TIPO ABSTRACTO DE DATOS ***COLA***

Las *Colas* son secuencias de elementos que pueden crecer y contraerse siguiendo la política : Primero en Entrar, Primero en Salir, por lo que entre los elementos de la Cola existe un *orden temporal.* A las colas se las suele llamar, además, *Listas FIFO* -First In First Out, o *Listas "primero en entrar-primero en salir"*.

Construiremos el TAD Cola, utilizado para resolver problemas que responden a la política mencionada. Nuevamente, las distintas aplicaciones permitirán determinar el tipo de cada elemento particular.

a) *Especificación*

Cola: Secuencia de cero o mas elementos de un tipo determinado que obedece a la política FIFO.

$C = (a_1, a_2, ..., a_n)$, n>=0

Si la cola tiene elementos, es decir n > 0, entonces

a_1 = Primer elemento

a_n = Ultimo elemento

Dado que esta secuencia puede crecer y contraerse, un nuevo elemento ingresa en la cola detrás de a_n y el elemento que mas tiempo ha permanecido en la cola, y por lo tanto el próximo a ser retirado, es a_1.

Podemos suponer que **C** representa a una cola de clientes esperando a ser atendidos por un servidor -un cajero en un supermercado o en una entidad bancaria por ejemplo-. En este caso cada elemento a_i correspondería a un cliente.

Consideramos las siguientes operaciones básicas para el TAD Cola:

- **Insertar** un elemento X en la cola C

Si $C = (a_1, a_2, ..., a_{n)}$, con n>=0 y X es el elemento a insertar

Entonces, luego de ejecutarse la operación:

$C = (a_1, a_2, ..., a_n , X)$

- **Suprimir** un elemento de la cola C

Si $C =(a_1, a_2, ..., a_{n)}$, con n>0

Entonces, luego de ejecutarse la operación:

$C = (a_2, ..., a_n)$ y $X = a_1$

- **Recorrer** la cola C.

Retomamos en este caso, las consideraciones realizadas respecto a la operación recorrer cuando presentamos el TAD Pila.

Sean C una Cola y X un elemento:

NOMBRE	*ENCABEZADO*	*FUNCION*	*ENTRADA*	*SALIDA*
Insertar	Insertar (C,X)	Ingresa el elemento X en la cola C	C y X	$C=(a_1,a_2,...,a_n,X)$
Suprimir	Suprimir(C,X)	Si C no está vacía, elimina el elemento que mas tiempo ha permanecido en la cola	C	si n>0: $C=(a_2,..., a_{n-1})$ y $X=a_1$; Error en caso contrario
Recorrer	Recorrer(C)	Procesa todos los elementos de C siguiendo la política FIFO	C	Está sujeta al proceso que se realice sobre los elementos de C
Crear	Crear(C)	Inicializa C	C	C=( )
Vacía	Vacía(C)	Evalúa si C tiene elementos	C	Verdadero si C No tiene elementos, Falso en caso contrario.

Simulación

Una de las áreas en las que se aplica el TAD Cola es la Simulación, que consiste en la imitación de un proceso del mundo-real, a lo largo del tiempo.

La simulación constituye una técnica económica que nos permite, además de ofrecer varios escenarios posibles de una situación, equivocarnos sin provocar efectos sobre el mundo real (por ejemplo un simulador de vuelo o de conducción de automóviles).

Los programas de simulación reproducen el sistema físico real, por medio de variables que representan el estado de ese sistema, para analizar, predecir, o mejorar su comportamiento.

En un ***sistema de simulación discreto***, las variables de estado cambian solo en puntos discretos o contables en el tiempo, mientras que en un ***sistema de simulación continuo*** esas variables cambian en forma continua a través del tiempo.

