UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Enunciado de la tercera práctica de programación I

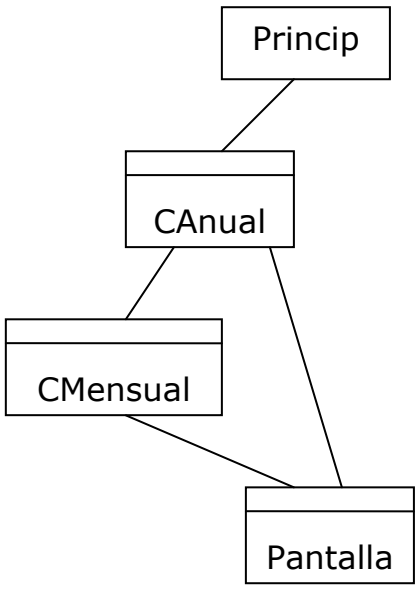
Departamento de Lenguajes y Sistemas Informáticos

Curso 2002-03

Se trata de realizar un programa en Modula-2 que muestre por pantalla una hoja de calendario de cualquier año comprendido entre 1601 y 3000. El formato de la hoja de calendario deberá ajustarse al que se indica a continuación para el ejemplo del año 2003.

```
¿Año (1601.3000)?2003

ENERO                2003    FEBRERO              2003    MARZO                2003
======================       ======================       ======================
LU MA MI JU VI | SA DO       LU MA MI JU VI | SA DO       LU MA MI JU VI | SA DO
======================       ======================       ======================
 .  .  1  2  3 |  4  5        .  .  .  .  .  |  1  2        .  .  .  .  .  |  1  2
 6  7  8  9 10 | 11 12        3  4  5  6  7  |  8  9        3  4  5  6  7  |  8  9
13 14 15 16 17 | 18 19       10 11 12 13 14  | 15 16       10 11 12 13 14  | 15 16
20 21 22 23 24 | 25 26       17 18 19 20 21  | 22 23       17 18 19 20 21  | 22 23
27 28 29 30 31 |  .  .       24 25 26 27 28  |  .  .       24 25 26 27 28  | 29 30
                                                         31  .  .  .  .  |  .  .

ABRIL                2003    MAYO                 2003    JUNIO                2003
======================       ======================       ======================
LU MA MI JU VI | SA DO       LU MA MI JU VI | SA DO       LU MA MI JU VI | SA DO
======================       ======================       ======================
 .  1  2  3  4 |  5  6        .  .  .  1  2  |  3  4        .  .  .  .  .  |  .  1
 7  8  9 10 11 | 12 13        5  6  7  8  9  | 10 11        2  3  4  5  6  |  7  8
14 15 16 17 18 | 19 20       12 13 14 15 16  | 17 18        9 10 11 12 13  | 14 15
21 22 23 24 25 | 26 27       19 20 21 22 23  | 24 25       16 17 18 19 20  | 21 22
28 29 30  .  . |  .  .       26 27 28 29 30  | 31  .       23 24 25 26 27  | 28 29
                                                         30  .  .  .  .  |  .  .

JULIO                2003    AGOSTO               2003    SEPTIEMBRE           2003
======================       ======================       ======================
LU MA MI JU VI | SA DO       LU MA MI JU VI | SA DO       LU MA MI JU VI | SA DO
======================       ======================       ======================
 .  1  2  3  4 |  5  6        .  .  .  .  1  |  2  3        1  2  3  4  5  |  6  7
 7  8  9 10 11 | 12 13        4  5  6  7  8  |  9 10        8  9 10 11 12  | 13 14
14 15 16 17 18 | 19 20       11 12 13 14 15  | 16 17       15 16 17 18 19  | 20 21
21 22 23 24 25 | 26 27       18 19 20 21 22  | 23 24       22 23 24 25 26  | 27 28
28 29 30 31  . |  .  .       25 26 27 28 29  | 30 31       29 30  .  .  .  |  .  .

OCTUBRE              2003    NOVIEMBRE            2003    DICIEMBRE            2003
======================       ======================       ======================
LU MA MI JU VI | SA DO       LU MA MI JU VI | SA DO       LU MA MI JU VI | SA DO
======================       ======================       ======================
 .  .  1  2  3 |  4  5        .  .  .  .  .  |  1  2        1  2  3  4  5  |  6  7
 6  7  8  9 10 | 11 12        3  4  5  6  7  |  8  9        8  9 10 11 12  | 13 14
13 14 15 16 17 | 18 19       10 11 12 13 14  | 15 16       15 16 17 18 19  | 20 21
20 21 22 23 24 | 25 26       17 18 19 20 21  | 22 23       22 23 24 25 26  | 27 28
27 28 29 30 31 |  .  .       24 25 26 27 28  | 29 30       29 30 31  .  .  |  .  .
```

