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_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 cola y su máximo correpondiente.
- void poner (int elemento)
  Función poner, añade un nuevo elemento a la cola y actualiza el máximo.

3.1.1 Detailed Description

Definition at line 52 of file Cola_max_Pila.h.

3.1.2 Member Function Documentation

3.1.2.1 elementos_size()

```
int Cola_max::elementos_size ( )
```

Funcion elementos_size.

Returns

Devuelve la cantidad de elementos que contiene la cola

Definition at line 18 of file Cola_max_Pila.cpp.

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

3.1.2.2 empty()

```
bool Cola_max::empty ( )
```

Función empty.

Returns

Devuelve true si la cola está vacía (las dos colas están vacías)

Definition at line 23 of file Cola_max_Pila.cpp.

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

3.1.2.3 frente()

```
Elemento Cola_max::frente ( )
```

Función frente.

Precondition

Se necesita mínimo un valor en la cola previo

Returns

Devuelve el elemento en el frente de la cola

Definition at line 12 of file Cola_max_Pila.cpp.

```
00012                           {
00013   return cola.tope();
00014 }
```

3.1.2.4 poner()

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

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

Parameters

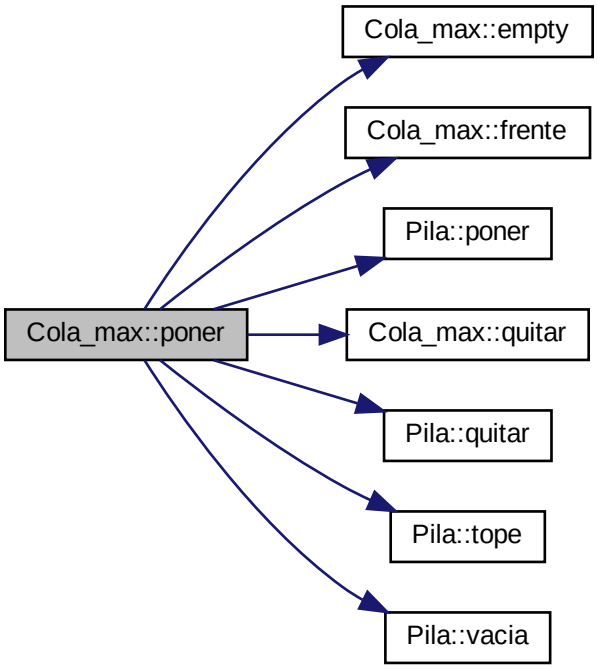
Elemento	a añadir

Definition at line 32 of file Cola_max_Pila.cpp.

```
00032                                   {
00033     if(empty()){
```

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

Here is the call graph for this function:



3.1.2.5 quitar()

```
void Cola_max::quitar ( )
```

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

Precondition

Se necesita mínimo un valor en la cola previo

Definition at line 27 of file Cola_max_Pila.cpp.

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

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

- src/Cola_max_Pila.h
- src/Cola_max_Pila.cpp

3.2 Elemento Struct Reference

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

`#include <Cola_max_Pila.h>`

Public Attributes

- int elemento
- int maximo

Friends

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

  Operador <<.

3.2.1 Detailed Description

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

Author

Yeray Lopéz Ramírez

Jaime Castillo Uclés

Date

27 NOV 2021

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

Definition at line 37 of file Cola_max_Pila.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
elemento	Elemento de salida

Returns

instancia del ostream

Definition at line 46 of file Cola_max_Pila.h.

```
00046                                                                   {
00047                 os « elem.elemento « " (" « elem.maximo « ")" « endl;
00048                 return os;
00049             };
```

3.2.3 Member Data Documentation

3.2.3.1 elemento

```
int Elemento::elemento
```

Definition at line 38 of file Cola_max_Pila.h.

3.2.3.2 maximo

```
int Elemento::maximo
```

Definition at line 39 of file Cola_max_Pila.h.

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

- src/Cola_max_Pila.h

3.3 Pila< T > Class Template Reference

T.D.A. Pila.

