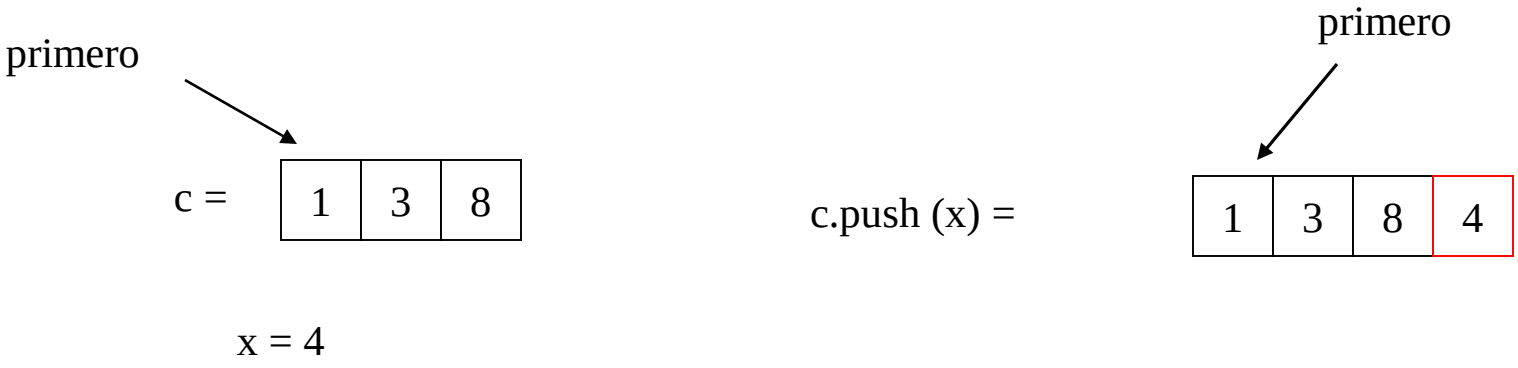
Módulos especiales

- Estructuras lineales:
  - Pilas
  - Colas
  - Listas
- Estructuras arborescentes

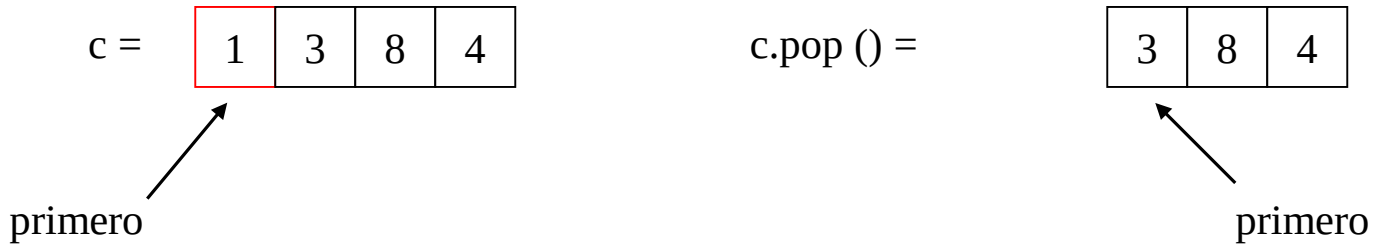
Colas

- Estructuras lineales
- Solo se puede consultar o borrar el primer elemento añadido (el más antiguo): estrategia FIFO (first in-first out)
- Su contenido se puede parametrizar (colas de enteros, de booleanos, etc)
- Especificación: ver `queue_esp.hh`

push


primero
c =
1
3
8
x = 4
primero
c.push (x) =
1
3
8
4

pop


c =
1
3
8
4
primero
c.pop () =
3
8
4
primero

Ejemplos de uso de colas

• suma de una cola

```
int suma_cua_int(queue<int>& c)
/* Pre: c=C */
/* Post: El resultat és la suma dels elements de C */
```

• La operación no podría ser genérica, ya que depende del *tipo* del contenido de la cola (ha de tener definido el operador)

