

INSTITUTO POLITÉCNICO NACIONAL
Escuela Superior de Cómputo
(ESCOM)



ESTRUCTURAS DE DATOS

Reporte de práctica II.

Simulaciones con el TAD Cola.

Prof.: Edgardo Adrián Franco Martínez

Ayona López Eugenio Milton



Benítez Morales Manuel Emilio



1CM8

Introducción

Una cola es una estructura de datos, caracterizada como una secuencia de elementos en la que se insertan datos por un extremo (el posterior o final) y se extraen por el otro (el anterior o frente).

Son un caso particular del TAD LISTA, el cual se verá en las prácticas posteriores, siendo la cola una lista FIFO (del inglés First In First Out), debido a que el primer elemento en entrar será también el primero en salir.

Su uso y utilidad más frecuentes son donde los datos requieren ser tomados tal como se almacenan y se guardan mediante colas para su posterior procesamiento. Un ejemplo claro es la lista de documentos a imprimir en un solo dispositivo impresor, donde sale primero, el que fue enviado en primer lugar, y análogamente el último.

Planteamiento del problema

En la práctica se pidió realizar tres simulaciones donde se nos presentan tres casos en los cuales las colas se utilizarán de distinta forma.

Las tres simulaciones fueron:

- Simulación de un Supermercado.
- Ejecución de los procesos en el sistema operativo.
- Banco.

En la primera simulación se debe simular la atención de los clientes de un supermercado, se debe atender al menos a 100 clientes, una vez que se hayan atendido a los 100 clientes y que no haya gente formada en las cajas se podrá cerrar la tienda en el caso contrario las personas podrán seguir llegando a las cajas.

En la siguiente se debe simular la ejecución de procesos gestionados por el sistema operativo en un equipo monoprocesador sin manejo de prioridades, manejando únicamente el cambio de la cola de listos a ejecución y una vez terminado el proceso este se envía a la cola de terminados.

La última simulación es sobre la atención de personas en un banco, cuidando que sean respetadas las políticas de atención del mismo y evitando que las personas no dejen de ser atendidas.

El banco cuenta con 1 a 10 cajas las cuales atienden a tres filas (clientes, usuarios y preferentes).

Cientes del Banco: Ellos no dejarán de ser atendidos por ninguna caja y pueden ser atendidos en cualquier cajero.

Usuarios del Banco: son atendidos según la disponibilidad de cada caja, nunca permitiendo que pasen más de 5 personas de las otras dos filas sin que una persona de esta fila sea atendida.

Clientes preferentes: serán atendidos con mucha mayor prioridad que a los clientes y usuarios.

Diseño y funcionamiento de la solución

1) Supermercado

Entradas:

- 1.1 Nombre del supermercado.
- 1.2 Número de cajas que lo atenderán ($0 < n < 11$).
- 1.3 Tiempo de atención en cada cajero en milisegundos.
- 1.4 Tiempo en milisegundos de la llegada de cada cliente a las cajas.

Salidas

- 1.5 Llegada de los clientes a las colas de las cajas.
- 1.6 Clientes en espera de cada cola
- 1.7 Cliente que es atendido en cada caja
- 1.8 Número de clientes atendidos en su totalidad
- 1.9 Nombre de la tienda y anuncio de cierre

2) Sistema operativo

Entradas:

- 2.1 La cantidad de procesos en la cola y sus propiedades (Nombre, actividad, ID, tiempo).

Salidas:

- 2.2 Mostrar de manera atractiva la simulación de manera que se vea:
- 2.3 Proceso en ejecución actual y sus datos (Nombre, ID, Actividad y Tiempo total que lleva ejecutándose) ->Tiempo en la cola de listos + tiempo de ejecución total.
- 2.4 ID y Nombre del ultimo proceso ejecutado y el tiempo de procesador que falta para que este proceso concluya.
- 2.5 ID y Nombre del proceso siguiente a ser ejecutado y el tiempo que falta para que este proceso concluya.
- 2.6 Cuando un proceso termina este se coloca en la cola de finalizados almacenando su tiempo total (Tiempo en la cola de listos + tiempo de ejecución total) .
- 2.7 Cuando terminen todos los procesos mostrar en el orden de finalización el Nombre, ID y tiempo total que tardo en terminar cada proceso.

3) Banco

Entradas:

- 3.1 Cantidad de cajeros en el banco ($0 < n < 10$).
- 3.2 Tiempo de atención en los cajeros en milisegundos.
- 3.3 Tiempo de llegada de los clientes del banco.
- 3.4 Tiempo de llegada de los usuarios del banco.
- 3.5 Tiempo de llegada de los clientes preferenciales del banco.

Salidas

- 3.6 Impresión de los cajeros.
- 3.7 Llegada de los clientes, clientes preferenciales, usuarios a la cola.
- 3.8 Movimiento del cursor respecto a la persona recién llegada.
- 3.9 Cantidad de las personas que se encuentran en la cola.

Implementación de la solución

Supermercado

Primeramente, fueron separados los procesos de petición de datos por parte del usuario, es decir, el número de cajas, número de clientes y los tiempos de atención y llegada. Cada función implementada para este objetivo, tiene especificado un mensaje de error en caso de que los datos ingresados no cumplan con lo establecido en la especificación.

Lo anterior lo presentan las funciones: `Entrada_Cajeras`, `Entrada_Atencion`, `Entrada_LlegadaCliente` y `N_Clientes`.