`#include <pila.h>`

Public Member Functions

- Pila ()
  Constructor por defecto.
- Pila (const Pila< T > &otra)
  Constructor de copias.
- ~Pila ()
  Destructor.
- Pila & operator= (const Pila< T > &otra)
  Operador de asignación.
- bool vacia () const
  Comprueba si la pila está vacía.
- T & tope ()
  Devuelve el elemento del tope de la pila.
- const T & tope () const
  Devuelve el elemento del tope de una pila constante.
- void poner (const T &elem)
  Añade un elemento "encima" del tope de la pila.
- void quitar ()
  Quita el elemento del tope de la pila.
- int num_elementos () const
  Devuelve el número de elementos de la pila.

3.3.1 Detailed Description

template<class T>
class Pila< T >

T.D.A. Pila.

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 esté vacía.

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

Author

J.Fdez-Valdivia.

Date

Octubre 2011

Definition at line 43 of file pila.h.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Pila() [1/2]

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

Constructor por defecto.

Definition at line 71 of file pila.h.

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

3.3.2.2 Pila() [2/2]

```
template<class T >
Pila< T >::Pila (
             const Pila< T > & otra )
```

Constructor de copias.

Parameters

otra	La pila de la que se hará la copia.

Definition at line 12 of file pila.cpp.

```
00012                                {
00013   if (otra.primera!=0){            //Si la pila original no está vacía
00014     Celda *p = otra.primera;       //Copio el puntero al primer nodo
00015     Celda *nueva;
00016     primera =
00017     nueva = new Celda(p->elemento,0); //Creamos el primer nodo
00018     p = p->siguiente;                 //Avanzamos el puntero
00019     while (p!=0){                                  //Mientras queden elementos
00020       nueva->siguiente = new Celda(p->elemento,0); //Creamos un nuevo nodo
00021       nueva = nueva->siguiente;                    //Avanzamos los punteros
00022       p = p->siguiente;
00023     }
00024   }
00025   else                             //Si la pila original está vacía
00026     primera = 0;
00027   num_elem = otra.num_elem;     //En cualquier caso, copiamos num_elem
00028 }
```

3.3.2.3 ~Pila()

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

Destructor.

Definition at line 33 of file pila.cpp.

```
00033              {
00034   Celda *aux;
00035   while (primera!=0){              //Mientras la pila no está vacía,
00036     aux = primera;                 //Copiamos el puntero al tope de la pila
00037     primera = primera->siguiente;  //Avanzamos primera
00038     delete aux;                    //Borramos el nodo de la lista
00039   }
00040 }
```

3.3.3 Member Function Documentation

3.3.3.1 num_elementos()

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

Devuelve el número de elementos de la pila.

Definition at line 121 of file pila.h.

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

3.3.3.2 operator=()

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

Operador de asignación.

Parameters

otra	La pila que se va a asignar.

Definition at line 45 of file pila.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 pila *this
00050       p = primera;
00051       primera = primera->siguiente;
00052       delete p;
00053     }
00054     if (otra.primera!=0){        //Si la otra pila tiene elementos
00055       p = otra.primera;          //Copiamos el puntero al primero nodo
00056       Celda *nueva;
00057       primera = nueva = new Celda(p->elemento,0);//Reservamos el primer nodo
00058       p = p->siguiente;                          //Avanzamos el puntero
00059       while (p!=0){                              //Mientras queden elementos
00060         nueva->siguiente = new Celda(p->elemento,0); //Creamos un nuevo nodo
00061         nueva = nueva->siguiente;                  //Avanzamos nueva (auxiliar)
00062         p = p->siguiente;                          //Avanzamos el puntero
00063       }
00064     }
```

```
00065     num_elem=otra.num_elem;      //En cualquier caso, copiamos num_elem
00066   }
00067   return *this; //Devolvemos el objeto para permitir el encadenamiento (a=b=c)
00068 }
```

3.3.3.3 poner()

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

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

Parameters

elem	Elemento que se va a añadir.

Definition at line 73 of file pila.cpp.

```
00073                                {
00074   primera = new Celda(elem,primera); //Creamos un nuevo nodo en el tope
00075   num_elem++;                        //Actualizamos num_elem
00076 }
```

3.3.3.4 quitar()

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

Quita el elemento del tope de la pila.

Definition at line 81 of file pila.cpp.

