

STL - Standard Template Library

Bixecamp 2025

09 de maio de 2025



O que é a STL?

Biblioteca padrão do C++

- Contém funções e classes prontas para algoritmos e estruturas de dados
- Evita reinventar a roda

O que é a STL?

Biblioteca padrão do C++

- Contém funções e classes prontas para algoritmos e estruturas de dados
- Evita reinventar a roda

Componentes da STL

- **Containers:** estruturas que armazenam coleções de dados
- **Iteradores:** ponteiros para percorrer containers
- **Algoritmos:** funções que operam sobre containers

O que é um container?

Um container é como uma caixa organizadora

- Armazena elementos (objetos)
- Fornece funções para inserir, acessar, remover e percorrer elementos

O que é um container?

Um container é como uma caixa organizadora

- Armazena elementos (objetos)
- Fornece funções para inserir, acessar, remover e percorrer elementos

Containers que veremos hoje

- `vector`
- `set`
- `map`

Vector

```
vector<tipo> meuVector;
```

Array melhorada com tamanho dinâmico

- Acesso por colchetes, como em arrays

```
vector<int> vec;  
vec.push_back(5);  
cout << vec[0]; // 5
```

Vector

```
vector<tipo> meuVector;
```

Array melhorada com tamanho dinâmico

- Acesso por colchetes, como em arrays

```
vector<int> vec;  
vec.push_back(5);  
cout << vec[0]; // 5
```

push_back(x)	Insere x no final $O(1)$
pop_back()	Remove o último elemento $O(1)$
size()	Retorna a quantidade de elementos $O(1)$
empty()	Verifica se o vector está vazio $O(1)$
begin()	Iterador para o primeiro elemento $O(1)$
end()	Iterador para a posição após o último elemento $O(1)$

Exercício 1

Ler números até que 0 seja inserido. Imprimir todos e a quantidade de números.

Exercício 1

Ler números até que 0 seja inserido. Imprimir todos e a quantidade de números.

```
vector<int> numeros;

int input;
cin >> input;

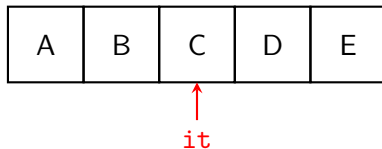
while (input != 0) {
    numeros.push_back(input);
    cin >> input;
}

cout << "foram inseridos " << numeros.size();
cout << " numeros" << endl;

for (int i = 0; i < numeros.size(); ++i) {
    cout << numeros[i] << " ";
}
```

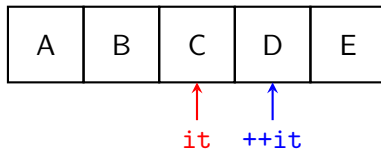
Iteradores

Objeto que aponta para um elemento do container



Iteradores

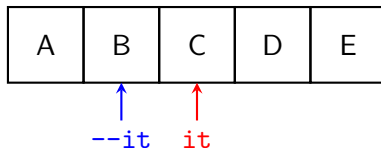
Objeto que aponta para um elemento do container



- `++it` move o iterador para o próximo

Iteradores

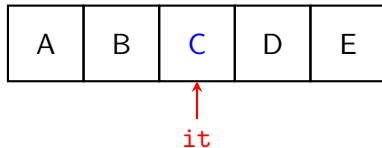
Objeto que aponta para um elemento do container



- `++it` move o iterador para o próximo
- `--it` move o iterador para o anterior

Iteradores

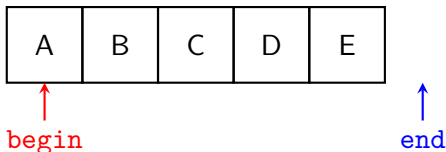
Objeto que aponta para um elemento do container



- `++it` move o iterador para o próximo
- `--it` move o iterador para o anterior
- `*it` acessa o elemento

Iteradores `begin()` e `end()`

Objeto que aponta para um elemento do container



- `c.begin()` iterador para o primeiro elemento
- `c.end()` iterador para a posição após o último elemento

Para percorrer um container C usando iteradores fazemos:

```
for (auto it = C.begin(); it != C.end(); ++it)
```

Exercício 2

Modifique o exercício 1 para imprimir usando iteradores:

Exercício 2

Modifique o exercício 1 para imprimir usando iteradores:

```
vector<int> numeros;

int input;
cin >> input;

while (input != 0) {
    numeros.push_back(input);
    cin >> input;
}

cout << "foram inseridos " << numeros.size();
cout << " numeros" << endl;

for (auto it = numeros.begin(); it != numeros.end(); ++it) {
    cout << *it << " ";
}
```


