Prog18. Conjuntos con listas doblemente ligadas.

System and Reference Manual, created on 15/Dec/03.

This documentation was automatically generated using **Doc-O-Matic 2**.

Content

1 Symbol Reference

These are all symbols available in this documentation.

1.1 Classes

These are all classes that are contained in this documentation.

1.1.1 CppListPos

```
template <class TObj> class CppListPos : public ListPos<TObj>;
```

Class Hierarchy

ListPos (see page 23)
  CppListPos (see page 1)

File

cpplistpos.h (see page 33)

CppListPos Members

Data Members

list

Methods

~CppListPos
addLast
getLast
isFull
removeAllElements
removeLast
addFirst
CppListPos
isEmpty
print
removeFirst
size

Legend

private

Description

public muestra miembros error friend ostream& operator<<(ostream &outStr, const CppListPos (see CppListPos::CppListPos, page 2)<TObj> &list (see CppListPos::list, page 1)); // ???const <TObj>

CppListPos::list

```
vector<TObj> list;
```

Description

Conjunto de elementos.

CppListPos::~CppListPos

```
~CppListPos();
```

Description

Destructor. Libera memoria.

Destructor. Libera nodos.

CppListPos::addFirst

```
void addFirst(const TObj item);
```

Parameters

```
const TObj item
```

Elemento. @throw FullListExceptionPos (see page 20) No hay capacidad de almacenamiento.

Description

Inserta un elemento al inicio de la lista.

CppListPos::addLast

```
void addLast(const TObj item);
```

Parameters

`const TObj item`

Elemento. @throw FullListExceptionPos (see page 20) No hay capacidad de almacenamiento.

Description

Inserta un elemento al final de la lista.

CppListPos::CppListPos

```
CppListPos();
```

Parameters

`theCapacity`

Numero de elementos. 10% del INT_MAX_VALUE (see page 32).

Description

Constructor sin elementos, capacidad predeterminada

Construye una lista estatica de taman~o fijo.

See Also

INT_MAX_VALUE (see page 32)

CppListPos::getLast

```
TObj getLast();
```

Return Value

item Elemento revisado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Regresa el contenido del elemento al final de la lista sin eliminarlo.

Version

26/12/2001

CppListPos::isEmpty

```
bool isEmpty() const;
```

Return Value

true No hay elementos en la lista

Description

Checa si la lista esta vacia.

CppListPos::isFull

```
bool isFull() const;
```

Return Value

true Se alcanzo el limite maximo.

Description

Checa si la lista esta llena. En teoria, la lista puede crecer indefinidamente, pero para conservar la consistencia con "size (see CppListPos::size, page 3)()" se limita al maximo soportado por un entero.

CppListPos::print

```
void print() const;
```

Return Value

Cadena con los elementos

Description

CppListPos::removeAllElements

```
void removeAllElements();
```

Description

Elimina todos los elementos de la lista. Libera memoria. No lanza "Exception".

CppListPos::removeFirst

```
TObj removeFirst();
```

Return Value

item Elemento retirado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Toma el elemento al inicio de la lista y lo elimina. Libera memoria.

CppListPos::removeLast

```
TObj removeLast();
```

Return Value

item Elemento retirado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Toma el elemento al final de la lista y lo elimina. Libera memoria.

CppListPos::size

```
int size() const;
```

Return Value

Numero entero de elementos.

Description

Cuenta el numero de elementos en la lista.

1.1.2 CppQueuePos

```
template <class TObj> class CppQueuePos : public QueuePos<TObj>;
```

Class Hierarchy

QueuePos (see page 25)
  CppQueuePos (see page 3)

File

allstackqueuepos.h (see page 33)

CppQueuePos Members

Methods

CppQueuePos

Description

Construye una cola dinamica basada en un "Vector" (libreria de C++).

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

CppQueuePos::CppQueuePos

```
CppQueuePos();
```

Description

Construye una cola dinamica basada en un "Vector" (libreria de C++).

1.1.3 CppStackPos

```
template <class TObj> class CppStackPos : public StackPos<TObj>;
```

Class Hierarchy

StackPos (see page 29)
  CppStackPos (see page 4)

File

allstackqueuepos.h (see page 33)

CppStackPos Members

Methods

CppStackPos

Description

Construye una pila dinamica basada en un "Vector" (libreria de C++).

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

CppStackPos::CppStackPos

```
CppStackPos();
```

Description

Construye una pila dinamica basada en un "Vector" (libreria de C++).

1.1.4 DLinkListPos

```
template <class TObj> class DLinkListPos : public ListPos<TObj>;
```

Class Hierarchy

ListPos (see page 23)
  DLinkListPos (see page 4)

File

dlinklistpos.h (see page 34)

DLinkListPos Members

Data Members

first
itemCount
last

Methods

~DLinkListPos
addFirst
addLast
DLinkListPos
find
get
getFirst
getFirstNode
getLast
getLeftNode
getRightNode
isEmpty
isFull
print
remove
removeAllElements
removeFirst
removeLast
size

Friends

ostream & operator<<(ostream &outStr, const DLinkNodePos<TObj> &list)

Legend

protected

Description

public muestra miembros

DLinkListPos::first

```
DLinkNodePos<TObj> * first;
```

Description

Primer nodo de la lista. Si es igual a null, esta vacia.

DLinkListPos::itemCount

```
int itemCount;
```

Description

Numero de nodos.

DLinkListPos::last

```
DLinkNodePos<TObj> * last;
```

Description

Ultimo nodo de la lista. Si es igual a null, esta vacia.

DLinkListPos::~DLinkListPos

```
~DLinkListPos();
```

Description

Destructor. Libera memoria.

Destructor. Libera nodos.

DLinkListPos::addFirst

```
void addFirst(const TObj item);
```

Parameters

`const TObj item`

Elemento.

Description

Inserta un elemento al inicio de la lista.

Exceptions

FullListExceptionPos (see page 20) No hay capacidad de almacenamiento.

DLinkListPos::addLast

```
void addLast(const TObj item);
```

Parameters

`const TObj item`

Elemento.

Description

Inserta un elemento al final de la lista.

