Cola_max

Generated by Doxygen 1.9.2

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Cola< T > Class Template Reference

T.D.A. Cola.

`#include <cola.h>`

Public Member Functions

- Cola ()
  Constructor por defecto.
- Cola (const Cola< T > &original)
  Constructor de copias.
- ~Cola ()
  Destructor.
- Cola & operator= (const Cola< T > &otra)
  Operador de asignación.
- bool 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.

3.1.1 Detailed Description

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 inserción de un nuevo elemento se hará 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.

Author

J. Fdez-Valdivia

Date

Octubre 2020

Definition at line 41 of file cola.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Cola() [1/2]

```
template<class T >
Cola< T >::Cola ( )  [inline]
```

Constructor por defecto.

Definition at line 71 of file cola.h.

```
00071           : primera(0), ultima(0), num_elem(0){
00072     }
```

3.1.2.2 Cola() [2/2]

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

Constructor de copias.

Parameters

original	La cola de la que se hará la copia.

Definition at line 13 of file cola.cpp.

```
00013                                   {
00014   if (original.primera!=0){           //Si tiene elementos
00015     Celda *p = original.primera;      //Copiamos el puntero a la primera celda
00016     primera =
00017     ultima = new Celda(p->elemento,0); //Inicializamos la lista de nodos
00018     p = p->siguiente;                     //Avanzamos el puntero
00019     while (p!=0){                         //Mientras queden elementos
00020       ultima->siguiente = new Celda(p->elemento,0); //Copiamos el elemento
00021       ultima = ultima->siguiente;                   //Avanzamos los punteros
00022       p = p->siguiente;
00023     }
00024   }
00025   else
00026     primera = ultima = 0;            //Si no tiene elementos
00027   num_elem = original.num_elem;
00028 }
```

3.1.2.3 ~Cola()

```
template<class T >
Cola< T >::~Cola
```

Destructor.

Definition at line 33 of file cola.cpp.

```
00033                {
00034   Celda * aux;
00035   while (primera!=0){             //Mientras queden elementos
00036     aux = primera;                //Copiamos el puntero
00037     primera = primera->siguiente; //Avanzamos primera
00038     delete aux;                   //Borramos el nodo
00039   }
00040 }
```

3.1.3 Member Function Documentation

3.1.3.1 frente() [1/2]

```
template<class T >
T & Cola< T >::frente ( )  [inline]
```

Devuelve el elemento del frente de la cola.

Definition at line 98 of file cola.h.

```
00098                 {
00099       assert(primera!=0);         //Si la cola está vacía, abortar
00100       return primera->elemento;  //Devuelve el elemento del frente de la cola
00101     }
```

3.1.3.2 frente() [2/2]

```
template<class T >
const T & Cola< T >::frente ( ) const  [inline]
```

Devuelve el elemento del frente de una cola constante.

Definition at line 105 of file cola.h.

```
00105                              {
00106       assert(primera!=0);         //Si la cola está vacía, abortar
00107       return primera->elemento;  //Devuelve el elemento del frente de la cola
00108     }
```

3.1.3.3 num_elementos()

```
template<class T >
int Cola< T >::num_elementos ( ) const  [inline]
```

Devuelve el número de elementos de la cola.

Definition at line 121 of file cola.h.

```
00121                                 {
00122       return num_elem;
00123     }
```

3.1.3.4 operator=()

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

Operador de asignación.

Parameters

otra	La cola que se va a asignar.

Definition at line 45 of file cola.cpp.

```
00045                                              {
00046   Celda * p;
00047
00048   if (this != &otra){    //Comprobación de rigor. Si son diferentes objetos
00049     while (primera!=0){  //Borramos la lista de nodos de la cola *this
00050       p = primera;
00051       primera = primera->siguiente;
00052       delete p;
00053     }
00054     if (otra.primera!=0){        //Si la otra cola tiene elementos
00055       p = otra.primera;          //Copiamos el puntero al primer nodo
00056       primera=ultima= new Celda(p->elemento,0); //Reservamos el primer nodo
00057       p=p->siguiente;                           //Avanzamos el puntero
00058       while (p!=0){                             //Mientras queden elementos
00059         ultima->siguiente=new Celda(p->elemento,0); //Creamos un nuevo nodo
00060         ultima=ultima->siguiente;                   //Actualizamos ultima
00061         p=p->siguiente;                             //Avanzamos el puntero
00062       }
00063     }
```

```
00064     else primera=ultima=0;      //Si la otra cola está vacía
00065     num_elem=otra.num_elem;
00066   }
00067   return *this; //Devolvemos el objeto para permitir el encadenamiento (a=b=c)
00068 }
```

3.1.3.5 poner()

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

Añade un elemento al final de la cola.

