PIlaMax

1.0

Generado por Doxygen 1.8.13

Índice general

Capítulo 1

Representativo del struct Elemento

1.1. Invariante de la representación

El invariante de la representación es: 0 <= utilizados <= reservados

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 plantilla de la Clase Cola< T >

T.D.A. Cola.

`#include <cola.h>`

Métodos públicos

- Cola ()

  Constructor por defecto.
- Cola (const Cola< T > &original)

  Constructor de copias.
- ~Cola ()

  Destructor.
- Cola & operator= (const Cola< T > &otra)

  Operador de asignación.
- bool esta_vacia () const

  Comprueba si la cola está vacía.
- T & frente ()

  Devuelve el elemento del frente de la cola.
- const T & frente () const

  Devuelve el elemento del frente de una cola constante.
- void poner (const T &elem)

  Añade un elemento al final de la cola.
- void quitar ()

  Quita el elemento del frente de la cola.
- int num_elementos () const

  Devuelve el número de elementos de la cola.

4.1.1. Descripción detallada

template<class T>
class Cola< T >

T.D.A. Cola.

Una instancia *c* del tipo de dato abstracto Cola sobre un dominio *T* es una sucesión finita de elementos del mismo con un funcionamiento *FIFO* (First In, First Out}). En una cola, las operaciones de inserción tienen lugar en uno de los extremos, denominado *final* de la cola, mientras que el borrado y consulta se lleva a cabo en el otro extremo, denominado *frente* de la cola. Una cola de longitud *n* la denotamos

- <a1,a2,a3,..,an<

En esta cola, tendremos acceso únicamente al elemento del *Frente*, es decir, a *a1*. El borrado o consulta de un elemento será sobre *a1*, mientras que la inserciñon de un nuevo elemento se haré después de *an* (final de la cola).

Si n=0 diremos que la cola está vacía.

El espacio requerido para el almacenamiento es O(n), donde n es el número de elementos de la cola.

Autor

J. Fdez-Valdivia

Fecha

Octubre 2011

4.1.2. Documentación del constructor y destructor

4.1.2.1. Cola()

```
template<class T>
Cola< T >::Cola (
            const Cola< T > & original )
```

Constructor de copias.

Parámetros

original	La cola de la que se hará la copia.

4.1.3. Documentación de las funciones miembro

4.1.3.1. operator=()

```
template<class T>
Cola< T > & Cola< T >::operator= (
            const Cola< T > & otra )
```

Operador de asignación.

Parámetros

otra	La cola que se va a asignar.

4.1.3.2. poner()

```
template<class T>
void Cola< T >::poner (
            const T & elem )
```

Añade un elemento al final de la cola.

Parámetros

elem	Elemento que se va a añadir.

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

- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/cola.h
- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/src/cola.cpp

4.2. Referencia de la Estructura Elemento

struct Elemento Una instancia *p* del tipo Elemento es un elemento formado por el valor asociado dicha `Elemento` y el máximo de los valores de todas las Elementos almacenadas

`#include <pila_max_cola.h>`

Atributos públicos

- int valor = 0
- int max = 0

4.2.1. Descripción detallada

struct Elemento Una instancia *p* del tipo Elemento es un elemento formado por el valor asociado dicha `Elemento` y el máximo de los valores de todas las Elementos almacenadas

4.2.2. Documentación de los datos miembro

4.2.2.1. max

```
int Elemento::max = 0
```

valor máximo almacenado en la pila

4.2.2.2. valor

```
int Elemento::valor = 0
```

valor del elemento

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

- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/pila_max_cola.h
- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/pila_max_VD.h

4.3. Referencia de la Clase PilaMax

T.D.A. PilaMax.

```
#include <pila_max_cola.h>
```

Métodos públicos

- PilaMax ()
  Constructor por defecto.
- PilaMax (const PilaMax &otra)
  Constructor de copias.
- PilaMax & operator= (const PilaMax &otra)
  Sobrecarga del operador de asignación.
- bool esta_vacia () const
  Comprueba si la pila está vacía.
- Elemento & tope ()
  Devuelve el elemento del tope de la pila.
- const Elemento & tope () const
  Devuelve el elemento del tope de una pila constante.
- void poner (const int &n)
  Añade un elemento "encima" del tope de la pila.
- void quitar ()
  Quita el elemento del tope de la pila.

