

---

# Reporte de Proyecto

## Administrador de clientes con colas de prioridad



---

Instituto Politécnico Nacional

Escuela Superior de Cómputo

Materia: Estructuras de Datos

Grupo: 1CM2

Proyecto final, Administrador de clientes con colas de prioridad

Alumno:

Salinas Hernández Luis Angel      2016630359

Profesor:

Saucedo Delgado Rafael Norman

---

## Objetivo

Implementar en C la estructura de datos “Cola de prioridad”, así como sus funciones básicas para aplicarlas en un administrador de clientes para negocios pequeños.

El programa deberá ser capaz de:

1. Ingresar nuevos clientes junto con la prioridad para atender a cada uno.
2. Mostrar de manera ordenada a todos los clientes formados en el orden en el que deberán ser atendidos.
3. Indicar cuál es el siguiente cliente en atender y al hacerlo, deberá desformarlo.

En caso de tener dos o más clientes con la misma prioridad, se deberá seguir el principio fundamental del funcionamiento de las colas y atender al primero de ellos en llegar

Cabe recalcar que los primero clientes en ser atendidos serán los que tengan una prioridad mas cercana al cero. Ejemplo.

Un cliente con prioridad 1 deberá ser atendido antes que uno con prioridad 6

## Desarrollo

Para el programa se implementaron las funciones básicas de una cola de prioridad, además de algunas funciones complementarias, como un menú para el usuario.

```
void encolar(struct cola , char, int);  
void mostrar(struct cola);  
void ordenar_elementos(struct cola *);  
void insertar(struct cola *, char, int);  
void menu(void);  
void atender(struct cola *);
```

Se realizaron las debidas validaciones, en caso de que la cola con la que fuéramos a trabajar estuviera vacía, la función malloc no pudiera reservar la memoria para trabajar, etc.

```
if (q->adelante == NULL)
    puts("Cola Vacía");
    exit(0);
```

Validación por si se recibe un apuntador nulo

Para utilizar la función malloc correctamente, se aplicó un casting, como lo vimos en clase.

```
struct nodo *
crear_nodo(char cliente, int pr)
{
    struct nodo *nuevo_nodo = (struct nodo *)malloc(sizeof(struct nodo));
    nuevo_nodo->dato = cliente;
    nuevo_nodo->prioridad = pr;
    return nuevo_nodo;
}
```

casting de la función malloc

Ya que se tenían ingresados los clientes junto con su prioridad, había que ordenarlos, para eso utilizamos el algoritmo de ordenamiento por burbuja (bubble sort)

```
void
ordenar_elementos(struct cola *q)
{
    struct nodo *aux1;
    struct nodo *aux2;
    int p_auxiliar;
    int c_auxiliar;

    aux1 = q->adelante;

    while (aux1->siguiente != NULL) {
        aux2 = aux1->siguiente;

        while (aux2 != NULL) {
            if (aux1->prioridad > aux2->prioridad) {
                p_auxiliar = aux1->prioridad;
                c_auxiliar = aux1->dato;

                aux1->prioridad = aux2->prioridad;
                aux1->dato = aux2->dato;

                aux2->prioridad = p_auxiliar;
                aux2->dato = c_auxiliar;
            }
            aux2 = aux2->siguiente;
        }
        aux1 = aux1->siguiente;
    }
}
```

Función para ordenar elementos con bubble sort

---

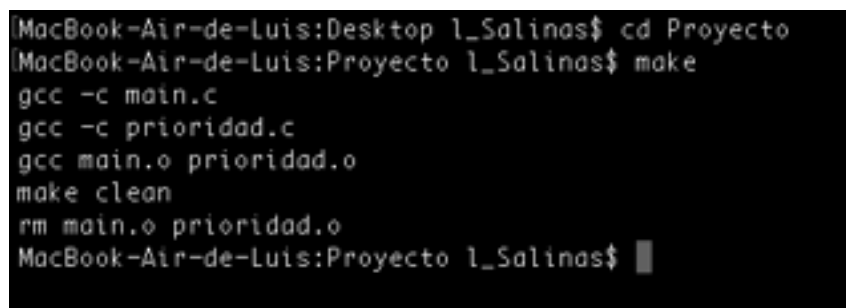
Para el menú, se dieron 4 opciones:

Ingresar cliente, Mostrar clientes, Atender y Salir, la cuales serán elegidas por el usuario mediante la estructura switch.