Parameters

elem	Elemento que se va a añadir.

Definition at line 73 of file cola.cpp.

```
00073                                {
00074   Celda *aux = new Celda(elem,0);    //Creamos un nuevo nodo
00075   if (primera==0)                    //Si la lista está vacía,
00076     primera = ultima = aux;          //primera y ultima apuntan a ese nodo
00077   else{                           //Si la lista ya tenia nodos,
00078     ultima->siguiente = aux; //Actualizamos el puntero siguiente del ultimo nodo
00079     ultima = aux;            //Actualizamos ultima
00080   }
00081   num_elem++;                 //Incrementamos el numero de elementos
00082 }
```

3.1.3.6 quitar()

```
template<class T >
void Cola< T >::quitar
```

Quita el elemento del frente de la cola.

Definition at line 87 of file cola.cpp.

```
00087                     {
00088   assert(primera!=0);             //Si la cola está vacía, abortar
00089   Celda *aux = primera;           //Copiamos el puntero al primer nodo
00090   primera = primera->siguiente;   //Actualizamos primera
00091   delete aux;                     //Borramos el primer nodo
00092   if (primera == 0)               //Si no quedan nodos,
00093     ultima=0;                     //actualizamos ultima
00094   num_elem--;                     //Actualizamos el número de elementos
00095 }
```

3.1.3.7 vacia()

```
template<class T >
bool Cola< T >::vacia ( ) const  [inline]
```

Comprueba si la cola está vacía.

Definition at line 92 of file cola.h.

```
00092                         {
00093       return num_elem==0;
00094     }
```

The documentation for this class was generated from the following files:

- src/cola.h
- src/cola.cpp

3.2 Elemento Struct Reference

T.D.A Pila con máximo. Una instancia *A* del tipo abstracto de dato @A pila con máximo es un objeto que representa una pila tipo que guarda el máximo de todos sus elementos en una pila paralela a la pila en la se guardan los datos, de forma que se pueda consultar el máximo directamente en una de las pilas y en la otra se guardan los elementos de por sí. Basado en el tipo de dato propio proporcionado por Prof. Joaquín Valdivia "cola.h" y parcialmente en std::queue de la STL.

`#include <Pila_max_Cola.h>`

Public Attributes

- int elemento
- int maximo

Friends

- ostream & operator<< (ostream &os, const Elemento &elem)

  Operador <<.

3.2.1 Detailed Description

T.D.A Pila con máximo. Una instancia *A* del tipo abstracto de dato @A pila con máximo es un objeto que representa una pila tipo que guarda el máximo de todos sus elementos en una pila paralela a la pila en la se guardan los datos, de forma que se pueda consultar el máximo directamente en una de las pilas y en la otra se guardan los elementos de por sí. Basado en el tipo de dato propio proporcionado por Prof. Joaquín Valdivia "cola.h" y parcialmente en std::queue de la STL.

Author

Yeray Lopéz Ramírez

Jaime Castillo Uclés

Date

27 NOV 2021

T.D.A. Elemento de la pila Una instancia del tipo abstracto Elemento de la pila contiene dos elementos, un dato y un máximo de la pila hasta ese punto de la pila.

Definition at line 38 of file Pila_max_Cola.h.

3.2.2 Friends And Related Function Documentation

3.2.2.1 operator<<

```
ostream & operator<< (
            ostream & os,
            const Elemento & elem )  [friend]
```

Operador <<.