Para una separación más atractiva del código implementado, existe una función que realiza toda la simulación de las cajas de supermercado; hace uso de las funciones del TAD Cola, inicia las colas que funcionarán como las cajas, y por medio de una serie de ciclos, imprime caracteres específicos para representar las filas obtenidas, así como su paso hasta ser atendido, al final muestra cuantos clientes fueron atendidos antes de que la tienda cierre.

Esto lo representa la función: `Simulacion`.

Existen también funciones encargadas de mostrar la interfaz en sí, se encargan de esperar un tiempo especificado en milisegundos, limpiar la pantalla de consola, mover el cursor de la terminal e imprimir las representaciones de las cajas, respectivamente representadas por:

`Tiempo_Espera`, `Borrar_Pantalla`, `Mover_Cursor`, `Imprimir_Caja`.

En el programa principal, únicamente se llaman las funciones de entrada y la de salida, ya que la salida (`Simulacion`) hace llamada de las funciones de interfaz.

Sistema operativo

Entradas:

- La cantidad de procesos en la cola y sus propiedades (Nombre, Actividad, ID, tiempo).

Para poder simular de una manera eficiente como se ejecutan los procesos se hará una interfaz gráfica para ver como se visualiza el uso de las Colas. En la simulación se tiene tres bloques uno donde se visualizan los procesos que están “listos” para ejecutar, el bloque donde se estará ejecutando el proceso y el bloque de los procesos finalizados.

Mostraremos de una manera atractiva el proceso que está en ejecución actual y sus datos como lo es nombre, ID, Actividad y tiempo total que lleva ejecutándose.

Cada que el proceso que esté en el bloque de ejecución si el tiempo de despachar un proceso concluye, este proceso se debe volver a encolar en los procesos que están en el bloque de listos para ejecutar.

Para tener un buen monitoreo de cada proceso se mostrará el último proceso con su ID y el nombre del propio al igual que el tiempo que le falta para que el proceso concluya. También se realizará lo mismo para el proceso que continúe en la Cola.

Cuando un proceso termine este se coloca en la cola de finalizados y almacenando su tiempo total.

Al finalizar se mostrará en orden de cómo fue terminando cada proceso.

Banco

Entradas:

(3.1) Es una condición ya que nuestro máximo de cajeros debe ser menor a 10, cuando sobrepasa esa cantidad el programa te pide que se deba asignar un número correcto.

(3.2) utilizamos este dato para dar atención a los clientes con respecto al número de cajeros que haya, si tenemos un tiempo (n) y una cantidad de cajeros (k) cada vez que se llegue a n de la cola se deben vaciar k personas y de nuevo dar la disponibilidad (k) personas.

(3.3, 3.4, 3.5) Cada vez que se llegue a este tiempo se ingresará en la cola clientes, clientes preferenciales y usuarios del banco respectivamente uno nuevo.

Salidas:

(3.6) Es la cantidad de cajeros que estarán en consola.

(3.7) Cada vez que llegue una persona nueva muestra de que tipo es.

(3.8) Imprime una barra en la posición del tipo de cliente que ha llegado.

(3.9) Imprime la cantidad del tipo de cliente que está en la cola, actualizando cuando encolamos y desencolamos.

Condiciones

- Los clientes del banco (personas con cuenta en ese banco), son atendidos por cualquier cajero y nunca dejan de ser atendidos por alguna caja.
- Los usuarios del banco (personas sin cuenta en ese banco), son atendidos según la disponibilidad de alguna caja, nunca permitiendo que pasen más de 5 personas de las otras dos filas sin que una persona de esta fila sea atendida.
- Los clientes preferentes (personas con más de una cuenta en ese banco y privilegios preferenciales), serán atendidos por cualquier cajero disponible con mayor prioridad que a los clientes y usuarios.

Se han creado tres colas diferentes para cada tipo de cliente para que al momento de checar las condiciones sea más sencillo atender un cliente. Existen tres condiciones que verifican que el tipo de cliente sea ingresado a la cola específica con respecto al tiempo que se le ha ingresado.

Cada persona debe ser atendida cada vez que el cajero esté disponible esto sucede cuando el tiempo que anteriormente le pusimos al cajero es un múltiplo del tiempo desde que inició la ejecución del programa, cuando esto sucede, se puede atender a la cantidad de clientes con respecto a los cajeros disponibles, pero los clientes no son atendidos aleatoriamente, debemos seguir las condiciones que están arriba hay tres condiciones en el programa que verifican lo anterior.

Además para que sea el programa sea más visual se añadió una sencilla interfaz gráfica que muestra la cantidad de cajeros que están en el banco, la cantidad de tipo de clientes que han sido atendidos y como último hay un cursor o barra que muestra al tipo de cliente actual que ha llegado a la cola.

Funcionamiento

Supermercado

Modo de compilación: gcc supermercado.c TADCola(Din)(Est).c Bib_Supermercado.c -o archivo_ejecutable

```
C:\WINDOWS\system32\cmd.exe
SuperESCOM
Cajas en esta tienda:
```

FIGURA 1. Salida inmediata al ejecutar programa.

```
C:\WINDOWS\system32\cmd.exe
SuperESCOM
Cajas en esta tienda: 10
Tiempo de atencion (ms): 80
Tiempo de llegada de los clientes (ms): 10
Numero de clientes a atender: 210
```

FIGURA 2. Ingreso de los datos necesarios.

```
Seleccionar C:\WINDOWS\system32\cmd.exe
SuperESCOM
Tiempo de atencion en cada caja: 80 ms
Tiempo de llegada de clientes: 10 ms

Clientes restantes: 210
Fila:1  Fila:1  Fila:6  Fila:4  Fila:2  Fila:2  Fila:1  Fila:1  Fila:10 Fila:1
|||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||
|Caja #1|Caja #2|Caja #3|Caja #4|Caja #5|Caja #6|Caja #7|Caja #8|Caja #9|Caja #10
|||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||  |||||
-      *      *      *      *      *      -      *      *      *
TIENDA CERRADA
Atendidos: 408 clientes *      *      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *      *
*      *      *      *      *      *      *      *      *
```

FIGURA 3. Pantalla de salida de la simulación.

