



INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA  
PARAÍBA  
Campus João Pessoa



**CNPq**  
Conselho Nacional de Desenvolvimento  
Científico e Tecnológico

## Projeto Olímpico de Programação

# [ C++ ] STL: Container List

# STL: Container List

- Lista duplamente encadeada;
- Permite acesso sequencial aos elementos armazenados;
- **Complexidade das operações básicas:**

| Operação                                   | Complexidade |
|--|--------------|
| Inserção e remoção no início, meio e final | $O(1)$       |
| Busca                                      | $O(n)$       |

- Iteradores são válidos após alteração na lista;
- Inserção no código:

```
#include <list>
```

- Declaração:

```
list<int> lista;
```

# STL: Container List

## assign

```
#include <iostream>

#include <list>

using namespace std;

void print_lista(list<int> lista);

int main(){
    list<int> lista;
    int numeros[] = {16,15,14,13,12};

    lista.assign(numeros, numeros + 5);

    print_lista(lista);
}

void print_lista(list<int> lista){
    list<int>::iterator it;

    for (it = lista.begin(); it != lista.end(); ++it){
        cout << *it << ' ';
    }
    cout << endl;
}
```



# STL: Container List

## clear e erase

```
#include <iostream>

#include <list>

using namespace std;

void print_lista(list<int> lista);

int main(){
    list<int> lista;
    int numeros[] = {16,15,14,13,12,11,10,9};

    lista.assign(numeros, numeros + 8);
    print_lista(lista);

    lista.erase(lista.begin());
    print_lista(lista);

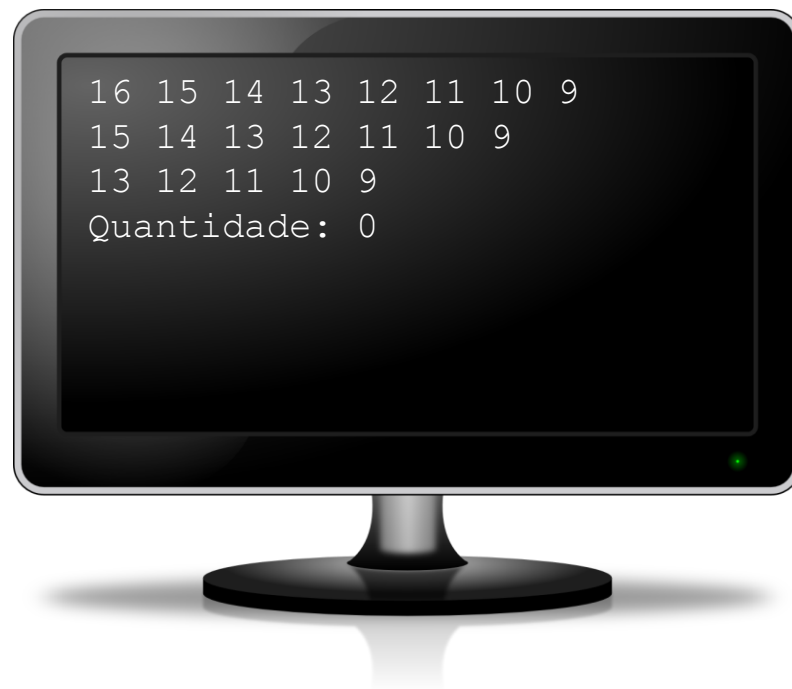
    lista.erase(lista.begin(), +++lista.begin());
    print_lista(lista);

    lista.clear();
    cout << "Quantidade: " << lista.size() << endl;

    return 0;
}

void print_lista(list<int> lista){
    list<int>::iterator it;

    for (it = lista.begin(); it != lista.end(); ++it){
        cout << *it << ' ';
    }
    cout << endl;
}
```



# STL: Container List

## insert

```
#include <iostream>
#include <list>

using namespace std;

void print_lista(list<int> lista);

int main(){
    list<int> lista;

    lista.insert(lista.begin(), 16);
    print_lista(lista);

    lista.insert(lista.begin(), 2, 17);
    print_lista(lista);

    lista.insert(lista.end(), 15);
    print_lista(lista);

    lista.insert(lista.begin(), lista.begin(), lista.end());
    print_lista(lista);

    return 0;
}

void print_lista(list<int> lista){
    list<int>::iterator it;

    for (it = lista.begin(); it != lista.end(); ++it){
        cout << *it << ' ';
    }
    cout << endl;
}
```



16  
17 17 16  
17 17 16 15  
17 17 16 15 17 17 16 15

# STL: Container List

## push\_back

```
#include <iostream>
#include <list>

using namespace std;

void print_lista(list<int> lista);

int main(){
    list<int> lista;

    lista.push_back(10);
    lista.push_back(20);
    lista.push_back(30);
    lista.push_back(40);
    print_lista(lista);

    return 0;
}

void print_lista(list<int> lista){
    list<int>::iterator it;

    for (it = lista.begin(); it != lista.end(); ++it){
        cout << *it << ' ';
    }
    cout << endl;
}
```



