STL - Standard Template Library

Aula 14 - Programação orientada a objetos

Professor Daniel Marques

INSTITUTO FEDERAL
Paraíba
Campus Campina Grande

Introdução

A STL faz parte da biblioteca padrão do C++ que inclui suporte estruturas de dados básicas e algoritmos que processam estas estruturas

- Criada por Alexander Stepanov e Meng Lee (Hewlett-Packard), adicionada ao C++ em 1994
- A STL é composta, basicamente, de containeres, iteradores e algoritmos

STL

Container: representação de estruturas de dados genérica

- Implementados usando Classes Templates
- Programadores especificam quais tipos de elementos suas coleções particulares contêm

Iteradores

São semelhantes a ponteiros, utilizados para percorrer e manipular os elementos de um contêiner

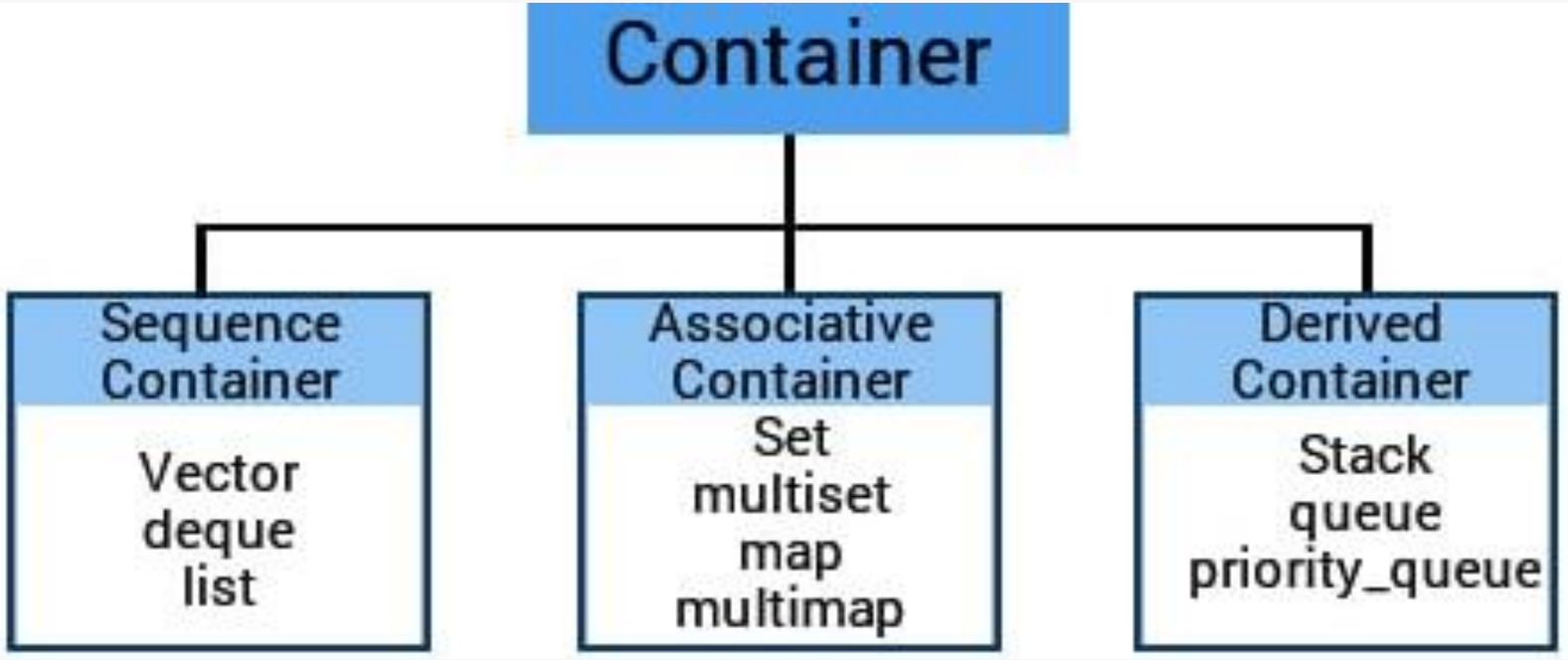
Algoritmos

São as funções/métodos que realizam operações tais como buscar, ordenar e comparar elementos ou containeres inteiros