```
00081                    {
00082   assert(primera!=0);           //Si la pila no tiene elementos, abortar
00083   Celda *p = primera;           //Copiamos el puntero al tope
00084   primera = primera->siguiente; //Actualizamos primera
00085   delete p;                     //Borramos el nodo que ocupaba el tope
00086   num_elem--;                   //Actualizamos num_elem
00087 }
```

3.3.3.5 tope() [1/2]

```
template<class T >
T & Pila< T >::tope ( )  [inline]
```

Devuelve el elemento del tope de la pila.

Definition at line 98 of file pila.h.

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

3.3.3.6 tope() [2/2]

```
template<class T >
const T & Pila< T >::tope ( ) const  [inline]
```

Devuelve el elemento del tope de una pila constante.

Definition at line 105 of file pila.h.

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

3.3.3.7 vacia()

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

Comprueba si la pila está vacía.

Definition at line 92 of file pila.h.

```
00092                         {
00093       return (primera==0);
00094     }
```

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

- src/pila.h
- src/pila.cpp

Chapter 4

File Documentation

4.1 Cola_max_Pila.cpp

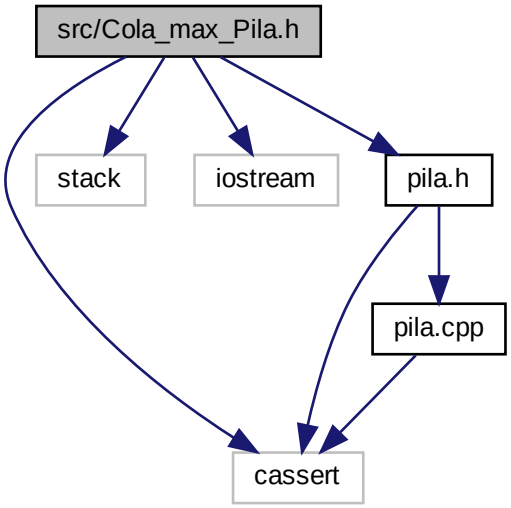
```
00001
00006 #include <cassert>
00007 #include "Cola_max_Pila.h"
00008
00009 /*-------------------------------------------------------------------------*/
00010
00011
00012 Elemento Cola_max::frente(){
00013   return cola.tope();
00014 }
00015 /*-------------------------------------------------------------------------*/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

4.2 src/Cola_max_Pila.h File Reference

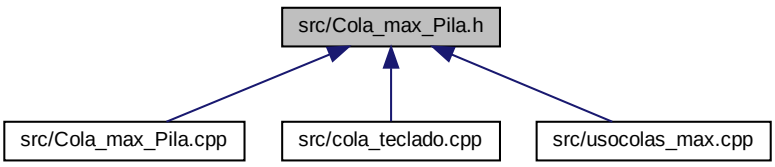
Fichero cabecera del T.D.A Cola con máximo.

```
#include <cassert>
#include <stack>
#include <iostream>
#include "pila.h"
```

Include dependency graph for Cola_max_Pila.h:



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



Classes

- struct Elemento

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

4.2.1 Detailed Description

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

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

Definition in file Cola_max_Pila.h.

4.3 Cola_max_Pila.h

Go to the documentation of this file.

```
00001
00009 #ifndef __COLA_MAX_PILA__
00010 #define __COLA_MAX_PILA__
00011
00012 #include <cassert>
00013 #include <stack>
00014 #include <iostream>
00015 #include "pila.h"
00016
00017
00018 using namespace std;
00019
00037 struct Elemento{
00038     int elemento;
00039     int maximo;
00046         friend ostream& operator«(ostream &os, const  Elemento& elem){
00047                 os « elem.elemento « " (" « elem.maximo « ")" « endl;
00048                 return os;
00049             };
00050     };
00051
00052 class Cola_max{
00053 private:
00054
00055     Pila<Elemento> cola;
00056 public:
00057
00063    Elemento frente();
00064
00069    int elementos_size();
00070
00075    bool empty();
00076
00081    void quitar();
00082
00087    void poner(int elemento);
00088
00089 };
00090
00091 #endif
00092
00093
00094
00095
```

