```
{Módulo que implementa el TAD "cola genérica".
 Versión: 1.0
 Fecha: 2015-10-07
 Autor: Javier Campos}

módulo genérico colasGenéricas

parámetro
  tipo elemento

exporta

  tipo cola {Los valores del TAD cola representan secuencias de elementos con
             acceso FIFO (first in, first out), esto es, el primer elemento añadido
             será el primero en ser borrado.}

  procedimiento crearVacía(sal c:cola)
  {Devuelve en c la cola vacía, sin elementos}

  procedimiento encolar(e/s c:cola; ent e:elemento)
  {Devuelve en c la cola resultante de añadir e a c}

  función esVacía(c:cola) devuelve booleano
  {Devuelve verdad si y sólo si c no tiene elementos}

  procedimiento primero(ent c:cola; sal e:elemento; sal error:booleano)
  {Si c es no vacía, devuelve en e el primer elemento añadido a c y error=falso.
   Si c es vacía, devuelve error=verdad y e queda indefinido}

  procedimiento desencolar(e/s c:cola)
  {Si c es no vacía, devuelve en c la cola resultante de eliminar de c el primer
   elemento que fue añadido. Si c es vacía, la deja igual}

  función longitud(c:cola) devuelve natural
  {Devuelve el número de elementos de c}

  procedimiento duplicar(sal colaSal:cola; ent colaEnt:cola)
  {Devuelve en colaSal una cola igual a colaEnt, duplicando la representación
   en memoria}

  función iguales(cola1,cola2:cola) devuelve booleano
  {Devuelve verdad si y sólo si cola1 y cola2 tienen los mismos elementos y en
   las mismas posiciones}

  procedimiento liberar(e/s c:cola)
  {Devuelve en c la cola vacía y además libera la memoria utilizada previamente por c}

  {Las tres operaciones siguientes conforman un iterador interno para la cola}

  procedimiento iniciarIterador(e/s c:cola)
  {Prepara el iterador para que el siguiente elemento a visitar sea un primer
   elemento de c, si existe (situación de no haber visitado ningún elemento)}

  función existeSiguiente(c:cola) devuelve booleano
  {Devuelve falso si ya se han visitado todos los elementos de c; devuelve cierto en
   caso contrario}

  procedimiento siguiente(e/s c:cola; sal e:elemento; sal error:booleano)
  {Si existe algún elemento de c pendiente de visitar, devuelve en e el siguiente
   elemento a visitar y error=falso, y además avanza el iterador para que a
   continuación se pueda visitar otro elemento de c. Si no quedan elementos pendientes
   de visitar devuelve error=verdad y e queda indefinido}
```

implementación

```
  tipos ptDato = ↑unDato;
        unDato = registro
                   dato:elemento;
                   sig:ptDato
                 freg;
        cola = registro
                  pri,ult:ptDato;
                  long:natural;
                  iter:ptDato {se utiliza para implementar el iterador}
                freg

  procedimiento crearVacía(sal c:cola)
  principio
    c.pri:=nil;
    c.ult:=nil;
    c.long:=0
  fin

  procedimiento encolar(e/s c:cola; ent e:elemento)
  principio
    si c.long=0 entonces
      nuevoDato(c.ult);
      c.pri:=c.ult
    sino
      nuevoDato(c.ult↑.sig);
      c.ult:=c.ult↑.sig
    fsi;
    c.ult↑.dato:=e;
    c.ult↑.sig:=nil;
    c.long:=c.long+1
  fin

  función esVacía(c:cola) devuelve booleano
  principio
    devuelve c.pri=nil
  fin

  procedimiento primero(ent c:cola; sal e:elemento; sal error:booleano)
  principio
    si esVacía(c) entonces
      error:=verdad
    sino
      error:=falso;
      e:=c.pri↑.dato
    fsi
  fin

  procedimiento desencolar(e/s c:cola)
  variable aux:ptDato
  principio
    si not esVacía(c) entonces
      aux:=c.pri;
      c.pri:=c.pri↑.sig;
      disponer(aux);
      c.long:=c.long-1;
      si c.long=0 entonces c.ult:=nil fsi
    fsi
  fin

  función longitud(c:cola) devuelve natural
  principio
    devuelve c.long
  fin
```

```
procedimiento duplicar(sal colaSal:cola; ent colaEnt:cola)
variables ptSal,ptEnt:ptDato
principio
  si esVacía(colaEnt) entonces
    crearVacía(colaSal);
  sino
    ptEnt:=colaEnt.pri;
    nuevoDato(colaSal.pri);
    colaSal.pri↑.dato:=ptEnt↑.dato;
    ptSal:=colaSal.pri;
    ptEnt:=ptEnt↑.sig;
    mientrasQue ptEnt≠nil hacer
      nuevoDato(ptSal↑.sig);
      ptSal:=ptSal↑.sig;
      ptSal↑.dato:=ptEnt↑.dato;
      ptEnt:=ptEnt↑.sig
    fmq;
    ptSal↑.sig:=nil;
    colaSal.ult:=ptSal;
    colaSal.long:=colEnt.long
  fsi
fin

función iguales(cola1,cola2:cola) devuelve booleano
variables pt1,pt2:ptDato; iguales:booleano:=verdad
principio
  si cola1.long≠cola2.long entonces
    devuelve falso;
  sino
    pt1:=cola1.pri;
    pt2:=cola2.pri;
    mientrasQue pt1≠nil and iguales hacer
      iguales:=pt1↑.dato=pt2↑.dato;
      pt1:=pt1↑.sig;
      pt2:=pt2↑.sig
    fmq;
    devuelve iguales
  fsi
fin

procedimiento liberar(e/s c:cola)
variable aux:ptDato
principio
  aux:=c.pri;
  mientrasQue aux≠nil hacer
    c.pri:=c.pri↑.sig;
    disponer(aux);
    aux:=c.pri
  fmq;
  c.ult:=nil;
  c.long:=0
fin

procedimiento iniciarIterador(e/s c:cola)
principio
  c.iter:=c.pri
fin

función existeSiguiente(c:cola) devuelve booleano
principio
  devuelve c.iter≠nil
fin
```

```
  procedimiento siguiente(e/s c:cola; sal e:elemento; sal error:booleano)
  principio
    si existeSiguiente(c) entonces
      error:=falso;
      e:=c.iter↑.dato;
      c.iter:=c.iter↑.sig
    sino
      error:=verdad
    fsi
  fin

fin
```