Se aconseja organizar la solución según el siguiente **diagrama de estructura**:



A continuación, se presenta el **módulo principal** y los **módulos de definición** sugeridos:

Princip.mod

```
MODULE Princip;

(*********************************************************************
*
*   Fichero: Módulo principal de la tercera práctica de PROGRAMACIÓN 1
*
*   Autores: Equipo Docente
*
*   Descripción:
*     Este módulo constituye el programa principal que, utilizando al
*     módulo CAnual (ver diagrama de estructura), resuelve el enunciado
*     de la tercera práctica de PROGRAMACIÓN 1
*
*********************************************************************)


FROM InOut IMPORT WriteString, WriteLn, ReadCard;
IMPORT CAnual;

VAR
   cal: CAnual.TipoCalendario;
   anno: CARDINAL;

BEGIN
   WriteString("¿Año (1601..3000)?");
   ReadCard(anno);
   WriteLn;
   CAnual.Crear(cal, anno);
   CAnual.Imprimir(cal);
END Princip.
```

CAnual.def

```
DEFINITION MODULE CAnual;

(*********************************************************************
*
*   Fichero: Módulo de definición CAnual
*
*   Autores: Equipo Docente
*
*   Descripción:
*     Este módulo es un TAD (Tipo Abstracto de Datos) capaz de construir
*     e imprimir la hoja de calendario de cualquier año comprendido entre
*     1601 y 3000, según el formato enunciado en la tercera práctica de
*     PROGRAMACIÓN 1
*
*   Utiliza los módulos:
*     - CMensual
*     - Pantalla
*
*********************************************************************)

IMPORT CMensual, Pantalla;

CONST
    AnnoInicial = 1601; (* Hoja de calendario desde 1601 *)
    AnnoFinal = 3000;   (* Hoja de calendario hasta 3000 *)

TYPE
    RangoAnnos = [AnnoInicial..AnnoFinal]; (* Rango de años de 1601 a 3000 *)

    (*-- TAD para hoja de calendario anual: --*)

    TipoCalendario = Pantalla.TipoPantalla;

    (* Una hoja de calendario anual se corresponde con una pantalla *)


PROCEDURE Crear (VAR calendario: TipoCalendario; anno: RangoAnnos);

   (*============================================================
      Procedimiento que crea una hoja de calendario de cualquier
      año comprendido entre 1601 y 3000, según el formato
      enunciado en la tercera práctica de PROGRAMACIÓN 1
    ============================================================*)

PROCEDURE Imprimir (VAR calendario: TipoCalendario);

   (*============================================================
      Procedimiento que imprime un hoja de calendario anual

      OBSERVACION: el valor de <<calendario>> no cambia; se pasa
      por referencia por eficiencia. Es más, si no se hiciera así y
      la compilación se realiza con el FST se produciría
      desbordamiento de la pila (stack overflow) en tiempo de
      ejecución.
    ============================================================*)

END CAnual.
```

CMensual.def

```
DEFINITION MODULE CMensual;

(*********************************************************************
*
*   Fichero: Módulo de definición CMensual
*
*   Autores: Equipo Docente
*
*   Descripción:
*     Este módulo es un TAD (Tipo Abstracto de Datos) capaz de construir
*     e imprimir una hoja de calendario de cualquier mes y año comprendido
*     entre 1601 y 3000, según el formato enunciado en la segunda práctica
*     de PROGRAMACIÓN 1
*
*   Utiliza el módulo:
*     - Pantalla
*
*********************************************************************)

IMPORT Pantalla;

CONST
    AnnoInicial = 1601; (* Hoja de calendario desde 1601 *)
    AnnoFinal = 3000;   (* Hoja de calendario hasta 3000 *)

TYPE
    RangoMeses = [1..12];                    (* Rango de meses del 1 al 12 *)
    RangoAnnos = [AnnoInicial..AnnoFinal];  (* Rango de años de 1601 a 3000 *)

    (*-- TAD para hoja de calendario mensual: --*)
    TipoCalendario = Pantalla.TipoPantalla;
    (* Una hoja de calendario mensual se corresponde con una pantalla *)

PROCEDURE Crear (VAR calendario: TipoCalendario;
                     mes: RangoMeses;
                     anno: RangoAnnos);

   (*============================================================
      Procedimiento que crea una hoja de calendario de cualquier
      mes y año comprendido entre 1601 y 3000, según el formato
      enunciado en la segunda práctica de PROGRAMACIÓN 1
    ============================================================*)


PROCEDURE Imprimir (VAR calendario: TipoCalendario);

   (*============================================================
      Procedimiento que imprime un hoja de calendario mensual

      OBSERVACION: el valor de <<calendario>> no cambia; se pasa
      por referencia por eficiencia. Es más, si no se hiciera así y
      la compilación se realiza con el FST se produciría
      desbordamiento de la pila (stack overflow) en tiempo de
      ejecución.
    ============================================================*)


END CMensual.
```

