Cabeceras de procedimientos

SEMANA 1

1. **Convierte un valor de tiempo en segundos a horas minutos y segundos.**
2. **Algoritmo para convertir tiempo en segundo a horas.**
3. **Algoritmo para convertir las horas a segundos.**

SEMANA 2

1. **Obtiene el valor entero correspondiente a dígito en formato carácter.**
2. **Área de un triángulo.**
3. **Estimación de la altura.**
4. **Coordenadas diferentes.**
5. **Prueba de impresión. de pi**
6. **Expresión que comprueba si la afirmación se cumple.**
7. **Traducir a coordenada carácter.**
8. **Determinar si le toca a Joan dar de comer a sus perros.**
9. **Algoritmo para convertir los grados Celsius en Fahrenheit y viceversa**

SEMANA 3

1. **Ordena dos valores, el primero el mínimo y el segundo el máximo.**
2. **Obtiene el número de días del mes.**
3. **Calcula la nota final.**

SEMANA 4

1. **Calcula el cociente y el resto de la división entera de dos valores naturales (n>0). (acció)**
2. **¿Cómo está el agua?**
3. **Comprueba si la letra del NIF es correcta.**
4. **Calcula el tamaño de una cadena.**
5. **Determina si hay alguna letra mayúscula en una secuencia de teclado.**

Semana 5:

1. **Comprueba si la letra del NIF es correcta.**
2. **Descompón un número natural en sus dígitos y guárdalo en una tabla.**
3. **Calcula el tamaño de una cadena.**

4. **Determina si hay alguna letra mayúscula en una secuencia de teclado.**
5. **Cuarta prueba, ¿qué tal se os dan los bucles?**

Semana 6:

1. **Frecuencia de cada vocal en un texto.**
2. **Determinar si un carácter pertenece o no a una cadena dada.**
3. **Cuántas montañas y valles hay en la secuencia?**
4. **Todos son triángulos escalenos.**
5. **"Cuantos impares hay."**

Semana 7:

1. **Calcula el perímetro de un polígono irregular.**
2. **Encripta una cadena.**
3. **Ordenación descendente: algortimoorden DESCENDENTE, una tabla de enteros**
4. **Búsqueda dicotómica o binaria: orden descendente.**
5. **Hay suspendidos?**
6. **Número total de aprobados.**
7. **"¿Hay dos pares seguidos?"**

Semana 9:

1. **Calcula el factorial el número que recibe por parámetro.**
2. **Calcula el las permutaciones con repeticiones.**
3. **Calcula las variaciones de m elementos cogidos de n en n (sin repeticiones).**
4. **Lee un número entre [min..max]**
5. **"¿Cuál es el valor mayor?"**

SEMANA 10:

1. **Determina los incrementos de fila y columna para recorrer una matri siguiendo la orientación indicada.**
2. **Comprueba si el carácter pertenece a la cadena.**
3. **Lee un carácter de teclado. El carácter ha de pertenecer a una cadena.**
4. **Inicializa una matriz con un carácter.**
5. **Busca en una matriz un determinado carácter e indica la fila y la columna donde se encuentra.**

SEMANA 11:

1. **Algoritmo para generar un número entero aleatorio acotado.**

2. **Determina un carácter aleatorio acotado [inf..sup]. Los valores inferiores y superiores se determinan según el código ascii correspondiente. Pueden no estar ordenados.**

3. **Cambiar de un jugador al siguiente**

4. **enterercamba el color: si el color era negro, nos devuelve blanco y si era blanco, nos devuelve negro. Si la entrada es incorrecta, nos devuelve la ficha vacía.**

5. **Determinan si una fila y columna dada están dentro de los límites de una matriz [nfilas][mcols]**

6. **Busca un carácter en una matriz de nfilas x mcols, a partir de una determinada posicion (fila,col). La busqueda se realiza en una determinada orientacion. No se evalua la casilla inicial (fila, col). El procedimiento devuelve si existe o no. En caso de que existe, la casilla donde se encuentra el elemento se devuelve a partir de los mismos parametros de entrada fila y col.**

7. **Busca un carácter en una matriz de nfilas x mcols, a partir de una determinada posicion (fila,col). La busqueda se realiza en una determinada orientacion. No se evalua la casilla inicial (fila, col) El procedimiento devuelve si existe o no. En caso de que existe, la casilla donde se encuentra el elemento se devuelve a partir de los mismos parametros de entrada fila y col.**

SEMANA 12:

