UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE CIENCIAS DE LA COMPUTACIÓN

LABORATORIO 12 – Punteros, POO y Colas.

DOCENTE:

Enzo Edir Velásquez Lobatón

ALUMNO:

Owen Haziel Roque Sosa.

FECHA:

19/06/2022

Arequipa – Perú

```cpp
#define COLA_H
#include <iostream>
#include <vector>
using namespace std;

template <class T>
class Cola;

template <class T>
class Nodo {
public:
    Nodo<T>(T, Nodo* =nullptr);
    friend class Cola<T>;
private:
    T value;
    Nodo* sig;
};

template <class T>
Nodo<T>::Nodo(T _value, Nodo* _sig) {
    sig = _sig;
    value = _value;
}

template <class T>
class Cola {
public:
    Cola<T>();
    ~Cola<T>();
    void insertar(T);
    void eliminar();
    void imprimir();
    void search(T);
    unsigned int getSize();
private:
    Nodo<T> * first = nullptr;
    unsigned int size = 0;
};
```

Clase Nodo y Clase Lista

```cpp
//constructor
template <class T>
Cola<T>::Cola(){}

//destructor
template <class T>
Cola<T>::~Cola() {
    Nodo<T> *aux = first;
    Nodo<T> *delete_node;
    while(aux != nullptr) {
        delete_node = aux;
        aux = aux->sig;
        delete delete_node;
    }
    first = nullptr;
    size = 0;
}

template <class T>
    unsigned int Cola<T>::getSize(){
    return size;
}

// imprimir Cola
template <class T>
    void Cola<T>::imprimir(){
    Nodo<T> *aux = first;
    while(aux != nullptr) {
        cout << aux->value << " -> ";
        aux = aux->sig;
    }
    cout << endl;
}
```

Métodos auxiliares: Constructor, Destructor, Imprimir

1. Defina una Cola que permita insertar elementos utilizando clases.

```
// ej 1
template <class T>
void Cola<T>::insertar(T val){
    Nodo<T> *nuevo = new Nodo<T>(val);
    if (first == nullptr) { // si es el primer elemento
        nuevo->sig = nullptr; // a añadir a la cola
        first = nuevo;
    }
    else { // si hay elementos en la cola, agrega
        nuevo->sig = first; //en primera posicion (first)
        first = nuevo;
    }
    size++;
}
```

```
#include<iostream>
#include "Cola.h"
using namespace std;

int main (int argc, char *argv[]) {
    Cola<float> B;
    B.insertar(2.76);
    B.insertar(5.14);
    B.insertar(7.2);
    B.insertar(4.35);
    B.insertar(8.88);
    B.insertar(12.03);
    B.imprimir();
    return 0;
}
```

```
12.03 -> 8.88 -> 4.35 -> 7.2 -> 5.14 -> 2.76 ->


<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

2. Sobre el ejercicio anterior, adecue el programa para eliminar elementos de una Cola.

```cpp
// ej 2
template <class T>
void Cola<T>::eliminar(){
    if (first == nullptr) {
        cout << "No existen elementos en la Cola.\n";
        return;
    }
    else {
        Nodo<T> * delete_node = first; // asigno delete a first
        for(unsigned int idx = 0; idx < size-2; idx++){ // recorre
            delete_node = delete_node->sig; // hasta el penultimo
        }
        delete_node->sig = nullptr; // establece que es el ultimo, apunta a nada
        delete delete_node->sig;
    }
    size--;
}
```

```cpp
#include<iostream>
#include "Cola.h"
using namespace std;

int main (int argc, char *argv[]) {
    Cola<float> B;
    B.insertar(2.76);
    B.insertar(5.14);
    B.insertar(7.2);
    B.insertar(4.35);
    B.insertar(8.88);
    B.insertar(12.03);
    B.insertar(0.54);
    B.imprimir();
    B.eliminar();
    B.imprimir();
    B.eliminar();
    B.eliminar();
    B.imprimir();
    return 0;
}
```

```
0.54 -> 12.03 -> 8.88 -> 4.35 -> 7.2 -> 5.14 -> 2.76 ->
0.54 -> 12.03 -> 8.88 -> 4.35 -> 7.2 -> 5.14 ->
0.54 -> 12.03 -> 8.88 -> 4.35 ->