Exceptions

FullListExceptionPos (see page 20) No hay capacidad de almacenamiento.

DLinkListPos::DLinkListPos

```
DLinkListPos();
```

Description

Constructor sin elementos

Constructor. Inicializa a "first (see DLinkListPos::first, page 5)" y "last (see DLinkListPos::last, page 5)".

DLinkListPos::find

```
DLinkNodePos<TObj> * find(const TObj item) const;
```

Parameters

`const TObj item`

Objeto a comparar.

Return Value

un apuntador al nodo donde se encontro. NULL si no se encontro

Description

Busca la primer ocurrencia en que el objeto es igual.

DLinkListPos::get

```
TObj get(const DLinkNodePos<TObj> * node) const;
```

Return Value

item Elemento revisado.

Description

Regresa el contenido del elemento actual de la lista sin eliminarlo.

Exceptions

EmptyListExceptionPos (see page 15) La lista esta vacia.

DLinkListPos::getFirst

```
TObj getFirst() const;
```

Return Value

item Elemento revisado.

Description

Solo de DLinkListPos (see DLinkListPos::DLinkListPos, page 6)

Regresa el contenido del elemento al inicio de la lista sin eliminarlo.

Exceptions

EmptyListExceptionPos (see page 15) La lista esta vacia.

DLinkListPos::getFirstNode

```
DLinkNodePos<TObj> * getFirstNode() const;
```

Return Value

Primer DLinkNodePos (see page 9).

Description

Regresa el primer nodo de la lista sin eliminarlo.

Exceptions

EmptyListExceptionPos (see page 15) La lista esta vacia.

DLinkListPos::getLast

```
TObj getLast();
```

Return Value

item Elemento revisado.

Description

Regresa el contenido del elemento al final de la lista sin eliminarlo.

Exceptions

EmptyListExceptionPos (see page 15) La lista esta vacia.

DLinkListPos::getLeftNode

```
DLinkNodePos<TObj> * getLeftNode(const DLinkNodePos<TObj> * node) const;
```

Description

usar protected en first (see DLinkListPos::first, page 5) y last (see DLinkListPos::last, page 5) DLinkNodePos (see page 9)<TObj> * getLastNode() const throw(EmptyListExceptionPos (see page 15)); void setFirstNode(DLinkNodePos (see page 9)<TObj> * node); void setLastNode(DLinkNodePos (see page 9)<TObj> * node);

template <class TObj> DLinkNodePos (see page 9)<TObj> * DLinkListPos (see DLinkListPos::DLinkListPos, page 6)<TObj> :: getLastNode() const throw(EmptyListExceptionPos (see page 15)) { if (isEmpty (see DLinkListPos::isEmpty, page 7)()) { throw EmptyListExceptionPos (see page 15)(); } return last (see DLinkListPos::last, page 5); }

/**

DLinkListPos::getRightNode

```
DLinkNodePos<TObj> * getRightNode(const DLinkNodePos<TObj> * node) const;
```

Parameters

```
const DLinkNodePos<TObj> * node
```

Nodo del que se tomara el siguiente (derecha).

Return Value

DLinkNodePos (see page 9) con el nodo siguiente.

Description

Regresa el contenido del siguiente elemento de la lista sin eliminarlo (derecha).

Exceptions

EmptyListExceptionPos (see page 15) La lista esta vacia. El nodo es NULL.

DLinkListPos::isEmpty

```
bool isEmpty() const;
```

Return Value

true No hay elementos en la lista

Description

Checa si la lista esta vacia.

DLinkListPos::isFull

```
bool isFull() const;
```

Return Value

true Se alcanzo el limite maximo.

Description

Checa si la lista esta llena. En teoria, la lista puede crecer indefinidamente, pero para conservar la consistencia con "size (see

DLinkListPos::size, page 8)()" se limita al maximo soportado por un entero.

DLinkListPos::print

```
void print() const;
```

Description

Regresa en un renglon a los elementos de la lista. En orden ascendente (del primero al ultimo) usando el metodo "TObj. toString()" y separado por espacios.

DLinkListPos::remove

```
TObj remove(DLinkNodePos<TObj> * node);
```

Return Value

item Elemento retirado.

Description

Elimina el nodo. Se asume que el nodo esta en la lista.

Exceptions

EmptyListExceptionPos (see page 15) La lista esta vacia, o el nodo es NULL.

DLinkListPos::removeAllElements

```
void removeAllElements();
```

Description

Elimina todos los elementos de la lista. Libera memoria. No lanza "Exception".

DLinkListPos::removeFirst

```
TObj removeFirst();
```

Return Value

item Elemento retirado.

Description

Toma el elemento al inicio de la lista y lo elimina. Libera memoria.

Exceptions

EmptyListExceptionPos (see page 15) La lista esta vacia.

DLinkListPos::removeLast

```
TObj removeLast();
```

Return Value

item Elemento retirado.

Description

Toma el elemento al final de la lista y lo elimina. Libera memoria.

Exceptions

EmptyListExceptionPos (see page 15) La lista esta vacia.

DLinkListPos::size

```
int size() const;
```

Return Value

Numero entero de elementos.

Description

Cuenta el numero de elementos en la lista.

friend ostream & operator<<(ostream &outStr, const DLinkNodePos<TObj> &list)

```
friend ostream & operator<<(ostream &outStr, const DLinkNodePos<TObj> &list);
```

Description

The text for this friend has been generated automatically. This means that it is not documented.

1.1.5 DLinkNodePos

```
template <class TObj> class DLinkNodePos;
```

Class Hierarchy

DLinkNodePos (see page 9)

File

dlinknodepos.h (see page 34)

DLinkNodePos Members

Data Members

item
left
right

Methods

DLinkNodePos
getItem
getLeft
getRight
setLeft
setRight

Legend

private

Description

Almacena al nodo de la lista dinamica doblemente ligada.

Author

Omar Posada Villarreal

Version

2.0, 02/01/2002 C++ 1.0, 26/10/2001

DLinkNodePos::item

```
TObj item;
```

Description

Contiene la informacion del nodo.

DLinkNodePos::left

```
DLinkNodePos<TObj> * left;
```

Description

Apunta al nodo a la izquierda.

