A biblioteca sl

PROFESSOR: EDUARDO HABIB BECHELANE MAIA
HABIB@CEFETMG.BR

Biblioteca STL em c++

STL significa Standard Template Library.

Template:

- Funcionalidade avançada em c++
- Permite tipos genéricos
- Significa que você pode, por exemplo, criar uma lista de qualquer tipo de dado, sem a necessidade de criar um nó e uma lista, como fizemos até agora
- Veremos nessa aula algumas estruturas da biblioteca STL.
- Economia de tempo
- Fazem alocação dinâmica automaticamente.

Vector

Introdução

- Estende a noção de vetor, tornando-a mais poderosa.
- Permite o armazenamento de uma sequência de objetos arbitrários
- Os elementos de um vector, assim como de um vetor, podem ser acessados especificando seu índice.
- **Principal vantagem:** Não é necessário saber o tamanho que você deseja que o vetor tenha ao declará-lo;
- Pode-se adicionar novos elementos ao final de um vetor usando a função push_back.
  - Permite que se insira novos elementos em qualquer posição do vetor, mas esta é uma operação muito ineficiente
    - Use Lista

Funções da classe Vector

- **size :** retorna o número de elementos no vetor
- **clear :** remove todos os elementos do vetor
- **empty :** retorna **true** se o vetor não tem elementos
- **push_back (x) :** adiciona x ao final do vetor
- **pop_back :** Remove o objeto no final do vetor
- **back :** Retorna o objeto no final do vetor
- **front :** retorna o objeto na frente do vetor
- **erase:** Remove o elemento em uma posição do vetor
- **erase**: Remove um intervalo de elementos, incluindo o elemento first, mas não o last
- **Insert:** inserir um item em **uma posição específica.**
  - Ex: v.insert(v.begin() + posicao, valor);

Exemplos de Vector

```
#include <iostream>
#include <vector>
using namespace std;
int main(){
   vector<int> vetor;
   int i;
   // INSERINDO 5 VALORES DE 1 A 5  USANDO PUSH_BACK
  for(i = 0; i < 5; i++){
    vetor.push_back(i);
  }
   // PEGANDO O TAMANHO DO VETOR
   cout << "Tamanho do Vetor = " << vetor.size() << endl;
   // ACESSANDO OS 5 VALORES DO VETOR PASSANDO PELO
INDEX
   for(i = 0; i < 5; i++){
    cout << "Valor do vetor [" << i << "] = " << vetor[i] << endl;
  }
  // OU USANDO O ITERATOR PARA ACESSAR OS VALORES.
  vector<int>::iterator v = vetor.begin();
  while( v != vetor.end()) {
    cout << "Valor do vetor = " << *v << endl;
    v++;
  }
   return 0;
}
```

Em java usando ArrayList

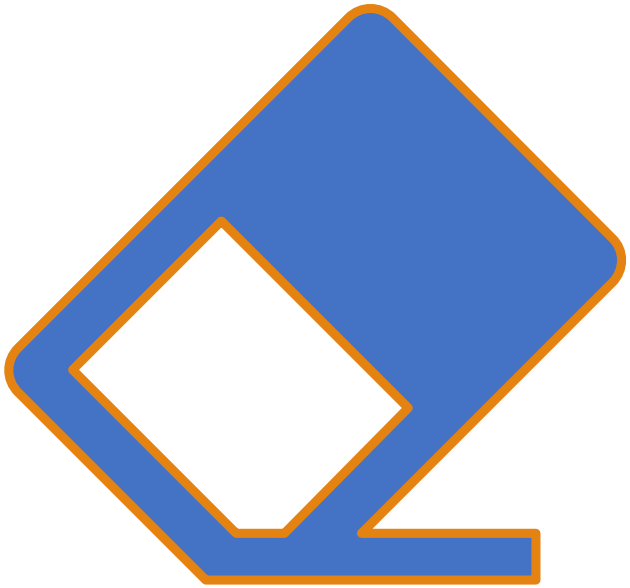
```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Main {
   public static void main(String[] args) {
      ArrayList<Integer> lista = new ArrayList<>();
      int i;

      // INSERINDO 5 VALORES DE 1 A 5 USANDO add
      for (i = 0; i < 5; i++) {
         lista.add(i);
      }

      // PEGANDO O TAMANHO DA LISTA
      System.out.println("Tamanho da Lista = " + lista.size());

      // ACESSANDO OS 5 VALORES DA LISTA PASSANDO PELO INDEX
      for (i = 0; i < 5; i++) {
         System.out.println("Valor da lista [" + i + "] = " + lista.get(i));
      }

      // OU USANDO O ITERATOR PARA ACESSAR OS VALORES.
      Iterator<Integer> it = lista.iterator();
      while (it.hasNext()) {
         System.out.println("Valor da lista = " + it.next());
      }
   }
}
```