1. **Compara dos fechas, f1 y f2, y devuelve cierto si son iguales.**
2. **Compara dos fechas, f1 y f2, y devuelve un valor entero según la relación existente**
3. **Muestra por pantalla la fecha formateada: dd-mmm-aaaa**
4. **Imprime por pantalla la información de un récord siguiendo el siguiente formato: dd-mmm-aa Color(Blanco/Negro) Puntuación Nombre. Los campos deben separarse con un tabulador y salto de línea al final.**
5. **Imprime un número determinado de los récords guardados en la tabla.**

SEMANA 13:

1. **Calcula y guarda el área de las esferas. El radio se enterroduce por teclado.**
2. **Calcula y guarda el área de las esferas. Los radios están en el fichero radios.txt**
3. **Lee las palabras almacenadas en un fichero y muestra las que superan un determinado tamaño.**

SEMANA 1

```
/** Convierte un valor de tiempo en segundos a horas minutos y segundos.
*
* @param temps (Valor: entero) Valor del tiempo en segundos.
* @param h (Ref: entero) Número de horas equivalentes.
* @param m (Ref: entero) Número de minutos equivalentes, siempre menor de 60.
* @param s (Ref: entero) Número de segundos equivalentes, siempre menor de 60.
*/
accion  convertir_tiempo (
        /* Parámetros de entrada */  temps: enter,
        /* Parámetros de salida */   var h: enter, var m: enter, var s: enter) es
*****************************************************************************

/** Algoritmo para converitir tiempo en segundo a horas.

 @param temps (Valor: entero) Tiempo en segundos a convetir.
 @return (entero) Devuelve el valor equivalente en horas.
*/
funcion seg_a_horas(
   /* Parámetros de entrada */ temps:enter) rotorna entero es

*****************************************************************************

/** Algoritmo para converitir las horas a segundos.

 @param horas (Valor: entero) Tiempo en horas a convetir.
 @return (entero) Devuelve el valor equivalente en segundos.
*/
funcion horas_a_seg(
   /* Parámetros de entrada */ horas:enter) retorna entero es


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

SEMANA 2

```
/**Obtiene el valor entero correspondiente a dígito en formato carácter.

 @param digito (Valor: caràcter) Caracter correspondiente a un dígito.
 @return (enter) Valor entero correspondiente al dígito.
 */
funcion digito_a_entero (
```

```
   /* Parámetros de entrada */ digito_car: carácter) retorna enter es

*****************************************************************************

/** Área de un triángulo.

 @param base (Valor: entero) Base del triángulo.
 @param altura (Valor: entero) Altura del triángulo.
 @return (real) Área del triángulo.
*/
funcion area_triangulo(
   /* Parámetros de entrada */ base: enter , altura:enter ) retorna real es

*****************************************************************************
/** Estimación de la altura.
 *
 * @param femur (Valor: real) Medida del femur en pulgadas.
 * @param altura_mujer (Ref: real) Estimación de la altura del una mujer a partir del
femur.
 * @param altura_hombre (Ref: real) Estimación de la altura del un hombre a partir del
femur.
 */
accion calcula_alturas (
   /* Parámetro de entrada */  real femur,
   /* Parámetros de salida */ var altura_mujer :real , var  altura_hombre:  real) es

*****************************************************************************
/** Coordenadas diferentes.

 @param x1 (Valor: entero) Valor de la coordenada x dada.
 @param y1 (Valor: entero) Valor de la coordenada y dada.
 @return (boolea) Devuelve cierto si la coordenada dada y
            la del usuario son distenteras y falso en caso contrario.
*/
fumcion coord_distenteras (
   /* Parámetros de entrada */ x1: enter, y1: enter ) retorna bolea es
*****************************************************************************

/** Prueba de impresión. de pi
*/
accion prueba_impresion ( ) es
*****************************************************************************

/** Expresión que comprueba si la afirmación se cumple.
*
* @param x (Valor: entero) Valor de la variable x.
```

```
* @param y (Valor: entero) Valor de la variable y.
* @param z (Valor: entero) Valor de la variable z.
* @return (booleaeano) Devuelve cierto si la afirmación se cumple
*              y falso en caso contrario.
*/
funcion expresion (
   /* Parámetros de entrada */  x:enter , y: enter , z:enter ) retorna bolea es


*****************************************************************************

/** Traducir a coordenada carácter.

