TDA VECTOR DISPERSO

JotaEle Díaz y Gregorio Vidoy

1.0

Dic 2014

Índice de clases

Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

- **vectorD< T > (Clase VectorD)**
- **vectorD< T >::const_iterator (Class const_iterator Iterador constante hacia delante sobre todas las posiciones del vector Disperso. Lectura const_iterator, const_iterator(const_iterator), const_iterator(iterator), operator*, operator++, ++operator, operator=(iterator), operator==, operator!=)**
- **vectorD< T >::const_stored_iterator (Class const_stored_iterator Iterador constante hacia delante que sólo recorre las posiciones que no son el valor por defecto del vector Disperso. Lectura. const_stored_iterator, const_stored_iterator(const_stored_iterator), const_stored_iterator(stored_iterator), operator*, operator++, ++operator, operator==, operator!=, operator=)**
- **vectorD< T >::iterator (Class iterator Iterador hacia delante sobre todas las posiciones del vector Disperso. iterator, operator*, operator++, ++operator operator=(iterator), operator==, operator!=)**
- **vectorD< T >::stored_iterator (Class stored_iterator Iterador hacia delante que sólo recorre las posiciones que no son el valor por defecto del vector Disperso. stored_iterator, stored_iterator(stored_iterator), operator*, operator++, ++operator, operator==, operator!=)**

Documentación de las clases

Referencia de la plantilla de la Clase vectorD< T >

Un vector disperso es un contenedor similar al vector de la STL en el que la mayoría de los elementos tienen el mismo valor (conocido como valor por defecto).

Gracias a la implementación interna del vector interno, el recorrido y acceso sobre este es eficiente y su almacenamiento ocupa una menor cantidad de memoria.

La gestión de memoria es dinámica y el tamaño del vector es modificable.

```
#include <vectorD.h>
```

Clases

class **const_iterator**

- *class **const_iterator** Iterador constante hacia delante sobre todas las posiciones del vector Disperso. Lectura **const_iterator**, const_iterator(const_iterator), const_iterator(iterator), operator*, operator++, ++operator, operator=(iterator), operator==, operator!=* class **const_stored_iterator**
- *class **const_stored_iterator** Iterador constante hacia delante que sólo recorre las posiciones que no son el valor por defecto del vector Disperso. Lectura. **const_stored_iterator**, const_stored_iterator(const_stored_iterator), const_stored_iterator(stored_iterator), operator*, operator++, ++operator, operator==, operator!=, operator=*
- *class iterator Iterador hacia delante sobre todas las posiciones del vector Disperso. iterator, operator*, operator++, ++operator operator=(iterator), operator==, operator!=*

- *class **stored_iterator** Iterador hacia delante que sólo recorre las posiciones que no son el valor por defecto del vector Disperso. **stored_iterator**, stored_iterator(stored_iterator), operator*, operator++, ++operator, operator==, operator!=*

Tipos públicos

typedef unsigned int **size_type**

Métodos públicos

- **vectorD** (const T &t=T())
  Constructor por defecto.
- **vectorD** (const **vectorD**< T > &b)
  Constructor primitivo que hace una copia de un vector disperso.
- **vectorD** (int numcomp, const T &t=T())
  *Constructor primitivo que crea un **vectorD** con numcomp componentes, todas ellas inicializadas al valor por defecto t.*
- **~vectorD** ()
  *Destructor primitivo de **vectorD**.*
- size_type **size** () const
  *Calcula el tamaño del **vectorD**.*
- T **default_value** () const
  Muestra el valor que tiene asignado el VectorD como valor por defecto del vector.
- bool **empty** ()
  Comprueba si el VectorD está vacío.
- void **assign** (int p, const T &t)
  Cambia a el valor t el valor de la posicion p-ésima del vector.
- void **push_back** (const T &t)
  *Inserta un elemento al final del **vectorD**.*
- void **pop_back** ()
  *Elimina el último elemento del **vectorD**.*