Exemplo com o Erase

```
#include <iostream>
#include <vector>

int main ()
{
  std::vector<int> myvector;

  // colocando valores no vetor
  for (int i = 1; i <= 10; i++)
      myvector.push_back(i);

  // Apagando o sexto elemento
  myvector.erase(myvector.begin() + 5);

  // Apagando os 3 primeiros elementos
  myvector.erase(myvector.begin(), myvector.begin() + 3);

  // Inserindo um elemento na 4ª posição (índice 3)
  myvector.insert(myvector.begin() + 3, 99); // Insere o valor 99

  std::cout << "Vetor:";
  for (unsigned i = 0; i < myvector.size(); ++i)
    std::cout << ' ' << myvector[i];
  std::cout << '\n';

  return 0;
}
```

Código similar em Java

```
import java.util.ArrayList;

public class Main {
   public static void main(String[] args) {
      ArrayList<Integer> myList = new ArrayList<>();

      // Colocando valores na lista
      for (int i = 1; i <= 10; i++)
         myList.add(i);

      // Apagando o sexto elemento (índice 5)
      myList.remove(5);

      // Apagando os 3 primeiros elementos (índices 0 a 2)
      myList.subList(0, 3).clear();

      // Inserindo um elemento na 4ª posição (índice 3)
      myList.add(3, 99); // Insere o valor 99 na posição 4

      // Exibindo o vetor
      System.out.print("Vetor:");
      for (int num : myList)
         System.out.print(" " + num);
      System.out.println();
   }
}
```



Mais exemplos de vector

HTTPS://WWW.CPLUSPLUS.COM/REFERENCE/VECTOR/VECTOR/

Vector de classes

- A utilização seria similar a de um vetor qualquer. Ex:
  - vector<Pais> vetor;

Pais.hpp

```
#ifndef __PAIS_HPP
#define __PAIS_HPP #include<vector>
#include <string>
using namespace std;
class Pais {
   private:
      string nome;
      float dimensao;
      string nomeCapital;
      vector <Pais> listaVizinhos;
   public:
      static int cont;
      int getCont();
      string getNome();
      void setNome(string nome);
      float getDimensao();
      void setDimensao(float dimensao);
      string getNomeCapital();
      void setNomeCapital(string nome);
      void setListaVizinhos(vector <Pais> listaPais);
      void verificarSePaisEIgual(Pais x);
      void imprimirFronteiras();};#endif
```

