PRÁCTICA Nº 4: 1 sesión

(S4: 5, 20, 21, 29 y 30 de abril de 2004)

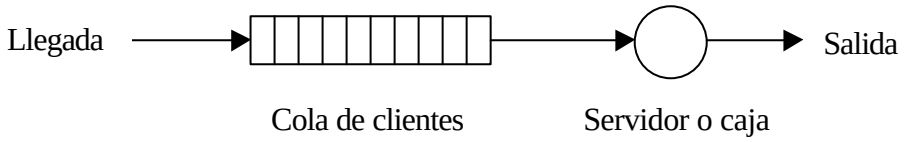
Simulación de colas

0. OBJETIVO

El objetivo de esta práctica es implementar y utilizar el tipo abstracto de datos "Cola" mediante las clases de C++. Para ello se va a modelizar el sistema de colas de un establecimiento de alimentación y obtener unos datos estadísticos.

1. INTRODUCCIÓN

El caso más simple de simulación de un sistema de colas consiste en un sistema con un único servidor o caja que atiende una única cola. Posteriormente se puede extender a escenarios más complicados con **n** servidores o cajas y sus correspondientes colas de clientes.



El establecimiento que se quiere simular está en funcionamiento durante 8 horas ininterrumpidas.

Los clientes llegan a intervalos de tiempo aleatorio, según una distribución exponencial, $f(x)= \lambda e^{-\lambda x}$ para $x>=0$, de media $\lambda=50$ segundos. Si U es un número aleatorio uniforme en el intervalo (0, 1), entonces se puede obtener una variable aleatoria distribuida exponencialmente $X=-\ln(1-U)/ \lambda$. A continuación se facilita la función que devuelve la hora(segundo) de llegada del siguiente cliente:

```
Cola::Valor ColaClientes::LlegadaSiguiente(Cola::Valor tiempo_actual)
{
   Cola::Valor x, r;

   r = rand();
   x = -log(1 - (r/Cola::Valor(RAND_MAX))) / LLEGADA;
   x = tiempo_actual + (LLEGADA * exp(-LLEGADA * x));
   return x;
}
```

Hay clientes que rehusan ponerse a la cola cuando esta es demasiado larga con la siguiente probabilidad:

$$Pr(rehuse)= \begin{cases} 0, & \text{si } m<=5 \text{ (siendo m el número de clientes en la cola)} \\ 0'5, & \text{en otro caso} \end{cases}$$

El tiempo empleado por el servidor o caja en atender al cliente también varía según una distribución exponencial de media $\lambda_2=60$ segundos, como se ha visto anteriormente.

Los datos de la simulación que interesa conocer de la cola de clientes son: clientes perdidos, tamaño medio y máximo de la cola durante la simulación, y el tiempo medio de espera para clientes en la cola. Y del servidor o caja: el número total de clientes atendidos, el tiempo medio de atención por cliente y el porcentaje de ocupación.

2. REALIZACIÓN DE LA PRÁCTICA

Se precisa implementar tres clases en C++, la clase que permite representar el TAD cola visto en clase, la clase cola de clientes y la clase servidor o caja.

Implementación de la clase *Cola*

La clase cola se implementa con la siguiente interfaz:

```
class Cola
{
   public:
      //Tipo de datos de los elementos de la cola
      typedef double Valor;
      //constructor
      Cola (void);
      bool ColaVacia (void);
      bool Encolar (Valor);
      bool Desencolar (Valor &);
      bool PrimeroCola (Valor &);
   private:
       ...
};
```