DLinkNodePos::right

```
DLinkNodePos<TObj> * right;
```

Description

Apunta al nodo a la derecha.

DLinkNodePos::DLinkNodePos

```
DLinkNodePos(const TObj element);
```

Description

The text for this method has been generated automatically. This means that it is not documented.

DLinkNodePos::getItem

```
TObj getItem() const;
```

Return Value

Contenido del objeto.

Description

No usa. Destructor predeterminado

Regresa el contenido del nodo actual.

DLinkNodePos::getLeft

```
DLinkNodePos<TObj> * getLeft() const;
```

Return Value

Regresa el nodo anterior. null: Si no hay nodo.

Description

Devuelve la referencia al nodo anterior al actual (izquierda).

DLinkNodePos::getRight

```
DLinkNodePos<TObj> * getRight() const;
```

Return Value

Regresa el siguiente nodo. null: Si no hay nodo.

Description

Devuelve la referencia al siguiente nodo del actual (derecha).

DLinkNodePos::setLeft

```
void setLeft(DLinkNodePos<TObj> * theLeft);
```

Parameters

`DLinkNodePos<TObj> * theLeft`

Nuevo apuntador.

Description

Cambia el apuntador al elemento anterior (izquierda). Modifica el apuntador al nodo anterior al actual.

DLinkNodePos::setRight

```
void setRight(DLinkNodePos<TObj> * theRight);
```

Parameters

`DLinkNodePos<TObj> * theRight`

Nuevo apuntador.

Description

Cambia el apuntador al siguiente elemento (derecha). Modifica el apuntador al nodo que le sigue al actual.

1.1.6 DynamicListPos

```
template <class TObj> class DynamicListPos : public ListPos<TObj>;
```

Class Hierarchy

ListPos (see page 23)
  DynamicListPos (see page 10)

File

dynamiclistpos.h (see page 35)

DynamicListPos Members

Data Members

first
last

Methods

~DynamicListPos
addFirst
addLast
DynamicListPos
get
getFirst
getFirstNode
getLast
getNextNode
isEmpty
isFull
print
removeAllElements
removeFirst
removeLast
size

Legend

private

Description

public muestra miembros error friend ostream& operator<<(ostream &outStr, const DynamicListPos (see DynamicListPos:: DynamicListPos, page 12)<TObj> &list); // ???const <TObj>

DynamicListPos::first

```
ListNodePos<TObj> * first;
```

Description

Primer nodo de la lista. Si es igual a null, esta vacia.

DynamicListPos::last

```
ListNodePos<TObj> * last;
```

Description

Ultimo nodo de la lista. Si es igual a null, esta vacia.

DynamicListPos::~DynamicListPos

```
~DynamicListPos();
```

Description

Destructor. Libera memoria.

Destructor. Libera nodos.

DynamicListPos::addFirst

```
void addFirst(const TObj item);
```

Parameters

`const TObj item`

Elemento. @throw FullListExceptionPos (see page 20) No hay capacidad de almacenamiento.

Description

Inserta un elemento al inicio de la lista.

DynamicListPos::addLast

```
void addLast(const TObj item);
```

Parameters

`const TObj item`

Elemento. @throw FullListExceptionPos (see page 20) No hay capacidad de almacenamiento.

Description

Inserta un elemento al final de la lista.

DynamicListPos::DynamicListPos

```
DynamicListPos();
```

Description

Constructor sin elementos

Constructor. Inicializa a "first (see DynamicListPos::first, page 11)" y "last (see DynamicListPos::last, page 11)".

DynamicListPos::get

```
TObj get(const ListNodePos<TObj> * node);
```

Return Value

item Elemento revisado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Regresa el contenido del elemento actual de la lista sin eliminarlo.

DynamicListPos::getFirst

```
TObj getFirst();
```

Return Value

item Elemento revisado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Regresa el contenido del elemento al inicio de la lista sin eliminarlo.

DynamicListPos::getFirstNode

```
ListNodePos<TObj> * getFirstNode();
```

Return Value

Primer ListNodePos (see page 22). @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Regresa el primer nodo de la lista sin eliminarlo.

DynamicListPos::getLast

```
TObj getLast();
```

Return Value

item Elemento revisado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Regresa el contenido del elemento al final de la lista sin eliminarlo.

DynamicListPos::getNextNode

```
ListNodePos<TObj> * getNextNode(const ListNodePos<TObj> * node);
```

Parameters

`const ListNodePos<TObj> * node`

Nodo del que se tomara el siguiente.

Return Value

ListNodePos (see page 22) con el nodo siguiente. @throw EmptyListExceptionPos (see page 15) La lista esta vacia. El nodo es NULL.

Description

Regresa el contenido del siguiente elemento de la lista sin eliminarlo.

DynamicListPos::isEmpty

```
bool isEmpty() const;
```

Return Value

true No hay elementos en la lista

Description

Checa si la lista esta vacia.

DynamicListPos::isFull

```
bool isFull() const;
```

Return Value

true Se alcanzo el limite maximo.

Description

Checa si la lista esta llena. En teoria, la lista puede crecer indefinidamente, pero para conservar la consistencia con "size (see DynamicListPos::size, page 13)()" se limita al maximo soportado por un entero.

DynamicListPos::print

```
void print() const;
```

Return Value

Cadena con los elementos

Description

DynamicListPos::removeAllElements

```
void removeAllElements();
```

Description

Elimina todos los elementos de la lista. Libera memoria. No lanza "Exception".

DynamicListPos::removeFirst

```
TObj removeFirst();
```

Return Value

item Elemento retirado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Toma el elemento al inicio de la lista y lo elimina. Libera memoria.

DynamicListPos::removeLast

```
TObj removeLast();
```

Return Value

item Elemento retirado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Toma el elemento al final de la lista y lo elimina. Libera memoria.

DynamicListPos::size

```
int size() const;
```

Return Value

Numero entero de elementos.

Description

Cuenta el numero de elementos en la lista.