- int getNelementos () const

  Devuelve el número de elementos de la pila.
- int maximo () const

  Devuelve el valor máximo almacenado en la pila.
- PilaMax ()

  Constructor por defecto.
- PilaMax (const PilaMax &otra)

  Constructor de copias.
- PilaMax & operator= (const PilaMax &otra)

  Sobrecarga del operador de asignación.
- bool esta_vacia () const

  Comprueba si la pila está vacía.
- Elemento & tope ()

  Devuelve el elemento del tope de la pila.
- const Elemento & tope () const

  Devuelve el elemento del tope de una pila constante.
- void poner (const int &n)

  Añade un elemento "encima" del tope de la pila.
- void quitar ()

  Quita el elemento del tope de la pila.
- int getNelementos () const

  Devuelve el número de elementos de la pila.
- int maximo () const

  Devuelve el valor máximo almacenado en la pila.

4.3.1. Descripción detallada

T.D.A. PilaMax.

Una instancia *v* del tipo de datos abstracto Pila sobre el tipo `T` es una lista de elementos del mismo con un funcionamiento *LIFO* (Last In, First Out). En una pila, las operaciones de inserción y borrado de elementos tienen lugar en uno de los extremos denominado *Tope*. Una pila de longitud *n* la denotamos

- [a1,a2,a3,..,an>

donde *ai* es el elemento de la posición i-ésima.

En esta pila, tendremos acceso únicamente al elemento del *Tope*, es decir, a *an*. El borrado o consulta de un elemento será sobre *an*, y la inserción de un nuevo elemento se hará sobre éste. Quedando el elemento insertado como *Tope* de la pila.

Si *n=0* diremos que la pila está vacía.

El espacio requerido para el almacenamiento es O(n). Donde n es el número de elementos de la Pila.

Además, esta Pila almacena Elementos de **valor** | **máximo**, donde máximo denota el valor máximo de los elementos inferiores en la Pila. Esto permite conocer el máximo de los valores de la pila en todo momento.

Para la implementación de la Pila, se ha usado una estructura Cola, y hemos situado el tope de la Pila en el frente de la Cola. Esto hace fácil la operación de quitar, pero complica el poner.

Autor

Francisco Rodriguez Jimenez

Fecha

Noviembre 2018

Una instancia *v* del tipo de datos abstracto Pila sobre el tipo `T` es una lista de elementos del mismo con un funcionamiento *LIFO* (Last In, First Out). En una pila, las operaciones de inserción y borrado de elementos tienen lugar en uno de los extremos denominado *Tope*. Una pila de longitud *n* la denotamos

- [a1,a2,a3,..,an>

donde *ai* es el elemento de la posición i-ésima.

En esta pila, tendremos acceso únicamente al elemento del *Tope*, es decir, a *an*. El borrado o consulta de un elemento será sobre *an*, y la inserción de un nuevo elemento se hará sobre éste. Quedando el elemento insertado como *Tope* de la pila.

Si *n=0* diremos que la pila está vacía.

El espacio requerido para el almacenamiento es O(n). Donde n es el número de elementos de la Pila.

Además, esta Pila almacena Elementos de **valor** | **máximo**, donde máximo denota el valor máximo de los elementos inferiores en la Pila. Esto permite conocer el máximo de los valores de la pila en todo momento.

Para la implementación de la Pila, se ha usado una estructura VectorDinamico, situando el *tope* de la Pila en la última posición ocupada del Vector.

Autor

Francisco Rodriguez

Fecha

Noviembre 2018

4.3.2. Documentación del constructor y destructor

4.3.2.1. PilaMax() [1/2]

```
PilaMax::PilaMax (
            const PilaMax & otra )
```

Constructor de copias.

Parámetros

otra	La pila de la que se hará la copia.

4.3.2.2. PilaMax() [2/2]

```
PilaMax::PilaMax (
            const PilaMax & otra )
```

Constructor de copias.

Parámetros

otra	La pila de la que se hará la copia.

4.3.3. Documentación de las funciones miembro

4.3.3.1. operator=() [1/2]

```
PilaMax& PilaMax::operator= (
            const PilaMax & otra )
```

Sobrecarga del operador de asignación.

Parámetros

otra	La pila que se va a asignar.

4.3.3.2. operator=() [2/2]

```
PilaMax & PilaMax::operator= (
            const PilaMax & otra )
```

Sobrecarga del operador de asignación.

Parámetros

otra	La pila que se va a asignar.

4.3.3.3. poner() [1/2]

```
void PilaMax::poner (
            const int & n )
```

Añade un elemento "encima" del tope de la pila.

Parámetros

n	Elemento que se va a añadir.

4.3.3.4. poner() [2/2]

```
void PilaMax::poner (
            const int & n )
```

Añade un elemento "encima" del tope de la pila.