 @param coord (Valor: entero) Coordenada como valor entero.
 @return (carácter) Devuelve el carácter correspondiente a la coordenada
            en formato letra.
*/
funcion coord_caracter (
   /* Parámetros de entrada */ coord: enter ) retorna caracter es

*****************************************************************************

/** Determinar si le toca a Joan dar de comer a sus perros.
*
* @param dia_setmana (Valor: carácter) Representa el día de la semana. Tiene los
posibles valores
*              de 'L', 'M', 'X', 'J', 'V', 'S', 'D' para cada día respectivamente.
* @param dia (Valor: entero) Representa el día del mes.
* @return (booleaea) Devuelve cierto si le toca a Joan dar de comer a los perros
*                y falso en caso contrario.
*/
funcion le_toca_a_Joan (
   /* Parámetros de entrada */  dia_setmana: carácter , dia: enter) retorna bolea es

*****************************************************************************

  /**
 * Primera prueba para "bug hunter":
 *                "Algoritmo para convertir los grados Celsius en Fahrenheit
 *                y viceversa"
 */
acció bug_hunters_01() és
*****************************************************************************
```

Semana 3

```
/** Ordena dos valores, el primero el mínimo y el segundo el máximo.
 *
 * @param min (Ref: enter) Primer valor. Al finalizar contendrá el mínimo de los dos.
 * @param max (Ref: enter) Segundo valor. Al finalizar contendrá el máximo de los
dos.
*/
accion min_max (
   /* Parámetros de entrada/salida */ var min: enter, var max:enter) es


*****************************************************************************

/** Obtiene el número de días del mes.
 *
 * @param mes (Valor: enter) Mes en formato numérico.
 * @param any (Valor: enter) Año.
 * @return (enter) Devuelve el número de días que tiene el mes.
*/
funcion dias_mes (
   /* Parámetros de entrada */ mes: enter , any: enter) retorna enter es

*****************************************************************************
/** Calcula la nota final.
 *
 * @param qt (Valor: real) Nota de teoría.
 * @param qp (Valor: real) Nota de problemas.
 * @param qpr (Valor: real) Nota de prácticas.
 * @param qpa (Valor: real) Nota de participación.
 */
funcion calcula_nota (
   /* Parámetros de entrada */  qt: real , qp: real , qpr: real , qpa: real) retorna real es


*****************************************************************************

/**
Segunda prueba "Bug hunter".
 *              "Programa para convertir los grados Celsius en Fahrenheit
 *               y viceversa"
 *   @param f (Ref:real) temperatura dada en Fahrenheit
 *   @param c (Ref:real) temperatura dada en Celsius
 */
acció bug_hunters_02(var f:real , var c: real ) és
```

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

Semana 4:

```
/** Calcula el cociente y el resto de la división entera de dos valores naturales (n>0).
 *
 * @param dividiendo (Valor: enter) Dividendo (>0)
 * @param divisor (Valor: enter) Divisor (>0)
 * @param cociente (Ref: enter) Cociente de realizar la división entera.
 * @param resto (Ref: enter) Resto de realizar la división entera.
*/
accion cociente_resto (
   /* Parámetros de entrada */     dividendo:enter , divisor:enter ,
   /* Parámetros de salida */     var  coc:enter, var rest:enter)  es

*****************************************************************************

/** Calcula el cociente y el resto de la división entera de dos valores naturales (n>0).
 *
 * @param n (Valor: entero) Número (>0) del que queremos calcular los 4 cuadrados.
 * @return (entero) Número de combinaciones de cuadrados que pueden darse.
*/
funcion cuatro_cuadrados_lagrange (
   /* Parámetros de entrada */     n:enter) retona enter es
*****************************************************************************

/** Calcula el cociente y el resto de la división entera de dos valores naturales (n>0).
 *
 * @param n (Valor: entero) Número (>0) del que queremos calcular los 4 cuadrados.
 * @return (entero) Número de combinaciones de cuadrados que pueden darse.
*/
funcion cuatro_cuadrados_lagrange (
   /* Parámetros de entrada */     n:enter) retorna enter es
*****************************************************************************

/**
Tercera prueba "Bug hunter".
 *              "¿Cómo está el agua?"