1.1.7 DynamicQueuePos

```
template <class TObj> class DynamicQueuePos : public QueuePos<TObj>;
```

Class Hierarchy

QueuePos (see page 25)
  DynamicQueuePos (see page 14)

File

allstackqueuepos.h (see page 33)

DynamicQueuePos Members

Methods

DynamicQueuePos

Description

Construye una cola que puede crecer basada en una lista dinamica.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

DynamicQueuePos::DynamicQueuePos

```
DynamicQueuePos();
```

Description

Construye una cola que puede crecer basada en una lista dinamica.

1.1.8 DynamicStackPos

```
template <class TObj> class DynamicStackPos : public StackPos<TObj>;
```

Class Hierarchy

StackPos (see page 29)
  DynamicStackPos (see page 14)

File

allstackqueuepos.h (see page 33)

DynamicStackPos Members

Methods

DynamicStackPos

Description

Construye una pila que puede crecer basada en una lista dinamica.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

DynamicStackPos::DynamicStackPos

```
DynamicStackPos();
```

Description

Construye una pila que puede crecer basada en una lista dinamica.

1.1.9 EmptyListExceptionPos

```
class EmptyListExceptionPos : public ListExceptionPos;
```

Class Hierarchy

ListExceptionPos (see page 21)
  EmptyListExceptionPos (see page 15)

File

listexceptionpos.h (see page 36)

EmptyListExceptionPos Members

Data Members

message

Methods

EmptyListExceptionPos(); EmptyListExceptionPos(string theMessage);
what

Legend

private

Description

Cuando la lista esta vacia.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

EmptyListExceptionPos::message

```
const string message;
```

Description

No se pudo heredar modificacion

EmptyListExceptionPos::EmptyListExceptionPos

```
EmptyListExceptionPos();
```

Description

Construye una excepcion para lista vacia.

EmptyListExceptionPos::EmptyListExceptionPos

```
EmptyListExceptionPos(string theMessage);
```

Parameters

`string theMessage`

Mensaje a mostrar.

Description

Construye una excepcion para lista vacia.

EmptyListExceptionPos::what

```
const string what() const;
```

Description

Regresa el mensaje especifico.

1.1.10 FixedListPos

```
template <class TObj> class FixedListPos : public ListPos<TObj>;
```

Class Hierarchy

ListPos (see page 23)
FixedListPos (see page 15)

File

fixedlistpos.h (see page 35)

FixedListPos Members

Data Members

capacity
first
last
list

Methods

~FixedListPos
addFirst
addLast
FixedListPos
getLast
isEmpty
isFull
print
removeAllElements
removeFirst
removeLast
size

Legend

private

Description

public muestra miembros error friend ostream& operator<<(ostream &outStr, const FixedListPos (see FixedListPos:: FixedListPos, page 17)<TObj> &list (see FixedListPos::list, page 16)); // ???const <TObj>

FixedListPos::capacity

```
int capacity;
```

Description

Taman~o maximo del arreglo.

FixedListPos::first

```
int first;
```

Description

Apunta al primer elemento. Si first == -1, la lista esta vacia.

FixedListPos::last

```
int last;
```

Description

Apunta al ultimo elemento. Si first (see FixedListPos::first, page 16) == -1, la lista esta vacia.

FixedListPos::list

```
TObj * list;
```

Description

Conjunto de elementos.

FixedListPos::~FixedListPos

```
~FixedListPos();
```

Description

Destructor. Libera memoria.

Destructor. Libera nodos.

FixedListPos::addFirst

```
void addFirst(const TObj item);
```

Parameters

`const TObj item`

Elemento. @throw FullListExceptionPos (see page 20) No hay capacidad de almacenamiento.

Description

Inserta un elemento al inicio de la lista.

FixedListPos::addLast

```
void addLast(const TObj item);
```

Parameters

`const TObj item`

Elemento. @throw FullListExceptionPos (see page 20) No hay capacidad de almacenamiento.

Description

Inserta un elemento al final de la lista.

FixedListPos::FixedListPos

```
FixedListPos(int capacity = INT_MAX_VALUE);
```

Parameters

`int capacity = INT_MAX_VALUE`

Numero de elementos.

Description

Constructor sin elementos, capacidad predeterminada

Construye una lista estatica de taman~o fijo. Para saber el taman~o predeterminado ver ListPos.INT_MAX_VALUE (see page 32).

See Also

INT_MAX_VALUE (see page 32)

FixedListPos::getLast

```
TObj getLast();
```

Return Value

item Elemento revisado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Regresa el contenido del elemento al final de la lista sin eliminarlo.

FixedListPos::isEmpty

```
bool isEmpty() const;
```

Return Value

true No hay elementos en la lista

Description

Checa si la lista esta vacia.

FixedListPos::isFull

```
bool isFull() const;
```

Return Value

true Se alcanzo el limite maximo.

Description

Checa si la lista esta llena. En teoria, la lista puede crecer indefinidamente, pero para conservar la consistencia con "size (see FixedListPos::size, page 18)()" se limita al maximo soportado por un entero.

FixedListPos::print

```
void print() const;
```

Return Value

Cadena con los elementos

Description

FixedListPos::removeAllElements

```
void removeAllElements();
```

Description

Elimina todos los elementos de la lista. Libera memoria. No lanza "Exception".

FixedListPos::removeFirst

```
TObj removeFirst();
```

Return Value

item Elemento retirado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Toma el elemento al inicio de la lista y lo elimina. Libera memoria.

FixedListPos::removeLast

```
TObj removeLast();
```

Return Value

item Elemento retirado. @throw EmptyListExceptionPos (see page 15) La lista esta vacia.

Description

Toma el elemento al final de la lista y lo elimina. Libera memoria.

FixedListPos::size

```
int size() const;
```

Return Value

Numero entero de elementos.

Description

Cuenta el numero de elementos en la lista.

1.1.11 FixedQueuePos

```
template <class TObj> class FixedQueuePos : public QueuePos<TObj>;
```

Class Hierarchy

QueuePos (see page 25)
  FixedQueuePos (see page 18)

File

allstackqueuepos.h (see page 33)

FixedQueuePos Members

Methods

FixedQueuePos(int itemCount);
FixedQueuePos();

Description

Construye una cola estatica basada en una lista estatica.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

FixedQueuePos::FixedQueuePos

```
FixedQueuePos(int itemCount);
```

Parameters

```
int itemCount
```

Numero fijo de elementos.