Ejemplos típicos de simulación discreta corresponden a sistemas de colas : colas de clientes que realizan el pago de servicios por medio de un cajero (automático o no), colas de vehículos que esperan en una estación de servicio, colas de aviones que esperan en un aeropuerto la orden para despegar, líneas de montaje dentro de una fábrica automotriz. Diversos Sistemas Operativos suelen mantener colas para regular el orden en el cual las tareas reciben procesamiento o les son asignados diferentes recursos del sistema. En general estamos hablando de un sistema de colas de espera asociadas a un servidor. En los sistemas mencionados los servidores son: el cajero, la bomba de combustible, la pista del aeropuerto, el procesador y los recursos del sistema (impresora por ejemplo).

La simulación permitirá obtener ciertos datos del comportamiento del sistema, que apoyarán cualquier instancia de toma de decisiones. En los ejemplos planteados es de interés conocer por ejemplo cual es la longitud máxima de la cola o el tiempo de espera promedio de los clientes antes de ser atendidos. Estos datos servirán al momento de

establecer si el número de cajeros o de bombas de expendio de combustible es adecuado, si el espacio físico asignado es apropiado para el tamaño máximo que puede alcanzar la cola, si los recursos del sistema son suficientes, etc.

La simulación de colas involucra además de la generación de los eventos artificiales que cambian el estado del sistema, la recolección de observaciones. Por ejemplo si estamos interesados solo en conocer la longitud de una cola de espera, esta medida solo cambia cuando un cliente entra o sale del sistema; en todos los demás momentos, no ocurre nada en el sistema desde el punto de vista de la inferencia estadística. Los dos eventos principales que podemos considerar en un sistema de colas son: llegada de un cliente y realización del servicio.

La principal dificultad que surge al implementar una simulación informática consiste en salvar la distancia que existe entre el sistema físico y su implementación informática. Por ello, además de traducir lo que ocurre en el mundo real en un algoritmo, debemos establecer qué estructuras de datos es conveniente utilizar para representar a los componentes del sistema físico.

Ahora proponemos una aplicación del TAD Cola en un problema de simulación:

- Realizar la simulación informática correspondiente al comportamiento de una cola de clientes que esperan efectuar una transacción en un cajero automático. El objetivo de esta simulación es conocer el tiempo promedio de espera de los clientes en la cola.

La información obtenida servirá para realizar ajustes en el sistema que se modela; por ejemplo ayudará a decidir la adquisición de nuevos cajeros automáticos si el tiempo de espera de los clientes, calculado en la simulación, no es adecuado.

Para la ejecución de la simulación, realizamos un análisis previo del comportamiento del sistema real que se encuentra en funcionamiento y que se desea mejorar.

Los eventos principales que vamos a considerar tienen que ver con la realización de servicio y la llegada de un nuevo cliente, por lo que consideramos el tiempo que tarda el cajero en realizar una transacción y el tiempo aproximado que tarda un cliente en arribar a la cola en espera del servicio. Si bien estos datos responden a una distribución estadística, en este documento consideraremos que cada x minutos llega un cliente al sistema, y que el cajero tarda y minutos en realizar una transacción.

Construimos una simulación regida por el tiempo, monitoreando minuto a minuto lo que ocurre.

¿Qué eventos pueden ocurrir en un minuto particular?

- Puede o no llegar un cliente a la cola.
- Si el servidor está libre, debe comenzar a atender a un nuevo cliente.
- Los clientes que aun están en la cola esperan un minuto mas.

Los principales componentes de este sistema son el cajero, que es el servidor, y la cola de clientes.

¿Cómo representamos a cada uno de los componentes?

- Del servidor solo necesitamos saber si está ocupado o no, por lo que podríamos representarlo por medio de una variable booleana. Ahora bien, como el cajero tarda y minutos en atender a un cliente ($y>0$), lo representaremos por medio de un contador.

- De los clientes, nos interesa registrar solamente el tiempo que han permanecido en la cola, por lo que cada cliente puede ser representado a través de una variable que permita registrar el tiempo requerido.