•No dice cómo queda la cola al final de la operación

•La primera versión que veremos será recursiva

```
int suma_cua_int(queue<int>& c)
/* Pre: c=C */
/* Post: El resultat és la suma dels elements de C */
{
  int ret;
  if (c.empty()) ret = 0;
  else{
    int aux = c.front();
    c.pop();
    ret = suma_cua_int(c) + aux;
  }
  return ret;
}
```

c = | 1 | 3 | 8 |

suma(c) = 12

suma (| 1 | 3 | 8 |) = 1+ suma (| 3 | 8 |) = 4+ suma (| 8 |) = 12+ suma (c_nula) = 12+0 = 12

Versión iterativa

•si pasamos el parámetro por valor, garantizamos que la cola no se destruye (la operación recibe una copia)

•si hiciéramos eso en la versión recursiva, tendríamos mucha ineficiencia

```
int suma_cua_int(queue<int> c)
/* Pre: c=C */
/* Post: El resultat és la suma dels elements de C */
{
  int ret=0;
  while(not c.empty()){
    ret += c.front();
    c.pop();
  }
  return ret;
}
```

Más ejemplos básicos de uso de colas

- Búsqueda de un elemento en una cola
- Determinar si dos colas son iguales
- Sumar un valor k a todos los elementos de una cola

`(ver fichero ejemplos_cola.cc)`

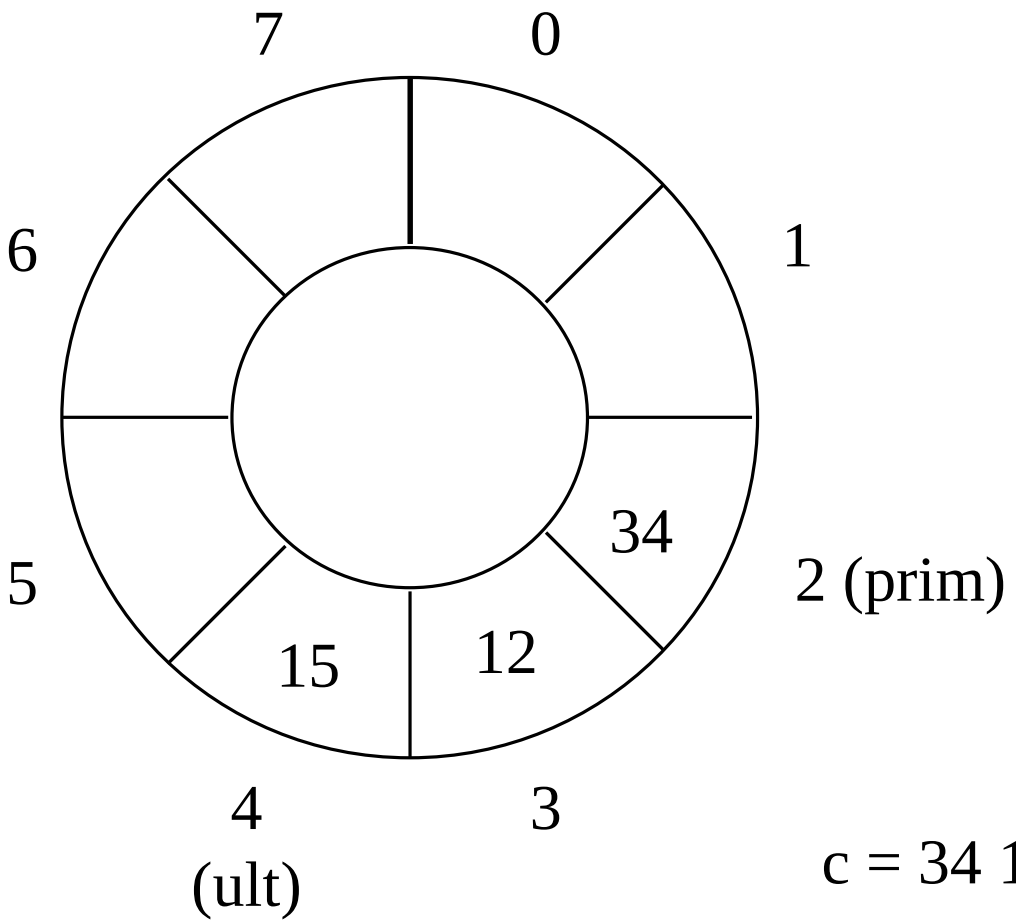
Ejemplos avanzados de uso de colas

- Cola de impresión
- Cola de procesos
- Turnos en las cajas de un supermercado, taquillas de un cine, etc.
- Cola con prioridad: los elementos están clasificados según la prioridad con la que deben ser tratados; si dos elementos tienen la misma prioridad se atienden según su orden de llegada

