




PARTE PRÁCTICA

PROFESOR: Alejandro Lara Caballero

ALUMNO: Leonardo Aguilar Martínez

MATRICULA: 2203025005



E S T R U C T U R A
D E D A T O S
L I N E A L E S

Void invierte () .Invierte el contenido de una pila, de forma que el elemento que esté en el fondo ahora sea el tope. En el proceso de inversión, quita todos los elementos repetidos. Emplea una pila mediante arreglos.

```
C ColaDinamica.h U • C Secc1_2_PILA.h U X
Estructuras de datos lineales > Segundo examen > Seccion1 > C Secc1_2_PILA.h
81 //----- EJERCICIO DE INVERSION-----//
82 template <typename T>
83 void PilaA<T>::invierte(){
84     int aux, j = 0;
85     int array[MAX];
86
87     for(int i=tope-1; i>=0 ; i--){
88         array[j] = arreglo[i];
89         j++;
90     }
91     for(int i=0; i<tope; i++){
92         arreglo[i] = array[i];
93     }
94 }
```

Explicación: La función utiliza un template para utilizar cualquier tipo de dato, es una función vacía de la clase PilaA, utiliza una variable para iterar y un arreglo para apoyarse en el cambio de orden.

El for inicial nos permite rellenar desde el tope hasta la base la pila inicial y apoyarnos con el arreglo definido. Más adelante, solo nos queda guardar esos valores en el arreglo principal.

Void invierte ().Invierte el contenido de una cola, de forma que el elemento que esté en el frente ahora sea la cola. En el proceso de inversión, quita todos los elementos repetidos. Emplea una cola mediante arreglos.

```
//----- Inversión de cola -----//
template <typename E>
void ColaArreglo<E>::invierte(){
    int arrayAux[fin+1], j=0;

    for(int i=fin-1; i>=0; --i){
        arrayAux[j] = elementos[i];
        j++;
    }

    for(int i=0; i<fin; i++){
        elementos[i] = arrayAux[i];
    }
}
```

Explicación: Basado en la función anterior, ambas estructuras están definidas por arreglos, es por ello que haciendo los correspondientes cambios para este metodo, podemos reutilizar código. La lógica es la misma.

Void sustituye(), la cual modifique la pila p, para que si valor original aparece en algún lugar de la pila, sea reemplazado por nuevo.

```
8
9 //----- Función de intercambio de datos -----/
0 template<typename E>
1 void PilaDinamica<E>::sustituye(int original, int nuevo){
2     Nodo<E> *p = this->getTope();
3     while (p != nullptr) {
4         if(p->getElemento() == original){
5             p->setElemento(nuevo);
6         }
7     }
8 }
9
```

Explicación: Al trabajar con este metodo de la clase PilaDinamica es necesario usar un nodo para iterar en la estructura, este nodo inicia desde el tope de la pila y lo recorre para comparar e intercambiar el elemento por el nuevo.

a) **Void sustituye()**, la cual modifique la cola c, para que si valor original aparece en algún lugar de la cola, sea reemplazado por nuevo.

```
1 //----- Funcion de intercambio de datos -----//
2 template<typename E>
3 void QueueDinamica<E>::sustituye(int original, int nuevo){
4     Nodo<E> *p = this->getTope();
5     while (p != nullptr) {
6         if(p->getElemento() == original){
7             p->setElemento(nuevo);
8         }
9         p = p->setSiguiente();
10    }
11 }
```

Explicación: Este metodo trabaja igual que el anterior, ya que ambas estructuras están hechas a base de nodos. Es por ello que de igual forma se itera la cola y se sustituyen los valores en caso de coincidir.