Description

Construye una pila estatica basada en una lista estatica.

FixedQueuePos::FixedQueuePos

```
FixedQueuePos();
```

Description

Construye una pila estatica basada en una lista estatica.

1.1.12 FixedStackPos

```
template <class TObj> class FixedStackPos : public StackPos<TObj>;
```

Class Hierarchy

StackPos (see page 29)
  FixedStackPos (see page 19)

File

allstackqueuepos.h (see page 33)

FixedStackPos Members

Methods

FixedStackPos(int itemCount); FixedStackPos();

Description

Construye una pila estatica basada en una lista estatica.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

FixedStackPos::FixedStackPos

```
FixedStackPos(int itemCount);
```

Parameters

```
int itemCount
```

Numero fijo de elementos.

Description

Construye una pila estatica basada en una lista estatica.

FixedStackPos::FixedStackPos

```
FixedStackPos();
```

Description

Construye una pila estatica basada en una lista estatica.

1.1.13 FullListExceptionPos

```
class FullListExceptionPos : public ListExceptionPos;
```

Class Hierarchy

ListExceptionPos (see page 21)
  FullListExceptionPos (see page 20)

File

listexceptionpos.h (see page 36)

FullListExceptionPos Members

Data Members

message

Methods

FullListExceptionPos();
FullListExceptionPos(string theMessage);
what

Legend

private

Description

Cuando la lista esta llena.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

FullListExceptionPos::message

```
const string message;
```

Description

No se pudo heredar modificacion

FullListExceptionPos::FullListExceptionPos

```
FullListExceptionPos();
```

Description

Construye una excepcion para lista llena.

FullListExceptionPos::FullListExceptionPos

```
FullListExceptionPos(string theMessage);
```

Parameters

`string theMessage`

Mensaje a mostrar.

Description

Construye una excepcion para lista llena.

FullListExceptionPos::what

```
const string what() const;
```

Description

Regresa el mensaje especifico.

1.1.14 ListExceptionPos

```
class ListExceptionPos;
```

Class Hierarchy

ListExceptionPos (see page 21)

File

listexceptionpos.h (see page 36)

ListExceptionPos Members

Data Members

message

Methods

ListExceptionPos(); ListExceptionPos(string theMessage);
what

Legend

private

Description

Conjunto de excepciones de las listas.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

ListExceptionPos::message

```
const string message;
```

Description

No se pudo heredar modificacion

ListExceptionPos::ListExceptionPos

```
ListExceptionPos();
```

Description

Construye una excepcion para listas.

ListExceptionPos::ListExceptionPos

```
ListExceptionPos(string theMessage);
```

Parameters

`string theMessage`

Mensaje a mostrar.

Description

Construye una excepcion para listas.

ListExceptionPos::what

```
const string what() const;
```

Description

Regresa el mensaje especifico.

1.1.15 ListNodePos

```
template <class TObj> class ListNodePos;
```

Class Hierarchy

ListNodePos (see page 22)

File

listnodepos.h (see page 36)

ListNodePos Members

Data Members

item

next

Methods

getItem

getNext

ListNodePos(TObj element);

ListNodePos(TObj element, ListNodePos * nextNode);

setNext

Legend

private

Description

Almacena al nodo de la lista dinamica.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

ListNodePos::item

```
TObj item;
```

Description

Contiene la informacion del nodo. Friendly para poder ser accedido por "DynamicListPos (see page 10)".

ListNodePos::next

```
ListNodePos<TObj> * next;
```

Description

Apunta al siguiente nodo. Friendly para poder ser accedido por "DynamicListPos (see page 10)".

ListNodePos::getItem

```
TObj getItem();
```

Return Value

Contenido del objeto.

Description

No usa. Destructor predeterminado

Regresa el contenido del nodo actual.

ListNodePos::getNext

```
ListNodePos<TObj> * getNext();
```

Return Value

Regresa el siguiente nodo. null: Si no hay nodo.

Description

Devuelve la referencia al nodo que le sigue al actual.

ListNodePos::ListNodePos

```
ListNodePos(TObj element);
```

Parameters

`TObj element`

Dato o item (see ListNodePos::item, page 22) a insertar.

Description

Constructor. Crea un nodo conteniendo a un "TObj". Usado cuando la lista esta vacia.

ListNodePos::ListNodePos

```
ListNodePos(TObj element, ListNodePos * nextNode);
```

Parameters

`TObj element`

Dato o item (see ListNodePos::item, page 22) a insertar.

`ListNodePos * nextNode`

Nodo para conectarlo antes.

Description

Constructor. Liga el nuevo elemento con la lista.

ListNodePos::setNext

```
void setNext(ListNodePos<TObj> * theNext);
```

Parameters

`ListNodePos<TObj> * theNext`

Nuevo apuntador.

Description

Cambia el apuntador al siguiente elemento. Modifica el apuntador al nodo que le sigue al actual.

1.1.16 ListPos

```
template <class TObj> class ListPos;
```

Class Hierarchy

ListPos (see page 23)

File

listpos.h (see page 36)

ListPos Members

Data Members

itemCount

Methods

addFirst
addLast
getLast
isEmpty
isFull
print
removeAllElements
removeFirst
removeLast
size

Legend

abstract protected

Description

Permite implementar metodos uniformes para listas. No hay "cpp" debido a que son virtuales puras.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

ListPos::itemCount

```
unsigned int itemCount;
```

Description

Numero de elementos en la lista.

ListPos::addFirst

```
virtual void addFirst(const TObj item) = 0;
```

Description

Agrega un elemento al inicio de la lista.

ListPos::addLast

```
virtual void addLast(const TObj item) = 0;
```

Description

Agrega un elemento al final de la lista.

ListPos::getLast

```
virtual TObj getLast() = 0;
```

Description

Obtiene el ultimo elemento sin quitarlo de la lista.

ListPos::isEmpty

```
virtual bool isEmpty() = 0 const;
```

Description

bool 0:false, 0!=:true C

ListPos::isFull

```
virtual bool isFull() = 0 const;
```

Description

Checa si se pueden agregar mas elementos.

