```
módulo genérico conjuntosGenéricos
usa booleanos, naturales
parámetros
  tipo elemento  {Nombre de un tipo cualquiera}
  con función iguales(e1,e2:elemento) devuelve booleano

exporta

  tipo conjunto {Conjunto de datos de tipo elemento}

  procedimiento vacío(sal C:conjunto)
  {Devuelve en C el conjunto vacío}

  función esVacío(C:conjunto) devuelve booleano
  {Devuelve verdad si y sólo si C está vacío}

  procedimiento poner(ent e:elemento; e/s C:conjunto)
  {Si e no está en C, devuelve el conjunto resultante de añadir e a C;
   en caso contrario, C queda como estaba}

  procedimiento quitar(ent e:elemento; e/s C: conjunto)
  {Si e está en C, devuelve el conjunto resultante de eliminar e de C;
   en caso contrario, C queda como estaba}

  función pertenece(e:elemento; C:conjunto) devuelve booleano
  {Devuelve verdad si y sólo si e está en C}

  procedimiento unión(ent A,B:conjunto; sal C:conjunto)
  {Devuelve en C la unión de A y B}

  procedimiento intersección(ent A,B:conjunto; sal C: conjunto)
  {Devuelve en C la intersección de A y B}

  función cardinal(C:conjunto) devuelve natural
  {Devuelve el cardinal del conjunto C}

  {Las tres siguientes operaciones implementan un Iterador}

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

  función existeSiguiente(C:conjunto) devuelve booleano
  {Devuelve falso si ya se han visitado todos los elementos de C.
   Devuelve verdad en caso contrario.}

  procedimiento siguiente(e/s C:conjunto; sal e:elemento; sal error:booleano)
  {Implementa las operaciones "siguiente" y "avanza" de la especificación, es decir:
   Si existeSiguiente(C), error toma el valor falso, e toma el valor del siguiente
   elemento del conjunto, y se avanza el iterador al elemento siguiente del conjunto.
   Si no existeSiguiente(C), error toma el valor verdad, e queda indefinido y C
   queda como estaba.}

implementación

  ...

fin
```

¿Cómo en un programa principal que utilice este módulo podríamos recorrer todos los elementos de un conjunto C para hacer algo con ellos? (por ejemplo, escribirlos en pantalla)