Set

```
set<tipo> meuSet;
```

Conjunto de elementos únicos

- Trabalho em grupo: mesmo que mais de uma pessoa entregue, ele é considerado uma única vez
- Armazena os elementos em ordem crescente

Set

```
set<tipo> meuSet;
```

Conjunto de elementos únicos

- Trabalho em grupo: mesmo que mais de uma pessoa entregue, ele é considerado uma única vez
- Armazena os elementos em ordem crescente

<code>insert(x)</code>	Inserir o elemento x $O(\log n)$
<code>erase(x)</code>	Remove o elemento x $O(\log n)$
<code>find(x)</code>	Retorna um iterador para x (ou <code>end()</code> se não encontrar) $O(\log n)$
<code>size()</code>	Retorna a quantidade de elementos $O(1)$
<code>empty()</code>	Verifica se o set está vazio $O(1)$
<code>begin()</code>	Iterador para o primeiro elemento $O(1)$
<code>end()</code>	Iterador para a posição após o último elemento $O(1)$

Exercício 3

Ler palavras de um texto e imprimir as palavras usadas em ordem alfabética.

Exercício 3

Ler palavras de um texto e imprimir as palavras usadas em ordem alfabética.

```
set<string> palavras;

int tamanho;
cin >> tamanho;

string pal;
for (int i = 0; i < tamanho; ++i) {
    cin >> pal;
    palavras.insert(pal);
}

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

Map

```
map<tipo1, tipo2> meuMap;
```

Mapeia uma chave a um valor

- Agenda de contatos: cada nome está associado a um telefone
- Cada chave é única
- Ordena os elementos pela chave

```
map<string, int> agenda;  
agenda["maysa"] = 91234;  
cout << agenda["maysa"]; // 91234
```

Map

```
map<tipo1, tipo2> meuMap;
```

Mapeia uma chave a um valor

- Agenda de contatos: cada nome está associado a um telefone
- Cada chave é única
- Ordena os elementos pela chave

```
map<string, int> agenda;
agenda["maysa"] = 91234;
cout << agenda["maysa"]; // 91234
```

erase(x)	Remove o par com chave x $O(\log n)$
size()	Retorna a quantidade de pares chave-valor $O(1)$
empty()	Verifica se o map está vazio $O(1)$
begin()	Iterador para o primeiro par ordenado $O(1)$
end()	Iterador para a posição após o último par $O(1)$

Iterando sobre um map

```
map<string, int> agenda;  
agenda["paulo"] = 94567;  
agenda["maysa"] = 91234;  
  
for (auto it = agenda.begin(); it != agenda.end(); it++)  
    cout << it->first << ": " << it->second << "\n";
```

- `it->first` acessa a **chave**
- `it->second` acessa o **valor**

Saída:

```
maysa: 91234  
paulo: 94567
```

Exercício 4

Dado um texto, contar quantas vezes cada palavra apareceu

Exercício 4

Dado um texto, contar quantas vezes cada palavra apareceu

```
map<string, int> palavras;  
int tamanho;  
cin >> tamanho;  
  
string pal;  
for (int i = 0; i < tamanho; ++i) {  
    cin >> pal;  
    if (palavras[pal])  
        palavras[pal] += 1;  
    else  
        palavras[pal] = 1;  
}  
  
for (auto it = palavras.begin(); it != palavras.end(); ++it)  
    cout << it->first << ": " << it->second;  
    cout << " vezes" << endl;
```

Outras funções úteis da STL

<code>sort(v.begin(), v.end())</code>	Ordena os elementos do vector v
<code>max(a, b)</code>	Retorna o maior entre a e b
<code>min(a, b)</code>	Retorna o menor entre a e b
<code>swap(a, b)</code>	Troca os valores de a e b

Outras funções úteis da STL

<code>sort(v.begin(), v.end())</code>	Ordena os elementos do vector v
<code>max(a, b)</code>	Retorna o maior entre a e b
<code>min(a, b)</code>	Retorna o menor entre a e b
<code>swap(a, b)</code>	Troca os valores de a e b

Onde aprender mais?

Mais containers e funções estão descritos na documentação do C++

- <https://en.cppreference.com/w/cpp/container>
- <https://www.cplusplus.com/reference/stl/>
- Deem uma olhada em `unordered_set` e `unordered_map`