- void **clear** ()
  *Elimina todos los elementos del **vectorD**.*
- void **resize** (int s)
  *Redimensiona el **vectorD** al número de posiciones dado.*
- **vectorD** & **operator=** (const **vectorD** &x)
  *Operador de asignación. Asigna una copia del **vectorD** al recibido como argumento.*
- const T & **operator[]** (int c) const
  Devuelve la componente c-ésima del VectorD. (Ej: vector[5])
- const T & **at** (int c) const
  Devuelve la componente c-ésima del VectorD.
- bool **operator==** (const **vectorD** &x)
  Operador de igualdad. Comprueba si dos vectores son iguales. Es decir, si su tamaño y todos sus elementos uno a uno coinciden.
- bool **operator!=** (const **vectorD** &x)
  Operador de desigualdad. Comprueba si dos vectores no son iguales. Es decir, si su tamaño y todos sus elementos uno a uno no coinciden.
- **iterator begin** ()
  *Devuelve un iterador que apunta al primer elemento del **vectorD**.*
- **iterator end** ()
  *Devuelve un iterador que apunta al final (es decir, al elemento siguiente al último) del **vectorD**.*
- **const_iterator cbegin** () const
  *Devuelve un iterador constante que apunta al primer elemento del **vectorD**.*
- **const_iterator cend** () const
  *Devuelve un iterador constante que apunta al final (es decir, al elemento siguiente al último) del **vectorD**.*
- **stored_iterator sbegin** ()
  Devuelve un iterador sobre elementos no nulos que apunta al primer elemento que no sea el valor por defecto.
- **stored_iterator send** ()
  Devuelve un iterador sobre elementos no nulos que apunta al final (el elemento siguiente al último) de los elementos no nulos.
- **const_stored_iterator csbegin** () const
  Devuelve un iterador constante sobre elementos no nulos que apunta al primer elemento que no sea el valor por defecto.
- **const_stored_iterator csend** () const
  Devuelve un iterador constante sobre elementos no nulos que apunta al final (el elemento siguiente al último) de los elementos no nulos.

Documentación del constructor y destructor

- **template<typename T> vectorD< T >::vectorD (const T & *t* = `T ()`)**

  Constructor por defecto.

  Parámetros:

in	*t*	Valor que se desea que se interprete como valor por defecto del vector. (Por defecto, será 0)

- **template<typename T> vectorD< T >::vectorD (const vectorD< T > & *b*)**

  Constructor primitivo que hace una copia de un vector disperso.

  Parámetros:

in	*b*	Vector disperso que se desea copiar

- **template<typename T> vectorD< T >::vectorD (int *numcomp*, const T & *t* = `T ()`)**

  Constructor primitivo que crea un **vectorD** con numcomp componentes, todas ellas inicializadas al valor por defecto t.

  Parámetros:

in	*numcomp*	Número de componentes que tendrá el vector.
in	*t*	Valor que se desea que se interprete como valor por defecto del vector. (Por defecto, 0)

Documentación de las funciones miembro

- **template<typename T> void vectorD< T >::assign (int *p*, const T & *t*)**

  Cambia a el valor t el valor de la posicion p-ésima del vector.

  Parámetros:

in	*p*	Posición en el vector que se desea modificar.
in	*t*	Nuevo valor que se desea dar a la posición del vector.

  Postcondición:
  No modifica el tamaño del vector

- **template<typename T > const T & vectorD< T >::at (int *c*) const**

  Devuelve la componente c-ésima del VectorD.

  Parámetros:

in	*c*	Número de la posición del **vectorD** a la que se desea acceder.

  Devuelve:
  Valor c-ésimo del vector

  Precondición:
  c debe ser mayor o igual que el número de componente del vector. Se hace comprobación de rango. Genera un error en caso de no cumplir la condición.

- **template<typename T > vectorD< T >::iterator vectorD< T >::begin ()**

  Devuelve un iterador que apunta al primer elemento del **vectorD**.

  Devuelve:
  Iterador a la primera posición.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > vectorD< T >::const_iterator vectorD< T >::cbegin () const**

  Devuelve un iterador constante que apunta al primer elemento del **vectorD**.

