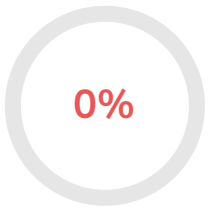
Plagiarism Scan Report

Report Generated on: Jun 10,2022



Plagiarised

Unique

Total Words:	**316**
Total Characters:	**2903**
Plagiarized Sentences:	**0**
Unique Sentences:	**0 (100%)**

Content Checked for Plagiarism

#include
#include
#include "Nodo.h"

// clase lista doble circular
using namespace std;
template
class Lista
{
private:
int dimension;
Nodo *cabeza = nullptr;
Nodo *cola = nullptr;
Nodo *obtenerUltimoNodo();
Nodo *obtenerNodo(int indice);

public:
Lista() = default;
void insertar(T dato);
void Buscar();
void recorrer(function callback);
void recorrerNodos(function *nodo)> callback);
void vaciarLista();
void insertEntre(T dato, int indice);
void eliminarNodo(int indice);
void imprimirListaInicio();
void imprimirListaFin();
};

template
void Lista::insertar(T dato)
{
Nodo *nuevo = new Nodo(dato);

if (cabeza == nullptr)
{
this->cabeza = nuevo;
this->cola = nuevo;
this->cabeza->siguiente = cabeza;
this->cabeza->anterior = cola;
}
else

```
{
this->cola->siguiente = nuevo;
nuevo->anterior = this->cola;
nuevo->siguiente = this->cabeza;
this->cola = nuevo;
this->cabeza->anterior = this->cola;
}

dimension++;
}

template
void Lista::Buscar()
{

Nodo *actual = this->cabeza;

if (this->cabeza != nullptr)
{
do
{
cout << actual->dato << "->";

actual = actual->siguiente;
} while (actual != this->cabeza);
}
else
{
cout << "\nLista Vacia";
}
}

template
void Lista::recorrer(function callback)
{

Nodo *actual = this->cabeza;

if (this->cabeza != nullptr)
{
do
{
callback(actual->dato);
actual = actual->siguiente;
} while (actual != this->cabeza);
}
else
{
cout << "\nLista Vacia";
}
}

template
void Lista::vaciarLista(){
cabeza = nullptr;
cola = nullptr;
}

template
```

```
void Lista::eliminarNodo(int indice)
{
}

// Imprime los datos desde el primero al ultimo
template
void Lista::imprimirListaInicio()
{
Nodo *actual = this->cabeza;

if (this->cabeza != nullptr)
{
do
{
cout << actual->dato << "->";

actual = actual->siguiente;
} while (actual != this->cabeza);
}
else
{
cout << "\nLista Vacia";
}
}

template
void Lista::imprimirListaFin()
{
Nodo *actual = this->cola;

if (this->cabeza != nullptr)
{
do
{
cout << actual->dato << "->";

actual = actual->anterior;
} while (actual != this->cola);
}
else
{
cout << "\nLista Vacia";
}
}

template
Nodo *Lista::obtenerUltimoNodo()
{
}

template
Nodo *Lista::obtenerNodo(int indice)
{
}
```

No Plagiarism Found