Colas (Queue):

Una cola es un tipo especial de lista abierta en la que sólo se pueden insertar nodos en uno de los extremos de la lista y sólo se pueden eliminar nodos en el otro. Además, como sucede con las pilas, las escrituras de datos siempre son inserciones de nodos, y las lecturas siempre eliminan el nodo leído.

Este tipo de lista es conocido como lista FIFO (First In First Out), el primero en entrar es el primero en salir.

El símil cotidiano es una cola para comprar, por ejemplo, las entradas del cine. Los nuevos compradores sólo pueden colocarse al final de la cola, y sólo el primero de la cola puede comprar la entrada.

El nodo típico para construir pilas es el mismo que vimos en los capítulos anteriores para la construcción de listas y pilas:

```
struct nodo {
  int dato;
  struct nodo *siguiente;
};
```

Declaraciones de tipos para manejar colas en C:

Los tipos que definiremos normalmente para manejar colas serán casi los mismos que para manejar listas y pilas, tan sólo cambiaremos algunos nombres:

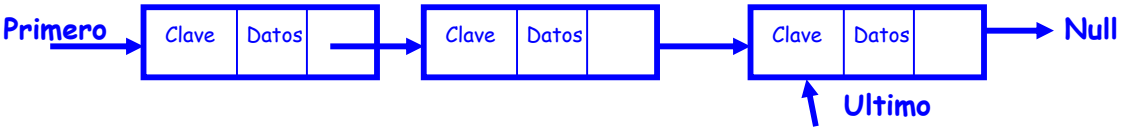
```
typedef struct _nodo {
  int dato;
  struct _nodo *siguiente;
} tipoNodo;

typedef tipoNodo *pNodo;
typedef tipoNodo *primero;
typedef tipoNodo *ultimo;
```

tipoNodo es el tipo para declarar nodos, evidentemente.

pNodo es el tipo para declarar punteros a un nodo.

Cola es el tipo para declarar colas.



Es evidente, a la vista del gráfico, que una cola es una lista abierta. Así que sigue siendo muy importante que nuestro programa nunca pierda el valor del puntero al primer elemento, igual que pasa con las listas abiertas. Además, debido al funcionamiento de las colas, también deberemos mantener un puntero para el último elemento de la cola, que será el punto donde insertemos nuevos nodos.

Teniendo en cuenta que las lecturas y escrituras en una cola se hacen siempre en extremos distintos, lo más fácil será insertar nodos por el final, a continuación del nodo que no tiene nodo siguiente, y leerlos desde el principio, hay que recordar que leer un nodo implica eliminarlo de la cola.

Operaciones básicas con colas:

De nuevo nos encontramos ante una estructura con muy pocas operaciones disponibles. Las colas sólo permiten añadir y leer elementos:

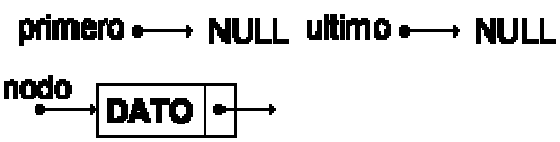
- Añadir: Inserta un elemento al final de la cola.
- Leer: Lee y elimina un elemento del principio de la cola.

ENCOLAR: Añadir un elemento:

Las operaciones con colas son muy sencillas, prácticamente no hay casos especiales, salvo que la cola esté vacía.

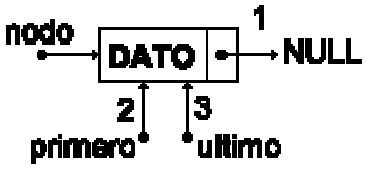
Añadir elemento en una cola vacía:

Partiremos de que ya tenemos el nodo a insertar y, por supuesto un puntero que apunte a él, además los punteros que definen la cola, primero y ultimo que valdrán NULL:



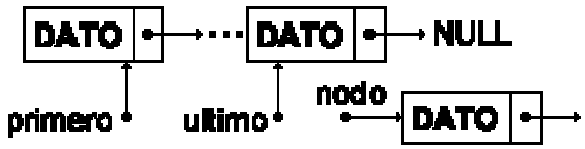
El proceso es muy simple, bastará con que:

1. nodo->siguiente apunte a NULL.
2. Y que los punteros primero y último apunten a nodo.