  Devuelve:
  Iterador constante a la primera posición.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > vectorD< T >::const_iterator vectorD< T >::cend () const**

  Devuelve un iterador constante que apunta al final (es decir, al elemento siguiente al último) del **vectorD**.

  Devuelve:
  Iterador constante al elemento siguiente al último.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > void vectorD< T >::clear ()**

  Elimina todos los elementos del **vectorD**.

  Postcondición:
  El tamaño del vector queda a 0

- **template<typename T > vectorD< T >::const_stored_iterator vectorD< T >::csbegin () const**

  Devuelve un iterador constante sobre elementos no nulos que apunta al primer elemento que no sea el valor por defecto.

  Devuelve:
  Iterador constante sobre elementos no nulos al primer elemento no nulo.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > vectorD< T >::const_stored_iterator vectorD< T >::csend () const**

  Devuelve un iterador constante sobre elementos no nulos que apunta al final (el elemento siguiente al último) de los elementos no nulos.

  Devuelve:
  Iterador constante sobre elementos no nulos al último elemento no nulo.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > T vectorD< T >::default_value () const**

  Muestra el valor que tiene asignado el VectorD como valor por defecto del cector.

  Devuelve:
  Valor por defecto

- **template<typename T > bool vectorD< T >::empty ()**

  Comprueba si el VectorD está vacío.

  Devuelve:
  True si está vacío, false si no lo está.

- **template<typename T > vectorD< T >::iterator vectorD< T >::end ()**

  Devuelve un iterador que apunta al final (es decir, al elemento siguiente al último) del **vectorD**.

  Devuelve:
  Iterador al elemento siguiente al último.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > bool vectorD< T >::operator!= (const vectorD< T > & *x*)**

  Operador de desigualdad. Comprueba si dos vectores no son iguales. Es decir, si su tamaño y todos sus elementos uno a uno no coinciden.

  Parámetros:

in	*x*	VectorD a comparar.

  Devuelve:
  Devuelve true si ambos vectores no son iguales, false en caso de que sí lo sean.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > vectorD< T > & vectorD< T >::operator= (const vectorD< T > & *x*)**

  Operador de asignación. Asigna una copia del **vectorD** al recibido como argumento.

  Parámetros:

in	*x*	VectorD a copiar.

  Devuelve:
  Vector copiado

  Postcondición:
  Los valores y tamaño del vector quedan iguales que los del **vectorD** x.

- **template<typename T > bool vectorD< T >::operator== (const vectorD< T > & *x*)**

  Operador de igualdad. Comprueba si dos vectores son iguales. Es decir, si su tamaño y todos sus elementos uno a uno coinciden.

  Parámetros:

in	*x*	VectorD a comparar.

  Devuelve:
  Devuelve true si ambos vectores son iguales, false en caso contrario.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > const T & vectorD< T >::operator[] (int *c*) const**

  Devuelve la componente c-ésima del VectorD. (Ej: vector[5])

  Parámetros:

in	*c*	Número de la posición del **vectorD** a la que se desea acceder.

  Devuelve:
  Valor c-ésimo del vector

- **template<typename T > void vectorD< T >::pop_back ()**

  Elimina el último elemento del **vectorD**.

  Postcondición:
  El tamaño del vector disminuye en 1

- **template<typename T> void vectorD< T >::push_back (const T & *t*)**

  Inserta un elemento al final del **vectorD**.

  Parámetros:

in	*t*	Nuevo valor que se desea añadir al **vectorD**.

  Postcondición:
  El tamaño del vector aumenta en 1

- **template<typename T > void vectorD< T >::resize (int *s*)**

  Redimensiona el **vectorD** al número de posiciones dado.

  Parámetros:

in	*s*	Nuevo tamaño del **vectorD**

  Postcondición:
  Si s es menor que el tamaño actual, se eliminan los restantes. Si es mayor, se amplia el tamaño poniendo al valor por defecto todos los elementos nuevos.