Para conocer el tiempo promedio de espera de los clientes en la cola, necesitamos contar los clientes atendidos y acumular sus tiempos de espera, para ello, proponemos el siguiente algoritmo en pseudocódigo. El proceso se realizará durante una cierta cantidad de unidades de tiempo, que caracterizaremos como *tiempo máximo de simulación.*

```
Ingresar :  Frecuencia de llegada de Clientes (x), Tiempo de atención de Cajero (y) y
Tiempo máximo de Simulación.
Inicializar Reloj
MIENTRAS (Reloj no alcance el Tiempo máximo de Simulación) HACER
   SI (llega Cliente)
        ENTONCES  Insertar(Cola, Cliente)
   FIN SI
   SI (Cajero libre)
        ENTONCES
                SI (No (Cola-vacía(Cola)))
                     ENTONCES
                            Suprimir (Cola, Cliente)
                            Computar el Tiempo de espera del Cliente en la Cola
                            Acumular Tiempo de espera del Cliente
                            Contar cliente atendido
                             Inicializar Cajero Ocupado
                FIN SI
    FIN SI
   Actualizar Reloj
   SI (Cajero Ocupado )
            ENTONCES  Actualizar Cajero
   FIN SI
FIN MIENTRAS
Tiempo Promedio de espera =  Tiempo de espera Acumulado
                             Cantidad de Clientes Atendidos
Reportar (Tiempo Promedio de espera)
```

Analizamos como pueden representarse las condiciones, en el algoritmo:

- Llega cliente: Podemos representar esta condición a través de un número aleatorio (random) que varíe entre 0 y 1/ Frecuencia de llegada de un cliente

  0 <= Nro. aleatorio <= 1 / Frecuencia de llegada

- Cuando el cliente llega, se lo ingresa a la Cola con el tiempo en el que arriba al sistema (Reloj)

- Cajero ocupado/ libre : Sabemos que el cajero tarda ***y*** minutos en atender a un cliente, por ello representaremos al cajero por medio de un contador:

  Cajero = 0 representa Cajero libre
  Cajero > 0 representa Cajero Ocupado
  Inicializar Cajero Ocupado: Cajero ← Tiempo de atención de cajero

- Para controlar el Tiempo máximo de Simulación inicializamos un Reloj, representado por un contador

  Reloj = 0 Comienzo de la simulación
  Reloj = Tiempo Máximo de Simulación Fin Simulación

La simulación planteada responde a un modelo de Una Cola con Un Servidor. De hecho, hay sistemas mas complejos en los que aumentan la cantidad de Colas y / o de Servidores.

Aunque la simulación propuesta responde a un sistema existente, también se realizan simulaciones a partir de modelos de sistemas físicos que por su magnitud o características no pueden ser manipulados de manera directa. Ejemplos de ello son modelos atómicos de constitución de la materia, sistemas nucleares, planetarios etc. Estas simulaciones no necesariamente se apoyan en el TAD Cola.

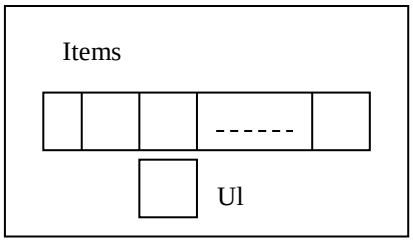
Actividad

1. A partir de un modelo de una cola de espera con un servidor:
   a) Diseñar el algoritmo que reporte la longitud máxima alcanzada por la cola.
   b) Diseñar el algoritmo que reporte el tiempo de espera máximo de un cliente en la cola.
2. ¿Las operaciones propuestas para el TAD Cola son suficientes para resolver el punto 1 o considera adecuado especificar nuevas operaciones?