Parameters

os	stream de salida
elem	Elemento de salida

Returns

instancia del ostream

Definition at line 47 of file Pila_max_Cola.h.

```
00047                                                                   {
00048                 os « elem.elemento « " (" « elem.maximo « ")" « endl;
00049                 return os;
00050             };
```

3.2.3 Member Data Documentation

3.2.3.1 elemento

```
int Elemento::elemento
```

Definition at line 39 of file Pila_max_Cola.h.

3.2.3.2 maximo

```
int Elemento::maximo
```

Definition at line 40 of file Pila_max_Cola.h.

The documentation for this struct was generated from the following file:

- src/Pila_max_Cola.h

3.3 Pila_max Class Reference

Public Member Functions

- Elemento frente ()
  Función frente.
- int elementos_size ()
  Funcion elementos_size.
- bool empty ()
  Función empty.
- void quitar ()
  Función quitar, elimina el elemento que se encuentra en el frente de la pila y su máximo correpondiente.
- void poner (int elemento)
  Función poner, añade un nuevo elemento a la pila y actualiza el máximo.

3.3.1 Detailed Description

Definition at line 53 of file Pila_max_Cola.h.

3.3.2 Member Function Documentation

3.3.2.1 elementos_size()

```
int Pila_max::elementos_size ( )
```

Funcion elementos_size.

Returns

Devuelve la cantidad de elementos que contiene la pila

Definition at line 18 of file Pila_max_Cola.cpp.

```
00018                               {
00019     return pila.num_elementos();
00020 }
```

3.3.2.2 empty()

```
bool Pila_max::empty ( )
```

Función empty.

Returns

Devuelve true si la pila está vacía

Definition at line 23 of file Pila_max_Cola.cpp.

```
00023                       {
00024     return pila.vacia();
00025 }
```

3.3.2.3 frente()

```
Elemento Pila_max::frente ( )
```

Función frente.

Precondition

Se necesita mínimo un valor en la pila previo

Returns

Devuelve el elemento en el frente de la pila

Definition at line 12 of file Pila_max_Cola.cpp.

```
00012                           {
00013   return pila.frente();
00014 }
```

3.3.2.4 poner()

```
void Pila_max::poner (
            int elemento )
```

Función poner, añade un nuevo elemento a la pila y actualiza el máximo.

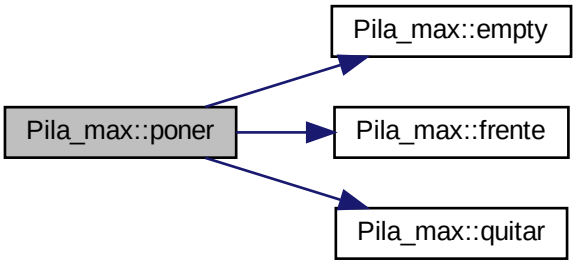
Parameters

Elemento	a añadir

Definition at line 33 of file Pila_max_Cola.cpp.

```
00033                                   {
00034     if(empty()){
00035         Elemento nuevo = {elemento, elemento};
00036         pila.poner(nuevo);
00037         return;
00038     }
00039     queue<Elemento> aux;
00040     while(!empty()){
00041         aux.push(frente());
00042         quitar();
00043     }
00044     int maximo = aux.front().maximo;
00045     Elemento nuevo;
00046     if(elemento>maximo){
00047      nuevo = {elemento, elemento};
00048     }
00049     else{
00050      nuevo = {elemento, maximo};
00051     }
00052     pila.poner(nuevo);
00053
00054     while(!aux.empty()) {
00055         pila.poner(aux.front());
00056         aux.pop();
00057     }
00058 }
```

Here is the call graph for this function:



3.3.2.5 quitar()

```
void Pila_max::quitar ( )
```

Función quitar, elimina el elemento que se encuentra en el frente de la pila y su máximo correpondiente.