Containeres

Os containeres são divididos em três categorias principais:

- **Containeres Sequenciais**
- **Containeres Associativos**
- **Adaptadores de Containeres**



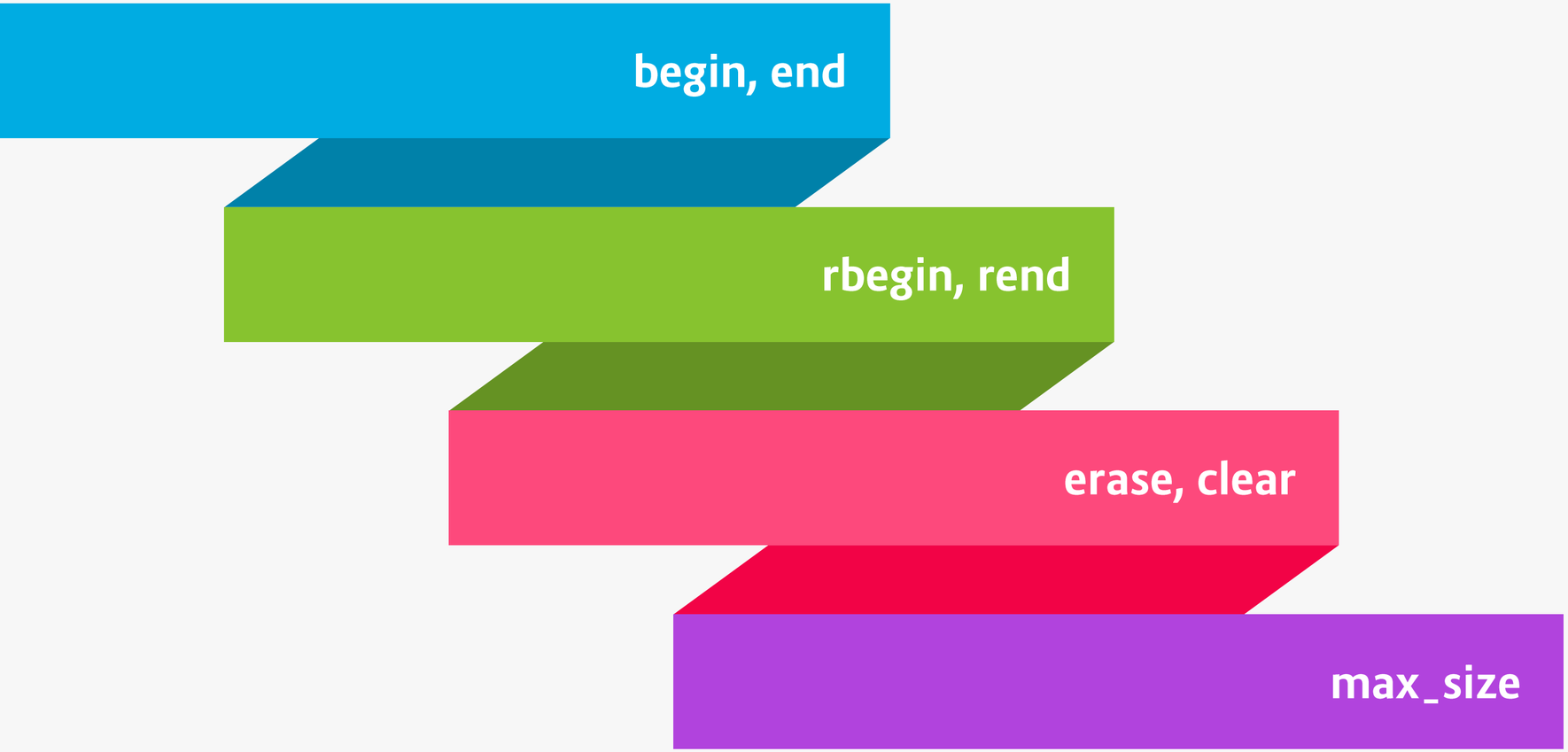
- Containeres Sequenciais: estruturas de dados lineares.
- Containeres Associativos: estruturas de dados não lineares (pares chave/valor)
- Adaptadores de Containeres: são containeres sequenciais, porém, limitados