Sistema operativo

```
Ingrese el numero de procesos a ejecutar: 4
Ingrese datos del proceso 1

Ingrese nombre del proceso: Jugar
Ingrese actividad del proceso: Jugar

Ingrese id del proceso: 1

Ingrese tiempo requerido del proceso: 30

Ingrese datos del proceso 2

Ingrese nombre del proceso: Itunes
Ingrese actividad del proceso: Musica

Ingrese id del proceso: 2

Ingrese tiempo requerido del proceso: 28

Ingrese datos del proceso 3

Ingrese nombre del proceso: _
```

FIGURA 4. Ingreso de los procesos.

```
Ultimo procesado:
Nombre: Redes ID: 4 Tiempo restante: 19

Procesado:
Nombre: Jugar
Activad: Jugar
ID: 1
Tiempo en proceso: 4

Siguiente a procesar:
Nombre: Itunes ID: 2 Tiempo restante: 27
_
```

```
Ultimo procesado:
Nombre: Redes ID: 4 Tiempo restante: 16

Procesado:
Nombre: Jugar
Activad: Jugar
ID: 1
Tiempo en proceso: 16

Siguiente a procesar:
Nombre: Itunes ID: 2 Tiempo restante: 24
_
```

FIGURA 5. Pantalla de salida de la simulación.


```

Nombre de proceso      Redes
Actividad del proceso: Facebook
ID del proceso   4
Proceso finalizado en: 77
*****Finalizado

Nombre de proceso      Itunes
Actividad del proceso: Musica
ID del proceso   2
Proceso finalizado en: 102
*****Finalizado

Nombre de proceso      Jugar
Actividad del proceso: Jugar
ID del proceso   1
Proceso finalizado en: 107
*****Finalizado

Nombre de proceso      Visual
Actividad del proceso: Programacion
ID del proceso   3
Proceso finalizado en: 111
*****Finalizado

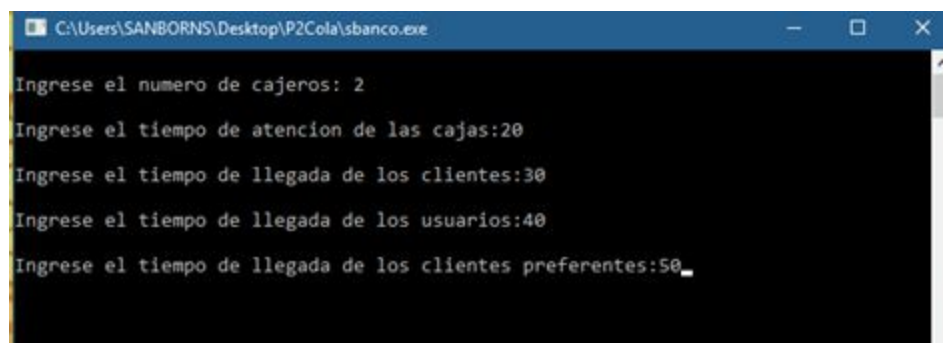
-----
Process exited after 264.8 seconds with return value 0
Presione una tecla para continuar . . .

```

FIGURA 6. Pantalla de Resultados Finales.

Banco

Funcionamiento para la simulación de la atención de personas en un banco, funcionamiento de la entrada de la simulación, entrada de datos con:



```

C:\Users\SANBORN\ Desktop\P2Cola\sbanco.exe
Ingrese el numero de cajeros: 2
Ingrese el tiempo de atencion de las cajas:20
Ingrese el tiempo de llegada de los clientes:30
Ingrese el tiempo de llegada de los usuarios:40
Ingrese el tiempo de llegada de los clientes preferentes:50_

```

FIGURA 7. Ingreso de datos necesarios para simular el banco.

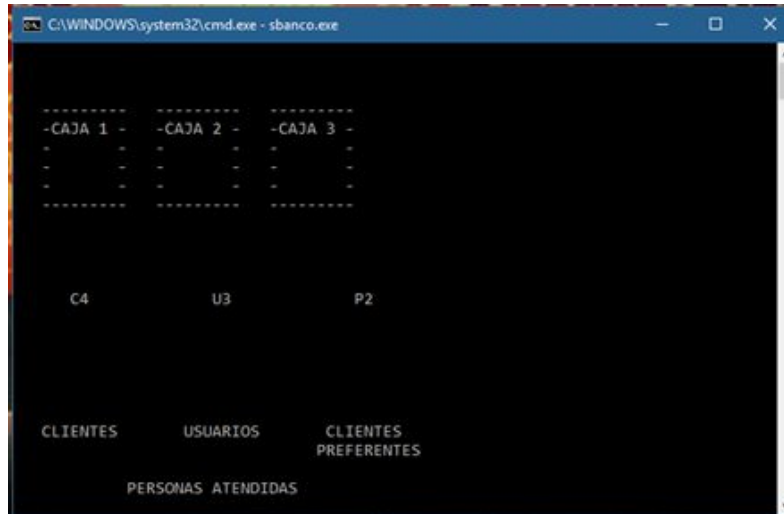


FIGURA 8. Simulación del banco.

