vectorD

Generado por Doxygen 1.8.13

Índice general

Capítulo 1

Rep del TDA vectorD

1.1. Invariante de la representación

El invariante es (vd,n_ele,v_nulo) : 0 <= vd.size() < n_ele; //Fallo tipo 1 vd[i].second != v_nulo, para todo i = 0, ..., vd.size()-1; // Fallo tipo 2 vd[i].first >=0, para todo i = 0, ..., vd.size()-1; // Fallo tipo 3 vd[i].first < vd[j].first si i<j // Fallo tipo 4

1.2. Función de abstracción

Un objeto válido *rep* del TDA FechaHistorica representa a

FA(rep): rep -- > vector (vd=[(a,v1), (b,v2), ..., (n,vn)] n_ele = M ------—> pos: 0 1 2 ... a-1 a ... x .... b.... ... n-1 n n+1 ..... M-1 val: t t t ....t v1 ... t .... v2 .. ... t vn t ..... t v_nulo = t

Capítulo 2

Índice de clases

2.1. Lista de clases

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

Capítulo 3

Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

Capítulo 4

Documentación de las clases

4.1. Referencia de la Clase vectorD< T >::iterator

Métodos públicos

- **iterator** (const iterator &d)
- const T & **operator*** ()
- iterator & **operator++** ()
- iterator **operator++** (int)
- iterator **operator+=** (const int inc)
- bool **operator==** (const iterator &d)
- bool **operator!=** (const iterator &d)
- iterator & **operator=** (const iterator &d)

Amigas

- class **vectorD< T >**

La documentación para esta clase fue generada a partir del siguiente fichero:

- include/vectorD.h

4.2. Referencia de la Clase vectorD< T >::stored_iterator

Métodos públicos

- **stored_iterator** (const stored_iterator &d)
- const pair< int, T > & **operator*** ()
- stored_iterator & **operator++** ()
- stored_iterator **operator++** (int)
- bool **operator==** (const stored_iterator &d)
- bool **operator!=** (const stored_iterator &d)

Amigas

- class **vectorD< T >**

La documentación para esta clase fue generada a partir del siguiente fichero:

- include/vectorD.h

4.3. Referencia de la plantilla de la Clase vectorD< T >

T.D.A. vectorD.

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

Clases

- class iterator
- class stored_iterator

Tipos públicos

- typedef unsigned int **size_type**

Métodos públicos

- vectorD (const T &t=T())
  Constructor por defecto de la clase.
- vectorD (const vectorD< T > &v)
  Constructor de copia de la clase.
- vectorD (int numcomp, const T &t=T())
  Constructor de la clase.
- ~vectorD ()
  desctructor
- size_type size () const
  size
- T default_value () const
  default_value
- bool empty ()
  Comprueba si esta vacio el vector disperso.
- list< pair< int, T > > **getList** ()
- void set (int p, const T &t)
  Inserta un elemento en una posicion del vector disperso.
- void push_back (const T &t)
  Inserta un elemento al final del vector disperso.
- void pop_back ()
  Elimina un elemento del vector disperso.
- void clear ()

Elimina los elemento del vector disperso.

- void resize (int s)

  Cambia el tamaño del vector.
- vectorD< T > & operator= (const vectorD< T > &x)

  Sobrecarga del operador =.
- const T & operator[ ] (int c)

  Sobrecarga del operador [].
- bool operator== (const vectorD< T > &x)

  Sobrecarga del operador ==.
- bool operator!= (const vectorD< T > &x)

  Sobrecarga del operador !=.
- iterator begin ()

  begin del vector disperso
- iterator end ()

  end del vector disperso
- stored_iterator sbegin ()

  sbegin del vector disperso(solo del storage)
- stored_iterator send ()

  send del vector disperso(solo del storage)

4.3.1. Descripción detallada

template<typename T>
class vectorD< T >

T.D.A. vectorD.

Una instancia *c* del tipo de datos abstracto `vectorD` es un objeto, el cual es un conjunto de elementos ordenados. El vector vendra dado por un valor nulo, quedando asi el contenido del vector "dividido" en dos, podiendo iterar sobre el vector de dos formas: Sobre su contenido no nulo, y sobre todo su contenido.