Ejemplos de implementación de colas

- Con un vector estático, particularizando el contenido (`queue_int.hh, queue_int.cc`)
- Con un vector estático, genérica (`queue.hh`)
- Lo mismo, con un vector de tamaño variable (`push_back`)
- Con punteros (lo veremos más adelante)
- Por adaptación de estructuras más complejas (`STL: deque`)

Cola como vector circular



Tamaño acotado

El siguiente de una posición `i` es `(i+1) % v.size()`

Ej: siguiente de 3 = 4
siguiente de 7 = 0

c = 34 12 15

Cola como vector circular

```
class cua_int {

private:
   vector<int> elems;
   int prim,ult;
   static const int MAX_SIZE = 20;
```

- Distintos programas pueden manejar distintas tamaños máximos editando el `.hh`

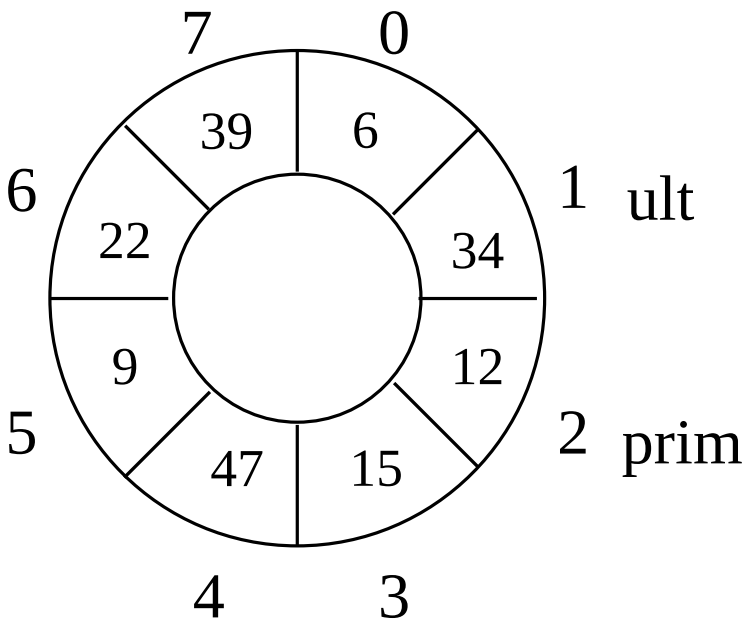
Cola como vector circular

• Añadir un elemento es ponerlo después del último (y pasa a ser el nuevo último)

• Eliminar un elemento es quitar el primero (y su siguiente pasa a ser el nuevo primero)