Errores detectados

Supermercado

1. La tienda sigue atendiendo después de haber que dato todas vacías y con más de 100 clientes.
2. si el contador de clientes X en la fila (Fila X) llega a un valor mayor o igual que 10, y luego disminuye, el segundo dígito queda escrito y parece un número muy elevado.

Sistema operativo

En la simulación del Administrador de procesos se notó que si pones demasiados procesos los tiempos de cada proceso no se restan cada que termina un Quantum.

Banco

En la solución para la simulación de la atención de clientes en un banco con prioridades, hasta el momento se detectó que si el programa o la ejecución, se deja procesando por un largo periodo, provoca un desbordamiento en la pila, que posiblemente pueda verse creado por la asignación estática que se le da al modificar la estructura cola para aplicarla en este problema.

Otro problema que se detectó con el desbordamiento, es que, si los tiempos son muy pequeños, genera que exista uno de igual forma bajo algunas condiciones ingresadas durante la demostración en clase, se detectó que solo le da prioridades y atención a clientes preferentes, este error se debe posiblemente al planteamiento del algoritmo de manera incorrecta.

Posibles mejoras

Supermercado

1. Quitar el salto de línea extra al comenzar a imprimir los ‘*’ de la fila, ya que no representa algo necesario.
2. Replantear la condición en donde debe cerrarse la tienda, para que se detenga realmente en el caso donde ya se atendieron 100 clientes o más, y las cajas quedan vacías.
3. Se puede dar un mejor diseño a las cajas.

Sistema operativo

Se podría mejorar un poco más la interfaz de usuario para que se vea un poco más claro el movimiento de los Queues y Dequeues al igual que corregir el error mencionado anteriormente.

Banco

Se debe plantear de mejor manera la solución del problema debido a que ocurrían muchos errores que al momento de analizarlo no se daban, la solución parecía funcionar en su gran mayoría pero en algunas cuestiones fallaba por lo que se debe hacer una mejora en eso y corregirlo en una versión futura.

Conclusiones

Ayona López Eugenio Milton

Cuando nos enseñan algo a base de teoría resulta muy difícil asimilarlo como conocimiento cuando hacemos este tipo de prácticas aplicamos esa teoría y adquirimos con mayor facilidad el conocimiento.

Esta práctica le dio un uso más real TAD Cola entendimos cómo podemos utilizarlo para resolver el problema planteado con base al uso de la abstracción de la cola.

Benítez Morales Manuel Emilio

El TAD Cola, así como los diferentes tipos de dato abstracto existentes, es muy útil para la resolución de varios problemas específicos, e incluso es posible observar su comportamiento natural en muchas actividades diarias. Entender el funcionamiento de este tipo de lista es importante en la implementación de soluciones que requieren una prioridad de operación.

Gracias a esta práctica, fue posible comprender el funcionamiento de una estructura tipo cola, y cómo puede abstraerse de manera similar a la realidad, siendo que, al igual que el TAD visto anteriormente, basta con pensar con lógica las operaciones y su uso.

Tellez Pérez Juan Manuel

Con esta práctica pudimos entender el funcionamiento del TAD Cola para la resolución de algunos problemas como los fueron las tres simulaciones donde tuvimos que aplicar de diferente forma el uso de las Colas, igual tuvimos que practicar el control de tiempo en las tres simulaciones al igual que la parte gráfica que representa de forma visual todas las operaciones que realizan con la Cola.

Anexo

Supermercado

- Bib_Supermercado.h

```
/*
```

```
LIBRERIA
```

```
Simulacion de la atencion a clientes en un supermercado, implementando el TAD Cola.  
Realiza la simulacion de la llegada y atencion de las cajas a las personas, atendiendo a  
por lo menos 100 clientes.
```

```
Fecha: 01/04/2019
```

```
Version 1.0
```

```
Autores: Ayona Lopez Eugenio Milton, Benitez Morales Manuel Emilio
```

```
*/
```

```
#include "TADColaDin.h"
```

```
#define ALTO_CAJA 3
```

```
#define ANCHO_CAJA 8
```

```
#define TIEMPO_BASE 10
```

```
/*
```

```
Pide la cantidad de cajas que hay en la tienda y  
retorna el valor ingresado cuando este es correcto.
```

```
*/
```

```
int Entrada_Cajeras(int ncajas);
```

```
/*
```

```
Pide el tiempo en milisegundos de atencion a los clientes y  
retorna el valor ingresado cuando este es correcto.
```

```

*/
int Entrada_Atencion(int tAtencion);

/*
Pide el tiempo en milisegundos de llegada de los clientes y
retorna el valor ingresado cuando este es correcto.
*/
int Entrada_LlegadaCliente(int tLlegada);

/*
Pide el numero de clientes que atenderan las cajas y
retorna esa cantidad.
*/
int N_Clientes(int nClientes);

/*
Realiza todo el proceso de la representacion del servicio de las cajas,
no devuelve valor, y recibe los enteros ingresados como tiempo de atencion, de llegada
y el numero de clientes y cajas para poder realizar el proceso.
*/
void Simulacion(int cajas, int tAtencion, int tLlegada, int nClientes);

```

/////////FUNCIONES PARA LA INTERFAZ

```

/*
Recibe el tiempo en milisegundos ingresados por el usuario, y se detiene
precisamente para realizar las operaciones pausadas.
*/
void Tiempo_Espera(int tiempo);

/*
No recibe ni devuelve valores; se encarga de limpiar la pantalla de la consola
para realizar de manera atractiva la simulacion.
*/
void Borrar_Pantalla();

/*

```

Se encarga de mover el cursor a las posiciones especificadas de la consola, para relizar una atractiva simulacion.