# STL: Container List

## push\_front

```
#include <iostream>
#include <list>

using namespace std;

void print_lista(list<int> lista);

int main(){
    list<int> lista;

    lista.push_front(10);
    lista.push_front(20);
    lista.push_front(30);
    lista.push_front(40);
    print_lista(lista);

    return 0;
}

void print_lista(list<int> lista){
    list<int>::iterator it;

    for (it = lista.begin(); it != lista.end(); ++it){
        cout << *it << ' ';
    }
    cout << endl;
}
```



# STL: Container List

## pop\_back

```
#include <iostream>
#include <list>

using namespace std;

int main(){
    list<int> lista;
    int numeros[] = {16,15,14,13,12};

    lista.assign(numeros, numeros + 5);

    while (!lista.empty()){
        cout << lista.back() << endl;
        lista.pop_back();
    }

    return 0;
}
```





# STL: Container List

## pop\_front

```
#include <iostream>
#include <list>

using namespace std;

int main() {
    list<int> lista;
    int numeros[] = {16, 15, 14, 13, 12};

    lista.assign(numeros, numeros + 5);

    while (!lista.empty()) {
        cout << lista.front() << endl;
        lista.pop_front();
    }

    return 0;
}
```



# STL: Container List

## Operadores Relacionais

```
#include <iostream>
#include <list>

using namespace std;

int main(){
    list<int> l1, l2;

    l1.push_back(10);
    l1.push_back(20);

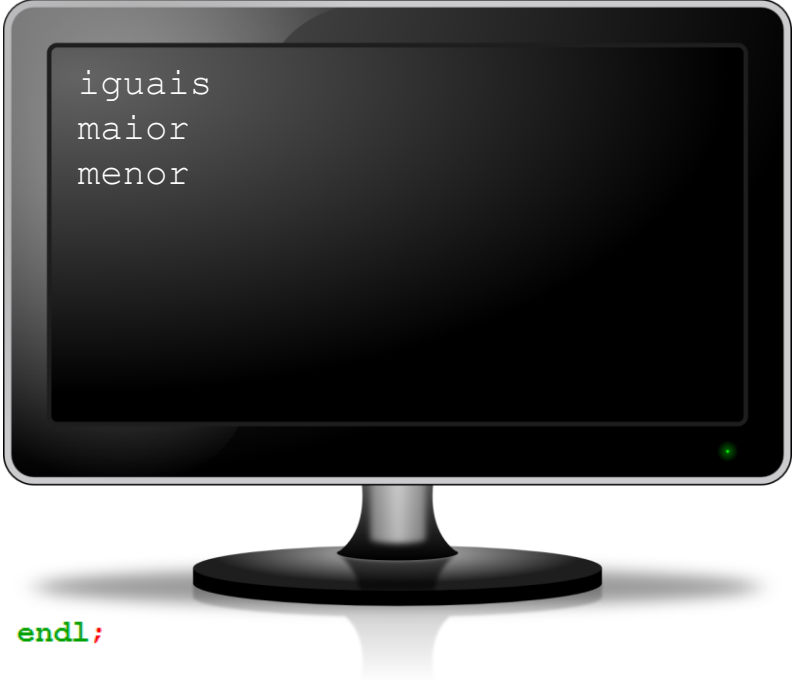
    l2.push_back(10);
    l2.push_back(20);

    cout << ((l1 == l2)?("iguais"):(("diferentes"))) << endl;

    l1.front() = 12;
    cout << ((l1 < l2)?("menor"):(("maior"))) << endl;

    l2.front() = 14;
    cout << ((l1 < l2)?("menor"):(("maior"))) << endl;

    return 0;
}
```



iguais  
maior  
menor

# STL: Container List

## unique

```
#include <iostream>
#include <list>

using namespace std;

void print_lista(list<int> lista);

int main() {
    list<int> lista;

    lista.push_front(10);
    lista.push_front(10);
    lista.push_front(20);
    lista.push_front(20);
    lista.push_front(30);
    lista.push_front(30);
    lista.push_front(20);
    lista.push_front(20);
    print_lista(lista);
    lista.unique();
    print_lista(lista);

    return 0;
}

void print_lista(list<int> lista) {
    list<int>::iterator it;

    for (it = lista.begin(); it != lista.end(); ++it) {
        cout << *it << ' ';
    }
    cout << endl;
}
```



20 20 30 30 20 20 10 10  
20 30 20 10

# STL: Container List

sort

```
#include <iostream>
#include <set>
#include <list>

using namespace std;

int main(){
    list<int> lista;
    set<int> dados;
    set<int>::iterator it;

    lista.push_back(10);
    lista.push_back(10);
    lista.push_back(20);
    lista.push_back(20);
    lista.push_back(30);
    lista.push_back(30);
    lista.push_back(10);
    lista.push_back(10);

    dados.insert(lista.begin(), lista.end());
    dados.insert(dados.begin(), 10);
    dados.insert(10);

    for (it = dados.begin(); it != dados.end(); ++it){
        cout << *it << endl;
    }

    return 0;
}
```



# STL: Container List

## Desafio!

Escrever um programa, em C++, para obter vários números, encerrando quando for digitado “0” (zero). O programa deverá remover todos os valores duplicados.

Calcular e exibir:

- Menor e maior valores digitados;
- Média dos números lidos;
- Remover os elementos que possuem valor abaixo da média.

# STL: Container List

Desafio! Resposta