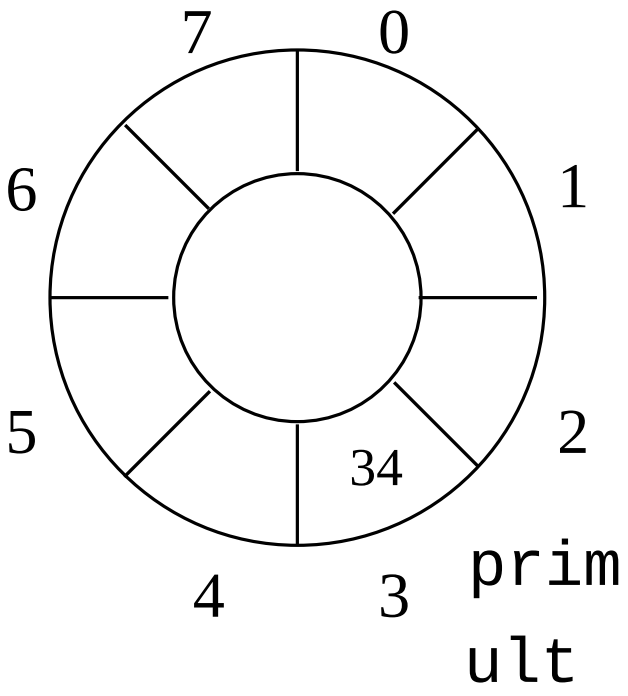
Cola como vector circular

• En principio, si una cola está llena, el siguiente del último es el primero



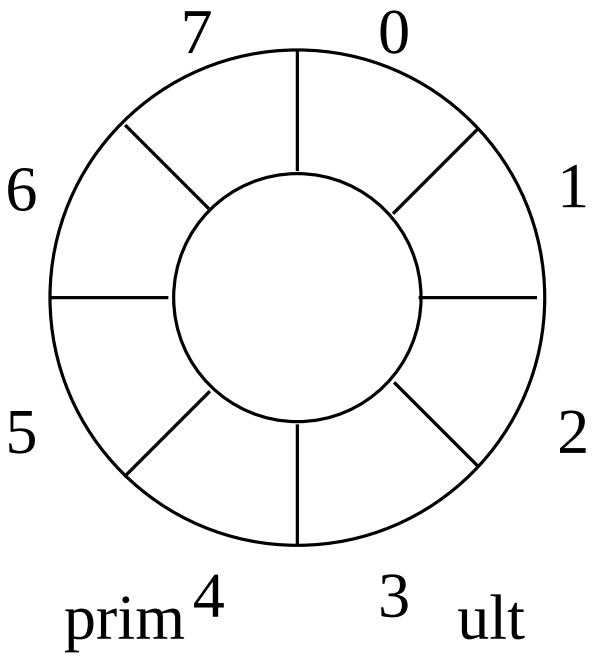
Cola como vector circular

•Y cómo es una cola vacía? Si una cola con un solo elemento cumple que `prim = ult` …



Cola como vector circular

• Si quitamos el primero y lo avanzamos, queda lo mismo que una cola llena!!!!



El siguiente del último es el primero

Cola como vector circular

- Para diferenciar los dos casos, hay dos opciones:
  - Dejar siempre una posición libre (en una cola llena, el primero está **dos** posiciones después del último)
  - Introducir un campo nuevo para la longitud
    - cola llena = longitud igual a v.size();
    - cola vacía = longitud igual a 0

  (en ambos casos el primero es el siguiente del último)

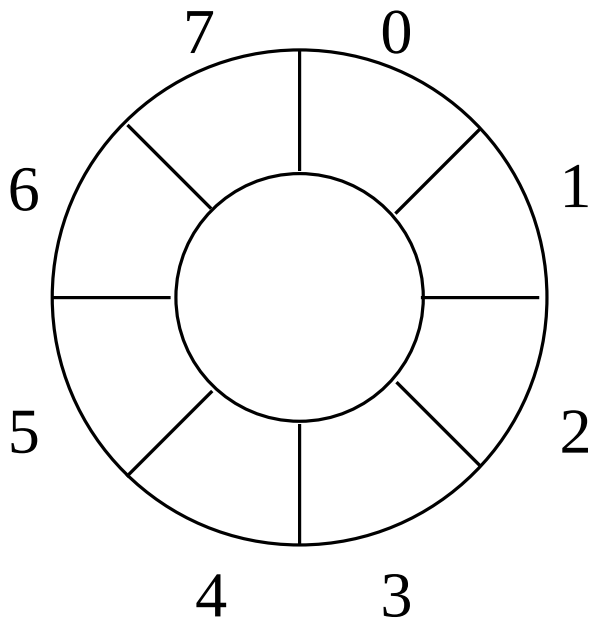
Independencia de la implementación

Aunque sepamos que una cola `c` se implementa con un vector, al usar la clase no podemos hacer cosas como

`c.prim=0` ni `c.elems[i]=14`

Tampoco podemos contar con una operación que vuelque la cola en un vector

```
vector<T> volcar_elementos(const queue<T> &c);
```


7
0
6
1
5
2
4
3