Funções comuns na STL

Construtor	
Empty	Ver se a estrutura está vazia
Size	Retorna a quantidade de elementos da estrutura de dados
Operações de comparação	= < <= > >= == !=
Swap	

Funções comuns na STL

Funções para containeres de primeira classe



Iteradores

- Funcionalidade similar a dos ponteiros
- Apontam para elementos em containeres de primeira classe

Certas operações com iteradores são as mesmas para todos os containeres

* desferencia

++ aponta para o próximo elemento

begin(): retorna o iterador do primeiro elemento

end() retorna iterador do elemento depois do último

Containeres sequenciais

Tipo	Descrição
vector	Inserções e remoções no final, acesso direto a qualquer elemento.
deque	Fila duplamente ligada, inserções e remoções no início ou no final, sem acesso direto a qualquer elemento.
list	Lista duplamente ligada, inserção e remoção em qualquer ponto.

Vector

- Definido na biblioteca #include<vector>
- Estrutura de dados com alocação de memória sequencial
- Mais eficientes se inserções forem feitas apenas no final
- Uma boa opção se os dados estão ordenados

Propriedades da classe vector

- Rápida inserção/remoção de dados no final do vetor: O(1)
- Lenta inserção/remoção de dados no inicio ou no meio: O(n)
- Pesquisa lenta: O(n)

Vector e seus construtores

vector()

Cria um vetor de tamanho zero

vector(size_type n, const T &val = T())

Cria um vetor de tamanho n e os elementos são inicializados com o valor T

vector(const T &V)

Cria uma cópia do vetor V

Vector e seus métodos

Size()

Retorna o número de elementos do vetor

empty()

Testa se o vetor está vazio ou não

operator =

Faz uma cópia do vetor

at(size_type i)

Se i estiver nos limites do vetor, retorna o elemento i; caso contrário, lança uma exceção

Vector e seus métodos

push_back(const T &val):

Insere um valor no final do vetor

pop_back()

Remove o último elemento do vetor

front()

Retorna uma referência para o 1º elemento

back()

Retorna uma referência para o último elemento

List

- Cabeçalho: #include<list>
- Implementado como Lista duplamente encadeada (dois ponteiros por nó)

Estrutura começa vazia, e novos elementos vão sendo alocados ou liberados dinamicamente, à medida que as operações de inserção e remoção vão sendo invocadas

- Inserção/remoção eficiente em qualquer lugar no container
- Iteradores bidirecionais

Declaração:

std::list <type> name;

List: principais características

- Uso mais eficiente da memória (só aloca o que utiliza)
- Acesso a elementos em tempo linear (O(n))
- Inserção/remoção no início em tempo constante. (O(1))

List e seus métodos

size()	Retorna o número de elementos da lista
empty()	Testa se a lista está vazia ou não
operator =	Faz uma cópia da lista
push_front()	Insere um elemento no início da lista
pop_front()	Remove o primeiro elemento da lista

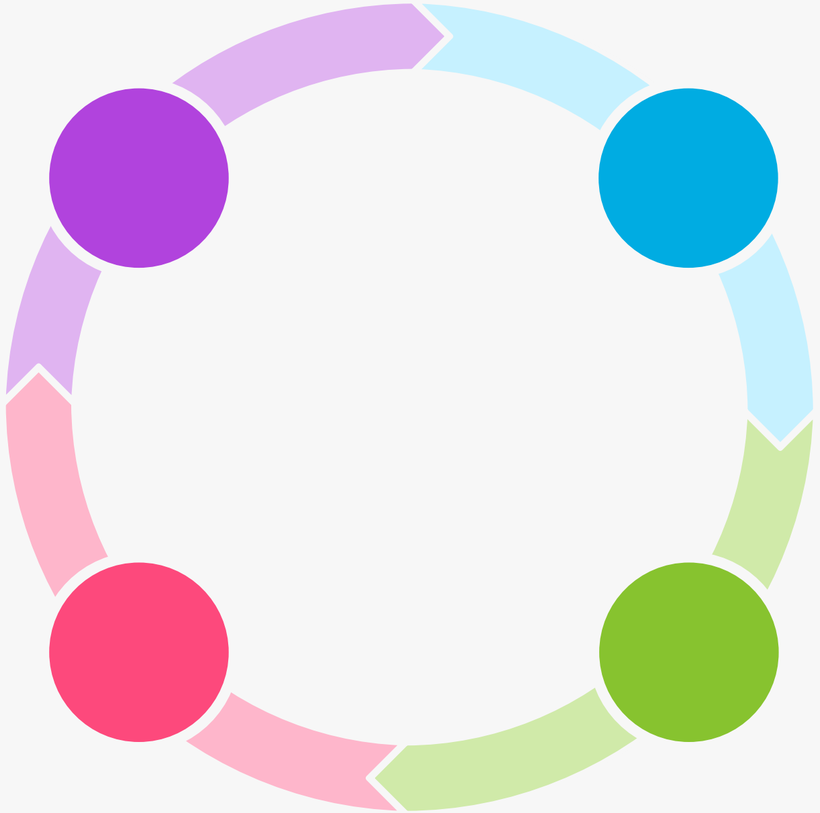
List e suas funções

push_back()	Adiciona um elemento no final da lista
pop_back()	Remove o último elemento da lista
front()	Retorna uma referência para o 1o elemento
back()	Retorna uma referência para o último elemento
sort()	Ordena os elementos da lista

Deque

Cabeçalho

#include<deque>



deque ("deek")

fila com final duplo (double-ended queue)

Acesso indexado

Acesso indexado usando []

Similar ao vector

entretanto, também possui como nas listas

- push_front (insere na frente do deque)
- pop_front (remove da frente)

Containeres adaptadores

stack, queue e priority_queue

- Não são containeres de primeira classe
- Não suportam iteradores
- Programador pode selecionar implementação
- Funções membro push e pop

Stack

Sobre stack

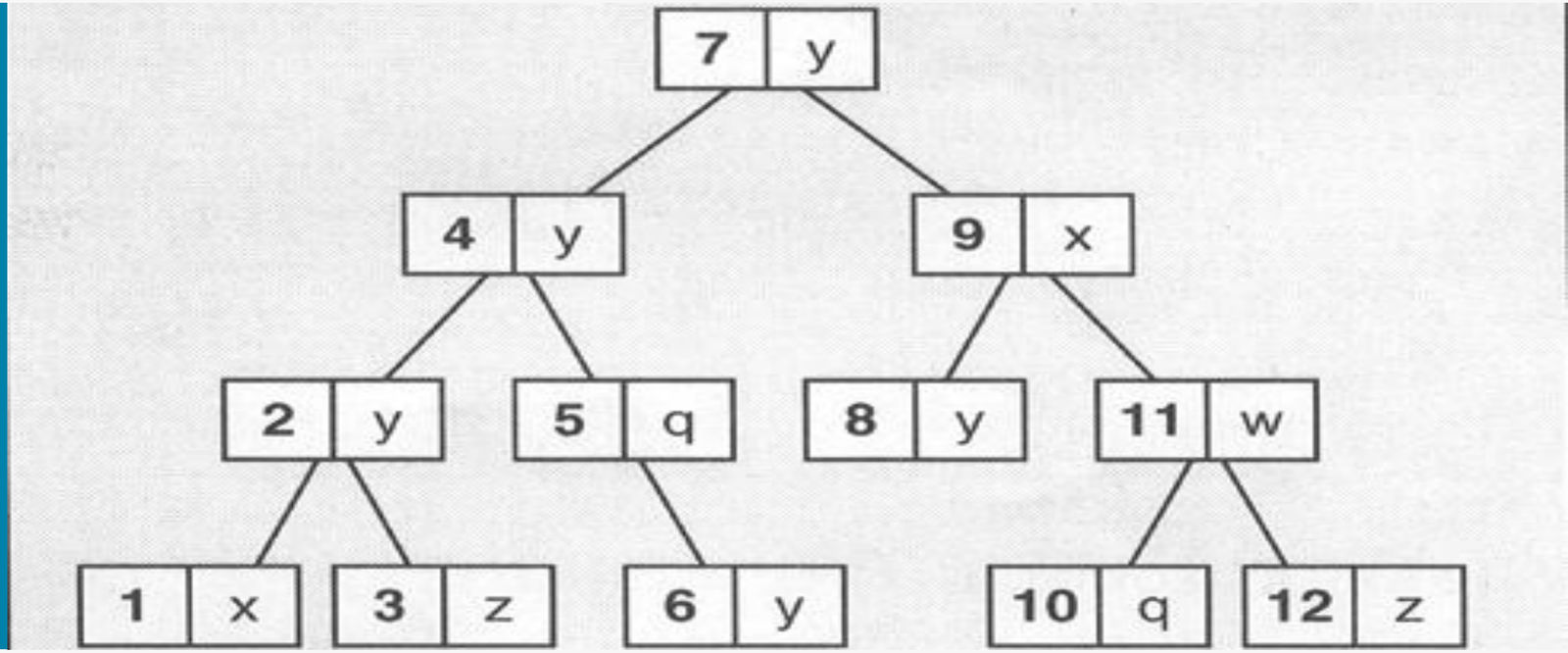
- Cabeçalho #include <stack>
- Inserções e remoções em uma extremidade
- Estrutura de dados last-in, first-out (LIFO)
- Pode usar vector, list, ou deque (padrão)

Declarações

```
stack<type, vector<type>> myStack;
stack<type, list <type>> myOtherStack;
stack<type> anotherStack; // padrão deque
```

Containeres associativos

- Acesso direto para armazenar/recuperar elementos
- Realizam buscas por chaves



Quatro tipos: multiset, set, multimap e map

- Ordenados por chaves

Containeres associativos

multiset e set

- manipulam conjunto de valores
- Valores são as próprias chaves

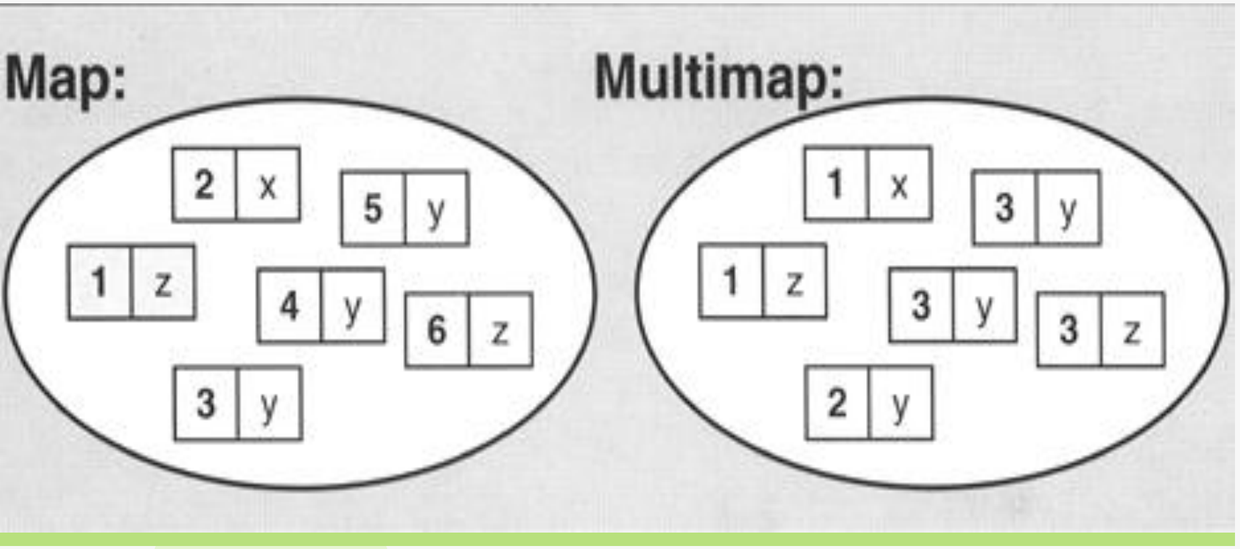
multimap e map

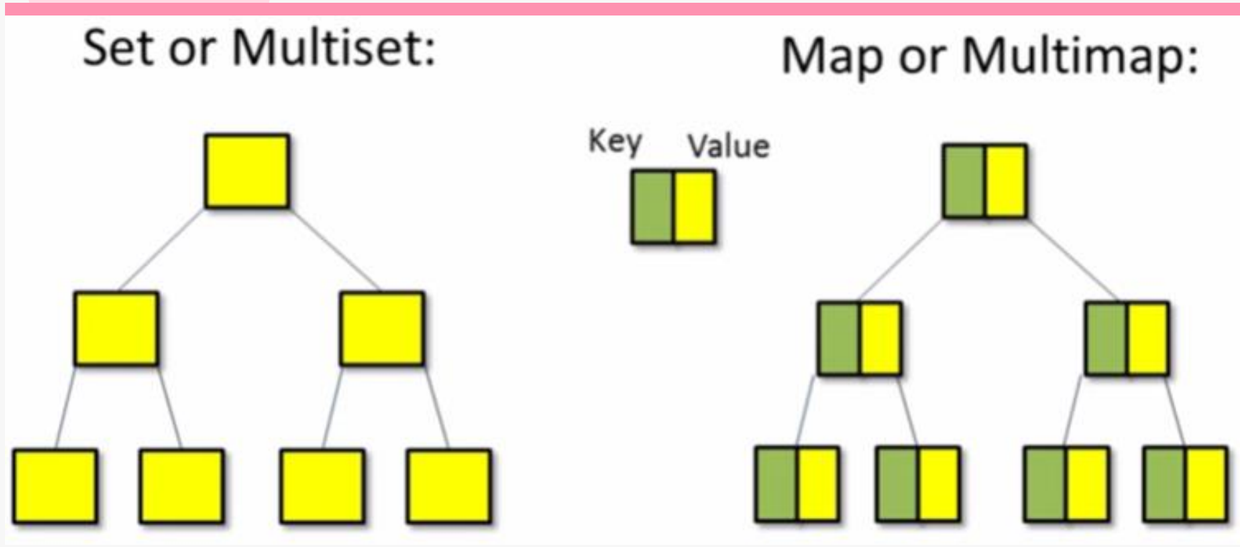
- manipulam valores associados com chaves
- Possuem chaves e valores

multimap e multiset

permitem chaves duplicadas enquanto set e map não permitem

Containeres associativos





Alguma dúvida?

Não guardem dúvidas, perguntem

• • •

Referências

1. DA COSTA, Anderson Fabiano F. **STL: Standard Templates Library**. Instituto Federal da Paraíba. 2022

2. Materiais de aula dos professores Guillermo Camara-Chavez e Tiago Maritan.

3. Material da IV Escola de Inverno, Maratona de Programação UNIFEI 2016.

4. DEITEL, C++ Como Programar, 5ª edição, Editora Prentice Hall, 2006

5.