ListPos::print

```
virtual void print() = 0 const;
```

Description

Muesta la lista en consola.

ListPos::removeAllElements

```
virtual void removeAllElements() = 0;
```

Description

Elimina todos los elementos de la lista.

ListPos::removeFirst

```
virtual TObj removeFirst() = 0;
```

Description

Elimina el elemento al inicio de la lista.

ListPos::removeLast

```
virtual TObj removeLast() = 0;
```

Description

Elimina el elemento al final de la lista.

ListPos::size

```
virtual int size() = 0 const;
```

Description

0 virtual pura

1.1.17 QueuePos

```
template <class TObj> class QueuePos;
```

Class Hierarchy

QueuePos (see page 25)

File

queuepos.h (see page 37)

QueuePos Members

Data Members

pList

Methods

dequeue	isEmpty
isFull	print
queue	QueuePos
removeAllElements	size

Legend

private

Description

Implementa los metodos comunes a las colas. Representacion "LIFO", ultimo en entrar, primero en salir ("Last in, first out"). Usa las implementaciones de "ListPos (see page 23)".

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

QueuePos::pList

```
ListPos<TObj> * pList;
```

Description

The text for this field has been generated automatically. This means that it is not documented.

QueuePos::dequeue

```
TObj dequeue();
```

Return Value

item Elemento retirado.

Description

Retira del inicio de la cola. Desencola el primer elemento. Uso:

itemString = (String)myQueue.dequeue();

Exceptions

EmptyListExceptionPos (see page 15) La fila esta vacia.

QueuePos::isEmpty

```
bool isEmpty() const;
```

Return Value

true No hay elementos en la cola

Description

Checa si la cola esta vacia.

QueuePos::isFull

```
bool isFull() const;
```

Return Value

true Se alcanzo el limite maximo.

Description

Checa si la cola esta llena.

QueuePos::print

```
void print() const;
```

Description

Regresa en un renglon a los elementos de la lista. En orden ascendente (del primero al ultimo) usando el metodo "Object.print()" y separado por espacios.

QueuePos::queue

```
void queue(const TObj item);
```

Description

The text for this method has been generated automatically. This means that it is not documented.

QueuePos::QueuePos

```
QueuePos(ListPos<TObj> * pTheList);
```

Parameters

`ListPos<TObj> * pTheList`

Lista concreta de un cierto tipo.

Description

Inicializa el tipo de lista Destructor predeterminado. La lista libera memoria

constructor, ~, friend----------

Instancia o crea la lista a manejar. Uso: pTheList = new DynamicListPos (see page 10)();

QueuePos::removeAllElements

```
void removeAllElements();
```

Description

Elimina todos los elementos de la cola. No lanza "Exception".

QueuePos::size

```
int size() const;
```

Return Value

Numero entero de elementos.

Description

public----------

Cuenta el numero de elementos en la cola.

1.1.18 SetPos

```
template <class TObj> class SetPos : public DLinkListPos<TObj>;
```

Class Hierarchy

ListPos (see page 23)
DLinkListPos (see page 4)
SetPos (see page 27)

File

setpos.h (see page 38)

SetPos Members

Methods

difference
intersection
removeDuplicates
SetPos(const SetPos<TObj> & theSet);
setSet
input
print
SetPos(TObj theSet[], int count);
SetPos();
unite

Description

Implementa el TDA de conjuntos con operaciones basicas. El conjunto es una lista para las operaciones de referencia y de insercion/eliminacion de una lista.

Author

Omar Posada Villarreal

Version

2.0, 03/01/2002 C++ 1.0, 28/10/2001

SetPos::difference

```
SetPos<TObj> * difference(const SetPos<TObj> & setB);
```

Parameters

`const SetPos<TObj> & setB`

Conjunto a quitar del actual.

Return Value

Conjunto diferencias.

Description

Elementos que pertenecen tanto al actual pero no a setB.

SetPos::input

```
int input();
```

Parameters

`name`

Nombre del conjunto.

Return Value

Numero de elementos.

Description

Destructor predeterminado de la lista

Entrada de conjuntos.

SetPos::intersection

```
SetPos<TObj> * intersection(const SetPos<TObj> & setB);
```

Parameters

```
const SetPos<TObj> & setB
```

Conjunto a unir con el actual.

Return Value

conjunto interseccion.

Description

Elementos que pertenecen tanto al actual como a setB.

SetPos::print

```
void print();
```

Description

Salida en un renglon a los elementos del conjunto. Formato: { hola, mundo }.

SetPos::removeDuplicates

```
int removeDuplicates();
```

Return Value

Numero de repeticiones eliminadas.

Description

Eliminar repeticiones de contenidos.

SetPos::SetPos

```
SetPos(TObj theSet[], int count);
```

Parameters

```
int count
```

Numero de elementos.

```
int count
```

Representacion del conjunto.

Description

constructor, ~, friend----------

Construye un conjunto a partir de un arreglo. NULL es el conjunto vacio. Se puede tener un conjunto con un elemento nulo. Del contador se encarga la lista.

Exceptions

FullListExceptionPos (see page 20) No hay capacidad de almacenamiento.

SetPos::SetPos

```
SetPos(const SetPos<TObj> & theSet);
```

Parameters

```
const SetPos<TObj> & theSet
```

Conjunto del que se copiaria la informacion.

Description

Construye un conjunto copiando los elementos de otro. NULL es el conjunto vacio.

SetPos::SetPos

```
SetPos();
```

Description

Construye un conjunto sin elementos. NULL es el conjunto vacio.

SetPos::setSet

```
int setSet(TObj theSet[], int count);
```

Parameters

`TObj theSet[]`

Arreglo de objetos.

Return Value

Numero de repeticiones eliminadas.

Description

Convierte un conjunto de objetos en un conjunto.

SetPos::unite

```
SetPos<TObj> * unite(const SetPos<TObj> & setB);
```

Parameters

`const SetPos<TObj> & setB`

Conjunto a unir con el actual.

Return Value

Conjunto union.

Description

Elementos que pertenecen al conjunto actual, a setB o a ambos conjuntos. Agrega los elementos de setB al final.