Precondition

Se necesita mínimo un valor en la pila previo

Definition at line 27 of file Pila_max_Cola.cpp.

```
00027                       {
00028     pila.quitar();
00029 }
```

The documentation for this class was generated from the following files:

- src/Pila_max_Cola.h
- src/Pila_max_Cola.cpp

Chapter 4

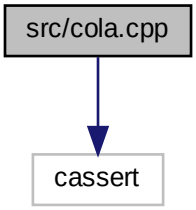
File Documentation

4.1 src/cola.cpp File Reference

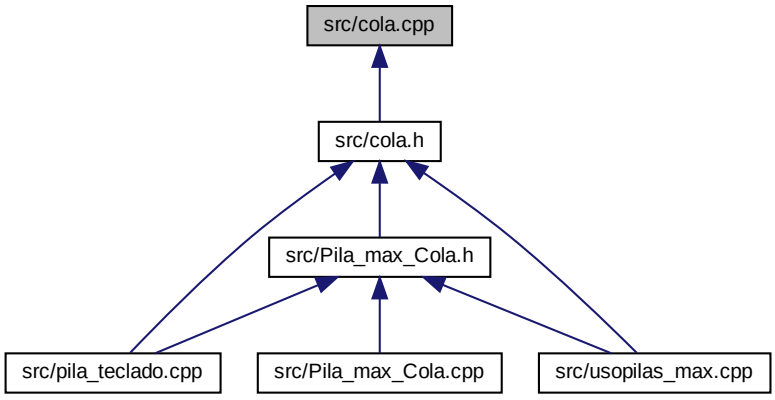
Implementación del TDA Cola.

```
#include <cassert>
```

Include dependency graph for cola.cpp:



This graph shows which files directly or indirectly include this file:



4.1.1 Detailed Description

Implementación del TDA Cola.

Definition in file cola.cpp.

4.2 cola.cpp

Go to the documentation of this file.

```
00001
00006 #include <cassert>
00007 // #include <cola.h>  El codigo ya se incluye en cola.h
00008
00009 /* ------------------------------------------------------------------------ */
00010
00011
00012 template <class T>
00013 Cola<T>::Cola(const Cola<T> & original){
00014   if (original.primera!=0){           //Si tiene elementos
00015     Celda *p = original.primera;      //Copiamos el puntero a la primera celda
00016     primera =
00017     ultima = new Celda(p->elemento,0); //Inicializamos la lista de nodos
00018     p = p->siguiente;                    //Avanzamos el puntero
00019     while (p!=0){                        //Mientras queden elementos
00020       ultima->siguiente = new Celda(p->elemento,0); //Copiamos el elemento
00021       ultima = ultima->siguiente;                   //Avanzamos los punteros
00022       p = p->siguiente;
00023     }
00024   }
00025   else
00026     primera = ultima = 0;            //Si no tiene elementos
00027   num_elem = original.num_elem;
00028 }
00029
00030 /* ------------------------------------------------------------------------ */
00031
00032 template <class T>
00033 Cola<T>::~Cola(){
00034   Celda * aux;
00035   while (primera!=0){              //Mientras queden elementos
00036     aux = primera;                 //Copiamos el puntero
00037     primera = primera->siguiente; //Avanzamos primera
00038     delete aux;                    //Borramos el nodo
00039   }
00040 }
00041
00042 /* ------------------------------------------------------------------------ */
00043
00044 template <class T>
00045 Cola<T>& Cola<T>::operator=(const Cola<T> & otra){
00046   Celda * p;
00047
00048   if (this != &otra){    //Comprobación de rigor. Si son diferentes objetos
00049     while (primera!=0){  //Borramos la lista de nodos de la cola *this
00050       p = primera;
00051       primera = primera->siguiente;
00052       delete p;
00053     }
00054     if (otra.primera!=0){        //Si la otra cola tiene elementos
00055       p = otra.primera;          //Copiamos el puntero al primer nodo
00056       primera=ultima= new Celda(p->elemento,0); //Reservamos el primer nodo
00057       p=p->siguiente;                           //Avanzamos el puntero
00058       while (p!=0){                             //Mientras queden elementos
00059         ultima->siguiente=new Celda(p->elemento,0); //Creamos un nuevo nodo
00060         ultima=ultima->siguiente;                   //Actualizamos ultima
00061         p=p->siguiente;                             //Avanzamos el puntero
00062       }
00063     }
00064     else primera=ultima=0;      //Si la otra cola está vacía
00065     num_elem=otra.num_elem;
00066   }
00067   return *this; //Devolvemos el objeto para permitir el encadenamiento (a=b=c)
00068 }
00069
00070 /*------------------------------------------------------------------------ */
00071
00072 template <class T>
00073 void Cola<T>::poner(const T & elem){
```

