1. Investigar cómo se hace un **Queue** y sus operaciones correspondientes en C++. Hacer un ejemplo completo.

Una cola es una estructura de datos que almacena elementos en una lista y permite acceder a los datos por uno de los dos extremos de la lista. Un elemento se inserta en la cola (parte final) de la lista y se suprime o elimina por la frente (parte inicial, cabeza) de la lista.

Los elementos se eliminan de la cola en el mismo orden en que se almacenan y, por consiguiente, una cola es una estructura de tipo FIFO (first-in first-out).

Las acciones que están permitidas en una cola son:

- Creación de una cola vacía.
- Verificación de que una cola este vacía.
- Añadir un dato al final de una cola.
- Eliminación de los datos al inicio de la cola.

Ejemplo de una cola en la vida real

Un ejemplo cotidiano de una cola es al momento de pagar en una tienda comercial, a continuación, se explicará cómo se pueden emplear las operaciones básicas de una cola.

- Creación de una cola: en una tienda la cola se crea una vez que hay un trabajador que atienda la caja, si no existiera este trabajador no habría una fila para atender.

Ilustración 1 Cola no creada

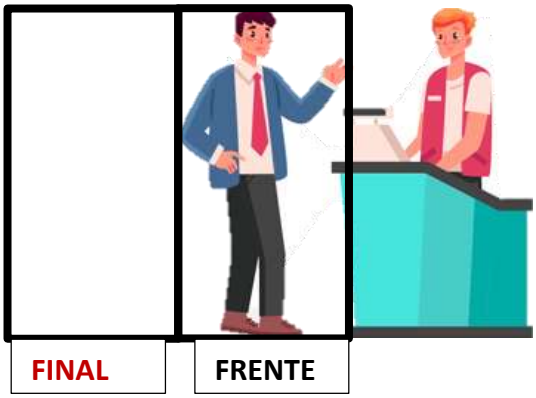
Ilustración 2 Cola creada

- Verificación de que una cola este vacía: para esta operación solo es cuestión de ver si hay clientes en la fila, si no hay ninguno podemos decir que la fila está vacía.

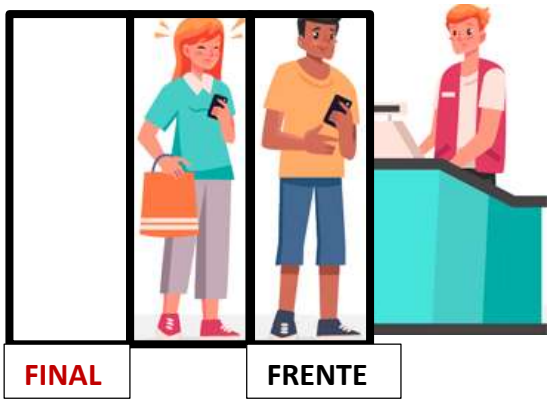
Ilustración 3 Cola vacía

Ilustración 4 Cola con un elemento

- Añadir un dato al final de una cola: la inserción de un dato (en este caso un cliente) se realizará por la parte final



- Eliminación de los datos al inicio de la cola: para eliminar un dato, este se eliminará por la parte principal, en este caso podemos decir que una vez atendido el cliente será sacado de la fila.



Ahora bien, en programación las colas se pueden implementar utilizando arrays o listas enlazadas. Para este caso se utilizará listas enlazadas

Una lista enlazada consta de un número de nodos con dos campos, un valor de cualquier tipo y un enlace al siguiente nodo. Para este ejemplo se hará una cola de números enteros.

```
 //Nodo de una lista enlazada
struct Nodo{
    int dato;
    Nodo *siguiente;
};
```

Para insertar un nodo a la cola se necesita crear un nuevo nodo, una vez creado se le asignara un valor, después el campo de siguiente apuntara a NULL. Ahora, para unir un nodo con otro nodo se verificará si la cola está vacía, si está vacía el nodo creado será igual a frente y a fin, en caso contrario el nuevo nodo será enlazado con el nodo final.

```
void insertarQ(Nodo *&frente, Nodo *&final , int n){
    Nodo *nuevoNodo = new Nodo();
    nuevoNodo -> dato = n;
    nuevoNodo -> siguiente = NULL;
    if(colaVacia(frente)){
        frente = nuevoNodo;
    }else{
        final -> siguiente = nuevoNodo;
    }
    final = nuevoNodo;
    printf("Elemento insertado\n");
}
```

Para eliminar un nodo de la cola se necesita saber el valor a eliminar, además de tener un nodo auxiliar que permita copiar el nodo de enfrente (nodo a eliminar), antes de eliminar un dato debemos verificar si hay elementos en la cola, si solo existe el elemento a eliminar debemos de asignar frente y fin a NULL, en caso contrario debemos asignar frente al siguiente nodo de modo que el nodo segundo sea ahora el frente y fin del nodo.

```
void eliminarNodo(Nodo *&frente, Nodo *&final , int &n){
    n = frente -> dato;
    Nodo *aux = frente;

    if(frente == final){
        frente = NULL;
        final = NULL;
    }else{
        frente = frente ->siguiente;
    }
}
```

Algoritmo completo

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

 //Nodo de una lista enlazada
struct Nodo{
    int dato;
    Nodo *siguiente;
};

//Prototipos de función
void insertarQ(Nodo *&,Nodo *&, int);          //Introduce un nodo a la cola
void eliminarNodo(Nodo *&,Nodo *&, int &);    // Elimina un nodo de la cola
bool colaVacia(Nodo *);                      // Permite saber si una cola tiene elementos
```

```
int main(){
    int dato;
    // Declracion de punteros que indican el inicio y fin de una cola
    Nodo *frente = NULL;
    Nodo *final = NULL;

    printf("inserte un elemento a la cola\n");
    scanf("%d",&dato);
    insertarQ(frente, final, dato);

    printf("inserte un elemento a la cola\n");
    scanf("%d",&dato);
    insertarQ(frente, final, dato);

    printf("inserte un elemento a la cola\n");
    scanf("%d",&dato);
    insertarQ(frente, final, dato);

    printf("Eliminacion de un elemento de la cola\n");
    while(frente !=NULL){
        eliminarNodo(frente, final, dato);
        if(frente != NULL){
            printf("%d ,",dato);
        }else{
            printf("%d .",dato);
        }
    }

}
```

```
45  void insertarQ(Nodo *&frente, Nodo *&final , int n){
46      Nodo *nuevoNodo = new Nodo();
47      nuevoNodo -> dato = n;
48      nuevoNodo -> siguiente = NULL;
49      if(colaVacia(frente)){
50          frente = nuevoNodo;
51      }else{
52          final -> siguiente = nuevoNodo;
53      }
54      final = nuevoNodo;
55      printf("Elemento insertado\n");
56  }
57
58  void eliminarNodo(Nodo *&frente, Nodo *&final , int &n){
59      n = frente -> dato;
60      Nodo *aux = frente;
61
62      if(frente == final){
63          frente = NULL;
64          final = NULL;
65      }else{
66          frente = frente ->siguiente;
67      }
68  }
69
70  bool colaVacia(Nodo *frente){
71      //Condicion if de una solo linea
72      return (frente == NULL)? true: false;
73  }
```