- **template<typename T > vectorD< T >::stored_iterator vectorD< T >::sbegin ()**

  Devuelve un iterador sobre elementos no nulos que apunta al primer elemento que no sea el valor por defecto.

  Devuelve:
  Iterador sobre elementos no nulos al primer elemento no nulo.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > vectorD< T >::stored_iterator vectorD< T >::send ()**

  Devuelve un iterador sobre elementos no nulos que apunta al final (el elemento siguiente al último) de los elementos no nulos.

  Devuelve:
  Iterador sobre elementos no nulos al último elemento no nulo.

  Postcondición:
  No se modifica el **vectorD**.

- **template<typename T > vectorD< T >::size_type vectorD< T >::size () const**

  Calcula el tamaño del **vectorD**.

  Devuelve:
  Tamaño del **vectorD**

Referencia de la Clase vectorD< T >::const_iterator

class **const_iterator** Iterador constante hacia delante sobre todas las posiciones del vector Disperso. Lectura **const_iterator**, const_iterator(const_iterator), const_iterator(iterator), operator*, operator++, ++operator, operator=(iterator), operator==, operator!=

```
#include <vectorD.h>
```

Métodos públicos

- **const_iterator** ()
  Constructor primitivo del iterador sobre el VectorD.
- **const_iterator** (const **const_iterator** &d)
  Constructor primitivo que crea un iterador constante sobre un VectorD copia de otro iterador constante.
- **const_iterator** (const **iterator** &d)
  Constructor primitivo que crea un iterador constante en base a un iterador no constante.
- const T & **operator*** ()
  Operador de indirección. Devuelve el elemento del VectorD al cual hace referencia el iterador constante.
- **const_iterator** & **operator++** ()
  Operador de incremento (++it). Incrementa el iterador constante, es decir referencia el iterador a la posición siguiente a la actual.
- **const_iterator operator++** (int)
  Operador de incremento (it++). Incrementa el iterador constante, es decir referencia el iterador a la posición siguiente a la actual.
- bool **operator==** (const **const_iterator** &d)
  Operador de igualdad. Comprueba si dos iteradores hacen referencia a una misma posición del VectorD.
- bool **operator!=** (const **const_iterator** &d)
  Operador de desigualdad. Comprueba si dos iteradores hacen referencia a distintas posiciones del VectorD.

Amigas

class **vectorD**

Descripción detallada

- **template<typename T>class vectorD< T >::const_iterator**

class **const_iterator** Iterador constante hacia delante sobre todas las posiciones del vector Disperso. Lectura **const_iterator**, const_iterator(const_iterator), const_iterator(iterator), operator*, operator++, ++operator, operator=(iterator), operator==, operator!=

Documentación del constructor y destructor

- **template<typename T > vectorD< T >::const_iterator::const_iterator (const const_iterator & *d*)**

  Constructor primitivo que crea un iterador constante sobre un VectorD copia de otro iterador constante.

  Parámetros:

in	*d*	Iterador al vector disperso que se desea copiar

- **template<typename T > vectorD< T >::const_iterator::const_iterator (const iterator & *d*)**

  Constructor primitivo que crea un iterador constante en base a un iterador no constante.

  Parámetros:

in	*d*	Iterador al vector disperso que se desea copiar

Documentación de las funciones miembro

- **template<typename T > bool vectorD< T >::const_iterator::operator!= (const const_iterator & *d*)**

  Operador de desigualdad. Comprueba si dos iteradores hacen referencia a distintas posiciones del VectorD.

  Parámetros:

in	*d*	Iterador a comparar.

  Devuelve:
  True si hacen referencia distintas posiciones, false en caso contrario

- **template<typename T > const T & vectorD< T >::const_iterator::operator* ()**

  Operador de indirección. Devuelve el elemento del VectorD al cual hace referencia el iterador constante.

  Devuelve:
  Elemento al cual hace referencia el iterador constante.