Parámetros

n	Elemento que se va a añadir.

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

- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/pila_max_cola.h
- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/pila_max_VD.h
- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/src/pila_max_cola.cpp
- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/src/pila_max_VD.cpp

4.4. Referencia de la plantilla de la Clase VectorDinamico< T >

TDA VectorDinamico.

```
#include <VDG.h>
```

Métodos públicos

- VectorDinamico ()
  Constructor por defecto de la clase. Crea un vector vacío.
- VectorDinamico (int n)
  Constructor. Reserva n elementos.
- VectorDinamico & operator= (const VectorDinamico &original)
  Operador de asignación de la clase.
- VectorDinamico (const VectorDinamico &original)
  Constructor de copias de la clase.
- ~VectorDinamico ()
  Destructor de la clase. Llama al método liberar_memoria.
- void liberar_memoria ()
  Libera la memoria dinámica reservada y pone a 0 todos los datos.
- int getReservados () const
  Devuelve el número de datos reservados.
- int getUtilizados () const
  Devuelve el número de datos utilizados.
- int & getUtilizados ()
  Devuelve la referenca al número de datos reservados.
- T & operator[ ] (int i)
  Devuelve la referenca al dato de la posición i.
- const T & operator[ ] (int i) const
  Devuelve el dato de la posición i.
- void insertar (int i, T elemento)
  Inserta el dato elemento en la posición i.
- void aniade (T elemento)
  Añade el dato elemento al final del vector.
- void elimina (int i)
  Elimina el elemento de la posición i.
- void resize (int n)
  Reserva n elementos (si n < utilizados, se eliminan el resto de elementos)
- bool esta_vacio () const
  Dice si el vector está vacío.

4.4.1. Descripción detallada

template<class T>
class VectorDinamico< T >

TDA VectorDinamico.

La clase `VectorDinamico` representa una estructura de datos con *templates*, esto permite montar sobre ella otro TDA y evitar redundancia de código. Esta estructura permite acceder a la posiciones en un tiempo reducido, sacrificando eficiencia a la hora de insertar. Permite redimensionar el vector a medida que se añaden o eliminan datos. Consta de un vector de datos y dos enteros que indican el número de posiciones reservadas y utilizadas.

Autor

J. Fdez-Valdivia mod cazz

Fecha

Noviembre 2018

4.4.2. Documentación del constructor y destructor

4.4.2.1. VectorDinamico() [1/2]

```
template<class T >
VectorDinamico< T >::VectorDinamico (
            int n )
```

Constructor. Reserva *n* elementos.

Parámetros

n	número de elementos a reservar

Precondición

n >= 0

4.4.2.2. VectorDinamico() [2/2]

```
template<class T >
VectorDinamico< T >::VectorDinamico (
            const VectorDinamico< T > & original )
```

Constructor de copias de la clase.

Parámetros

original	`VectorDinamico` a copiar

4.4.3. Documentación de las funciones miembro

4.4.3.1. aniade()

```
template<class T>
void VectorDinamico< T >::aniade (
            T elemento )
```

Añade el dato *elemento* al final del vector.

Parámetros

elemento	a insertar

4.4.3.2. elimina()

```
template<class T >
void VectorDinamico< T >::elimina (
             int i )
```

Elimina el elemento de la posición *i*.

Parámetros

i	posición del elemento a borrar

Precondición

utilizados > i >= 0

4.4.3.3. esta_vacio()

```
template<class T >
bool VectorDinamico< T >::esta_vacio ( ) const
```

Dice si el vector está vacío.

Devuelve

true si está vacío, **false** si no lo está

4.4.3.4. insertar()

```
template<class T>
void VectorDinamico< T >::insertar (
             int i,
             T elemento )
```

Inserta el dato *elemento* en la posición *i*.

Parámetros

i	posición en la que se inserta el dato
elemento	a insertar

4.4.3.5. operator=()

```
template<class T >
VectorDinamico< T > & VectorDinamico< T >::operator= (
            const VectorDinamico< T > & original )
```

Operador de asignación de la clase.

Parámetros

original	`VectorDinamico` a copiar

Devuelve

el propio objeto, ∗**this**

4.4.3.6. operator[]() [1/2]

```
template<class T >
T & VectorDinamico< T >::operator[] (
            int i )
```

Devuelve la referenca al dato de la posición *i*.

Parámetros

i	posición a modificar

Precondición

utilizados > i >= 0

4.4.3.7. operator[]() [2/2]

```
template<class T >
const T & VectorDinamico< T >::operator[] (
            int i ) const
```

Devuelve el dato de la posición *i*.