 */
funcion bug_hunters_03(
   /* Parámetros de entrada */ temperatura: real) retorna enter es

*****************************************************************************

/** Comprueba si la letra del NIF es correcta.
```

```
 *
 * @param dni (Valor: entero) DNI, número del NIF.
 * @param letra (Valor: carácter) Letra del NIF.
 * @param letras_nif (Ref: tabla de carácter) Tabla con las letras del nif.
 * @return (boolea) Devuelve cierto si la letra se corresponde con el DNI y
 *                falso en caso contrario.
*/
funcion nif_correcto (
        /* Parámetros de entrada */     dni: enter , letra: carácter,var  :letras_nif: taula[]
de carácter ) retorna bolea es

*****************************************************************************

/** Descompón un número natural en sus dígitos y guárdalo en una tabla.
 *
 * @param n (Valor: natural) Número natural (>0) que queremos descomponer.
 * @param digitos (Ref: taula de naturales) Tabla con los dígitos del n.
 * @param dim (Valor: enter) Tamaño máximo de la tabla digitos[]. Siempre
 *     será >= DIM_MAX.
*/

accion descompon_natural (
        /* Parámetros de entrada */ n:enter,
        /* Parámetros de salida */    var digitos: taula[] d'enter,
        /* Parámetros de entrada */ dim:enter)  es


*****************************************************************************

/** Calcula el tamaño de una cadena.
 *
 * @param cadena (Ref: taula de caràcters) Cadena.
 * @return (entero) Devuelve el tamaño de la cadena.
*/
funcion medida_cadena (
        /* Parámetros de entrada */ cadena: taula [] de carácter) retorna enter es

*****************************************************************************


/** Determina si hay alguna letra mayúscula en una secuencia de teclado.
 *
 * @retorna (booleà) Retorna cierto si hay una mayúscula y
 *            falso en caso contrario.
*/
funcionl hay_mayuscula ()  retorna bolea es
```

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

/**
 * @param temperatura (Valor: real) real que queremos cambiar de unidad.
 * @return (entero).
funcio bug_hunters_03(
   /* Parámetros de entrada */ temperatura: real ) retorna entero es


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

Semana 5:

```
/** Comprueba si la letra del NIF es correcta.
 *
 * @param dni (Valor: entero) DNI, número del NIF.
 * @param letra (Valor: carácter) Letra del NIF.
 * @param letras_nif (Ref: tabla de carácter) Tabla con las letras del nif.
 * @return (booleà) Devuelve cierto si la letra se corresponde con el DNI y
 *                falso en caso contrario.
*/
funcion nif_correcto (
         /* Parámetros de entrada */     dni: enter , letra: carácter , var letras_nif:taula[]
de  carácter) retorna bolea es
*****************************************************************************

/** Descompón un número natural en sus dígitos y guárdalo en una tabla.
 *
 * @param n (Valor: natural) Número natural (>0) que queremos descomponer.
 * @param digitos (Ref: taula de naturales) Tabla con los dígitos del n.
 * @param dim (Valor: enter) Tamaño máximo de la tabla digitos[]. Siempre
 *      será >= DIM_MAX.
*/

accion descompon_natural (
         /* Parámetros de entrada */ n: naturals,
         /* Parámetros de salida */    digitos: taula [] de naturals,
         /* Parámetros de entrada */ dim: enter) es
*****************************************************************************


/** Calcula el tamaño de una cadena.
 *
 * @param cadena (Ref: taula de caràcters) Cadena.
 * @return (entero) Devuelve el tamaño de la cadena.
```

```
*/
funcion medida_cadena (
        /* Parámetros de entrada */ cadena: taula[] de caràcters) retorna entero es

*****************************************************************************


/** Determina si hay alguna letra mayúscula en una secuencia de teclado.
 *
 * @retorna (booleà) Retorna cierto si hay una mayúscula y
 *            falso en caso contrario.
*/
funcion hay_mayuscula (void) retorna bolea es
*****************************************************************************
* @param notas (Ref: taula de reales) Tabla con los dígitos del n.
* @param dim (Valor: enter) Tamaño máximo de la tabla notas[]. Siempre
*     será >= DIM_MAX.
/** Cuarta prueba, ¿qué tal se os dan los bucles?

 * @retorna (Real) retorna les notes que hi ha a la taula notas[].

funcion bug_hunters_04 (
   /* Parámetros de entrada */ var notas: taula[] de real , dim:enter) retorna real es
*****************************************************************************
```

Semana 6:

```
/** Frecuencia de cada vocal en un texto.
 *
 * @param freq_vocales (Ref: taula[] de reales) Tabla para guardar la frecuencia
 *              de cada una de las vocales (dim = 5).
*/
accion frecuencia_vocales (
        /* Parámetros de entrada/salida */  var freq_vocales:taula[] de real) es


*****************************************************************************

/** Determinar si un carácter pertenece o no a una cadena dada.