*/

```
void Mover_Cursor(int x, int y); //simula el gotoxy de conio.h
```

/*

Realiza la escritura en pantalla de las representaciones de cajas; recibe las posiciones en x,y y el numero de cajas en la tienda, no devuelve valores.

*/

```
void Imprimir_Caja(int x,int y, int ncajas);
```

- Bib_Supermercado.c

/*

IMPLEMENTACION

Simulacion de la atencion a clientes en un supermercado, implementando el TAD Cola. Realiza la simulacion de la llegada y atencion de las cajas a las personas, atendiendo a por lo menso 100 clientes.

Fecha: 01/04/2019

Version 1.0

Autores: Ayona Lopez Eugenio Milton, Benitez Morales Manuel Emilio

*/

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

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

```
#include <windows.h>
```

```
#include "Bib_Supermercado.h"
```

```
int Entrada_Cajeras(int ncajas)
```

```
{
```

```
    printf("Cajas en esta tienda: ");
```

```
    scanf("%d", &ncajas);
```

```
    if(ncajas<=0 || ncajas>=11)
```

```
    {
```

```
        printf("Debes poner una cantidad de cajas mayor que 0 y menor que 11.\n");
```

```

        ncajas=-1;//ayuda a la condicion en el main
    }

    return ncajas;
}

int Entrada_Atencion(int tatencion)
{
    printf("Tiempo de atencion (ms): ");
    scanf("%d", &tatencion);

    if(tatencion%10!=0)
    {
        printf("El tiempo de atencion de la caja debe ser multiplo de 10.\n");
        tatencion=-1;//ayuda a la condicion en el main
    }

    return tatencion;
}

int Entrada_LlegadaCliente(int tllegada)
{
    printf("Tiempo de llegada de los clientes (ms): ");
    scanf("%d", &tllegada);

    if(tllegada%10!=0)
    {
        printf("El tiempo de llegada de los clientes debe ser multiplo de 10.\n");
        tllegada=-1;//ayuda a la condicion en el main
    }

    return tllegada;
}

int N_Clientes(int nclientes)
{
    printf("Numero de clientes a atender: ");
    scanf("%d", &nclientes);

```

```

        if(nclientes<100)
        {
            printf("Debe haber al menos 100 clientes.\n");
            nclientes=-1;//ayuda a la condicion en el main
        }

        return nclientes;
    }

void Simulacion(int cajas, int tAtencion, int tLlegada, int nClientes)
{
    int i, o, clientes_restantes=nClientes, numCaja, tamFila, formar,
    contador_clientes=0;
    int tiempo=0, cliente=0; //Tiempo de base para la espera y la llegada, cliente
    atendido
    cola filaParaCobrar;
    cola filaDeCaja[cajas]; //arreglo de colas, porque en un super hay varias filas
    elemento e;

    Borrar_Pantalla();

    printf("SuperESCOM\n\n");
    printf("Tiempo de atencion en cada caja: %d ms\n", tAtencion);
    printf("Tiempo de llegada de clientes: %d ms\n\n", tLlegada);
    printf("Clientes restantes: %d", clientes_restantes);

    //Se inician todas las colas
    for(i=0;i<cajas;i++)
    {
        Initialize(&filaDeCaja[i]);
    }

    for(i=0; i<cajas; i++)
    {
        Imprimir_Caja((i*10)+1, 8, cajas);
    }

    Tiempo_Espera(TIEMPO_BASE);

```



```

do
{
    tiempo++;
    Tiempo_Espera(TIEMPO_BASE);
    if(tiempo%tLlegada==0)
    {
        clientes_restantes--; //un cliente menos que atender
        cliente++;
        numCaja= rand()%cajas; //Se selecciona una caja al azar
        e.n= cliente;
        Queue(&filaDeCaja[numCaja], e); //encolar al nuevo cliente

        for(i=0; i<cajas; i++) //se recorren las cajas
        {
            tamFila= Size(&filaDeCaja[i]); //hay que saber cuantos hay
            formar= (5*(i+1) + 6*i - i)-1; //Moviendo el cursor sobre x

            for(o=1; o<=tamFila; o++) //se recorren los clientes
            {
                if(o==1)
                {
                    e= Element(&filaDeCaja[i], o);
                    Mover_Cursor(formar,12);
                    printf("*", e.n);
                    //Imprime el numero de clientes en espera a

                    Mover_Cursor(formar-4,7);
                    printf("Fila:%d", tamFila);
                    Mover_Cursor(formar,11);
                }
                else
                {
                    e=Element(&filaDeCaja[i], o);
                    Mover_Cursor(formar, o+12);
                    printf("*", e.n);
                }
            }
        }
    }
}

```

formados

para formar al cliente

formados

ser atendidos

```

//Imprime el numero de clientes en espera a
ser atendidos

Mover_Cursor(formar-4,7);
printf("Fila:%d", tamFila);
Mover_Cursor(formar, o+11);
    }
    }
}
//Rectifica el tiempo de cada una de las colas
for(i=0; i<cajas; i++)
{
    if(tiempo%tAtencion==0)
    {
        //Si hay alguien por atender
        if (!Empty(&filaDeCaja[i]))
        {
            e=Dequeue(&filaDeCaja[i]);
            //Aumenta el Numero de clientes atendidos
            contador_clientes++;
        }
        //Si la cola esta vacia(no hay clientes formados)
        else
        {
            formar=(5*(i+1) + 6*i - i)-1;
            Mover_Cursor(formar, 12);
            //Si la caja esta vacia, imprime "-"
            printf("-");
        }
    }
}

}while(clientes_restantes>0 || (tamFila!=0 && contador_clientes>=100));

printf("\n\nTIENDA CERRADA\n");
printf("Atendidos: %d clientes", contador_clientes);

return;
}