4.4 cola_teclado.cpp

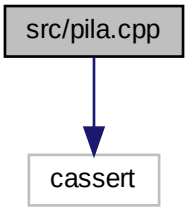
```
00001  #include <iostream>
00002  #include "Cola_max_Pila.h"
00003  #include "pila.h"
00004
00005  using namespace std;
00006
00007
00012 // class Cola_max;
00013 // class Pila<T>;
00014
00015  int main(){
00016      Cola_max q;
00017      int dato;
00018      /*
00019      int i;
```

```
00020      for ( i=10; i>=0 ; i--)
00021          q.poner(i);
00022     */
00023      cout « "Introduce un dato a la cola(0 para salir):";
00024     cin » dato;
00025     while(dato != 0){
00026         q.poner(dato);
00027         cin » dato;
00028     }
00029      while (!q.empty() ){
00030          Elemento x = q.frente();
00031          cout « x «endl;
00032          q.quitar();
00033     }
00034
00035     return 0;
00036  }
```

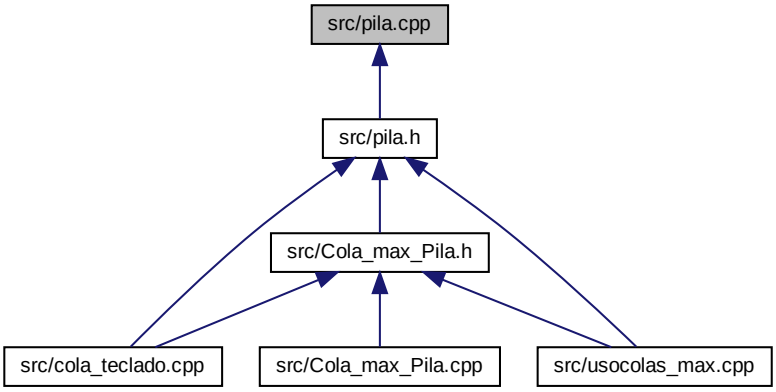
4.5 src/pila.cpp File Reference

Implementación del TDA Pila.

`#include <cassert>`
Include dependency graph for pila.cpp:



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



4.5.1 Detailed Description

Implementación del TDA Pila.

Definition in file pila.cpp.

4.6 pila.cpp

Go to the documentation of this file.

```
00001
00006 #include <cassert>
00007 //#include "pila.h"
00008
00009 /* ____________________________________________________________________ */
00010
00011 template <class T>
00012 Pila<T>::Pila(const Pila<T> & otra){
00013   if (otra.primera!=0){            //Si la pila original no está vacía
00014     Celda *p = otra.primera;       //Copio el puntero al primer nodo
00015     Celda *nueva;
00016     primera =
00017     nueva = new Celda(p->elemento,0); //Creamos el primer nodo
00018     p = p->siguiente;                 //Avanzamos el puntero
00019     while (p!=0){                                  //Mientras queden elementos
00020       nueva->siguiente = new Celda(p->elemento,0); //Creamos un nuevo nodo
00021       nueva = nueva->siguiente;                    //Avanzamos los punteros
00022       p = p->siguiente;
00023     }
00024   }
00025   else                              //Si la pila original está vacía
00026     primera = 0;
00027   num_elem = otra.num_elem;     //En cualquier caso, copiamos num_elem
00028 }
00029
00030 /* ____________________________________________________________________ */
00031
00032 template <class T>
00033 Pila<T>::~Pila(){
00034   Celda *aux;
00035   while (primera!=0){              //Mientras la pila no está vacía,
00036     aux = primera;                 //Copiamos el puntero al tope de la pila
00037     primera = primera->siguiente;  //Avanzamos primera
00038     delete aux;                    //Borramos el nodo de la lista
00039   }
00040 }
00041
00042 /* ____________________________________________________________________ */
00043
00044 template <class T>
00045 Pila<T>& Pila<T>::operator=(const Pila<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 pila *this
00050       p = primera;
00051       primera = primera->siguiente;
00052       delete p;
00053     }
00054     if (otra.primera!=0){        //Si la otra pila tiene elementos
00055       p = otra.primera;          //Copiamos el puntero al primero nodo
00056       Celda *nueva;
00057       primera = nueva = new Celda(p->elemento,0);//Reservamos el primer nodo
00058       p = p->siguiente;                          //Avanzamos el puntero
00059       while (p!=0){                              //Mientras queden elementos
00060         nueva->siguiente = new Celda(p->elemento,0); //Creamos un nuevo nodo
00061         nueva = nueva->siguiente;                  //Avanzamos nueva (auxiliar)
00062         p = p->siguiente;                          //Avanzamos el puntero
00063       }
00064     }
00065     num_elem=otra.num_elem;      //En cualquier caso, copiamos 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 Pila<T>::poner(const T & elem){
```