 * @param cadena (Ref: taula[] de caràcter) Tabla que contiene la cadena.
 * @param letra (Valor: caràcter) Caracter que vamos a comprobar.
 * @return (booleà) Devuelve cierto si letra pertenece a la cadena y
 *            falso en caso contrario (case sensitive).
```

```
*/

funcion pertenece_cadena (
        /* Parámetros de entrada */ var cadena: taula[] de carácter: , letra: carácter)
retorna bolea es

*****************************************************************************

/** Cuántas montañas y valles hay en la secuencia?
 *
 * @param m (Ref: enter) Devuelve el número de montañas de la secuencia.
 * @param v (Ref: enter) Devuelve el número de valles de la secuencia.
 *
*/
accion montanyas_valles (
   /* Parámetros de salida */   var m: enter , var v:enter ) es
*****************************************************************************


/** Todos son triángulos escalenos.
 *
 * @return (booleá) Devuelve cierto si todos son escalenos
 *             y falso en caso contrario.
*/
funcion todos_escalenos () retorna booleà es


*****************************************************************************
 /**Quentera prueba, entre recorridos y búsquedas.
 *              "Cuantos impares hay."

* @return (entero) Devuelve el número de enteros que se ha encontrado.

funcion bug_hunters_05 () retorna entero es
*****************************************************************************
```

Semana 7:

```
/** Calcula el perímetro de un polígono irregular.
 *
 * @return (real) Devuelve el valor del perímetro calculado
 *         a partir de la secuencia ordenada de los puntos, (x, y),
 *         que forman el polígono.
```

```
*/
funcion calcula_perimetro (void) retora real es
*****************************************************************************

/** Encripta una cadena.
 *
 * @param cadena (Ref: taula[] de caràcter) Tabla que contiene la cadena a encriptar.
 *
*/
accion encripta_cadena (
        /* Parámetros de entrada/salida */  var cadena: taula[[] de carácter ) és

*****************************************************************************

/** Ordenación descendente: algortimoorden DESCENDENTE, una tabla de enteros
 *
 * @param t (Ref: taula[] de enter) Tabla a ordenar.
 * @param dim (Valor: enter) Tamaño de la tabla. Dimensión máxima.
 *
*/
accion  ordena_descendente (
        /* Parámetros de entrada/salida */  var t:taula[] de enter:,
        /* Parámetros de entrada */         dim: enter ) es

*****************************************************************************

/** Búsqueda dicotómica o binaria: orden descendente.
 *
 * @param t (Ref: taula[] de enter) Tabla donde buscamos.
 * @param dim (Valor: enter) Tamaño de la tabla. Dimensión máxima.
 * @param valor (Valor: enter) Dato que vamos a buscar.
 * @return (enter) Devuelve la posición, dentro de la tabla,
 *          donde se encuentra el valor o -1 si no lo encuentra.
 *
*/
funcion cerca_binaria_desc (
        /* Parámetros de entrada */ var t: taula[] d'enters, dim:enter, valor:enter)
retorna entero es

*****************************************************************************

/** Hay suspendidos?.
 *
 * @retorna (enter) Devuelve 1 si hay algún suspendido,
 *         0 si no hay ninguno y -20 si la secuencia está vacía.
*/
```

```
funcion hay_suspendidos () retorna entero es
********************************************************************************


/** Número total de aprobados.
 *
 * @retorna (enter) Devuelve el número total de aprobados en la secuencia.
*/
funcion total_aprobados () retorna entero es

********************************************************************************


/**Sexta prueba, más secuencias y subsecuencias.
 *              "¿Hay dos pares seguidos?"

@return (booleá) Devuelve cierto hay dos pares sequidos y falso en caso contrario.

funcion bug_hunters_06 (void) retorna bolea es
********************************************************************************
```

Semana 9:

```
/* Calcula el factorial el número que recibe por parámetro.
 *
 * @param n (Valor: enter) Número de entrada (n >= 0).
 * @return (enter) Devuelve el factorial del número de entrada (n).
 */
funcion factorial (
        /* Parámetro de entrada*/   n:enter) retorna entero es