```

/////////FUNCIONES PARA LA INTERFAZ

```
void Tiempo_Espera(int tiempo)
{
    Sleep(tiempo);//tiempo en milisegundos
    return;
}
```

```
void Borrar_Pantalla()
{
    system("cls");
    return;
}
```

```
void Mover_Cursor(int x, int y) //simula el gotoxy de conio.h
{
    HANDLE salida = GetStdHandle(STD_OUTPUT_HANDLE); //"objeto" de la
"clase" handle
    COORD posicion = {x, y};
    SetConsoleCursorPosition(salida, posicion);
    return;
}
```

```
void Imprimir_Caja(int x,int y, int ncajas)
{
    int fila,columna,i;
    int aux=x;    //Guarda la posicion del cursor
    for(fila=1; fila<=ALTO_CAJA; fila++)
    {
        x=aux;

        for(columna=1; columna<=ANCHO_CAJA; columna++)
        {
            Mover_Cursor(x,y); //Se posiciona en una coordenada
especificada
            if(y==9 && columna>1 && columna<10)
            {
```

```

        for(i=0; i<ncajas; i++)
        {
            if(x==(i*10)+2 && i<15)
            {
                printf("Caja #%d", i+1);
                x=x+5;
                columna=columna+8;
            }
            if(x==(i*10)+2 && i>=15)
            {
                printf("Caja #%d", i+1);
                x=x+5;
                columna=columna+9;
            }
        }
    }
    else
    printf("|");    //Margen de la caja
    Tiempo_Espera(TIEMPO_BASE);
    x++;    //Aumentando la posicion en la consola en el eje x
}
y++;    //Aumentando la posicion en la consola en el eje y
}
}

```

- supermercado.c

```

/*
PROGRAMA PRINCIPAL
Modo de compilación: gcc supermercado.c TADCola(Din)(Est).c Bib_Supermercado.c -o
archivo_ejecutable
Fecha: 01/04/2019
Version 1.0
Autores: Ayona Lopez Eugenio Milton, Benitez Morales Manuel Emilio
*/

```

```

#include <stdio.h>
#include "Bib_Supermercado.h"

int main()
{
    int n_cajeras, t_atencion, t_llegada, n_clientes;

    Borrar_Pantalla();

    printf("SuperESCOM\n\n");

    do
    {
        //Pide los datos hasta que todo sea correcto
        n_cajeras= Entrada_Cajeras(n_cajeras);
        t_atencion= Entrada_Atencion(t_atencion);
        t_llegada= Entrada_LlegadaCliente(t_llegada);
        n_clientes= N_Clientes(n_clientes);

        //Y sia lgo no es correcto, entonces pide precisamente que se ingresen de
nuevo
        if(n_cajeras==-1 || t_atencion==-1 || t_llegada==-1 || n_clientes==-1)
        {
            printf("\n\nVuelva a ingresar los datos, alguno de ellos sale de la
especificacion!!!\n\n");
        }
    }while(n_cajeras==-1 || t_atencion==-1 || t_llegada==-1 || n_clientes==-1);

    printf("\n\nTIENDA ABIERTA\n\n");

    Simulacion(n_cajeras, t_atencion, t_llegada, n_clientes);

    return 0;
}

```

Sistema operativo

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include "TADColaEst.h"
// #include "TADColaDin.h"
#include <windows.h>
#define T 1000
// #define T 1

void Ingresar_datos(cola *proc, int n);
void Imprimir_cola(cola *proc);
void Procesador(cola *proc, cola *fin);
void Imprimir_Proceso(elemento *i, cola *j);
int main(void)
{
    // Variable e inicializacion de colas
    cola procesos;
    cola finalizados;
    Initialize (&procesos);
    Initialize (&finalizados);
    elemento p,f;
    int np;
    cola *ColaP;
    cola *ColaF;
    ColaP = &procesos;
    ColaF = &finalizados;
```

```

printf("Ingrese el numero de procesos a ejecutar: "); //Solicitud inicial del numero de procesos
a ejecutar
scanf("%d",&np); // almacena el numero de procesos a ejecutar
Ingresar_datos(ColaP,np); //Llama a funcion para ingresar datos de los procesos
Procesador(ColaP, ColaF); //Llama a funcion para ejecutar procesos
Imprimir_cola(ColaF); //Llama a funcion para imprimir la lista de procesos finalizados
return 0;
}

```

//Funcion para recibir los datos principales del proceso

/*

void Ingresar_datos(cola *proc, int n);

Descripción: Ingresa los datos requerido para el funcionamiento de cada elemento

Recibe: cola *proc (Ya inicializada, cola de procesos) y int n (La cantidad de procesos que se van a ejecutar)

Devuelve:

Observaciones: El usuario a creado una cola y proc tiene la referencia a ella, si esto no ha pasado se ocasionara un error.

*/

void Ingresar_datos(cola *proc, int n){

 elemento p;

 int x;

 char nombre[45];

 char actividad[200];

 char id[45];

 int tiempo;

 for (x=0;x<n;x++){