Añadir elemento en una cola no vacía:

De nuevo partiremos de un nodo a insertar, con un puntero que apunte a él, y de una cola, en este caso, al no estar vacía, los punteros primero y último no serán nulos:



El proceso sigue siendo muy sencillo:

1. Hacemos que nodo->siguiente apunte a NULL.
2. Después que ultimo->siguiente apunte a nodo.
3. Y actualizamos ultimo, haciendo que apunte a nodo.

Añadir elemento en una cola, caso general:

Para generalizar el caso anterior, sólo necesitamos añadir una operación:

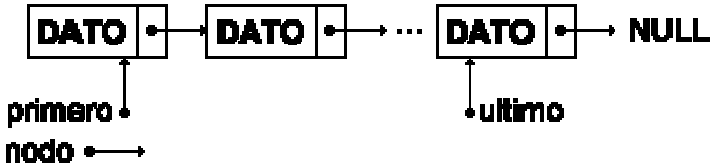
1. Hacemos que nodo->siguiente apunte a NULL.
2. Si ultimo no es NULL, hacemos que ultimo->siguiente apunte a nodo.
3. Y actualizamos ultimo, haciendo que apunte a nodo.
4. Si primero es NULL, significa que la cola estaba vacía, así que haremos que primero apunte también a nodo.

DESENCOLAR: Leer un elemento de una cola, implica eliminarlo:

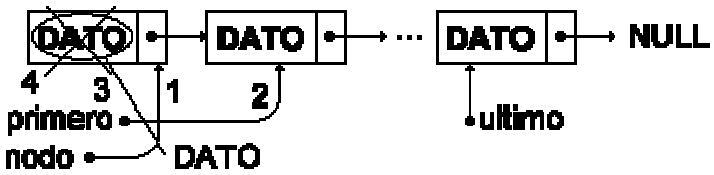
Ahora también existen dos casos, que la cola tenga un solo elemento o que tenga más de uno.

Leer un elemento en una cola con más de un elemento:

Usaremos un puntero a un nodo auxiliar:

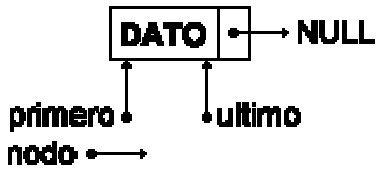


1. Hacemos que nodo apunte al primer elemento de la cola, es decir a primero.
2. Asignamos a primero la dirección del segundo nodo de la pila: primero->siguiente.
3. Guardamos el contenido del nodo para devolverlo como retorno, recuerda que la operación de lectura en colas implican también borrar.
4. Liberamos la memoria asignada al primer nodo, el que queremos eliminar.



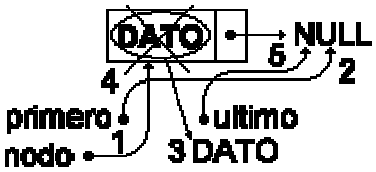
Leer un elemento en una cola con un solo elemento:

También necesitamos un puntero a un nodo auxiliar:



1. Hacemos que nodo apunte al primer elemento de la pila, es decir a primero.
2. Asignamos NULL a primero, que es la dirección del segundo nodo teórico de la cola: primero->siguiente.

3. Guardamos el contenido del nodo para devolverlo como retorno, recuerda que la operación de lectura en colas implican también borrar.
4. Liberamos la memoria asignada al primer nodo, el que queremos eliminar.
5. Hacemos que ultimo apunte a NULL, ya que la lectura ha dejado la cola vacía.



Leer un elemento en una cola caso general:

1. Hacemos que nodo apunte al primer elemento de la pila, es decir a primero.
2. Asignamos a primero la dirección del segundo nodo de la pila: primero→siguiente.
3. Guardamos el contenido del nodo para devolverlo como retorno, recuerda que la operación de lectura en colas implican también borrar.
4. Liberamos la memoria asignada al primer nodo, el que queremos eliminar.
5. Si primero es NULL, hacemos que ultimo también apunte a NULL, ya que la lectura ha dejado la cola vacía.

Ejemplo de cola en C:

Construiremos una cola para almacenar números enteros. Haremos pruebas insertando varios valores y leyéndolos alternativamente para comprobar el resultado.