********************************************************************************


/* Calcula el las permutaciones con repeticiones.
 *
 * @param m (Valor: enter) Número de elementos.
 * @param a (Valor: enter) Repetición del primer elemento (a > 0).
 * @param b (Valor: enter) Repetición del segundo elemento (b > 0).
 * @param c (Valor: enter) Repetición del tercer elemento (c > 0).
 * @return (enter) Devuelve las permutaciones con repetición.
 */
funcio permutaciones_repeticion (
                                                        /* Parámetros de entrada*/
m:enter, a:enter, b:enter, c:enter) retorna enter es
```

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

/* Calcula las variaciones de m elementos cogidos de n en n (sin repeticiones).
 *
 * @param m (Valor: enter) Número de elementos.
 * @param n (Valor: enter) Cuantos elementos cogemos.
 * @return (enter) Devuelve las variaciones sin repetición.
 */
funcion variaciones ( m:enter, n:enter) retorna entero es

*****************************************************************************


/** Lee un número entre [min..max]
 *
 * @param min (Valor: enter) Valor mínimo del número.
 * @param max (Valor: enter) Valor máximo que puede tener el número.
 * @return (enter) Devuelve un valor entero acotado [min..max].
*/
funcion lee_entero_acotado (
   /* Parámetros de entrada */ min:enter, max:enter) retorna entero es


*****************************************************************************


/* Muestra un menú y devuelve la opción elegida.
 * La opción siempre será válida.
 *
 * @return (enter) Devuelve la opción elegida.
 */
funcion menu() retorna entero es

*****************************************************************************

/** Séptima prueba, empezamos con procedimientos!!
 *              "¿Cuál es el valor mayor?"

 * @param valor1 (Valor: enter) Primer valor.
 * @param valor2 (Valor: enter) Segundo valor.
 * @return (enter) Devuelve el máximo de los dos valores.
 */
funcion mayor_2 ( valor1:enter, valor2: enter) retorna entero es
```

****

Semana 10:

```
/** Determina los incrementos de fila y columna para recorrer una matriz
*  siguiendo la orientación indicada.
*
* @param orientacion (Valor: enter) Identificación de la orientación.
* @param incf (Ref: enter) Incremento aplicable a las filas.
* @param incc (Ref: enter) Incremento aplicable a las columnas.
*/
aicció tb_calcula_incrementos ( ori:enter, var incf: enter, var incc:enter)
```

****

```
/** Comprueba si el carácter pertenece a la cadena.
*
* @param caracter (Valor: caràcter) Carácter que queremos comprobar.
* @param cadena (Ref: taula[] de caràcter) Cadena estándard que contiene
*         los carácteres dónde vamos a buscar.
* @return (booleà) Retorna cierto si el caràcter se encuentra en la cadena
*                  y falso en caso contrario.
*/
funció pertenece_cadena( caracter:carácter, var cadena: taula[] de carácter) retorna
booleà és
```

****

```
/** Lee un carácter de teclado. El carácter ha de pertenecer a una cadena.
 *
 * @param cadena (Ref: taula[] de caràcter) Cadena que contine los caracteres
válidos.
 * @param texto (Ref: taula[] de caràcter) Texto informativo que se muestra al usuario.
 * @return (caràcter) Devuelve un carácter que se encuentra entre los de la cadena.
 */
funció  lee_caracter_cadena (var cadena:taula[] de carácter: ,var texto:taula[] de
carácter: ) restorna caracter és
```

****

```
/** Inicializa una matriz con un carácter.
*
* @param matriz (Ref: taula[][] de caràcter) Matriz a inicializar.
* @param nfilas (Valor: enter) Número de filas de la matriz (nfilas > 0).
```

```
* @param mcols (Valor: enter) Número de columans de la matriz (mcols > 0).
* @param caracter (Valor: caràcter) Caracter con el que se inicializa.
*/
aicció tb_inicializa_matriz (
        /* Parámetros de salida */    var matriz:taula[][MCOLS_MAX] de carácter ,
        /* Parámetros de entrada */ nfilas:enter , mcols:enter , caracter: carácter )

*****************************************************************************

/** Busca en una matriz un determinado carácter e indica la fila y la columna donde se
encuentra.
*
* @param matriz (Ref: taula[][] de caràcter) Matriz donde buscamos.
* @param nfilas (Valor: enter) Número de filas de la matriz (nfilas > 0).
* @param mcols (Valor: enter) Número de columans de la matriz (mcols > 0).
* @param f (Ref: enter) Fila donde se ha encontrado el caracter buscado.
* @param c (Ref: enter) Columna donde se ha encontrado el caracter buscado.
* @param caracter (Valor: caràcter) Caracter que se está buscando.
* @return (booleà) Retorna cierto si lo ha encontrado y falso en caso contrario.
*/
funció tb_busca_matriz (
        /* Parámetros de entrada */ var matriz:taula[][MCOLS_MAX] de carácter,
        /* Parámetros de entrada */ nfilas:enter , mcols:enter ,
        /* Parámetros de salida */  var f:enter , var c:enter ,
        /* Parámetros de entrada */ caracter:carácter) retorna boolea és
```