```

printf("Ingrese datos del proceso %d \n",x+1 );
printf("\nIngrese nombre del proceso: ");
scanf ("%s",p.nombre);
printf("Ingrese actividad del proceso: ");
fflush (stdin);
gets(p.actividad); // Inseguro
//scanf("%s", p.actividad); //Funciona con Linux pero no guarda espacios
//fgets(p.actividad,200,stdin); //
//scanf("%[!\n]",p.actividad); //No funciona ni linux ni windows
fflush (stdin);
printf("\nIngrese id del proceso: ");
scanf ("%s",p.id);
printf("\nIngrese tiempo requerido del proceso: ");
scanf ("%d",&p.tiempo);
p.tiempoEnProceso=0;
Queue(proc,p);
printf("\n \n ");

}

return ;
}

```

//Funcion que realiza el trabajo de un nucleo del procesador

/*

void Procesador(cola *proc, cola *fin);

Descripción: Realiza el trabajo de proceso de los elementos, a traves de 2 colas, de finalizado y procesos

Recibe: cola *proc (Ya inicializada, cola de procesos) y cola *fin (Ya inicializada, cola de procesos finalizados)

Devuelve:

Observaciones: No administra prioridades, y una vez que inicia el procesamiento no se puede incluir otro proceso

*/

```
void Procesador(cola *proc,cola *fin){
    elemento i,o;
    elemento *aux;
    aux=&i;
    while (Empty(proc)!=TRUE){
        i=Dequeue(proc);
        Imprimir_Proceso (aux,proc);
        Sleep(T);
        i.tiempo--;
        i.tiempoEnProceso++;
        if (i.tiempo>0){
            i.tiempoEnProceso=i.tiempoEnProceso+Size(proc);
            Queue(proc,i);

        }
        else {
            Queue(fin,i);
        }
    }
    return ;
}
```

//Funcion que realiza la impresion del trabajo del procesador

/*

void Imprimir_Proceso(elemento *i,cola *j);

Descripción: imprime en pantalla el proceso anterior ejecutado, el siguiente y el actual

Recibe: cola *j (Ya inicializada, cola de procesos) y elemento *i (Elemento extraido del proceso actual)

Devuelve:

Observaciones: Recibe la cola de procesos para obtener los procesos siguientes y anterior
para trabajar con linux se requiere modificar lineas de clear

*/

void Imprimir_Proceso(elemento *i,cola *j){

 //system("clear"); //Para uso en Linux CLEAR

 system("cls"); //Para uso de Windows

 elemento s,a;

 s=Front(j);

 a=Final(j);

 printf("\tUltimo procesado:\n");

 printf("Nombre:\t%s\tID:\t%s\tTiempo restante:\t%i\n",a.nombre,a.id,a.tiempo);

 printf("\n\n");

 printf("\tProcesado:\n");

 printf("Nombre:\t%s",i->nombre);

 printf("\nActivad:\t");

 puts (i->actividad);

 printf("ID:\t %s",i->id);

 printf("\nTiempo en proceso:\t %d",i->tiempoEnProceso);

 printf("\n");

 printf("\n\n");

 printf("\tSiguiente a procesar:\n");

 printf("Nombre:\t%s\tID:\t%s\tTiempo restante:\t%i\n",s.nombre,s.id,s.tiempo);

```
return ;  
}
```

```
//Funcion que imprime los procesos finalizados  
/*
```

```
void ImprimirCola(cola *fin);
```

Descripción: imprime en pantalla los procesos finalizados y el orden con el que terminaron y el tiempo requerido

Recibe: cola *fin (Ya inicializada, cola de procesos finalizados)

Devuelve:

Observaciones: para trabajar con linux se requiere modificar lineas de clear

```
*/
```

```
void ImprimirCola(cola *fin){  
    elemento f;  
    //system("clear"); //Para uso en Linux  Clear  
    system("cls"); //Para uso de Windows  
    while (Empty(fin)!=TRUE){  
        f=Dequeue (fin);  
        printf("Nombre de proceso\t %s \n",f.nombre);  
        printf("Actividad del proceso:\t");  
        puts(f.actividad);  
        printf("ID del proceso\t %s\n",f.id);  
        //printf("Tiempo de proceso %i \n",f.tiempo);  
        printf("Proceso finalizado en:\t %i\n",f.tiempoEnProceso);  
        printf("*****Finalizado\n\n");  
    }  
    return ;  
}
```

Banco

/*Simulacion de la atencion de personas en un banco implementando el TAD Cola.

Realiza la simulacion de la atencion de personas con prioridades en una institucion bancaria.

Fecha: 01/04/2019

Version 1.1

Autores: Ayona Lopez Eugenio Milton,Emilio ,Juan*/

//LIBRERIAS

#include<stdio.h>

#include "TADColaEst.h"

//CONSTANTES

#define TIEMPO_BASE 100

#include "presentacion.h"

void GraficarCajas(int numero_cajeros);

//PROGRAMA PRINCIPAL

int main(void)

{

int tiempo_cajeros,numero_cajeros;

int tiempo_clientes; //tiempo de llegada de los clientes

int tiempo_usuarios; //tiempo de llegada de los usuarios

int tiempo_clipref; //tiempo de llegada de los clientes preferentes

int columna,aux;

int tamCola;

int i,j;

int cont_clientes, cont_usuarios, cont_clipref;

int carga=0,estado=0;

int cliente=0,usuario=0,clienteP=0;

int ClienteCola=0, UsuarioCola=0,Pcola=0;

cola Clientes, Usuarios, ClientesP;

elemento e;

Initialize(&Clientes);

Initialize(&Usuarios);

Initialize(&ClientesP);

unsigned int tiempo = 0;

BorrarPantalla();

printf("\nIngrese el numero de cajeros: ");

scanf("%d",&numero_cajeros);