Algoritmo de la función "Añadir" (ENCOLAR):

1. Creamos un nodo para el valor que colocaremos en la cola.
2. Hacemos que nodo→siguiente apunte a NULL.
3. Si "ultimo" no es NULL, hacemos que ultimo→siguiente apunte a nodo.
4. Actualizamos "ultimo" haciendo que apunte a nodo.
5. Si "primero" es NULL, hacemos que apunte a nodo.

```
void Anadir(pNodo *primero, pNodo *ultimo, int v)
{
  pNodo nuevo;

  /* Crear un nodo nuevo */
  nuevo = (pNodo)malloc(sizeof(tipoNodo));
```

```
  nuevo->valor = v;
  /* Este será el último nodo, no debe tener siguiente */
  nuevo->siguiente = NULL;
  /* Si la cola no estaba vacía, añadimos el nuevo a continuación de ultimo */
  if(*ultimo) (*ultimo)->siguiente = nuevo;
  /* Ahora, el último elemento de la cola es el nuevo nodo */
  *ultimo = nuevo;
  /* Si primero es NULL, la cola estaba vacía, ahora primero apuntará también al nuevo nodo
*/
  if(!*primero) *primero = nuevo;
}
```

Algoritmo de la función "leer" (DESENCOLAR):

1. Hacemos que nodo apunte al primer elemento de la cola, es decir a primero.
2. Asignamos a primero la dirección del segundo nodo de la cola: primero->siguiente.
3. Guardamos el contenido del nodo para devolverlo como retorno, recuerda que la operación de lectura equivale a leer y borrar.
4. Liberamos la memoria asignada al primer nodo, el que queremos eliminar.
5. Si primero es NULL, haremos que último también apunte a NULL, ya que la cola habrá quedado vacía.

```
int Leer(pNodo *primero, pNodo *ultimo)
{
  pNodo nodo; /* variable auxiliar para manipular nodo */
  int v;      /* variable auxiliar para retorno */

  /* Nodo apunta al primer elemento de la pila */
  nodo = *primero;
  if(!nodo) return 0; /* Si no hay nodos en la pila retornamos 0 */
  /* Asignamos a primero la dirección del segundo nodo */
  *primero = nodo->siguiente;
  /* Guardamos el valor de retorno */
  v = nodo->valor;
  /* Borrar el nodo */
  free(nodo);
  /* Si la cola quedó vacía, ultimo debe ser NULL también*/
  if(!*primero) *ultimo = NULL;
  return v;
}
```

Código del ejemplo completo:

Tan sólo nos queda escribir una pequeña prueba para verificar el funcionamiento de las colas:

```
#include <stdlib.h>
#include <stdio.h>

typedef struct _nodo {
  int valor;
  struct _nodo *siguiente;
} tipoNodo;

typedef tipoNodo *pNodo;

/* Funciones con colas: */
void Anadir(pNodo *primero, pNodo *ultimo, int v);
int Leer(pNodo *primero, pNodo *ultimo);

int main()
{
  pNodo primero = NULL, ultimo = NULL;

  Anadir(&primero, &ultimo, 20);
  printf("Añadir(20)\n");
  Anadir(&primero, &ultimo, 10);
  printf("Añadir(10)\n");
  printf("Leer: %d\n", Leer(&primero, &ultimo));
  Anadir(&primero, &ultimo, 40);
  printf("Añadir(40)\n");
  Anadir(&primero, &ultimo, 30);
  printf("Añadir(30)\n");
  printf("Leer: %d\n", Leer(&primero, &ultimo));
  printf("Leer: %d\n", Leer(&primero, &ultimo));
  Anadir(&primero, &ultimo, 90);
  printf("Añadir(90)\n");
  printf("Leer: %d\n", Leer(&primero, &ultimo));
  printf("Leer: %d\n", Leer(&primero, &ultimo));

  system("PAUSE");
  return 0;
}

void Anadir(pNodo *primero, pNodo *ultimo, int v)
{
  pNodo nuevo;
```

```
  /* Crear un nodo nuevo */
  nuevo = (pNodo)malloc(sizeof(tipoNodo));
  nuevo->valor = v;
  /* Este será el último nodo, no debe tener siguiente */
  nuevo->siguiente = NULL;
  /* Si la cola no estaba vacía, añadimos el nuevo a continuación de ultimo */
  if(*ultimo) (*ultimo)->siguiente = nuevo;
  /* Ahora, el último elemento de la cola es el nuevo nodo */
  *ultimo = nuevo;
  /* Si primero es NULL, la cola estaba vacía, ahora primero apuntará también al nuevo nodo
*/
  if(!*primero) *primero = nuevo;
}

int Leer(pNodo *primero, pNodo *ultimo)
{
  pNodo nodo; /* variable auxiliar para manipular nodo */
  int v;      /* variable auxiliar para retorno */

  /* Nodo apunta al primer elemento de la pila */
  nodo = *primero;
  if(!nodo) return 0; /* Si no hay nodos en la pila retornamos 0 */
  /* Asignamos a primero la dirección del segundo nodo */
  *primero = nodo->siguiente;
  /* Guardamos el valor de retorno */
  v = nodo->valor;
  /* Borrar el nodo */
  free(nodo);
  /* Si la cola quedó vacía, ultimo debe ser NULL también*/
  if(!*primero) *ultimo = NULL;
  return v;
}
```

