Unidades en PASCAL (Units)

Aquellos que quieran probar los ejercicios de TAD en Pc pueden utilizar las UNITS de Pascal.
En este documento se explican los conceptos básicos para la creación de Units en Pascal.
Este tema esta relacionado a la construcción de TAD´s (Tipos Abstractos de Datos)
La forma que tiene Pascal de trabajar con tipos de datos abstractos es a través de la generación de Units.

Definición

Las unidades son grupos de funciones o procedimientos compilados que pueden ser llamados y utilizados desde cualquier programa sin necesidad de escribirlos de nuevo.

Estructura de las Unidades

Las unidades no son ejecutables por si solas, dependen del programa que las llama para poder realizar su función.

Las unidades están compuestas por cuatro partes:
1) DECLARACIÓN
2) INTERFAZ
3) IMPLEMENTACION
4) INICIALIZACION

1) DECLARACION

Es obligatoria, provee el nombre de la unidad que se creará.
Su sintaxis es:
UNIT nombre de la unidad;

2) INTERFAZ

En esta sección se encuentran las declaraciones de todos los elementos que pueden ser utilizados por el programa que llama a la unidad, por ejemplo otras unidades, variables, constantes, type, procedimientos y funciones. En esta sección no se incluyen los códigos de los procedimientos ni funciones, únicamente su declaración.
Observar que el TIPO (type exportado) se define en la interfaz a diferencia de la sintaxis (en pseudocódigo) de los TADS visto en la teoría y la práctica.

Su sintaxis es:

INTERFACE
{ declaración de funciones, procedimientos, otras unidades, variables constantes y el type }

3) IMPLEMENTACION

La sección de implementación es exclusiva para la unidad que se está creando, contiene declaraciones de etiquetas, variables, constantes, así como el contenido de los procedimientos y funciones declarados en la sección de interfaz.
Su sintaxis es:

IMPLEMENTATION
{implementaciones y declaraciones privadas}

4) INICIALIZACION

Esta sección se utiliza para inicializar los valores de las variables, no es indispensable y por lo mismo no siempre se incluye. Su sintaxis es parecida a la del cuerpo principal de un programa, se inicia con BEGIN y termina con un END. Observar que esto no es obligatorio y se puede obviar.

BEGIN
{inicialización de variables }
END

Uniendo todo lo visto hasta ahora

UNIT nombre de la unidad;

INTERFACE
declaración de funciones, procedimientos, otras unidades, variables, constantes y type.

IMPLEMENTATION
implementaciones y declaraciones privadas

BEGIN
inicialización de variables
END.

Compilación y uso de las Unidades

Para que una unidad este lista para su utilización en programas posteriores, es necesario que se realice la compilación de la misma, en forma separada al programa.

Una vez compilada la unidad se podrá incorporar al programa agregando la sentencia **USES.** Esta palabra reservada debe ir inmediatamente después de la sentencia **PROGRAM** del programa en el que desea utilizar la unidad.

```
PROGRAM nombre del programa
USES nombre de la unidad
        Begin
                {sentencias del programa}
        end.
```

Ejemplo: Unit para construir una "Cola de Enteros"

```
Unit Cola_enteros;

Interface

{declaración de tipos y variables }
Type
   Pun_Nodo_Cola = ^Tipo_Nodo_Cola;

   Tipo_Nodo_Cola = record
                       Elem: Integer;
                       Siguiente:Pun_Nodo_Cola;
                    end;

   Tipo_Cola =record
                 Primero,Ultimo:Pun_Nodo_Cola;
                 CantElementos: integer;
              end;
```

{declaración de procesos y procedimiento}

```
Procedure q_create(var C:Tipo_Cola);
        {se crea una cola vacia}

Function q_empty(C:Tipo_Cola):boolean;
        {consulta si la cola esta vacia}

Procedure q_push(var C:Tipo_Cola; E:Integer);
        {Agregar un elemento al final de la cola }

Procedure q_pop(var C:Tipo_Cola; var E:Integer);
        {Sacar un elemento del comienzo de la cola, la cola debe tener elementos}

Procedure  q_top(C:Tipo_Cola; var E:Integer);
        {Devuelve el elemento del comienzo de la cola sin sacarlo; la cola debe
tener elementos }

Procedure  q_bottom(C:Tipo_Cola; var E:Integer;
        {Devuelve el elemento del final de la cola sin sacarlo; la cola debe tener
elementos }

Function q_length(C:Tipo_Cola) : integer;
         {Devuelve la cantidad de elementos que hay en la cola}

Procedure asignar(c1:Tipo_Cola; var c2:Tipo_Cola);
         {recibe C1 y devuelve en C2 los mimos elementos que tiene c1 }
```