El menú se seguirá mostrando hasta que el usuario presione una opción diferente a 1, 2 o 3.


Para una mayor comodidad al trabajar con el programa en la terminal, se utilizó la función `system("clear")`.

Ahora mostramos capturas de pantalla de algunas pruebas con el programa:



```
MacBook-Air-de-Luis:Desktop l_Salinas$ cd Proyecto
MacBook-Air-de-Luis:Proyecto l_Salinas$ make
gcc -c main.c
gcc -c prioridad.c
gcc main.o prioridad.o
make clean
rm main.o prioridad.o
MacBook-Air-de-Luis:Proyecto l_Salinas$
```

Ejecución correcta del comando make



```
MacBook-Air-de-Luis:Proyecto l_Salinas$ ./a.out

COLA DE PRIORIDAD
-----
1. INGRESAR CLIENTE
2. MOSTAR CLIENTES
3. ATENDER
4. SALIR
INGRESE OPCION:
█
```

Menú de inicio del programa

```
COLA DE PRIORIDAD
-----
1. INGRESAR CLIENTE
2. MOSTAR CLIENTES
3. ATENDER
4. SALIR
INGRESE OPCION:
1
Ingreso inicial del cliente:
q
Ingreso prioridad del cliente:
1
```

```
COLA DE PRIORIDAD
-----
1. INGRESAR CLIENTE
2. MOSTAR CLIENTES
3. ATENDER
4. SALIR
INGRESE OPCION:
1
Ingreso inicial del cliente:
a
Ingreso prioridad del cliente:
1
```

```
COLA DE PRIORIDAD
-----
1. INGRESAR CLIENTE
2. MOSTAR CLIENTES
3. ATENDER
4. SALIR
INGRESE OPCION:
1
Ingreso inicial del cliente:
j
Ingreso prioridad del cliente:
6
```

```
COLA DE PRIORIDAD
-----
1. INGRESAR CLIENTE
2. MOSTAR CLIENTES
3. ATENDER
4. SALIR
INGRESE OPCION:
1
Ingreso inicial del cliente:
l
Ingreso prioridad del cliente:
2
```

Pruebas con la primera función para ingresar a un nuevo cliente

Ahora usamos la segunda opción para ver a los clientes en el orden en el que deberán ser atendidos (nótese que a pesar de que dos de los clientes ingresados tenían la misma prioridad, el segundo criterio que se usó para ordenarlo fue quién de ellos llegó antes)

```
2
```

Cliente	Prioridad
q	1
a	1
l	2
j	6

Posteriormente probamos la tercera opción, la cual nos dice cuál debe ser el siguiente cliente en ser atendido y luego lo desforma.

```
atender a: q
COLA DE PRIORIDAD
-----
1. INGRESAR CLIENTE
2. MOSTAR CLIENTES
3. ATENDER
4. SALIR
INGRESE OPCION:
█
```

```
INGRESE OPCION:
2
```

Cliente	Prioridad
a	1
l	2
j	6

Podemos ver que el primer cliente en ser atendido deberá ser 'q' y después veremos de nuevo el menú, al presionar la opción de mostrar clientes, notamos que 'q' ya no está

---

Utilizamos esta opción hasta atender al último de los clientes y la usamos de nuevo para ver que pasa.

```
atender a: j
COLA DE PRIORIDAD
-----
1. INGRESAR CLIENTE
2. MOSTAR CLIENTES
3. ATENDER
4. SALIR
INGRESE OPCION:
3
Cola Vacía
MacBook-Air-de-Luis:Proyecto 1_Salinas$
```

Nos damos cuenta de que después de atender al último cliente y tratar de usar la opción de atender nuevamente, el programa nos indica que la cola está vacía y ya no hay más clientes

Con estas pruebas comprobamos que el programa funciona correctamente según lo requerido.