Semana 11:

```
/** Algoritmo para generar un número entero aleatorio acotado.
*
* @param rango (Valor: entero) Número de valores aleatorios.
* @return (entero) Devuelve un número entero aleatorio: número >= 0 y número <
rango.
*/
funció entero_aleatorio_acotado (
   /* Parámetros de entrada */  min:enter , max:enter ) retorna enter és

*****************************************************************************

/**
 Determina un carácter aleatorio acotado [inf..sup]. Los valores inferiores y superiores
se determinan según el código ascii correspondiente. Pueden no estar ordenados.

 @param inf (Valor: carácter) Carácter inferior que limita los carácteres válidos.
```

```
 @param sup (Valor: carácter) Carácter superior que limita los carácteres válidos.
 @return (carácter) Devuelve un aleatorio perteneciente a [inf..sup].
 */
funció caracter_aleatorio_acotado (
   /* Parámetros de entrada */  carácter inf, carácter sup) retorna carácter és
*****************************************************************************

/** Cambiar de un jugador al siguiente

 @param jugador (Valor: entero) Identificador del jugador con el turno.
 @param num_jugadores (Valor: entero) Número total de jugadores.
 @return (entero) Devuelve el identificador del jugador siguiente.
*/
funció siguiente_jugador(
   /* Parámetros de entrada */ jugador: enter , num_jugadores:enter ) retorna enter és

*****************************************************************************

/** enterercamba el color: si el color era negro, nos devuelve blanco y si era blanco,
 * nos devuelve negro. Si la entrada es incorrecta, nos devuelve la ficha vacía.

 @param color_ficha (Valor: caràcter) Color actual de la ficha.
 @return (caràcter) Devuelve el color contrario al recibido como parámetro.
 */
funció ot_cambia_color(
   /* Parámetro de entrada */ color_ficha:carácter ) retorna caracter és

*****************************************************************************
/**
 Determinan si una fila y columna dada están dentro de los límites de una matriz
[nfilas][mcols]

 @param f (Valor: entero) Número de la fila.
 @param c (Valor: entero) Número de la columna.
 @param nfilas (Valor: entero) Número máximo de filas de la matriz.
 @param mcols (Valor: entero) Número máximo de columnas de la matriz.
 @return (booleà) Devuelve cierto si la fila pertenece a [0..nfilas) y
 columna pertenece a [0..mcols) y falso en caso contrario.
 */
funció tb_dentro_limites (
   /* Parámetros de entrada */ f:enter , c:enter , nfilas:enter , mcols:enter) retorna
booleà és

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

```
/** Busca un carácter en una matriz de nfilas x mcols, a partir de una determinada
posicion (fila,col). La busqueda se realiza en una determinada orientacion.
 No se evalua la casilla inicial (fila, col)
 El procedimiento devuelve si existe o no. En caso de que existe, la casilla donde se
encuentra el elemento se devuelve a partir de los mismos parametros de entrada fila y
col.

 @param matriz (Ref: tabla[][MCOLS_MAX] de carácter) Matriz de caracteres donde
buscamos.
 @param nfilas (Valor: entero) Número real de filas de la matriz (<= NFILAS_MAX).
 @param mcols (Valor: entero) Número real de columnas de la matriz (<=
MCOLS_MAX).
 @param fila (Ref: entero) Fila inicial y, fila en la que se encuentra.
 @param col (Ref: entero) Columna inicial y, columna en la que se encuentra.
 @param caracter (Valor: carácter) Carácter que se desea buscar.
 @param orientacion (Valor: enter) Orientación en la que se va a buscar.
 @return (booleà) Devuelve cierto si se ha encontrado el carácter y
                          falso en caso contrario.
 */
funció  tb_busca_orientacion(
   /* Parámetros de entrada */ var matriz[][MCOLS_MAX]: taula de carácter, nfilas:enter
, mcols:enter ,
   /* Parámetros de entrada/salida */     var fila: enter ,var col:enter ,
   /* Parámetros de entrada */    caracter:carácter , orientacion: enter) retorna
booleano és


**********

/** Busca un carácter en una matriz de nfilas x mcols, a partir de una determinada
posicion (fila,col). La busqueda se realiza en una determinada orientacion.
 No se evalua la casilla inicial (fila, col)
 El procedimiento devuelve si existe o no. En caso de que existe, la casilla donde se
encuentra el elemento se devuelve a partir de los mismos parametros de entrada fila y
col.