Implementación de la clase *Cola de Clientes*

La clase cola de clientes que se implementa contiene un objeto de tipo 'Cola' y las variables necesarias para el cálculo de las estadísticas de la simulación de la cola de clientes.

```
class ColaClientes
{
   public:
      //Constructor
      ColaClientes::ColaClientes(void);
      //Indica si la cola está vacia
      bool ColaVacia(void);
      //Entra un nuevo cliente y se coloca en la cola
      void Entrar(Cola::Valor);
      //Sale cliente de la cola
      void Salir(Cola::Valor);
      //Guarda cuando llegará el siguiente cliente
      void Siguiente(Cola::Valor);
      //Comprueba si ha llegado un cliente
      bool LlegaCliente(Cola::Valor);
      //Muestra las estadísticas de la cola de clientes
      void Estadisticas(Cola::Valor);
```

```
   private:
      Cola::Cola cola; //Objeto tipo cola
      // Variables para el cálculo de las estadísticas de la cola de clientes
      int longitud,  // nº de clientes que hacen cola en un momento dado
      tam_max_cola,  // longitud máxima que ha alcanzado la cola
      total_clientes, //nº total de clientes que se han acercado al establecimiento
      clientes_perdidos; //nº de clientes que han rehusado hacer cola
      Cola::Valor llega_en,  //cuando llegará el próximo cliente
      tiempo_ult_suceso, //momento en el que se realizó una entrada o una salida
      tiempo_total_espera, //tiempo total que han esperado los clientes
      tiempo_max_espera,  //tiempo máximo que ha esperado un cliente
      tam_total_cola; //Sirve para calcular el tamaño medio de cola. Es la suma
       //acumulada de los productos de los tamaños de la cola por la cantidad de
       //tiempo que la cola tuvo ese tamaño

      //Devuelve true si el cliente rehusa ponerse a la cola
      bool Rehusa();
};
```

De estos métodos se proporcionará al alumno 'Siguiente' y 'Rehusa';

Implementación de la clase *Servidor*

La clase servidor o caja que se implementa recoge los datos necesarios para el cálculo de las estadísticas de la simulación del servidor o caja.

```
class Servidor
{
   public:
      //Tipo de datos de los elementos de la cola
      typedef double Valor;
      //constructor
      Servidor (void);
      //El servidor atiende al cliente
      void AtenderCliente(Servidor::Valor);
      //Indica si el servidor está libre u ocupado
      bool Ocupado(Servidor::Valor);
      //Muestra las estadísticas del servidor
      void Estadisticas(Servidor::Valor, Servidor::Valor);
   private:
      // Variables para el cálculo de las estadísticas del servidor
      int clientes_atendidos; // número de clientes atendidos por el servidor
      Servidor::Valor ocupado_hasta, // cuando está libre el servidor
      tiempo_total_atencion; //tiempo total que el servidor pasa atendiendo clientes
```

```
        //Simula cuanto tardará el servidor en atender al cliente
        Servidor::Valor DisponibleEn(Servidor::Valor tiempo_actual);
};
```

De estos métodos se proporcionará al alumno el método privado 'DisponibleEn'

Implementación del programa

Se propone escribir un programa que simule un sistema con un servidor y una cola de clientes.

El proceso consiste, básicamente, en los siguientes pasos:

1.- Se simula la hora de llegada del primer cliente.

2.- Mientras el reloj de la simulación marque una hora inferior a la de cierre:

   a.- Se incrementa el reloj

   b.- Si llega un cliente (la hora actual es mayor que la hora en la que se preveía su llegada) se pone a la cola guardando la hora de entrada y se simula la hora de llegada del siguiente cliente.

   c.- Si el servidor no está ocupado (la hora actual es menor que la hora hasta la que está ocupado) y la cola no está vacía se saca un cliente de la cola, el servidor pasa a estar ocupado atendiéndolo y se simula la hora de salida.

3.- Finalmente se muestran las estadísticas de la cola de clientes y del servidor.

Una posible salida por pantalla del programa es:

```
ESTADÍSTICAS DE CLIENTES
        Clientes perdidos:                193
        Tamaño máximo de cola:            10
        Tamaño medio de cola:             1.00323
        Tiempo máximo de espera(seg.):    357
ESTADÍSTICAS DEL SERVIDOR
        Clientes atendidos:        935
        Tiempo medio de atención(seg.):   30.1524
        Ocupación servidor(%):            97.774
```

3. ENTREGA

En esta práctica, habrá que entregar una documentación de programa completa, el código del programa principal(**`##simulacion.cpp`**) y la implementación de las clases 'Cola' (**`##cola.h`** y **`##cola.cpp`**), 'ColaClientes' (**`##colaclientes.h`** y **`##colaclientes.cpp`**) y 'Servidor' (**`##servidor.h`** y **`##servidor.cpp`**).

La entrega puede realizarse por e-mail a la dirección del profesor encargado del grupo o en disquete al iniciar la siguiente sesión de prácticas, pero siempre ANTES de empezar la siguiente sesión de prácticas.

Cualquier práctica entregada fuera de plazo no será admitida para su corrección.

ENTREGA DE PROGRAMAS:
Al comenzar la sesión 5 de prácticas (26 , 27, 28 de abril y 6, 7 de mayo de 2004)