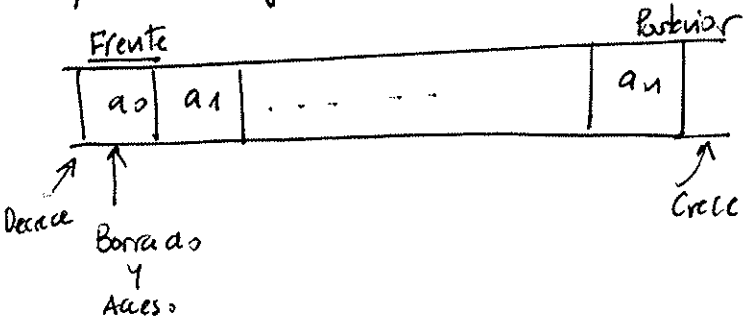
LECCION 9 : COLAS

COLAS : FIFO (First Input First output)

Especificación: Son E.D.L, contenedores, que contienen una secuencia de datos $\{a_0, a_1, \ldots, a_n\}$ especialmente diseñadas por hacer las inserciones por un extremo y borrados y accesos por el otro extremo.

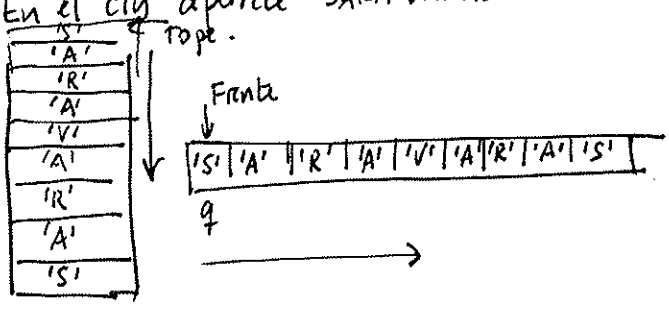


Operaciones

- Frente: accede al elemento en el frente
- Vacia: Devuelve true si la cola está vacia
- Quitar: Elimina el elemento en el frente
- Poner: Inserta un elemento al final de la cola

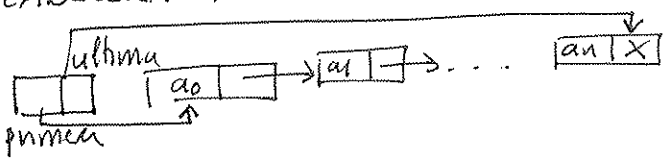
Ver transparencia 2
ejemplo de uso de una cola.

En el cin aparece "SARA VARAS"

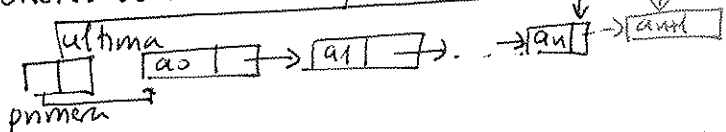


- Si al ir consultando el TOPE en p y el Frente en q coinciden la frase era un PALINDROMO.

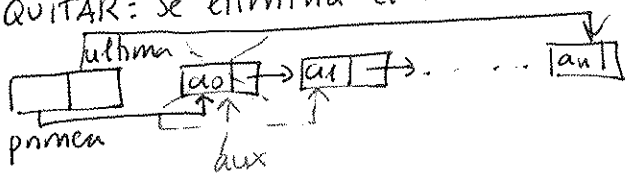
COLA : IMPLEMENTACIÓN CON CELDAS ~~CABECERA~~ Y DOS PUNTEROS



VACIA: primera y ultima son ceros O(1)
FRENTE: El dato al que apunta primera O(1)
PONER: Se inserta despues de ultima O(1)



QUITAR: Se elimina el elemento en el FRENTE



```
#ifndef _COLA_H
#define _COLA_H
template <class T>
class Cola {
  private:
    struct Celda {
      T ele;
      Celda *sig;
      Celda(const T &d, Celda *s)
      {
        ele = d;
        sig = s;
      }
    };

    Celda *primera, *ultima;
    void Copiar(const Cola<T> &C);
    void Borrar();
  public:
    Cola();
    Cola(const Cola<T> &C);
    ~Cola();
    Cola<T> &operator=(const Cola<T> &C);
    T Frente() const;
    void Quitar();
    void Poner(const T &d);
    int Size() const;
    bool Vacia() const;
};
#include "Cola.cpp"
#endif
```

Colas

Esquema de cola

Una posible *clase Cola* para almacenar datos de tipo *char* puede tener la siguiente sintaxis.

```
#ifndef __COLA_H__
# define __COLA_H__

  class Cola{
      ...                    // La implementación que deseamos
    public:
      Cola();
      Cola(const Cola& p);
      ~Cola();
      Cola& operator= (const Cola& p);

      bool vacia() const;
      void poner (char c);
      void quitar();
      char frente() const;
  };

#endif
```

Uso de una cola

```
#include <iostream>
#include <pila.h>
#include <cola.h>
using namespace std;
int main()
{
  Pila p;
  Cola q;
  char dato;

  cout << "Escriba una frase" << endl;
  while ((dato=cin.get())! ='\n')
    if (dato! =' ') {
      p.poner(dato);
      q.poner(dato);
    }
  bool palindromo=true;
  while (!p.vacia() && palindromo) {
    if (p.tope()! =q.frente())
      palindromo= false;
    p.quitar();
    q.quitar();
  }
  if (palindromo)
    cout << "La frase es un palíndromo" << endl;
  else cout << "La frase no es un palíndromo" << endl;
  return 0;
}
```

LECCIÓN 9

```
template <class T>
void Cola<T>::Copiar(const Cola<T> &C){
    if (C.primera==0){
        primera=ultima=0;
    }
    else{
        primera=new Celda(C.primera->dc,0);
        ultima=primera;
        Celda<T> *q=C.primera->sig;
        while (q!=0){
            ultima->sig=new Celda(q->elc,0);
            ultima=ultima->sig;
            q=q->sig;
        }
    }
}
```

```
template <class T>
void Cola<T>::Poner(const T &e){
    Celda<T> *aux=new Celda(e,0);
    if (primera==0){
        primera=ultima=aux;
    }
    else{
        ultima->sig=aux;
        ultima=aux;
    }
}

template <class T>
bool Cola<T>::Vacia()const{
    return primera==0;
}
```

```
template <class T>
void Cola<T>::Borrar(){
    while (primera!=0){
        Celda *aux=primera;
        primera=primera->sig;
        delete aux;
    }
    ultima=0;
}

template <class T>
Cola<T>::Cola(){
    primera=ultima=0;
}

template <class T>
Cola<T>::~Cola(){
    Borrar();
}
```

```
template <class T>
Cola<T> & Cola<T>::operator=(const Cola<T> &C){
    if (this!=&C){
        Borrar();
        Copiar(C);
    }
    return *this;
}

template <class T>
T Cola<T>::Frente()const{
    assert(primera!=0);
    return primera->elc;
}

template <class T>
void Cola<T>::Quitar(){
    assert(primera!=0);
    Celda<T> *aux=primera;
    primera=primera->sig;
    delete aux;
}
```