- **template<typename T > vectorD< T >::const_iterator & vectorD< T >::const_iterator::operator++ ()**

  Operador de incremento (++it). Incrementa el iterador constante, es decir referencia el iterador a la posición siguiente a la actual.

  Devuelve:
  Iterador constante a la posición siguiente a la actual.

- **template<typename T > vectorD< T >::const_iterator vectorD< T >::const_iterator::operator++ (int)**

  Operador de incremento (it++). Incrementa el iterador constante, es decir referencia el iterador a la posición siguiente a la actual.

  Devuelve:
  Iterador constante original antes de ser incrementado.

- **template<typename T > bool vectorD< T >::const_iterator::operator== (const const_iterator & *d*)**

  Operador de igualdad. Comprueba si dos iteradores hacen referencia a una misma posición del VectorD.

  Parámetros:

in	*d*	Iterador a comparar.

  Devuelve:
  True si hacen referencia a la misma posición, false en caso contrario

La documentación para esta clase fue generada a partir de los siguientes ficheros:
vectorD.h
vectorD.hxx

Referencia de la Clase vectorD< T >::const_stored_iterator

class **const_stored_iterator** Iterador constante hacia delante que sólo recorre las posiciones que no son el valor por defecto del vector Disperso. Lectura. **const_stored_iterator**, const_stored_iterator(const_stored_iterator), const_stored_iterator(stored_iterator), operator*, operator++, ++operator, operator==, operator!=, operator=

`#include <vectorD.h>`

Métodos públicos

- **const_stored_iterator** ()
  Constructor primitivo del iterador constante a posiciones no nulas del VectorD.
- **const_stored_iterator** (const **const_stored_iterator** &d)
  Constructor primitivo que crea un iterador constante sobre elementos no nulos copia de otro.
- **const_stored_iterator** (const **stored_iterator** &d)
  Constructor primitivo que crea un iterador constante sobre posiciones no nulas en base a uno no constante.
- const pair< int, T > & **operator*** ()
  Operador de indirección. Devuelve el elemento del VectorD al cual hace referencia el iterador constante.
- **const_stored_iterator** & **operator++** ()
  Operador ++. Devuelve un iterador constante a la posición no nula siguiente a la actual.
- **const_stored_iterator operator++** (int)
  Operador ++. Devuelve un iterador constante a la posición no nula siguiente a la actual.
- **const_stored_iterator** & **operator--** ()
  Operador de decremento (–it). Decrementa el iterador, es decir referencia el iterador a la posición no nula anterior a la actual.
- **const_stored_iterator operator--** (int)
  Operador de decremento (it–). Decrementa el iterador constante, es decir referencia el iterador a la posición no nula anterior a la actual.
- bool **operator==** (const **const_stored_iterator** &d)
  Operador de igualdad. Comprueba si dos iteradores constantes sobre posiciones no nulas hacen referencia a una misma posición del VectorD.
- bool **operator!=** (const **const_stored_iterator** &d)
  Operador de desigualdad. Comprueba si dos iteradores constantes sobre posiciones no nulas hacen referencia a distintas posiciones del VectorD.
- **const_stored_iterator** & **operator=** (const **const_stored_iterator** &d)
  Operador de asignación. Asigna al iterador constante sobre elementos no nulos a posición a la que apunta el recibido.

Amigas

class **vectorD**

Descripción detallada

- **template<typename T>class vectorD< T >::const_stored_iterator**

class **const_stored_iterator** Iterador constante hacia delante que sólo recorre las posiciones que no son el valor por defecto del vector Disperso. Lectura. **const_stored_iterator**, const_stored_iterator(const_stored_iterator), const_stored_iterator(stored_iterator), operator*, operator++, ++operator, operator==, operator!=, operator=

Documentación del constructor y destructor

- **template<typename T > vectorD< T >::const_stored_iterator::const_stored_iterator (const const_stored_iterator & *d*)**

  Constructor primitivo que crea un iterador constante sobre elementos no nulos copia de otro.