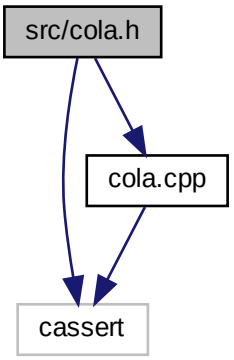
```
00074   Celda *aux = new Celda(elem,0);     //Creamos un nuevo nodo
00075   if (primera==0)                     //Si la lista está vacía,
00076     primera = ultima = aux;           //primera y ultima apuntan a ese nodo
00077   else{                       //Si la lista ya tenia nodos,
00078     ultima->siguiente = aux; //Actualizamos el puntero siguiente del ultimo nodo
00079     ultima = aux;            //Actualizamos ultima
00080   }
00081   num_elem++;                //Incrementamos el numero de elementos
00082 }
00083
00084 /* ____________________________________________________________________ */
00085
00086 template <class T>
00087 void Cola<T>::quitar(){
00088   assert(primera!=0);             //Si la cola está vacía, abortar
00089   Celda *aux = primera;           //Copiamos el puntero al primer nodo
00090   primera = primera->siguiente;   //Actualizamos primera
00091   delete aux;                     //Borramos el primer nodo
00092   if (primera == 0)               //Si no quedan nodos,
00093     ultima=0;                     //actualizamos ultima
00094   num_elem--;                     //Actualizamos el número de elementos
00095 }
```

4.3 src/cola.h File Reference

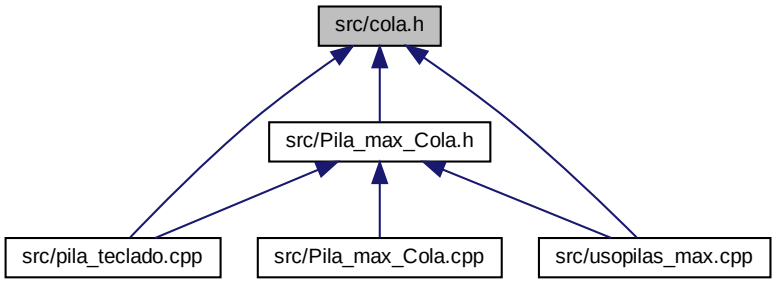
Fichero cabecera del TDA Cola.

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

Include dependency graph for cola.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Cola< T >

  T.D.A. Cola.

4.3.1 Detailed Description

Fichero cabecera del TDA Cola.

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

Definition in file cola.h.

4.4 cola.h

Go to the documentation of this file.

```
00001
00009 #ifndef __Cola_H__
00010 #define __Cola_H__
00011
00012 #include <cassert>
00013
00040 template <class T>
00041 class Cola{
00042
00043   private:
00044     struct Celda {
00045       T elemento;
00046       Celda * siguiente;
00047
00051       Celda() : siguiente(0){
00052       }
00058       Celda(const T & elem, Celda * sig): elemento(elem), siguiente(sig){
00059       }
00060     };
00061
00062     Celda * primera;
00063     Celda * ultima;
00064     int num_elem;
00065
00066   public:
00067     // ---------------  Constructores ----------------
00071     Cola(): primera(0), ultima(0), num_elem(0){
00072     }
```