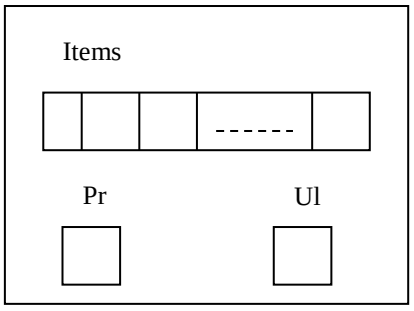
c) *Representación*:

El objeto de datos de nuestro TAD Cola está constituido como un agregado de componentes, por lo que recurriremos a las representaciones secuencial y vinculada.

- *Representación secuencial*: En un principio proponemos las siguiente estructuras de datos.



a)



b)

En ellas, consideramos que **Pr** almacena el subíndice del arreglo que contiene al primer elemento de la cola y **Ul** tiene el subíndice del arreglo que contiene al último elemento de C. En a) y b) la estructura de datos es un registro, compuesto por un arreglode elementos y uno-Pr-, o dos campos mas-Pr y Ul. En a) no se incorpora el campo Pr porque supone que el primer elemento de la cola siempre está almacenado en la primer componente del arreglo, mientras que en b) ese elemento puede estar en cualquier posición del arreglo, mantenida en Pr.

Actividad

1 – Partiendo de una cola vacía -C=( )-, grafique las distintas instancias de C al ejecutarse cada una de las siguientes operaciones:

Insertar(C, X) con X=5
Insertar(C, X) con X=2
Insertar(C, X) con X=7
Suprimir (C, X)
Suprimir (C, X)
Insertar(C, X) con X=4
Insertar(C, X) con X=6

2-

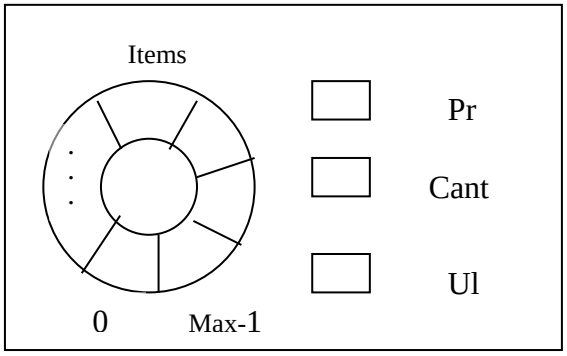
a) Para las representaciones a) y b) los elementos de la cola son almacenados en un arreglo. Suponiendo que el arreglo tiene 5 componentes, grafique para cada una de las representaciones, el estado de la estructura de datos luego de ejecutar cada una de las operaciones de la secuencia propuesta en el inciso 1.

b) Analice que ocurrirá en cada una de las propuestas de representación, si a la secuencia de operaciones del inciso 1 le agregamos:
Insertar(C, X) con X=3
Insertar(C, X) con X=9

Cola almacenada en un espacio circular

La Cola crece por un extremo y se contrae por el extremo opuesto. En la representación b), Pr se mueve hacia Ul y Ul se mueve hacia las posiciones mas altas, por lo que en algún momento puede ocurrir que se arribe a una situación en la que Ul alcanza el final del espacio asignado y una nueva inserción no pueda registrarse en el arreglo-Overflow, aún existiendo lugares libres en la posiciones mas bajas.
Para evitar este problema, se propone concebir el espacio de almacenamiento de manera circular, en el que a la última componente le sigue la primera. La cola se encuentra en alguna parte de este círculo ocupando posiciones consecutivas.

c)

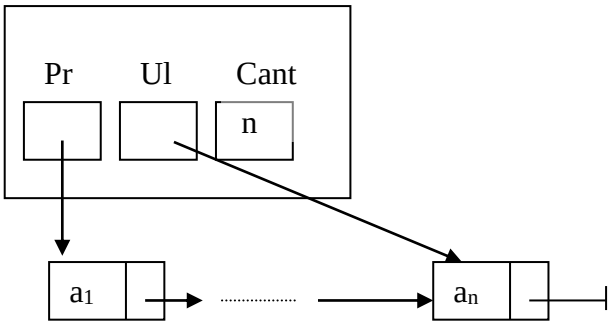


En esta alternativa, no pueden registrarse nuevas inserciones solo cuando todas las componentes del arreglo están ocupadas.