  Parámetros:

in	*d*	Iterador constante sobre elementos no nulos del vector disperso que se desea copiar

- **template<typename T > vectorD< T >::const_stored_iterator::const_stored_iterator (const stored_iterator & *d*)**

  Constructor primitivo que crea un iterador constante sobre posiciones no nulas en base a uno no constante.

  Parámetros:

in	*d*	Iterador sobre posiciones no nulas que se desea copiar.

Documentación de las funciones miembro

- **template<typename T > bool vectorD< T >::const_stored_iterator::operator!= (const const_stored_iterator & *d*)**

  Operador de desigualdad. Comprueba si dos iteradores constantes sobre posiciones no nulas hacen referencia a distintas posiciones del VectorD.

  Parámetros:

in	*d*	Iterador constante sobre posiciones no nulas a comparar.

  Devuelve:
  True si hacen referencia a la diferentes posiciones, false en caso contrario

- **template<typename T > const pair< int, T > & vectorD< T >::const_stored_iterator::operator* ()**

  Operador de indirección. Devuelve el elemento del VectorD al cual hace referencia el iterador constante.

  Devuelve:
  pair <int, T> que contiene la posición y el elemento del VectorD al cual hace referencia el iterador constante.

- **template<typename T > vectorD< T >::const_stored_iterator & vectorD< T >::const_stored_iterator::operator++ ()**

  Operador ++. Devuelve un iterador constante a la posición no nula siguiente a la actual.

  Devuelve:
  Iterador consante a la posición no nula siguiente a la actual.

- **template<typename T > vectorD< T >::const_stored_iterator vectorD< T >::const_stored_iterator::operator++ (int)**

  Operador ++. Devuelve un iterador constante a la posición no nula siguiente a la actual.

  Devuelve:
  Iterador constante a la posición no nula siguiente a la actual.

- **template<typename T > vectorD< T >::const_stored_iterator & vectorD< T >::const_stored_iterator::operator-- ()**

  Operador de decremento (–it). Decrementa el iterador, es decir referencia el iterador a la posición no nula anterior a la actual.

  Devuelve:
  Iterador constante a la posición no nula anterior a la actual.

- **template<typename T > vectorD< T >::const_stored_iterator vectorD< T >::const_stored_iterator::operator-- (int)**

  Operador de decremento (it–). Decrementa el iterador constante, es decir referencia el iterador a la posición no nula anterior a la actual.

  Devuelve:
  Iterador constante original antes de ser decrementado.

- **template<typename T > vectorD< T >::const_stored_iterator & vectorD< T >::const_stored_iterator::operator= (const const_stored_iterator & *d*)**

  Operador de asignación. Asigna al iterador constante sobre elementos no nulos a posición a la que apunta el recibido.

  Parámetros:

in	*d*	Iterador constante sobre posiciones no nulas a copiar.

- **template<typename T > bool vectorD< T >::const_stored_iterator::operator== (const const_stored_iterator & *d*)**

  Operador de igualdad. Comprueba si dos iteradores constantes sobre posiciones no nulas hacen referencia a una misma posición del VectorD.

  Parámetros:

in	*d*	Iterador constante sobre posiciones no nulas a comparar.

  Devuelve:
  True si hacen referencia a la misma posición, false en caso contrario

La documentación para esta clase fue generada a partir de los siguientes ficheros:

vectorD.h
vectorD.hxx

Referencia de la Clase vectorD< T >::iterator

class iterator Iterador hacia delante sobre todas las posiciones del vector Disperso. iterator, operator*, operator++, ++operator operator=(iterator), operator==, operator!=