1.1.19 StackPos

```
template <class TObj> class StackPos;
```

Class Hierarchy

StackPos (see page 29)

File

stackpos.h (see page 38)

StackPos Members

Data Members

pList

Methods

isEmpty	isFull
peek	pop
print	push
removeAllElements	size
StackPos	

Legend

private

Description

Implementa los metodos comunes a las pilas. Representacion "FIFO", primero en entrar, primero en salir ("First in, first out"). Usa las implementaciones de "ListPos (see page 23)".

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

StackPos::pList

```
ListPos<TObj> * pList;
```

Description

The text for this field has been generated automatically. This means that it is not documented.

StackPos::isEmpty

```
bool isEmpty() const;
```

Return Value

true No hay elementos en la cola

Description

Checa si la cola esta vacia.

StackPos::isFull

```
bool isFull() const;
```

Return Value

true Se alcanzo el limite maximo.

Description

Checa si la cola esta llena.

StackPos::peek

```
TObj peek();
```

Return Value

Elemento a revisar.

Description

Regresa el elemento del tope de la pila sin retirarlo.

Exceptions

EmptyListExceptionPos (see page 15) No hay elementos para revisar.

StackPos::pop

```
TObj pop();
```

Return Value

item Elemento retirado.

Description

Regresa el elemento en el tope de la pila. Uso:

itemString = (String)myStack.pop();

Exceptions

EmptyListExceptionPos (see page 15) No hay elementos para retirar.

StackPos::print

```
void print() const;
```

Description

Regresa en un renglon a los elementos de la lista. En orden ascendente (del primero al ultimo) usando el metodo "Object.print()" y separado por espacios.

StackPos::push

```
void push(const TObj item);
```

Description

The text for this method has been generated automatically. This means that it is not documented.

StackPos::removeAllElements

```
void removeAllElements();
```

Description

Elimina todos los elementos de la cola. No lanza "Exception".

StackPos::size

```
int size() const;
```

Return Value

Numero entero de elementos.

Description

public----------

Cuenta el numero de elementos en la cola.

StackPos::StackPos

```
StackPos(ListPos<TObj> * pTheList);
```

Parameters

`ListPos<TObj> * pTheList`

Lista concreta de un cierto tipo.

Description

Inicializa el tipo de lista Destructor predeterminado. La lista libera memoria

constructor, ~, friend----------

Instancia o crea la lista a manejar.

1.2 Functions

These are all functions that are contained in this documentation.

1.2.1 getString

```
void getString(const string message, string & theString);
```

File

interfacepos.h (see page 35)

Parameters

`const string message`

Mensaje a mostrar.

`string & theString`

Variable de entrada.

Description

Muestra un mensaje en la siguiente linea y coloca la entrada con espacios (blancos) de hasta 1000 caracteres, en un "string".

1.2.2 stringToCharArray

```
char * stringToCharArray(string s);
```

File

interfacepos.h (see page 35)

Return Value

arreglo de char con el contenido de s.

Description

Funciones para entrada y salida del usuario.

Author

Omar Posada Villarreal

Version

1.0, 08/12/2001

Convierte un string en un arreglo de char.

1.2.3 waitEnter

```
void waitEnter();
```

File

interfacepos.h (see page 35)

Description

Espera el Enter. Detiene la ejecucion del programa

Version

2.0, 04/01/2002

1.3 Variables

These are all variables that are contained in this documentation.

1.3.1 INT_MAX_VALUE

```
const int INT_MAX_VALUE = 32000;
```

File

listpos.h (see page 36)

Description

Maximo numero de elementos.

1.3.2 main

```
int main;
```

File

prog18.cpp (see page 37)

Description

Prueba la lista doblemente ligada para la representacion de Conjuntos.

Author

Omar Posada Villarreal

Version

1.0, 02/01/2002 C

1.3.3 SEPARATOR

```
const char SEPARATOR = ' ';
```

File

listpos.h (see page 36)

Description

Separador para "print()".

1.4 Files

These are all files that are contained in this documentation.

1.4.1 allstackqueuepos.h

File Overview

Classes in File allstackqueuepos.h

CppQueuePos (see page 3)
DynamicQueuePos (see page 14)
FixedQueuePos (see page 18)
CppStackPos (see page 4)
DynamicStackPos (see page 14)
FixedStackPos (see page 19)

@(#) listpos/allstackqueuepos.h

NOTA: Para un mejor mantenimiento se opto por tener todas las clases juntas. No se tiene archivo allstackqueuepos.cpp porque el cuerpo de los constructores es de una linea.

1.4.2 cpplistpos.cpp

@(#) listpos/cpplistpos.cpp

Implementa una lista basandose en <vector>.

Author

Omar Posada Villarreal

Version

1.2, 10/12/2001 C++ 1.1, 15/10/2001 getFirst(), getNext() 1.0, 03/10/2001

1.4.3 cpplistpos.h

File Overview

Classes in File cpplistpos.h

CppListPos (see page 1)

@(#) listpos/cpplistpos.h

Implementa una lista basandose en <vector>.

Author

Omar Posada Villarreal

Version

2.1, 10/12/2001 C++ getLast() 2.0, 10/12/2001 C++ 1.1, 15/10/2001 getFirst(), getNext() 1.0, 03/10/2001

1.4.4 dlinklistpos.cpp

@(#) listpos/dlinklistpos.cpp

Implementa la lista doblemente ligada con almacenamiento dinamico.

Author

Omar Posada Villarreal

Version

2.0, 02/01/2002 C++ 1.0, 26/10/2001

1.4.5 dlinklistpos.h

File Overview

Classes in File dlinklistpos.h

DLinkListPos (see page 4)

@(#) listpos/dlinklistpos.h

Implementa la lista doblemente ligada con almacenamiento dinamico.

Author

Omar Posada Villarreal

Version

2.0, 02/01/2002 C++ 1.0, 26/10/2001

1.4.6 dlinknodepos.cpp

@(#) listpos/dlinknodepos.cpp

Almacena al nodo de la lista dinamica doblemente ligada.