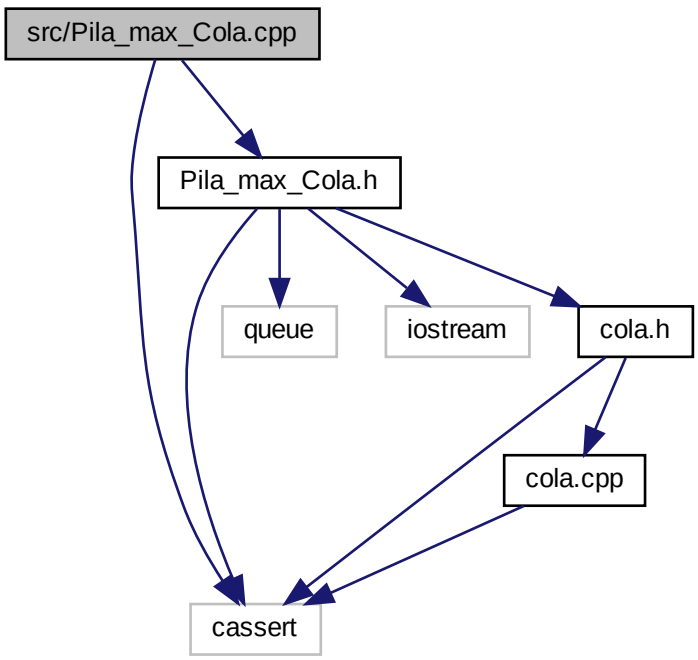
```
00077     Cola(const Cola<T> & original);
00078     // ------------------ Destructor ------------------
00082     ~Cola();
00083     // --------------- Otras funciones ---------------
00088     Cola& operator= (const Cola<T> & otra);
00092     bool vacia() const{
00093       return num_elem==0;
00094     }
00098     T& frente (){
00099       assert(primera!=0);         //Si la cola está vacía, abortar
00100       return primera->elemento;  //Devuelve el elemento del frente de la cola
00101     }
00105     const T & frente () const{
00106       assert(primera!=0);         //Si la cola está vacía, abortar
00107       return primera->elemento;  //Devuelve el elemento del frente de la cola
00108     }
00113     void poner(const T & elem);
00117     void quitar();
00121     int num_elementos() const{
00122       return num_elem;
00123     }
00124 };
00125
00126 #include "cola.cpp"
00127
00128 #endif // __Cola_H__
```

4.5 src/Pila_max_Cola.cpp File Reference

Implementación del T.D.A Pila con máximo.

```
#include <cassert>
#include "Pila_max_Cola.h"
```

Include dependency graph for Pila_max_Cola.cpp:



4.5.1 Detailed Description

Implementación del T.D.A Pila con máximo.

Definition in file Pila_max_Cola.cpp.

4.6 Pila_max_Cola.cpp

Go to the documentation of this file.

```
00001
00006 #include <cassert>
00007 #include "Pila_max_Cola.h"
00008
00009 /*-------------------------------------------------------------------------*/
00010
00011
00012 Elemento Pila_max::frente(){
00013   return pila.frente();
00014 }
00015 /*-------------------------------------------------------------------------*/
00016
00017
00018 int Pila_max::elementos_size(){
00019    return pila.num_elementos();
00020 }
00021
00022
00023 bool Pila_max::empty(){
00024    return pila.vacia();
00025 }
00026
00027 void Pila_max::quitar(){
00028    pila.quitar();
00029 }
00030
00031 /*-------------------------------------------------------------------------*/
00032
00033 void Pila_max::poner(int elemento){
00034     if(empty()){
00035         Elemento nuevo = {elemento, elemento};
00036         pila.poner(nuevo);
00037         return;
00038     }
00039     queue<Elemento> aux;
00040     while(!empty()){
00041         aux.push(frente());
00042         quitar();
00043     }
00044     int maximo = aux.front().maximo;
00045     Elemento nuevo;
00046     if(elemento>maximo){
00047      nuevo = {elemento, elemento};
00048     }
00049     else{
00050      nuevo = {elemento, maximo};
00051     }
00052     pila.poner(nuevo);
00053
00054     while(!aux.empty()) {
00055         pila.poner(aux.front());
00056         aux.pop();
00057     }
00058 }
00059
```

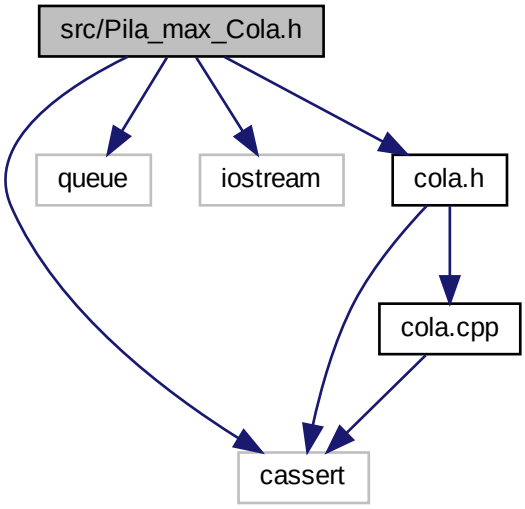
4.7 src/Pila_max_Cola.h File Reference

Fichero cabecera del T.D.A Pila con máximo.

