 8);
    obtener_posicion_cursor(&fila, &columna);
    printf("%d,%d\n", fila, columna);

    mi_pausa();
}
```

En el main, utilizamos gotoxy() para mover el cursor a diferentes posiciones y luego llamamos a obtener_posicion_cursor para comprobar que efectivamente se ha posicionado en las coordenadas esperadas, imprimiendo los valores en pantalla.



f. imprimirCadena()

La función tiene como objetivo imprimir una cadena de caracteres que se le pase como argumento, en este caso, las cadenas "Hola Mundo!" y "Periféricos y Dispositivos de Interfaz Humana".

La función recibe como parámetros las posiciones x e y donde debe comenzar a imprimir el texto, la longitud máxima de la cadena a imprimir, y el texto. Para ello, se utiliza un bucle for que recorre la cadena hasta el valor de length o hasta que se alcance el final de la cadena. En cada iteración del bucle, se imprime un carácter mediante la función cputchar, y la posición "y" se incrementa, de modo que el siguiente carácter de la cadena se imprime en la siguiente columna, en la misma fila.

De este modo, la función asegura que la cadena se imprima correctamente, empezando desde las coordenadas indicadas y avanzando horizontalmente a medida que se imprimen los caracteres.

```
void imprimir_cadena(int x, int y, int length, char *text) {
    int i;
    gotoxy(x, y); // Posiciona el cursor

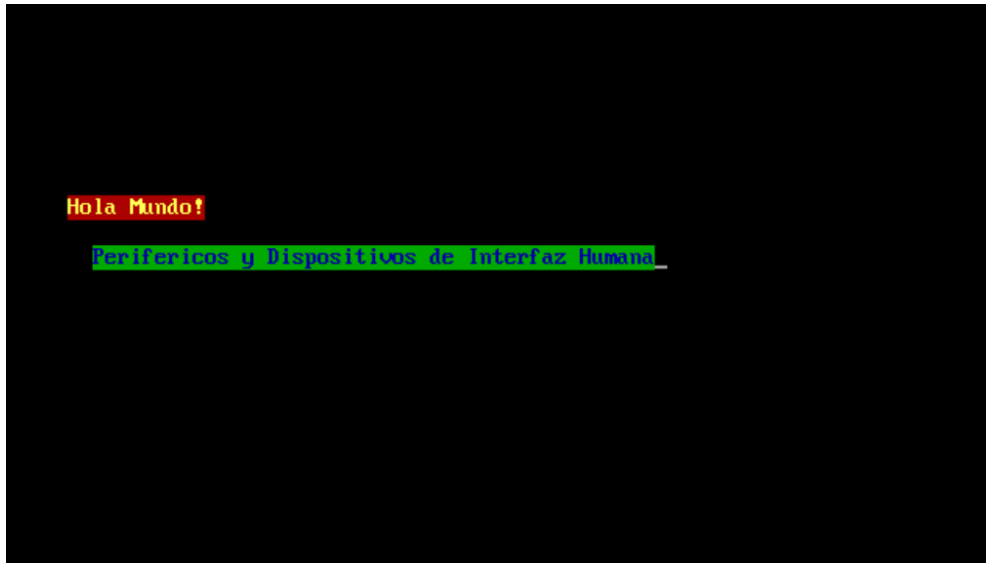
    for (i = 0; i < length && text[i] != '\0'; i++) {
        cputchar(text[i]); // Imprime cada carácter de la cadena
        y++;
        gotoxy(x, y);
    }
}

int main() {
    clrscr(); // Borra la pantalla

    textbackground(RED);
    textcolor(YELLOW);
    imprimir_cadena(10, 5, 20, "Hola Mundo!");

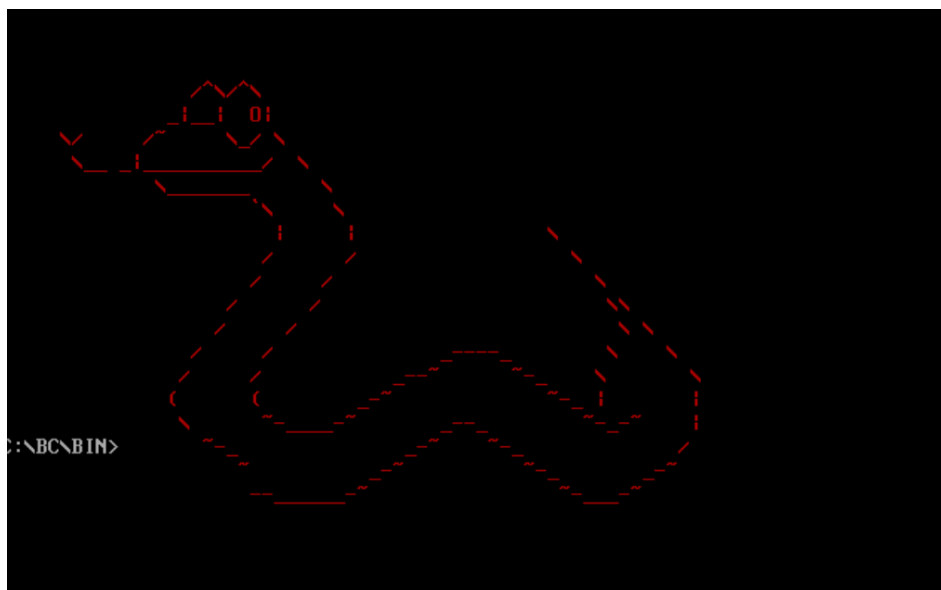
    gotoxy(12, 7);
    textbackground(GREEN);
    textcolor(BLUE);
    imprimir_cadena(12, 7, 60, "Periféricos y Dispositivos de Interfaz Humana");

    mi_pausa();
    return 0;
}
```



g. dibujaSerpiente()

Siguiendo el ejemplo de ascii art, hemos hecho el dibujo de una serpiente (el código es extenso, por lo tanto no lo vamos a insertar aquí, pero se encuentra en el código disponible y sigue el mismo procedimiento que la función `ascii_art`):



5. Bibliografía

P1 - Programación en C. Disponible en:

https://swad.ugr.es/swad/tmp/v2/MmG9Hy_Vj-xZPD8Fj1QIRZfGMPFGuuTvG4zUNVVes/P1-programacionC.pdf

Wikipedia (s.f.) Algoritmo de Bresenham. Disponible en:

https://es.wikipedia.org/wiki/Algoritmo_de_Bresenham (Accedido: 14 de marzo 2025).