Implementation

{Implementación de funciones y procedimientos}

```
Procedure q_create(var C:Tipo_Cola);
begin
    C.Primero:=nil;
    C.Ultimo:=nil;
    C.CantElementos:=0;
end;


Function q_empty(C:Tipo_Cola):boolean;
begin
    q_empty:=(C.Primero=nil);
end;
```

```
Procedure q_push(var C:Tipo_Cola; E:Integer);
var
  pAux:Pun_Nodo_Cola;
begin
    new(pAux);
    pAux^.Elem:=E;
    pAux^.Siguiente:=nil;
    if (C.Ultimo=nil) then
      C.Primero:=pAux
    else
       C.Ultimo^.Siguiente:=pAux;
    C.Ultimo:=pAux;
    C.CantElementos:= C.CantElementos +1;
end;



Procedure q_pop(var C:Tipo_Cola; var E:Integer);
var
  pAux:Pun_Nodo_Cola;
begin
    pAux:=C.Primero;
    C.Primero:=pAux^.siguiente;
    E:=pAux^.Elem;
    if (C.Primero=nil) then C.Ultimo:=nil;
    dispose(pAux);
    C.CantElementos:= C.CantElementos -1;

end;


Procedure  q_top(C:Tipo_Cola; var E:Integer);
  begin
       E:=C.Primero^.Elem;
  end;


Procedure  q_bottom(C:Tipo_Cola; var E:Integer);
  begin
       E:=C.Ultimo^.Elem;
  end;

Function q_length(C:Tipo_Cola) : integer;
```

```
Begin
  q_length:=C.CantElementos;
End;

Procedure asignar(c1:Tipo_Cola; var c2:Tipo_Cola);
Begin
c2.Primero:= c1.Primero;
c2.Ultimo:= c1.Ultimo;
c2.CantElementos:= c1.CantElementos;
End;
End.
```

Ejemplo del uso de la cola de enteros

Enunciado: Leer números enteros de teclado y generar una cola de enteros. Una vez generada la cola mostrar cuántos elementos se cargaron y cuantos elementos iguales a 5 se ingresaron.

{Observar en la solución que en ningún momento se trabaja con la estructura interna de la cola, esa definición es desconocida para el programa usuario de la UNIT (SERIA INCORRECTO ACCEDER DESDE AFUERA DE LA UNIT A LA ESTRUCTURA QUE LA MISMA TIENE). Solo puedo utilizar el tipo exportado y las funciones y procedimientos definidos en la interfaz.
También observar el grado de *modularización* logrado}

```
PROGRAM ejemploCola;
Uses Cola_enteros;

Var
colaE: Tipo_Cola;

Procedure CargarC(var colaE: Tipo_Cola);
Var
num: integer;
Begin
  Writeln (´Ingrese un número entero ó 0 (cero) para terminar: ');
  Readln(num);
  While (num <> 0) do begin
     q_push(colaE,num);
     Writeln (´Ingrese un número entero ó 0 (cero) para terminar: ');
     Readln(num);
  End;
End;
```

```
Function Cuantos5 (colaE: Tipo_Cola): integer;
Var
  nume: integer;
Begin
tot:=0;
While not q_empty(colaE) do begin
  q_pop (colaE,nume);
  if nume = 5 then tot:=tot+1;
 end;
 Cuantos5:= tot;
End;

Begin {del programa principal}
  Q_create(colaE);
  CargarC(colaE);
  Writeln('La cantidad de elementos ingresados es de:'; q_length(colaE));
  Writeln (' La cantidad de numeros iguales a 5 de la cola es de:'
              Cuantos5(colaE));
End.
```