Pantalla.def

```
DEFINITION MODULE Pantalla;

(*********************************************************************
*
*   Fichero: Módulo de definición Pantalla
*
*   Autores: Equipo Docente
*
*   Descripción:
*     Este módulo es un TAD (Tipo Abstracto de Datos) capaz de almacenar
*     y operar con una pantalla de caracteres
*
*********************************************************************)

CONST
    MaxLineas = 80;     (* La pantalla dispondrá como máximo de 80 líneas *)
    MaxColumnas = 80;   (* La pantalla dispondrá como máximo de 80 columnas *)

TYPE
    RangoLineas = [1..MaxLineas];
    RangoColumnas = [1..MaxColumnas];

    (*-- TAD para pantalla: --*)

    TipoPantalla = RECORD
                       numLineas: RangoLineas;
                               (* numero de líneas de la pantalla *)
                       numColumnas: RangoColumnas;
                               (* numero de columnas de la pantalla *)
                       linea: RangoLineas;
                               (* línea en la que se encuentra el
                                * cursor en un momento dado *)
                       columna: RangoColumnas;
                               (* columna en la que se encuentra el
                                  cursor en un momento dado *)
                       pantalla: ARRAY RangoLineas, RangoColumnas OF CHAR;
                               (* matriz de caracteres que almacena la
                  * pantalla *)
                     END; (* RECORD *)

    (*-- Fin del TAD para pantalla --*)

PROCEDURE CrearVacia(VAR pantalla: TipoPantalla;
                         numLineas: RangoLineas;
                         numColumnas: RangoColumnas);

   (*=========================================================
      Procedimiento que crea una "pantalla vacía". Es decir,
      inicializa los valores del parámetro pantalla del
      siguiente modo:
         - numLineas -> numLineas especificadas por el usuario
                    del procedimiento
         - numColumnas -> numColumnas especificadas por el usuario
                      del procedimiento
         - linea -> 1
         - columnas -> 1 (inicialmente, el cursor se encuentra
                       en la posición (1, 1))
         - pantalla -> se rellena de espacios en blanco
    ==========================================================*)


PROCEDURE CambiarCoordenadas(VAR pantalla: TipoPantalla;
                                 linea:RangoLineas;
                                 columna: RangoColumnas);
```

```
   (*=========================================================
      Procedimiento que modifica la posición del cursor de
      <pantalla> según los valores de <linea> y <columna>
    =========================================================*)


PROCEDURE EscribirRistra(VAR pantalla: TipoPantalla;
                             ristra: ARRAY OF CHAR);

   (*=========================================================
      Procedimiento que escribe una cadena de caracteres en
      <pantalla> en la posición donde se encuentre el cursor,
      actualizándolo, es decir, reajustando los valores de
      <linea> y <columna>
    =========================================================*)


PROCEDURE EscribirNumero(VAR pantalla: TipoPantalla;
                             numero: CARDINAL;
                              numEspacios: CARDINAL);

   (*=========================================================
      Procedimiento que escribe una número entero positivo
      en <pantalla> en la posición donde se encuentre
      el cursor, actualizándolo debidamente, es decir, reajustando
      los valores de <linea> y <columna>.

      El parámetro <numEspacios> equivale al segundo parámetro
      de WriteInt y WriteCard. Por ejemplo:

      EscribirNumero(pantalla, 123, 5);

      escribiría en <pantalla> dos espacios en blanco
      y 123: ' '' ''1''2''3'
    =========================================================*)


PROCEDURE EscribirPantalla(VAR destino: TipoPantalla;
                           VAR origen: TipoPantalla);

   (*=========================================================
      Procedimiento que escribe una pantalla origen dentro de
      una pantalla destino en la posición donde se encuentre
      el cursor

      OBSERVACION: el valor de <<origen>> no cambia; se pasa
      por referencia por eficiencia. Es más, si no se hiciera así y
      la compilación se realiza con el FST se produciría
      desbordamiento de la pila (stack overflow) en tiempo de
      ejecución.
    =========================================================*)


PROCEDURE Imprimir(VAR pantalla: TipoPantalla);

   (*=========================================================
      Procedimiento que imprime el contenido de la pantalla

      OBSERVACION: el valor de <<pantalla>> no cambia; se pasa
      por referencia por eficiencia. Es más, si no se hiciera así y
      la compilación se realiza con el FST se produciría
      desbordamiento de la pila (stack overflow) en tiempo de
      ejecución.
    =========================================================*)


END Pantalla.
```