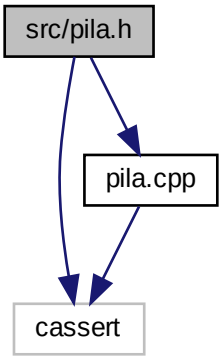
```
00074   primera = new Celda(elem,primera); //Creamos un nuevo nodo en el tope
00075   num_elem++;                        //Actualizamos num_elem
00076 }
00077
00078 /* ____________________________________________________________________ */
00079
00080 template <class T>
00081 void Pila<T>::quitar(){
00082   assert(primera!=0);           //Si la pila no tiene elementos, abortar
00083   Celda *p = primera;           //Copiamos el puntero al tope
00084   primera = primera->siguiente; //Actualizamos primera
00085   delete p;                     //Borramos el nodo que ocupaba el tope
00086   num_elem--;                   //Actualizamos num_elem
00087 }
```

4.7 src/pila.h File Reference

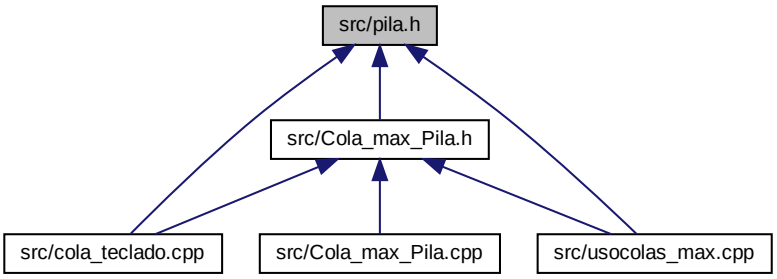
Fichero cabecera del TDA Pila.

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

Include dependency graph for pila.h:



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



Classes

- class Pila< T >

  T.D.A. Pila.

4.7.1 Detailed Description

Fichero cabecera del TDA Pila.

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

Definition in file pila.h.

4.8 pila.h

Go to the documentation of this file.

```
00001
00010 #ifndef __Pila_H__
00011 #define __Pila_H__
00012
00013 #include <cassert>
00014
00042 template <class T>
00043 class Pila{
00044   private:
00045     struct Celda {
00046       T elemento;
00047       Celda * siguiente;
00048
00052       Celda() : siguiente(0){
00053       }
00059       Celda(const T & elem, Celda * sig): elemento(elem), siguiente(sig){
00060       }
00061     };
00062
00063     Celda * primera;
00064     int num_elem;
00065
00066   public:
00067     // ---------------  Constructores ----------------
00071     Pila(): primera(0), num_elem(0){
00072     }
00077     Pila(const Pila<T> & otra);
00078     // ------------------ Destructor ------------------
00082     ~Pila();
00083     // --------------- Otras funciones ---------------
00088     Pila& operator= (const Pila<T>& otra);
00092     bool vacia() const{
00093       return (primera==0);
00094     }
00098     T& tope (){
00099       assert(primera!=0);       //Si la pila está vacía, abortar
00100       return primera->elemento; //Devuelve el elemento del tope de la pila
00101     }
00105     const T & tope () const{
00106       assert(primera!=0);       //Si la pila está vacía, abortar
00107       return primera->elemento; //Devuelve el elemento del tope de la pila
00108     }
00113     void poner(const T & elem);
00117     void quitar();
00121     int num_elementos() const{
00122       return num_elem;
00123     }
00124 };
00125
00126 #include "pila.cpp"
00127 #endif  //__Pila_H__
```

4.9 usocolas_max.cpp

```
00001  #include <iostream>
00002  #include "Cola_max_Pila.h"
00003  #include "pila.h"
00004
00005  using namespace std;
00006
00007
00012 // class Cola_max;
00013 // class Pila<T>;
00014
00015  int main(){
00016      Cola_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      while (!q.empty() ){
00030          Elemento x = q.frente();
00031          cout « x «endl;
00032          q.quitar();
00033     }
00034
00035     return 0;
00036  }
```

Index