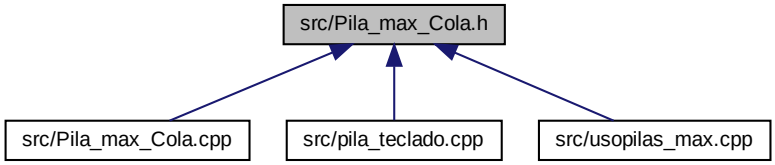
```
#include <cassert>
#include <queue>
```

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

Include dependency graph for Pila_max_Cola.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct Elemento

  T.D.A Pila con máximo. Una instancia A del tipo abstracto de dato @A pila con máximo es un objeto que representa una pila tipo que guarda el máximo de todos sus elementos en una pila paralela a la pila en la se guardan los datos, de forma que se pueda consultar el máximo directamente en una de las pilas y en la otra se guardan los elementos de por sí. Basado en el tipo de dato propio proporcionado por Prof. Joaquín Valdivia "cola.h" y parcialmente en std::queue de la STL.
- class Pila_max

4.7.1 Detailed Description

Fichero cabecera del T.D.A Pila con máximo.

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

Definition in file Pila_max_Cola.h.

4.8 Pila_max_Cola.h

Go to the documentation of this file.

```
00001
00009 #ifndef __PILA_MAX_COLA__
00010 #define __PILA_MAX_COLA__
00011
00012 #include <cassert>
00013 #include <queue>
00014 #include <iostream>
00015 #include "cola.h"
00016
00017
00018 using namespace std;
00019
00038 struct Elemento{
00039     int elemento;
00040     int maximo;
00047         friend ostream& operator«(ostream &os, const  Elemento& elem){
00048                 os « elem.elemento « " (" « elem.maximo « ")" « endl;
00049                 return os;
00050             };
00051     };
00052
00053 class Pila_max{
00054 private:
00055
00056     Cola<Elemento> pila;
00057 public:
00058
00064    Elemento frente();
00065
00070    int elementos_size();
00071
00076    bool empty();
00077
00082    void quitar();
00083
00088    void poner(int elemento);
00089
00090 };
00091
00092 #endif
00093
00094
00095
00096
```

4.9 pila_teclado.cpp

```
00001  #include <iostream>
00002  #include "Pila_max_Cola.h"
00003  #include "cola.h"
00004
00005  using namespace std;
00006
00007
00012 // class Cola_max;
00013 // class Pila<T>;
00014
00015  int main(){
00016      Pila_max q;
00017      /*
00018      int i;
00019      for ( i=10; i>=0 ; i--)
```

```
00020           q.poner(i);
00021       */
00022       int dato;
00023       cout « "Introduce un dato a la pila(0 para salir):";
00024       cin » dato;
00025       while(dato != 0){
00026           q.poner(dato);
00027           cin » dato;
00028       }
00029
00030        while (!q.empty()){
00031           cout « q.frente() « endl;
00032           q.quitar();
00033       }
00034
00035       return 0;
00036   }
00037
```

4.10 usopilas_max.cpp

```
00001   #include <iostream>
00002   #include "Pila_max_Cola.h"
00003   #include "cola.h"
00004
00005   using namespace std;
00006
00007
00012  // class Cola_max;
00013  // class Pila<T>;
00014
00015   int main(){
00016       Pila_max q;
00017       int i;
00018       /*
00019       for ( i=10; i>=0 ; i--)
00020           q.poner(i);
00021       */
00022       q.poner(4);
00023       q.poner(5);
00024       q.poner(6);
00025       q.poner(7);
00026       q.poner(22);
00027       q.poner(11);
00028
00029       Elemento y = q.frente();
00030       cout« y «endl;
00031
00032        while (!q.empty() ){
00033           Elemento x = q.frente();
00034           cout « x «endl;
00035           q.quitar();
00036       }
00037
00038       return 0;
00039   }
```

Index