Ejemplo de cola en C++:

Ya hemos visto que las colas son casos particulares de listas abiertas, pero más simples. Como en los casos anteriores, veremos ahora un ejemplo de cola usando clases.

Para empezar, y como siempre, necesitaremos dos clases, una para nodo y otra para cola. Además la clase para nodo debe ser amiga de la clase cola, ya que ésta debe acceder a los miembros privados de nodo.

```
class nodo {
  public:
   nodo(int v, nodo *sig = NULL) {
      valor = v;
      siguiente = sig;
   }

  private:
   int valor;
   nodo *siguiente;

  friend class cola;
};

typedef nodo *pnodo;

class cola {
  public:
   cola() : ultimo(NULL), primero(NULL) {}
   ~cola();

   void Anadir(int v);
   int Leer();

  private:
   pnodo primero, ultimo;
};
```

Los algoritmos para Anadir y Leer son los mismos que expusimos para el ejemplo C, tan sólo cambia el modo de crear y destruir nodos.

Código del ejemplo completo:

```
#include <iostream>
using namespace std;

class nodo {
  public:
   nodo(int v, nodo *sig = NULL) {
      valor = v;
      siguiente = sig;
   }

  private:
   int valor;
   nodo *siguiente;

  friend class cola;
};

typedef nodo *pnodo;

class cola {
  public:
   cola() : ultimo(NULL), primero(NULL) {}
   ~cola();

   void Push(int v);
   int Pop();

  private:
   pnodo ultimo;
};

cola::~cola() {
  while(primero) Leer();
}

void cola::Anadir(int v) {
  pnodo nuevo;

  /* Crear un nodo nuevo */
  nuevo = new nodo(v);
  /* Si la cola no estaba vacía, añadimos el nuevo a continuación de ultimo */
  if(ultimo) ultimo->siguiente = nuevo;
  /* Ahora, el último elemento de la cola es el nuevo nodo */
```

```
  ultimo = nuevo;
  /* Si primero es NULL, la cola estaba vacía, ahora primero apuntará también al nuevo nodo
*/
  if(!primero) primero = nuevo;
}

int cola::Leer() {
  pnodo nodo; /* variable auxiliar para manipular nodo */
  int v;      /* variable auxiliar para retorno */

  /* Nodo apunta al primer elemento de la pila */
  nodo = primero;
  if(!nodo) return 0; /* Si no hay nodos en la pila retornamos 0 */
  /* Asignamos a primero la dirección del segundo nodo */
  primero = nodo->siguiente;
  /* Guardamos el valor de retorno */
  v = nodo->valor;
  /* Borrar el nodo */
  delete nodo;
  /* Si la cola quedó vacía, ultimo debe ser NULL también*/
  if(!primero) ultimo = NULL;
  return v;
}

int main() {
  cola Cola;

  Cola.Anadir(20);
  cout << "Añadir(20)" << endl;
  Cola.Anadir(10);
  cout << "Añadir(10)" << endl;
  cout << "Leer: " << Cola.Leer() << endl;
  Cola.Anadir(40);
  cout << "Añadir(40)" << endl;
  Cola.Anadir(30);
  cout << "Añadir(30)" << endl;
  cout << "Leer: " << Cola.Leer() << endl;
  cout << "Leer: " << Cola.Leer() << endl;
  Cola.Anadir(90);
  cout << "Añadir(90)" << endl;
  cout << "Leer: " << Cola.Leer() << endl;
  cout << "Leer: " << Cola.Leer() << endl;
  cin.get();
  return 0;

}
```

Ejemplo de cola en C++ usando plantillas:

Veremos ahora un ejemplo sencillo usando plantillas. Ya que la estructura para colas es más sencilla que para listas abiertas, nuestro ejemplo también será más simple.