`#include <vectorD.h>`

Métodos públicos

- **iterator** ()
  Constructor primitivo del iterador sobre el VectorD.
- **iterator** (const **iterator** &d)
  Constructor primitivo que crea un iterador sobre un VectorD copia de otro iterador.
- const T & **operator*** ()
  Operador de indirección. Devuelve el elemento del VectorD al cual hace referencia el iterador.
- **iterator** & **operator++** ()
  Operador de incremento (++it). Incrementa el iterador, es decir referencia el iterador a la posición siguiente a la actual.
- **iterator operator++** (int)
  Operador de incremento (it++). Incrementa el iterador, es decir referencia el iterador a la posición siguiente a la actual.
- bool **operator==** (const **iterator** &d)
  Operador de igualdad. Comprueba si dos iteradores hacen referencia a una misma posición del VectorD.
- bool **operator!=** (const **iterator** &d)
  Operador de desigualdad. Comprueba si dos iteradores hacen referencia a distintas posiciones del VectorD.
- **iterator** & **operator=** (const **iterator** &d)
  Operador de asignación. Asigna al iterador la posición a la que apunta el iterador recibido.

Amigas

class **vectorD**

Descripción detallada

- **template<typename T>class vectorD< T >::iterator**

class iterator Iterador hacia delante sobre todas las posiciones del vector Disperso. iterator, operator*, operator++, ++operator operator=(iterator), operator==, operator!=

Documentación del constructor y destructor

- **template<typename T > vectorD< T >::iterator::iterator (const iterator & *d*)**

  Constructor primitivo que crea un iterador sobre un VectorD copia de otro iterador.

  Parámetros:

in	*d*	Iterador al vector disperso que se desea copiar

Documentación de las funciones miembro

- **template<typename T > bool vectorD< T >::iterator::operator!= (const iterator & *d*)**

  Operador de desigualdad. Comprueba si dos iteradores hacen referencia a distintas posiciones del VectorD.

  Parámetros:

in	*d*	Iterador a comparar.

  Devuelve:

  True si hacen referencia distintas posiciones, false en caso contrario

- **template<typename T > const T & vectorD< T >::iterator::operator* ()**

  Operador de indirección. Devuelve el elemento del VectorD al cual hace referencia el iterador.

  Devuelve:

  Elemento al cual hace referencia el iterador.

- **template<typename T > vectorD< T >::iterator & vectorD< T >::iterator::operator++ ()**

  Operador de incremento (++it). Incrementa el iterador, es decir referencia el iterador a la posición siguiente a la actual.

  Devuelve:

  Iterador a la posición siguiente a la actual.

- **template<typename T > vectorD< T >::iterator vectorD< T >::iterator::operator++ (int)**

  Operador de incremento (it++). Incrementa el iterador, es decir referencia el iterador a la posición siguiente a la actual.

  Devuelve:

  Iterador original antes de ser incrementado.

- **template<typename T > vectorD< T >::iterator & vectorD< T >::iterator::operator= (const iterator & *d*)**

  Operador de asignación. Asigna al iterador la posición a la que apunta el iterador recibido.

  Parámetros:

in	*d*	Iterador a copiar.

- **template<typename T > bool vectorD< T >::iterator::operator== (const iterator & *d*)**

  Operador de igualdad. Comprueba si dos iteradores hacen referencia a una misma posición del VectorD.

  Parámetros:

in	*d*	Iterador a comparar.

  Devuelve:

  True si hacen referencia a la misma posición, false en caso contrario

Referencia de la Clase vectorD< T >::stored_iterator

class **stored_iterator** Iterador hacia delante que sólo recorre las posiciones que no son el valor por defecto del vector Disperso. **stored_iterator**, stored_iterator(stored_iterator), operator*, operator++, ++operator, operator==, operator!=