<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

3. Implemente un algoritmo para buscar elementos de la Cola.

```cpp
// ej 3
template <class T>
void Cola<T>::search(T val){
    // lista de posiciones
    std::vector<int> res;
    unsigned int idx=0;
    Nodo<T> * aux = first;
    for (;idx < size; idx++){
        if (aux->value == val){
            res.push_back(idx);
        }
        // recorrer al siguiente nodo
        aux = aux->sig;
    }
    if (res.empty())
        std::cout << "No se encontró el valor en la Cola.\n";
    else  {
        for (unsigned int i=0; i<res.size(); i++){
            std::cout << val << " encontrado en posicion ";
            std::cout << res.at(i) << "\n";
        }
        std::cout << std::endl;
    }
}
```

```cpp
#include<iostream>
#include "Cola.h"
using namespace std;

int main (int argc, char *argv[]) {
    Cola<float> B;
    B.insertar(2.75);
    B.insertar(12.03);
    B.insertar(7.2);
    B.insertar(12.03);
    B.insertar(8.88);
    B.insertar(12.03);
    B.insertar(7.2);
    B.insertar(8.88);
    B.insertar(8.88);
    B.imprimir();
    B.search(8.88);
    B.search(12.03);
    return 0;
}
```

```
8.88 -> 8.88 -> 7.2 -> 12.03 -> 8.88 -> 12.03 -> 7.2 -> 12.03 -> 2.75 ->
8.88 encontrado en posicion 0
8.88 encontrado en posicion 1
8.88 encontrado en posicion 4

12.03 encontrado en posicion 3
12.03 encontrado en posicion 5
12.03 encontrado en posicion 7
```

4. Escribir un programa que permita comparar las edades de diferentes elementos. Debe utilizar el formato de colas en clases. Se debe definir· inicialmente el número de elementos y valores de cada cola (pudiendo ser de diferentes tamaños ej. 2-3). Se evaluará el resultado de la comparación de los primeros elementos de las colas, realizada en un número de iteraciones 'n', en cada iteración se debe realizar el procedimiento de inserción y eliminación de elementos (rotando los elementos definidos inicialmente)


```
// cambios segun al ej 4
template <class T>
void Cola<T>::reset(){ // envia el first al ultimo, recorre first
    Cola<T>::insertarFinal(first->value);
    Cola<T>::eliminarInicio();
}
```

```
template <class T>
void Cola<T>::insertarFinal(T val){
    Nodo<T> *nuevo = new Nodo<T>(val);
    nuevo->sig = nullptr;
    Nodo<T> * aux = first;
    while(aux->sig!=nullptr){ // recorre actualizando
        aux = aux->sig; // el valor de aux hasta llegar
    } // al ultimo valor, que apunta a nullptr por defecto
    aux->sig = nuevo;
}

template <class T>
void Cola<T>::eliminarInicio(){
    Nodo<T> * delete_node = first; // asigno delete a first
    first = first->sig; // first recorre al siguiente
    delete delete_node;
}
```

```cpp
#include "Cola.h"
using namespace std;
// ej 4
int main (int argc, char *argv[]) {
    int size_A, size_B, it=0;
    Cola<int> A, B;
    cout << "Ingrese tamanho de colas a comparar: ";
    cin >> size_A >> size_B;
    cout<<"Cola A: \n";
    fill(A,size_A);
    cout<<"Cola B: \n";
    fill(B,size_B);
    cout << endl;

    A.imprimir();
    B.imprimir();

    cout << "Nro de bucles: ";
    cin >> it;

    for (int i = 0; i < it; i++){
        int head_A = A.getFirstValue();
        int head_B = B.getFirstValue();
        if (head_A > head_B){
            A.reset();
        } else if (head_A == head_B){
            A.reset();
            B.reset();
        } else {
            B.reset();
        }
    }
    // para recuperar el mayor de la ultima iteración
    // y no resetear la cola
    cout << "Comparacion final\n";
    int res;
    A.imprimir();
    B.imprimir();
    int head_A = A.getFirstValue();
    int head_B = B.getFirstValue();
    if (head_A > head_B){
        res = head_A;
    } else if (head_A == head_B){
        res = head_A;
    } else {
        res = head_B;
    }
    cout << "Mayor luego de " << it << " iteraciones: " << res << endl;
    return 0;
```

```
Ingrese tamanho de colas a comparar: 2
3
Cola A:
Valor 1: 15
Valor 2: 20
Cola B:
Valor 1: 23
Valor 2: 18
Valor 3: 13

20 -> 15 ->
13 -> 18 -> 23 ->
Nro de bucles: 3
Comparacion final
15 -> 20 ->
13 -> 18 -> 23 ->
Mayor luego de 3 iteraciones: 15
```