Con el objetivo de simplificar la resolución de los ejercicios de entrada/salida, aunque sin alejarnos mucho de la realidad, vamos a trabajar en pseudocódigo C, suponiendo implementadas muchas de las funciones de bajo nivel necesarias para la total programación de estos ejercicios. La programación de este tipo de funciones en bajo nivel es un aspecto que ya se trabaja en los ejercicios de laboratorio.

Vamos a suponer ya programadas las siguientes funciones:

```
unsigned char InPort(registro);
```

Esta función recibe como parámetro la dirección de un registro del espacio de entrada/salida y devuelve el contenido de dicho registro. En los ejercicios no se proporcionan direcciones concretas de registros, sino simplemente nombres o etiquetas. Por ejemplo, el registro de datos del controlador del teclado se referenciará como R_DAT_Ktec, y, en este ejemplo, éste será el parámetro de la función: `InPort(R_DAT_Ktec)`.

```
void OutPort(registro, valor);
```

La función recibe como parámetros la dirección de un registro del espacio de entrada/salida y un valor, de tal forma que actualiza ese registro con dicho valor. Al igual que en el caso anterior, utilizaremos la etiqueta de un registro en lugar de su dirección concreta para, por ejemplo, actualizar el registro de control de un periférico: `OutPort(R_CON_Kper, valor)`.

```
void Eoi();
```

End Of Interruption. Esta función indica que ha finalizado el tratamiento de la interrupción: cuando el controlador de interrupciones recibe esta señal, examina el registro ISR para desactivar el bit de más peso entre aquellos que estén activados. De esta forma, se pueden servir interrupciones de menor prioridad a la que acaba de finalizar.

```
void Strobe(registro);
```

Realiza una secuencia de "*strobe*" en el registro de control de un periférico. En estos ejercicios no tendremos en cuenta el bit concreto sobre el que se realiza el *strobe*, aspecto que sí trabajaremos en los ejercicios de laboratorio.

```
int MAKE(código-tecla);
```

Cada vez que se pulsa/suelta un tecla del teclado, esta función lógica indica si el código de la tecla leído se corresponde con un MAKE (pulsación de la tecla) o BREAK (liberación de la tecla): devuelve un 1 si se ha pulsado la tecla (MAKE), o un 0 si se ha soltado (BREAK).

```
unsigned char LeerIRR();
```

Devuelve el contenido del registro IRR del controlador de interrupciones del PC.

Por otra parte, para inhibir o desinhibir las interrupciones de un periférico actuando sobre el registro IMR del controlador de interrupciones del PC, vamos a suponer implementadas las siguientes funciones:

```
void InhibirKper(entrada_IMR);
void DesinhibirKper(entrada_IMR);
```

Estas funciones modifican en el registro IMR del controlador de interrupciones el bit correspondiente al valor "Entrada_IMR" pasado como parámetro. De esta forma inhibe las interrupciones asociadas a dicho periférico (escribiendo un 1 en el bit correspondiente de IMR) o desinhibe las interrupciones (escribiendo un 0 en dicho bit). Por ejemplo, para inhibir las interrupciones correspondientes al teclado, que interrumpe por la línea IRQ1, hay que modificar el bit 1 de IMR, por lo que la llamada sería: `InhibirKper(1)`.

Por último, para ejecutar en nuestras aplicaciones las rutinas de atención que hemos codificado para los distintos periféricos, hay que cambiar las entradas correspondientes en el vector de interrupción del PC, guardando previamente los valores actuales de las entradas para poder recuperarlas al finalizar el programa. Para ello, supondremos implementadas las siguientes dos funciones:

```
void CambiaVI(entradas_VI);
void RecuperaVI(entradas_VI);
```

Estas funciones reciben como parámetro la lista de entradas del vector de interrupción que se desean cambiar o recuperar en el programa. Por ejemplo, para cambiar las entradas correspondientes al reloj y al teclado, llamaremos a la función de la forma `CambiaVI(0x1C,0x09)`, donde `0x1C` se corresponde con la entrada del reloj y `0x09` será la entrada correspondiente al teclado.

Por otra parte, supondremos también definido el vector TABLA_ASCII[], en el que podemos obtener el código ASCII correspondiente a la tecla pulsada. El índice de este vector es el código MAKE de la tecla pulsada, no el código BREAK, es decir, en el vector podremos obtener el código ASCII de una tecla cuando se pulsa la tecla, nunca al soltarla.