 @param matriz (Ref: tabla[][MCOLS_MAX] de carácter) Matriz de caracteres donde
buscamos.
 @param nfilas (Valor: entero) Número real de filas de la matriz (<= NFILAS_MAX).
 @param mcols (Valor: entero) Número real de columnas de la matriz (<=
MCOLS_MAX).

 @param fila (Ref: entero) Fila inicial y, fila en la que se encuentra.
 @param col (Ref: entero) Columna inicial y, columna en la que se encuentra.
 @param caracter (Valor: carácter) Carácter que se desea buscar.
 @param orientacion (Valor: enter) Orientación en la que se va a buscar.
```

```
 @return (booleà) Devuelve cierto si se ha encontrado el carácter y
                          falso en caso contrario.
 */
funció tb_busca_orientacion(
    /* Parámetros de entrada */ var matriz:taula[][MCOLS_MAX] de carácter, nfilas:enter
, mcols:enter ,
    /* Parámetros de entrada/salida */     var fila:enter var col:enter,
    /* Parámetros de entrada */    caracter: carácter, orientacion:enter) retorna boolea és

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

Semana 12:

```
 Compara dos fechas, f1 y f2, y devuelve cierto si son iguales.

 @param f1 (Ref: fecha_t) Primera fecha a evaluar.
 @param f2 (Ref: fecha_t) Segunda fecha a evaluar.
 @return (booleà) Devuelve cierto si las fechas son iguales y
 falso en caso contrario.
 */
funció fechas_iguales (var fecha_t f1, var fecha_t f2) retorna booleà és
 *****************************************************************************
Compara dos fechas, f1 y f2, y devuelve un valor entero según la relación existente
entre ellas:
1 - f1 > f2  La primera fecha es más reciente que la segunda.
0 - f1 = f2  Ambas fechas son iguales.
-1 - f1 < f2  La segunda fecha es más antigua que la segunda.

@param f1 (Ref: fecha_t) Primera fecha a evaluar.
@param f2 (Ref: fecha_t) Segunda fecha a evaluar.
@return (enter) Devuelve un valor según la relación existente entre las fechas:
- 1 si f1 es más reciente que f2.
- 0 si ambas son iguales.
- -1 si f1 es más antigua que f2.
*/
funció compara_fechas (
    /* Parámetros de entrada */var  f1: fecha_t , var f2: fecha_t ) retorna enter és

 *****************************************************************************
 Muestra por pantalla la fecha formateada: dd-mmm-aaaa

 @param fecha (Ref: fecha_t) Fecha a escribir por pantalla.
 */
acció escribe_fecha (var fecha_t fecha);
```

```
/**

 ****************************************************************************
 Imprime por pantalla la información de un récord siguiendo el siguiente formato:
 dd-mmm-aa  Color(Blanco/Negro)   Puntuación  Nombre
 Los campos deben separarse con un tabulador y salto de línea al final.

 @param record (Ref: record_t) Registro con toda la información del récord.
 */
acció ot_escribe_record(var record:record_t )
****************************************************************************
Imprime un número determinado de los récords guardados en la tabla.

@param records (Ref: tabla[] de record_t) Tabla con los records, ordenados o no.
@param num_records (Valor: entero) Número máximo de registros a imprimir.
*/
acció ot_escribe_records (
        /* Parámetros de entrada */ var records: taula[] de record_t ,
num_records:enter ) és
****************************************************************************
```

Semana 13:

```
/*
Calcula y guarda el área de las esferas. El radio se enterroduce por teclado.

@return (enter) Devuelve 1 si todo ha ido bien y 0 si se ha producido un error.
*/
funció guarda_areas () retorna enter és
****************************************************************************
Calcula y guarda el área de las esferas. Los radios están en el fichero
radios.txt

@return (enter) Devuelve el número de áreas calculadas o un código
                de error si se ha producido algún problema.
*/
funció  guarda_areas () retorna enter és
****************************************************************************
Lee las palabras almacenadas en un fichero y
muestra las que superan un determinado tamaño.

@param nom_fichero (Ref: taula[] de caràcter) Nombre del fichero dónde están las
palabras.
@param mida (Valor: enter) Medida a partir de la cual se mostrarán las palabras.
*/
```

acció lee_palabras (
/* Parámetros de entrada */ var nom_fichero: taula[] de carácter, mida: enter)

****