Lo representamos:

v[n_ele] = {elem, elem, elem... (n_ele)...}

Un ejemplo de su uso:

Autor

Pablo Pérez Méndez
Luis González

Fecha

Noviembre 2017

4.3.2. Documentación del constructor y destructor

4.3.2.1. vectorD() [1/2]

```
template<typename T>
vectorD< T >::vectorD (
            const vectorD< T > & v )
```

Constructor de copia de la clase.

Parámetros

v	vector del tipo vector disperso a construir

4.3.2.2. vectorD() [2/2]

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

Constructor de la clase.

Parámetros

numcomp	numero de elementos del vector disperso a construir
t	valor por defecto del vector disperso a constuir

Precondición

numcomp debe ser un natural positivo

4.3.3. Documentación de las funciones miembro

4.3.3.1. begin()

```
template<typename T>
iterator vectorD< T >::begin ( )
```

begin del vector disperso

Devuelve

Devuelve un iterador del vectorD al inicio de todos los elementos de este

4.3.3.2. default_value()

```
template<typename T>
T vectorD< T >::default_value ( ) const
```

default_value

Devuelve

Devuelve el valor por defecto del vector disperso

4.3.3.3. empty()

```
template<typename T>
bool vectorD< T >::empty ( )
```

Comprueba si esta vacio el vector disperso.

Devuelve

Devuelve si esta vacio

4.3.3.4. end()

```
template<typename T>
iterator vectorD< T >::end ( )
```

end del vector disperso

Devuelve

Devuelve un iterador del vectorD al final(siguiente al ultimo) de todos los elementos de este

4.3.3.5. operatori=()

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

Sobrecarga del operador !=.

Parámetros

x	un objeto de tipo vectorD que se copiara en el objeto implícito

4.3.3.6. operator=()

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

Sobrecarga del operador =.

Parámetros

x	un objeto de tipo vectorD que se copiara en el objeto implcito

4.3.3.7. operator==()

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

Sobrecarga del operador ==.

Parámetros

x	un objeto de tipo vectorD que se copiara en el objeto implícito

4.3.3.8. operator[]()

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

Sobrecarga del operador [].

Parámetros

c	posicion sobre la que indexar

Precondición

c c >= 0 && c < size

4.3.3.9. pop_back()

```
template<typename T>
void vectorD< T >::pop_back ( )
```

Elimina un elemento del vector disperso.

Parámetros

t	valor a insertar

4.3.3.10. push_back()

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

Inserta un elemento al final del vector disperso.

Parámetros

t	valor a insertar

4.3.3.11. resize()

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

Cambia el tamaño del vector.

Parámetros

s	valor a insertar

Precondición

s debe ser un natural positivo

4.3.3.12. sbegin()

```
template<typename T>
stored_iterator vectorD< T >::sbegin ( )
```

sbegin del vector disperso(solo del storage)

Devuelve

Devuelve un iterador(de los elementos almacenados) del vectorD al inicio(siguiente al ultimo) de todos los elementos de este

4.3.3.13. send()

```
template<typename T>
stored_iterator vectorD< T >::send ( )
```

send del vector disperso(solo del storage)

Devuelve

Devuelve un iterador(de los elementos almacenados) del vectorD al final(siguiente al ultimo) de todos los elementos de este

4.3.3.14. set()

```
template<typename T>
void vectorD< T >::set (
            int p,
            const T & t )
```

Inserta un elemento en una posicion del vector disperso.

Parámetros

p	posicion a insertar
t	valor a insertar

Precondición

p p >= 0

4.3.3.15. size()

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

size

Devuelve

Devuelve el tamaño del vector disperso

La documentación para esta clase fue generada a partir del siguiente fichero:

- include/vectorD.h

Capítulo 5

Documentación de archivos

5.1. Referencia del Archivo include/vectorD.h

Fichero cabecera del TDA vectorD.

```
#include <list>
#include <utility>
#include "vectorD.hxx"
```

Dependencia gráfica adjunta para vectorD.h: