Aula 1

STL

Maratona de Programação

FEI

April 29, 2025

O que é STL?

- Definição
  - A *Standard Template Library* (STL) do C++ é um conjunto de templates e funções que fornece implementações comuns de estruturas de dados e algoritmos.
- Componentes da STL
  - Containers
  - Algoritmos
  - Iteradores
  - Functors

Containers que serão estudados

- Containers sequenciais
  - Vector
- Containers associativos
  - Set
  - Map
- Containers adaptativos
  - Stack
  - Queue
  - Priority Queue

Vector

- Resumidamente, *vectors* são containers sequenciais que representam *arrays* dinâmicos.

- Diferentemente dos arrays tradicionais, seu tamanho pode ser alterado durante a execução do código, com o armazenamento sendo gerenciado automaticamente pelo container.

- Motivação: Vector

Maneiras de inicializar um vector

```cpp
// Maneira mais simples
vector<int> v;

// Informando um tamanho inicial
vector<int> v(10);

// Informando um tamanho e
// informando oss valores dos elementos
vector<int> v(10, 2);
```

Principais métodos

```cpp
// Retorna o tamanho do vector
v.size();

// Adiciona um elemento no final
v.push_back(10);

// Remove um elemento do final
v.pop_back();

// Remove todos os elementos do vector
v.clear();
```

Set

- São conjuntos que armazenam elementos únicos em alguma ordem de classificação, geralmente crescente.
- Geralmente implementados com Árvores Red-Black, que garantem complexidades logarítmicas.
- Motivação: Dijkstra

Set

```cpp
// Inicializacao
set<int> s;

// Insercao de elemento
s.insert(10);

// Remocao de um elemento
s.erase(10);

// Retorna se um elemento
// esta presente no conjunto
s.count(10);

// Retorna o tamanho do set
s.size();
```

Exemplo de execução

String: abbbcad

Set: { }

Exemplo de execução

String: abbbcad

Set: {a}

Exemplo de execução

String: abbbcad

Set: {a, b}

Exemplo de execução

String: abbbcad

Set: {a, b}

Exemplo de execução

String: abbbcad

Set: {a, b}

Exemplo de execução

String: abbbcad

Set: {a, b, c}

Exemplo de execução

String: abbbcad

Set: {a, b, c}

Exemplo de execução

String: abbbcad

Set: {a, b, c, d}

Exemplo em código

```
1 set<int> s;
2
3 for(int i = 0; i < 10; i++) s.insert(i);
4 for(int i = 0; i < 10; i++) s.insert(i);
5
6 cout<<"Tamanho: "<<s.size()<<endl;
7 s.erase(2);
8 cout<<"Tamanho: "<<s.size()<<endl;
9 cout<<"2 esta no conjunto? "<<s.count(2)<<endl;
```

- Saída:

Tamanho: 10
Tamanho: 9
2 está no conjunto? 0

Complexidades do Set

- Inserção de um elemento: $\mathcal{O}(\log n)$
- Remoção de um elemento: $\mathcal{O}(\log n)$
- Verificar se um elemento está contido: $\mathcal{O}(\log n)$
- Retornar tamanho do set: $\mathcal{O}(1)$

Map

- *Maps* são containers associativos que armazenam elementos baseados em pares de chave-valor.
- Organiza seus elementos com base no valor da chave
- Motivação: Ida à Feira

Exemplo

```cpp
// Inicializacao
map<string, double> m = {{"Joao", 1.15},{"Maria", 7}};

// Adicionando um novo elemento
m["Roberto"] = 10.0;

// Atualizando um elemento
m["Joao"] = 0.0;

// Procura por uma chave no map
auto it = m.find("Roberto");

// Excluindo elementos
m.erase("Maria");
```

Exemplo

```
1  map<string, int> m;
2
3  // Outra forma de inserir elementos
4  m.emplace("Fusca", 1975);
5  m.emplace("Santana", 2001);
6
7  string carro = "Jetta";
8  auto it = m.find(carro);
9
10 if(it != m.end()) cout<<"Ano do carro: "<<m[carro]<<endl
       ;
11 else cout<<"Carro nao cadastrado!"<<endl;
```

Nota:

Quando acessamos uma chave não existente no *map*, o construtor automaticamente cria um valor padrão para aquela chave.

Complexidades do Map

- Inserção de um elemento: $\mathcal{O}(\log n)$
- Remoção de um elemento: $\mathcal{O}(\log n)$
- Buscar por um elemento: $\mathcal{O}(\log n)$
- Atualizar um elemento: $\mathcal{O}(\log n)$
- Acessar um elemento: $\mathcal{O}(\log n)$
- Retornar tamanho do map: $\mathcal{O}(1)$

Stack

- Containers adaptativos onde os elementos são inseridos e removidos de um único lado.
- Segue o princípio *LIFO* (*Last In, First Out*)
- Motivação: Balanço de Parênteses I

Exemplo em código

```
1  stack<int> st;
2
3  // Verifica se a stacke esta vazia
4  cout<<st.empty()<<endl;
5
6  // Insercao de um novo elemento
7  st.push(10);
8
9  // Retorna o elemento do topo
10 cout<<st.top()<<endl;
11
12 // Remove um elemento
13 st.pop();
```

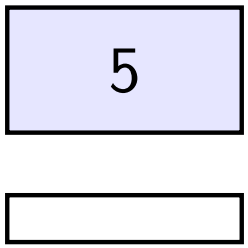
- Saída:

1
10

Visualização de Stack

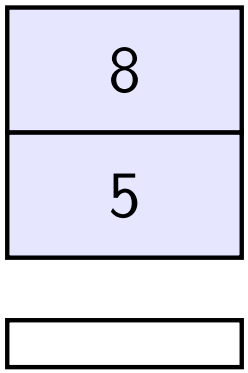
Visualização de Stack

- stack.push(5);



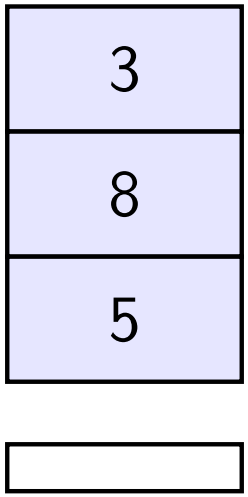
Visualização de Stack

- stack.push(8);



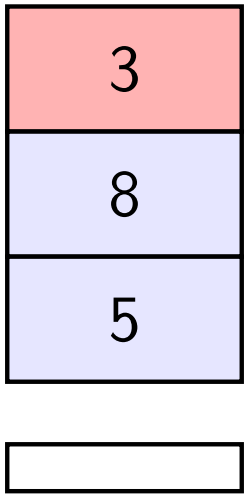
Visualização de Stack

- stack.push(3);



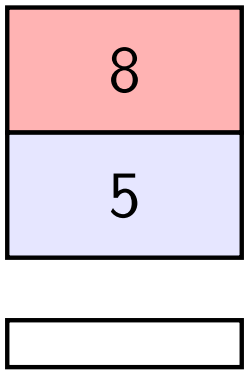
Visualização de Stack

- stack.pop();



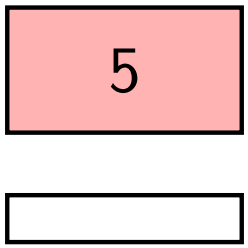
Visualização de Stack

- stack.pop();



Visualização de Stack

- stack.pop();



Visualização de Stack

Complexidades da Stack

- Inserção de um elemento: $\mathcal{O}(1)$
- Remoção de um elemento: $\mathcal{O}(1)$
- Verificar se está vazia: $\mathcal{O}(1)$
- Retornar tamanho: $\mathcal{O}(1)$

Queue

- Containers adaptativos onde os elementos são inseridos em um lado e removidos do outro.
- Segue o princípio *FIFO* (*First In, First Out*)
- Motivação: Restaurant Queue

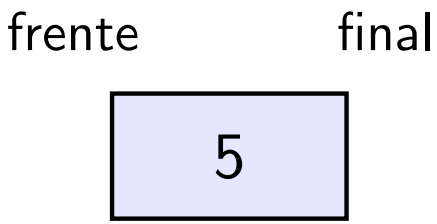
Visualização de Queue

frente

final

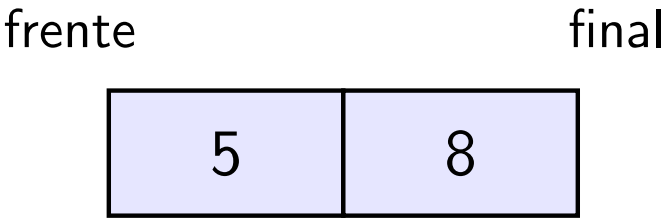
Visualização de Queue

- queue.push(5);



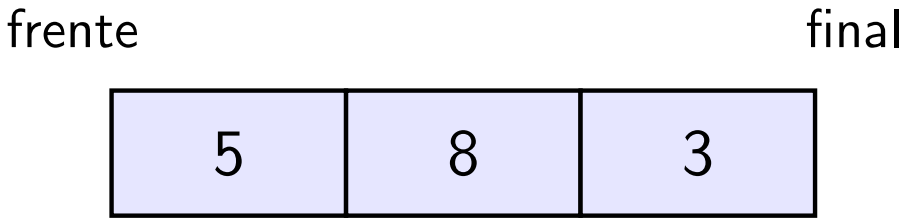
Visualização de Queue

- queue.push(8);



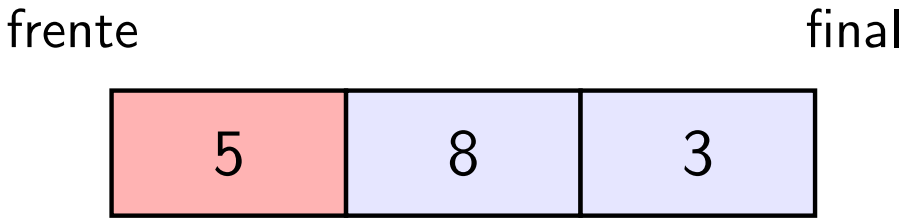
Visualização de Queue

- queue.push(3);



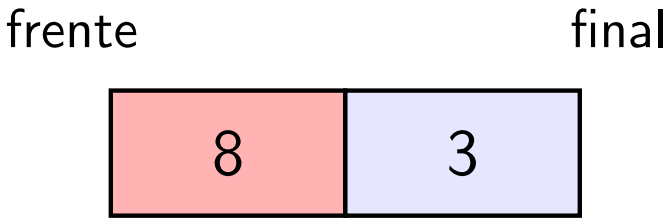
Visualização de Queue

- queue.pop();



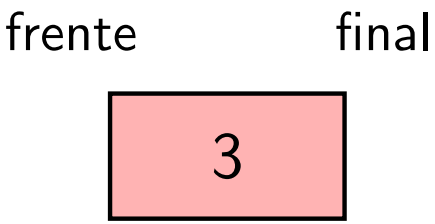
Visualização de Queue

- queue.pop();



Visualização de Queue

- queue.pop();



Visualização de Queue

frente

final

Exemplo de código

```
1  queue<int> q;
2
3  // Insercao de um elemento
4  q.push(3);
5
6  // Retorna o primeiro elemento
7  cout<<q.front()<<endl;
8
9  // Remove o primeiro elemento
10 q.pop();
```

- Saída:

3

Complexidades da Queue

- Inserção de um elemento: $\mathcal{O}(1)$
- Remoção de um elemento: $\mathcal{O}(1)$
- Retornar primeiro elemento: $\mathcal{O}(1)$

Priority Queue

- Containers adaptativos onde os elementos são ordenados por prioridade.
- Os elementos de maior valor são os primeiros a serem removidos
- Motivação: Use Priority Queue Please

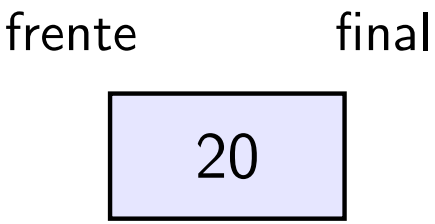
Visualização de Priority Queue

frente

final

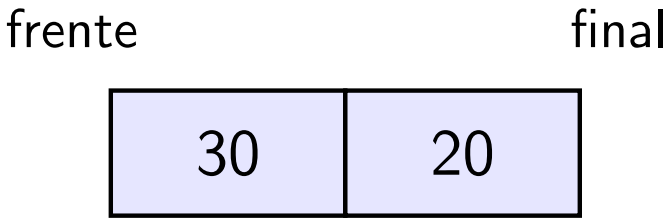
Visualização de Priority Queue

- pq.push(20);



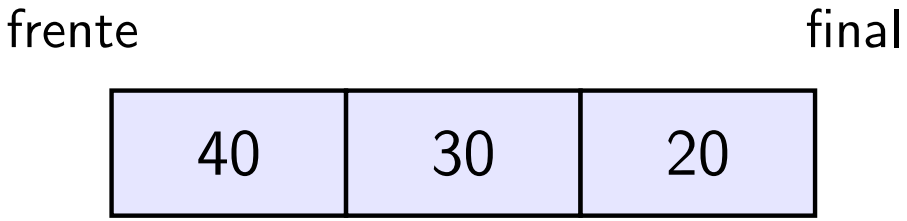
Visualização de Priority Queue

- pq.push(30);



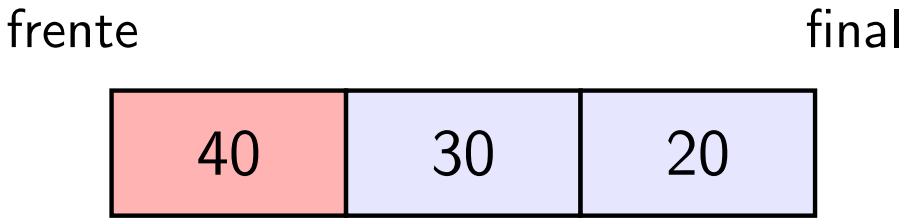
Visualização de Priority Queue

- pq.push(40);



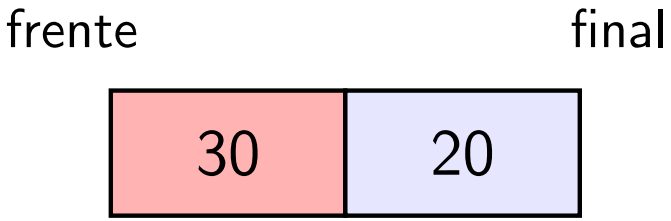
Visualização de Priority Queue

- pq.pop();



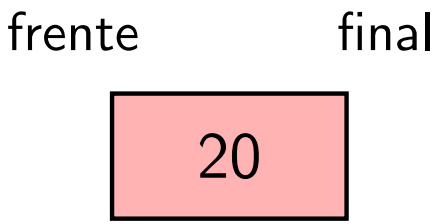
Visualização de Priority Queue

- pq.pop();



Visualização de Priority Queue

- pq.pop();



Visualização de Priority Queue

frente

final

Exemplo de código

```
1  priority_queue<int> pq;
2
3  // Insercao de elementos
4  pq.push(6);
5
6  // Retorna o proximo elemento
7  cout<<pq.top()<<endl;
8
9  // Remocao de elementos
10 pq.pop();
```

- Saída:

6

Complexidades da Priority Queue

- Inserção de um elemento: $\mathcal{O}(\log n)$
- Remoção de um elemento: $\mathcal{O}(\log n)$
- Retornar elemento de maior prioridade: $\mathcal{O}(1)$

Resumo das Estruturas

Container	**Inserção**	**Busca**	**Remoção**
vector	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$
set	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
map	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
stack	$\mathcal{O}(1)$	-	$\mathcal{O}(1)$
queue	$\mathcal{O}(1)$	-	$\mathcal{O}(1)$
priority queue	$\mathcal{O}(\log n)$	-	$\mathcal{O}(\log n)$

Referências

 GeeksforGeeks. https://www.geeksforgeeks.org.

 CPlusPlus. https://cplusplus.com.