Seguimos necesitando dos clases, una para nodo y otra para cola. Pero ahora podremos usar esas clases para construir listas de cualquier tipo de datos.

Código del un ejemplo completo:

Veremos primero las declaraciones de las dos clases que necesitamos:

```
template<class TIPO> class cola;

template<class TIPO>
class nodo {
  public:
    nodo(TIPO v, nodo<TIPO> *sig = NULL) {
      valor = v;
      siguiente = sig;
    }

  private:
    TIPO valor;
    nodo<TIPO> *siguiente;

    friend class cola<TIPO>;
};

template<class TIPO>
class cola {
  public:
    cola() : primero(NULL), ultimo(NULL) {}
    ~cola();

    void Anadir(TIPO v);
    TIPO Leer();

  private:
    nodo<TIPO> *primero, *ultimo;
};
```

La implementación de las funciones es la misma que para el ejemplo de la página anterior.

```
template<class TIPO>
cola<TIPO>::~cola() {
  while(primero) Leer();
}

template<class TIPO>
void cola<TIPO>::Anadir(TIPO v) {
  nodo<TIPO> *nuevo;

  /* Crear un nodo nuevo */
  /* Este será el último nodo, no debe tener siguiente */
  nuevo = new nodo<tipo>(v);
  /* Si la cola no estaba vacía, añadimos el nuevo a continuación de ultimo */
  if(ultimo) ultimo->siguiente = nuevo;
  /* Ahora, el último elemento de la cola es el nuevo nodo */
  ultimo = nuevo;
  /* Si primero es NULL, la cola estaba vacía, ahora primero apuntará también al nuevo nodo
*/
  if(!primero) primero = nuevo;
}

template<class TIPO>
TIPO cola<TIPO>::Leer() {
   nodo<TIPO> *Nodo; /* variable auxiliar para manipular nodo */
   TIPO v;      /* variable auxiliar para retorno */

  /* Nodo apunta al primer elemento de la pila */
  Nodo = primero;
  if(!Nodo) return 0; /* Si no hay nodos en la pila retornamos 0 */
  /* Asignamos a primero la dirección del segundo nodo */
  primero = Nodo->siguiente;
  /* Guardamos el valor de retorno */
  v = Nodo->valor;
  /* Borrar el nodo */
  delete Nodo;
  /* Si la cola quedó vacía, ultimo debe ser NULL también*/
  if(!primero) ultimo = NULL;
  return v;
}
```