Author

Omar Posada Villarreal

Version

2.0, 02/01/2002 C++ 1.0, 26/10/2001

1.4.7 dlinknodepos.h

File Overview

Classes in File dlinknodepos.h

DLinkNodePos (see page 9)

@(#) listpos/dlinknodepos.h

Almacena al nodo de la lista dinamica doblemente ligada.

Author

Omar Posada Villarreal

Version

2.0, 02/01/2002 C++ 1.0, 26/10/2001

1.4.8 dynamiclistpos.cpp

@(#) listpos/dynamiclistpos.cpp

Implementa la lista con almacenamiento dinamico.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.1, 15/10/2001 getFirst(), getNext() 1.0, 03/10/2001

1.4.9 dynamiclistpos.h

File Overview

Classes in File dynamiclistpos.h

DynamicListPos (see page 10)

@(#) listpos/dynamiclistpos.h

Implementa la lista con almacenamiento dinamico.

Author

Omar Posada Villarreal

Version

1.2, 10/12/2001 C++ 1.1, 15/10/2001 getFirst(), getNext() 1.0, 03/10/2001

1.4.10 fixedlistpos.cpp

@(#) listpos/fixedlistpos.cpp

Implementa una lista circular con arreglos de "Object" estatica.

Author

Omar Posada Villarreal

Version

1.2, 10/12/2001 C++ 1.1, 15/10/2001 getFirst(), getNext() 1.0, 03/10/2001

1.4.11 fixedlistpos.h

File Overview

Classes in File fixedlistpos.h

FixedListPos (see page 15)

@(#) listpos/fixedlistpos.h

Implementa una lista circular con arreglos de "Object" estatica.

Author

Omar Posada Villarreal

Version

1.2, 10/12/2001 C++ 1.1, 15/10/2001 getFirst(), getNext() 1.0, 03/10/2001

1.4.12 interfacepos.cpp

@(#) utilpos/interfacepos.cpp

Funciones para entrada y salida del usuario.

Author

Omar Posada Villarreal

Version

1.0, 08/12/2001

1.4.13 interfacepos.h

File Overview

Functions in File interfacepos.h

getString (see page 31)
stringToCharArray (see page 32)
waitEnter (see page 32)

@(#) utilpos/interfacepos.h

Funciones para entrada y salida del usuario.

Author

Omar Posada Villarreal

Version

1.1, 27/12/2001 getString (see page 31)() 1.0, 08/12/2001

1.4.14 listexceptionpos.h

File Overview

Classes in File listexceptionpos.h

EmptyListExceptionPos (see page 15) FullListExceptionPos (see page 20)
ListExceptionPos (see page 21)

@(#) listpos/listexceptionpos.h

Conjunto de excepciones de las listas.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C

1.4.15 listnodepos.cpp

@(#) listpos/listnodepos.cpp

Almacena al nodo de la lista dinamica.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

1.4.16 listnodepos.h

File Overview

Classes in File listnodepos.h

ListNodePos (see page 22)

@(#) listpos/listnodepos.h

Almacena al nodo de la lista dinamica.

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

1.4.17 listpos.h

File Overview

Classes in File listpos.h

ListPos (see page 23)

Variables in File listpos.h

INT_MAX_VALUE (see page 32) SEPARATOR (see page 33)

@(#) listpos/listpos.h

Permite implementar metodos uniformes para listas. No hay "cpp" debido a que son virtuales puras.

Author

Omar Posada Villarreal

Version

1.3, 24/12/2001 Comparacion de signed con unsigned 2.0, 10/12/2001 C++ 1.0, 03/10/2001

1.4.18 prog18.cpp

File Overview

Variables in File prog18.cpp

main (see page 32)

@(#) prog18.cpp

Prueba la lista doblemente ligada para la representacion de Conjuntos.

Author

Omar Posada Villarreal

Version

2.0, 02/01/2002 C++ 1.0, 26/10/2001

1.4.19 queuepos.cpp

@(#) listpos/queuepos.cpp

Implementa los metodos comunes a las colas. Representacion "LIFO", ultimo en entrar, primero en salir ("Last in, first out"). Usa las implementaciones de "ListPos (see page 23)".

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

1.4.20 queuepos.h

File Overview

Classes in File queuepos.h

QueuePos (see page 25)

@(#) listpos/queuepos.h

Implementa los metodos comunes a las colas. Representacion "LIFO", ultimo en entrar, primero en salir ("Last in, first out"). Usa las implementaciones de "ListPos (see page 23)".

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

1.4.21 setpos.cpp

@(#) listpos/setpos.cpp

Implementa el TDA de conjuntos con operaciones basicas. El conjunto es una lista para las operaciones de referencia y de insercion/eliminacion de una lista.

Author

Omar Posada Villarreal

Version

2.0, 03/01/2002 C++ 1.0, 28/10/2001

1.4.22 setpos.h

File Overview

Classes in File setpos.h

SetPos (see page 27)

@(#) listpos/setpos.h

Implementa el TDA de conjuntos con operaciones basicas. El conjunto es una lista para las operaciones de referencia y de insercion/eliminacion de una lista.

Author

Omar Posada Villarreal

Version

2.0, 03/01/2002 C++ 1.0, 28/10/2001

1.4.23 stackpos.cpp

@(#) listpos/stackpos.cpp

Implementa los metodos comunes a las pilas. Representacion "FIFO", primero en entrar, primero en salir ("First in, first out"). Usa las implementaciones de "ListPos (see page 23)".

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

1.4.24 stackpos.h

File Overview

Classes in File stackpos.h

StackPos (see page 29)

@(#) listpos/stackpos.h

Implementa los metodos comunes a las pilas. Representacion "FIFO", primero en entrar, primero en salir ("First in, first out"). Usa las implementaciones de "ListPos (see page 23)".

Author

Omar Posada Villarreal

Version

2.0, 10/12/2001 C++ 1.0, 03/10/2001

1.4.25 test_dlinklistpos.cpp

@(#) prog18.cpp (see page 37)

Prueba la lista doblemente ligada para la representacion de Conjuntos.

Author

Omar Posada Villarreal

Version

2.0, 02/01/2002 C++ 1.0, 26/10/2001

Index

~

A

C

D

I

L

M

N

O

P

Q

R

S

T

U

V

W