`#include <vectorD.h>`

Métodos públicos

- **stored_iterator** ()
  Constructor primitivo del iterador a posiciones no nulas del VectorD.
- **stored_iterator** (const **stored_iterator** &d)
  Constructor primitivo que crea un iterador sobre elementos no nulos copia de otro.
- const pair< int, T > & **operator*** ()
  Operador de indirección. Devuelve el elemento del VectorD al cual hace referencia el iterador.
- **stored_iterator** & **operator++** ()
  Operador de incremento (++it). Incrementa el iterador, es decir referencia el iterador a la posición no nula siguiente a la actual.
- **stored_iterator operator++** (int)
  Operador de incremento (it++). Incrementa el iterador, es decir referencia el iterador a la posición no nula siguiente a la actual.
- **stored_iterator** & **operator--** ()
  Operador de decremento (–it). Decrementa el iterador, es decir referencia el iterador a la posición no nula anterior a la actual.
- **stored_iterator operator--** (int)
  Operador de decremento (it–). Decrementa el iterador, es decir referencia el iterador a la posición no nula anterior a la actual.
- bool **operator==** (const **stored_iterator** &d)
  Operador de igualdad. Comprueba si dos iteradores sobre posiciones no nulas hacen referencia a una misma posición del VectorD.
- bool **operator!=** (const **stored_iterator** &d)
  Operador de desigualdad. Comprueba si dos iteradores sobre posiciones no nulas hacen referencia a distintas posiciones del VectorD.

Amigas

- class **vectorD**
- class **vectorD< T >::const_stored_iterator**

Descripción detallada

- **template<typename T>class vectorD< T >::stored_iterator**

class **stored_iterator** Iterador hacia delante que sólo recorre las posiciones que no son el valor por defecto del vector Disperso. **stored_iterator**, stored_iterator(stored_iterator), operator*, operator++, ++operator, operator==, operator!=

Documentación del constructor y destructor

- **template<typename T > vectorD< T >::stored_iterator::stored_iterator (const stored_iterator & *d*)**

  Constructor primitivo que crea un iterador sobre elementos no nulos copia de otro.

  Parámetros:

in	*d*	Iterador sobre elementos no nulos del vector disperso que se desea copiar

Documentación de las funciones miembro

- **template<typename T> bool vectorD< T >::stored_iterator::operator!= (const stored_iterator & *d*)**

  Operador de desigualdad. Comprueba si dos iteradores sobre posiciones no nulas hacen referencia a distintas posiciones del VectorD.

  Parámetros:

in	*d*	Iterador sobre posicionesn no nulas a comparar.

  Devuelve:
  True si hacen referencia a la diferentes posiciones, false en caso contrario

- **template<typename T > const pair< int, T > & vectorD< T >::stored_iterator::operator* ()**

  Operador de indirección. Devuelve el elemento del VectorD al cual hace referencia el iterador.

  Devuelve:
  pair <int, T> que contiene la posición y el elemento del VectorD al cual hace referencia el iterador.

- **template<typename T > vectorD< T >::stored_iterator & vectorD< T >::stored_iterator::operator++ ()**

  Operador de incremento (++it). Incrementa el iterador, es decir referencia el iterador a la posición no nula siguiente a la actual.

  Devuelve:
  Iterador a la posición no nula siguiente a la actual.

- **template<typename T > vectorD< T >::stored_iterator vectorD< T >::stored_iterator::operator++ (int)**

  Operador de incremento (it++). Incrementa el iterador, es decir referencia el iterador a la posición no nula siguiente a la actual.

  Devuelve:
  Iterador original antes de ser incrementado.

- **template<typename T > vectorD< T >::stored_iterator & vectorD< T >::stored_iterator::operator-- ()**

  Operador de decremento (–it). Decrementa el iterador, es decir referencia el iterador a la posición no nula anterior a la actual.

  Devuelve:
  Iterador a la posición no nula anterior a la actual.

- **template<typename T > vectorD< T >::stored_iterator vectorD< T >::stored_iterator::operator-- (int)**

  Operador de decremento (it–). Decrementa el iterador, es decir referencia el iterador a la posición no nula anterior a la actual.

  Devuelve:
  Iterador original antes de ser decrementado.

- **template<typename T> bool vectorD< T >::stored_iterator::operator== (const stored_iterator & *d*)**

  Operador de igualdad. Comprueba si dos iteradores sobre posiciones no nulas hacen referencia a una misma posición del VectorD.

  Parámetros:

in	*d*	Iterador sobre posiciones no nulas a comparar.

  Devuelve:
  True si hacen referencia a la misma posición, false en caso contrario

La documentación para esta clase fue generada a partir de los siguientes ficheros:
vectorD.h
vectorD.hxx