Eso es todo, ya sólo falta usar nuestras clases para un ejemplo práctico:

```
#include <iostream>
#include "CCadena.h"
using namespace std;
```

```
template<class TIPO> class cola;

template<class TIPO>
class nodo {
  public:
    nodo(TIPO v, nodo<TIPO> *sig = NULL) {
      valor = v;
      siguiente = sig;
    }

  private:
    TIPO valor;
    nodo<TIPO> *siguiente;

    friend class cola<TIPO>;
};

template<class TIPO>
class cola {
  public:
    cola() : primero(NULL), ultimo(NULL) {}
    ~cola();

    void Anadir(TIPO v);
    TIPO Leer();

  private:
    nodo<TIPO> *primero, *ultimo;
};

template<class TIPO>
cola<TIPO>::~cola() {
  while(primero) Leer();
}

template<class TIPO>
void cola<TIPO>::Anadir(TIPO v) {
  nodo<TIPO> *nuevo;

  /* Crear un nodo nuevo */
  /* Este será el último nodo, no debe tener siguiente */
  nuevo = new nodo<tipo>(v);
  /* Si la cola no estaba vacía, añadimos el nuevo a continuación de ultimo */
  if(ultimo) ultimo->siguiente = nuevo;
  /* Ahora, el último elemento de la cola es el nuevo nodo */
  ultimo = nuevo;
```

```
  /* Si primero es NULL, la cola estaba vacía, ahora primero apuntará también al nuevo nodo
*/
  if(!primero) primero = nuevo;
}

template<class TIPO>
TIPO cola<TIPO>::Leer() {
   nodo<TIPO> *Nodo; /* variable auxiliar para manipular nodo */
   TIPO v;      /* variable auxiliar para retorno */

  /* Nodo apunta al primer elemento de la pila */
  Nodo = primero;
  if(!Nodo) return 0; /* Si no hay nodos en la pila retornamos 0 */
  /* Asignamos a primero la dirección del segundo nodo */
  primero = Nodo->siguiente;
  /* Guardamos el valor de retorno */
  v = Nodo->valor;
  /* Borrar el nodo */
  delete Nodo;
  /* Si la cola quedó vacía, ultimo debe ser NULL también*/
  if(!primero) ultimo = NULL;
  return v;
}

int main() {
  cola <int> iCola;
  cola <float> fCola;
  cola <double> dCola;
  cola <char> cCola;
  cola <Cadena> sCola;

  // Prueba con <int>
  iCola.Anadir(20);
  cout << "Añadir(20)" << endl;
  iCola.Anadir(10);
  cout << "Añadir(10)" << endl;
  cout << "Leer: " << iCola.Leer() << endl;
  iCola.Anadir(40);
  cout << "Añadir(40)" << endl;
  iCola.Anadir(30);
  cout << "Añadir(30)" << endl;
  cout << "Leer: " << iCola.Leer() << endl;
  cout << "Leer: " << iCola.Leer() << endl;
  iCola.Anadir(90);
  cout << "Añadir(90)" << endl;
  cout << "Leer: " << iCola.Leer() << endl;
```

```
cout << "Leer: " << iCola.Leer() << endl;

// Prueba con <float>
fCola.Anadir(20.01);
cout << "Añadir(20.01)" << endl;
fCola.Anadir(10.02);
cout << "Añadir(10.02)" << endl;
cout << "Leer: " << fCola.Leer() << endl;
fCola.Anadir(40.03);
cout << "Añadir(40.03)" << endl;
fCola.Anadir(30.04);
cout << "Añadir(30.04)" << endl;
cout << "Leer: " << fCola.Leer() << endl;
cout << "Leer: " << fCola.Leer() << endl;
fCola.Anadir(90.05);
cout << "Añadir(90.05)" << endl;
cout << "Leer: " << fCola.Leer() << endl;
cout << "Leer: " << fCola.Leer() << endl;

// Prueba con <double>
dCola.Anadir(0.0020);
cout << "Añadir(0.0020)" << endl;
dCola.Anadir(0.0010);
cout << "Añadir(0.0010)" << endl;
cout << "Leer: " << dCola.Leer() << endl;
dCola.Anadir(0.0040);
cout << "Añadir(0.0040)" << endl;
dCola.Anadir(0.0030);
cout << "Añadir(0.0030)" << endl;
cout << "Leer: " << dCola.Leer() << endl;
cout << "Leer: " << dCola.Leer() << endl;
dCola.Anadir(0.0090);
cout << "Añadir(0.0090)" << endl;
cout << "Leer: " << dCola.Leer() << endl;
cout << "Leer: " << dCola.Leer() << endl;

// Prueba con <char>
cCola.Anadir('x');
cout << "Añadir(\'x\')" << endl;
cCola.Anadir('y');
cout << "Añadir(\'y\')" << endl;
cout << "Leer: " << cCola.Leer() << endl;
cCola.Anadir('a');
cout << "Añadir(\'a\')" << endl;
cCola.Anadir('b');
cout << "Añadir(\'b\')" << endl;
```

```
  cout << "Leer: " << cCola.Leer() << endl;
  cout << "Leer: " << cCola.Leer() << endl;
  cCola.Anadir('m');
  cout << "Añadir(\'m\')" << endl;
  cout << "Leer: " << cCola.Leer() << endl;
  cout << "Leer: " << cCola.Leer() << endl;

  // Prueba con <Cadena>
  sCola.Anadir("Hola");
  cout << "Añadir(\"Hola\")" << endl;
  sCola.Anadir("somos");
  cout << "Añadir(\"somos\")" << endl;
  cout << "Leer: " << sCola.Leer() << endl;
  sCola.Anadir("programadores");
  cout << "Añadir(\"programadores\")" << endl;
  sCola.Anadir("buenos");
  cout << "Añadir(\"buenos\")" << endl;
  cout << "Leer: " << sCola.Leer() << endl;
  cout << "Leer: " << sCola.Leer() << endl;
  sCola.Anadir("!!!!");
  cout << "Añadir(\"!!!!\")" << endl;
  cout << "Leer: " << sCola.Leer() << endl;
  cout << "Leer: " << sCola.Leer() << endl;

  cin.get();
  return 0;
}
```