```

//Verifica que los cajeros sean menor que 10
while(numero_cajeros<=0||numero_cajeros>10)
{
    BorrarPantalla();
    printf("\nNumero Incorrecto, ingrese otro:");
    scanf("%d",&numero_cajeros);
}
printf("\nTiempo de atencion en cajas:");
scanf("%d",&tiempo_cajeros);
printf("\nTiempo de llegada de clientes:");
scanf("%d",&tiempo_clientes);
printf("\nTiempo de llegada de los usuarios:");
scanf("%d",&tiempo_usuarios);
printf("\nTiempo de llegada de los usuarios preferentes:");
scanf("%d",&tiempo_clipref);
BorrarPantalla();
//Graficacion de los cajeros
GraficarCajas(numero_cajeros);
//Formacion de los clientes en la cola
while(1)
{
    EsperarMiliSeg(TIEMPO_BASE);

    tiempo++; //Aumenta el tiempo del reloj del banco

    //Si corresponde al tiempo de llegada de un cliente
    if(tiempo%tiempo_clientes==0)
    {
        ++ClienteCola;
        cliente++;
        //Encolando a un cliente
        e.n=cliente;
        Queue(&Clientes, e);
    }
}

```

```

//Si corresponde al tiempo de llegada de un Usuario
if(tiempo%tiempo_usuarios==0)
{
    ++UsuarioCola;
    usuario++;
    //Encolando a un usuario
    e.n=usuario;
    Queue(&Usuarios, e);
}
//Si corresponde al tiempo de llegada de un cliente preferente
if(tiempo%tiempo_clipref==0)
{
    ++Pcola;
    clienteP++;
    //Encolando a un clientepref
    e.n=clienteP;
    Queue(&ClientesP, e);
}

//ATENCION DE LAS PERSONAS FORMADAS

/*
Cuando el tiempo es igual al asignado en tiempo_cajero
Significa que la disponibilidad de los cajeros esta disponible
*/
if(tiempo%tiempo_cajeros==0)
{
    GraficarCajas(numero_cajeros);
    //Clientes y C.preferentes que han pasado
    for(i=0;i<numero_cajeros;i++)
        carga=0;    //se usa para contar los clientesP y normales que
han pasado por los cajeros
    //Cuando llega a cinco tiene que permitirle la entrada a un usuario del cliente

    for(j=0;j<numero_cajeros;j++)
    {

        estado=0;

```

```

        if(!Empty(&ClientesP))    //como la politica del banco dice que
los clientes Preferenciales se atienden en cualquier momento
    {
        //printf("Cliente:  %d    Usuario:  %d    Cliente  Prefe:
%d",ClienteCola,UsuarioCola,Pcola);

        e=Dequeue(&ClientesP);
        MoverCursor(36,13);
        printf("P%d",e.n);
        estado=1;
        ++carga;
        //
        getch();
    // BorrarPantalla();
    }
    if(!Empty(&Clientes)&&estado==0)
    {
        printf("Cliente:  %d    Usuario:  %d    Cliente  Prefe:
%d",ClienteCola,UsuarioCola,Pcola);
        e=Dequeue(&Clientes);
        MoverCursor(6,13);
        printf("C%d",e.n);
        estado=1;
        ++carga;
    }
    if(!Empty(&Usuarios) && estado==0 &&carga>5 )
    {
        e=Dequeue(&Usuarios);
        MoverCursor(21,13);
        printf("U%d",e.n);
        estado=1;
        carga=0;
    }
}

}

}

```

```

}
/*
void GraficarCajas(int numero_cajeros)
Descripcion: Funcion para graficar las cajas para las filas del banco
Recibe: Un entero int numero_cajeros que corresponden al numero de cajas que se van a generar
para la atencion
Observaciones: El numero deba estar contenido entre 0<numero_cajeros<11
*/
void GraficarCajas(int numero_cajeros)
{
    int j,i,k=0,columna,aux;
    //Dibujar caja a caja
    for(j=0;j<numero_cajeros;j++)
    {
        //Auxiliar para recorrer las posicion en x en la consola
        columna=(j*9)+3+k;
        aux=columna;
        //Espacios entre cajas
        k=k+3;
        //Dibujando la parte superior de las cajas
        while(columna<=aux+8)
        {
            MoverCursor(columna,3);
            printf("-");
            columna++;
        }
        columna--;
        //Especificamente se establece i=4 para la posicion en Y y variacion de la misma
        sin cambiar en X
        i=4;
        //Dibujando la parte derecha de las cajas
        while(i<=8)
        {
            MoverCursor(columna,i);
            printf("-");
            i++;
        }
        i--; //Colocacion correcta del cursor dentro del borde de la caja
        //Dibujando la parte inferior de las cajas
    }
}

```



```

while(columna>=aux)
{
    MoverCursor(columna,i);
    printf("-");
    columna--;
}
columna++;//Colocacion correcta del cursor dentro del borde de la caja
//Dibujando la parte izquierda de las cajas
while(i!=3)
{
    MoverCursor(columna,i);
    printf("-");
    i--;
}
//Nombre y numero de cada caja
MoverCursor(aux+1,4);
printf("CAJA %d",j+1);
}
MoverCursor(3,20);
printf("CLIENTES");
MoverCursor(18,20);
printf("USUARIOS");
MoverCursor(33,20);
printf("CLIENTES");
MoverCursor(32,21);
printf("PREFERENTES");
MoverCursor(12,23);
printf("PERSONAS ATENDIDAS");
}

```