Parámetros

i	posición a consultar

Precondición

utilizados > i >= 0

4.4.3.8. resize()

```
template<class T >
void VectorDinamico< T >::resize (
             int n )
```

Reserva *n* elementos (si *n* < *utilizados*, se eliminan el resto de elementos)

Parámetros

n	número de elementos a reservar

Precondición

n >= 0

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

- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/VDG.h
- /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/src/VDG.cpp

Capítulo 5

Documentación de archivos

5.1. Referencia del Archivo /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/cola.h

Fichero cabecera del TDA Cola.

```
#include <cassert>
#include "../src/cola.cpp"
```

Dependencia gráfica adjunta para cola.h:

5.2. Referencia del Archivo /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/pila_max.h

Fichero para compilar usopilas_max con VD o con cola.

```
#include "pila_max_VD.h"
```

Dependencia gráfica adjunta para pila_max.h:

defines

- #define CUAL_COMPILA 1

5.2.1. Descripción detallada

Fichero para compilar usopilas_max con VD o con cola.

5.2.2. Documentación de los 'defines'

5.2.2.1. CUAL_COMPILA

```
#define CUAL_COMPILA 1
```

Autor

Francisco Rodriguez Jimenez

Fecha

Noviembre 2018

5.3. Referencia del Archivo /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/pila_max_cola.h

Fichero cabecera del TDA Pila.

```
#include <cassert>
#include <iostream>
#include "cola.h"
```

Dependencia gráfica adjunta para pila_max_cola.h:

Clases

- struct Elemento

  struct Elemento Una instancia p del tipo Elemento es un elemento formado por el valor asociado dicha `Elemento` y el máximo de los valores de todas las Elementos almacenadas
- class PilaMax

  T.D.A. PilaMax.

Funciones

- ostream & operator<< (ostream &flujo, const Elemento &p)

  Sobrecarga del operador <<.

5.3.1. Descripción detallada

Fichero cabecera del TDA Pila.

Gestiona una secuencia de elementos con facilidades para la inserción y borrado de elementos en un extremo(FIFO)

5.3.2. Documentación de las funciones

5.3.2.1. operator<<()

```
ostream& operator<< (
            ostream & flujo,
            const Elemento & p )
```

Sobrecarga del operador <<.

Parámetros

flujo	Flujo de salida
p	`Elemento` que se quiere escribir

5.4. Referencia del Archivo /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/pila_max_VD.h

Fichero cabecera del TDA Pila.

```
#include <cassert>
#include <iostream>
#include "VDG.h"
```

Dependencia gráfica adjunta para pila_max_VD.h: Gráfico de los archivos que directa o indirectamente incluyen a este archivo:

Clases

- struct Elemento

  struct Elemento Una instancia p del tipo Elemento es un elemento formado por el valor asociado dicha `Elemento` y el máximo de los valores de todas las Elementos almacenadas
- class PilaMax

  T.D.A. PilaMax.

Funciones

- ostream & operator<< (ostream &flujo, const Elemento &p)

  Sobrecarga del operador <<.

5.4.1. Descripción detallada

Fichero cabecera del TDA Pila.

Gestiona una secuencia de elementos con facilidades para la inserción y borrado de elementos en un extremo

5.4.2. Documentación de las funciones

5.4.2.1. operator<<()

```
ostream& operator<< (
            ostream & flujo,
            const Elemento & p )
```

Sobrecarga del operador <<.

Parámetros

flujo	Flujo de salida
p	`Elemento` que se quiere escribir

5.5. Referencia del Archivo /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/include/VDG.h

Fichero cabecera del TDA VectorDinamico.

```
#include "../src/VDG.cpp"
```

Gráfico de los archivos que directa o indirectamente incluyen a este archivo:

Clases

- class VectorDinamico< T >

  TDA VectorDinamico.

5.5.1. Descripción detallada

Fichero cabecera del TDA VectorDinamico.

5.6. Referencia del Archivo /home/cazz/Documentos/3º CURSO/1 Cuatrimestre/ED/PRACTICA/Practica 3 - TDA Lineales/pilas-colas-pilamax/pila_max/src/cola.cpp

Implementación del TDA Cola.

```
#include <cassert>
#include "../include/cola.h"
```

Dependencia gráfica adjunta para cola.cpp: Gráfico de los archivos que directa o indirectamente incluyen a este archivo:

5.6.1. Descripción detallada

Implementación del TDA Cola.

Índice alfabético