Actividad

Enuncie las ventajas y desventajas que, a su criterio, tienen las representaciones a), b) y c) propuestas.

- *Representación Encadenada:* Sugerimos la siguiente alternativa para esta representacion



Actividad

En la representación encadenada presentada, proponemos que las celdas que contienen los elementos a_1, a_2, ...,a_n, se vinculen entre sí en ese sentido. ¿Qué diferencia se presentaría, a su criterio, si la vinculacion de dichas celdas fuese en sentido inverso (desde a_n se accede a a_{n-1}...)? Justifique su respuesta.

c) *Implementación*:

La implementación de las operaciones está estrechamente vinculada con la representación seleccionada.

- Propuesta de TAD Cola con representación secuencial.

```
class cola
{
  int *items;
  int pr;
  int ul;
  int cant;
  int max;
  public:
  cola(int xmax=0):
  max(xmax)
  {  pr=0;
     ul=0;
     cant=0;
     items=new int[max];
   }

  int vacia(void)
  {
  return (cant==0);
   }

  int insertar(int x)
 {
  if (cant<max)
    {
    items[ul]=x;
    ul=(ul+1)%max;
    cant++;
    return (x);
```

```
    }
    else return (0);
  }

 int suprimir(void)
 {
   int x;
   if (vacia())
   {
   printf("%s","Pila vacia");
   return(0);
   }
 else
   {
     x=items[pr];
     pr=(pr+1)%max;
     cant--;
      return(x);
    }
 }

void recorrer(void)
 { int i,j;
   if (!vacia())
   {   i=pr;
       j=0;
       for (i; j<cant ; i=(i+1)%max,j++)
      {
        cout<<items[i]<<endl;
      }
   }
 }
};
```

- Propuesta de TAD Cola con representación encadenada.

```
class celda
{
  int item;
  celda *sig;
  public:
  int obteneritem(void)
  {
    return(item);
  }
  void cargaritem(int xitem)
  {
    item=xitem;
  }
  void cargarsig(celda* xtope)
  {
    sig=xtope;
```

```
  }
  celda* obtenersig(void)
  {
    return(sig);
  }
};

class cola
{
  int cant;
  celda *pr;
  celda *ul;
  public:
  cola(celda* xpr=NULL,celda *xul=NULL,int xcant=0):
  pr(xpr),ul(xul),cant(xcant)
  { }

  int vacia(void)
  {
  return (cant==0);
   }

  int insertar(int x)
 {
  celda *ps1;
  ps1=new(celda);
  ps1->cargaritem(x);
  ps1->cargarsig(NULL);
  if (ul==NULL)
   {pr=ps1;}
   else
  {ul->cargarsig(ps1);}
  ul=ps1;
  cant++;
  return(ul->obteneritem());
  }

 int suprimir(void)
 { celda *aux;
 int x;
 if (vacia())
   {
   printf("%s","Cola vacia");
   return(0);
   }
 else
 {
  aux=pr;
  x=pr->obteneritem();
  pr=pr->obtenersig();
```

```
  cant--;
  free(aux);
  return(x);
  }
 }

 celda* recuperapr(void)
 {
  return(pr);
 }

 void recorrer(celda *aux)
 {
   if (aux!=NULL)
   {  cout<<aux->obteneritem()<<endl;
      recorrer(aux->obtenersig());
   }
 }
};
```

Actividad

1. En la implementación propuesta para la representación secuencial, se hace uso de la operación módulo-**%-**. Si en el lenguaje de alto nivel la operación módulo no estuviese disponible, ¿cual sería la secuencia de instrucciones que la reemplazarían?
2. Reconstruya la operación **recorrer**, para la representación encadenada, a través de un subprograma iterativo.
3. Implemente en C++ el algoritmo de simulación presentado en este documento