Pais.cpp

```
#include <iostream>
#include <string>
#include <math.h>
#include <vector>
#include "Pais.hpp"
using namespace std;

int Pais::cont;

void Pais::setListaVizinhos(vector <Pais>
lp){
	listaVizinhos=lp;
}

int Pais::getCont(){
	return cont;
}
string Pais::getNome(){
	return nome;
}
void Pais::setNome(string nome){
	this->nome=nome;
}

float Pais::getDimensao()
{
	return this->dimensao;
}
void Pais::setDimensao(float dimensao)
{
	this->dimensao=dimensao;
}

string Pais::getNomeCapital(){
	return nomeCapital;
}
void Pais::setNomeCapital(string
nomeCapital){
	this->nomeCapital=nomeCapital;
}

void Pais::verificarSePaisEIgual(Pais x){
	if ((x.getNome()== nome) &&
	(x.getNomeCapital() == nomeCapital))
	{
	   cout << "Paises Iguais";
	 } else  cout << "São diferentes";
}

void Pais::imprimirFronteiras(){
	cout << "\nVizinhos: \n";
	for (int i=0;i<listaVizinhos.size();i++){
	   cout <<
	    listaVizinhos[i].getNome()<<endl;
	 }
}
```

Main.cpp

```
#include <iostream>
#include <vector>
#include "Pais.hpp"
using namespace std;
int main(){
   int i;
   //Vetor de pais
   vector <Pais> vet;
   vector <Pais> vetVizinhos;
   Pais p, vizinho;
   string nome, capital;
   float dimensao;
   int numVizinhos;
   cout<< "Digite o nome do pais: ";
   getline(cin,nome);
   cout << "\nDigite o Dimensao: \n";
   cin >> dimensao;
   cout << "\nDigite a capital:";
   cin.ignore();
   getline(cin, capital);
   p.cont=0;
   p.setNome(nome);
   p.setDimensao(dimensao);
   p.setNomeCapital(capital);
   cout << "Digite numero de vizinhos: ";
   cin >> numVizinhos;
   i=0;
   cin.ignore();
   while (i<numVizinhos)
   {
      cout<< "Digite o nome do pais: \n";
      getline(cin,nome);
      cout << "\nDigite o Dimensao: \n";
      cin >> dimensao;
      cout << "\nDigite a capital:";
      cin.ignore();
      getline(cin, capital);
      vizinho.setNome(nome);
      vizinho.setDimensao(dimensao);
      vizinho.setNomeCapital(capital);
      vetVizinhos.push_back(vizinho);
      i++;
   }
   p.setListaVizinhos(vetVizinhos);
   vet.push_back(p);
   for (int i=0;i<vet.size();i++)
   {
      cout << "Nome: " << vet[i].getNome()
   << endl;
      vet[i].imprimirFronteiras();
    }
    return 0;
   }
```

Pilha (stack)

Stack

A stack é uma pilha, igual a que já vimos:

- o último que entra é o primeiro que sai (LIFO). Exemplo

```
#include <bits/stdc++.h>
using namespace std;
int main() {
        stack<int> s;
        s.push(2);
        s.push(3);
        cout << "Mostra o topo: " << s.top() << endl;
        cout<< "tamanho = " << s.size() << endl;
        cout << "mostra o último da pilha: " << s.top() << endl;
        cout << "vai retirar o último da pilha: " <<_endl;
        s.pop();
        cout << "Mostra o último agora: " << s.top() << endl;
        cout << "tamanho=" << s.size() << endl;
        cout << "está vazio? " << s.empty() << "\n";
}
```

Pilha em Java

Em Java, uma pilha pode ser implementada de várias maneiras, mas uma das formas mais comuns é usar a classe **Stack** que faz parte da Java Collection Framework.

- No entanto, é importante observar que Stack é uma classe legada
- Documentação oficial do Java recomenda o uso de um Deque
- Usar Deque para pilhas é preferível porque é mais consistente e fornece uma API mais completa.

Vou mostrar um exemplo usando Deque como uma pilha:

Exemplo

Código: https://onlinegdb.com/UaNC0aU3e

Fila

Queue

A **queue** é uma fila

- o primeiro que entra é o primeiro que sai (FIFO).

- Exemplo

```
#include <bits/stdc++.h>
using namespace std;
int main() {
        queue<int> q;
        q.push(5);
        q.push(6);
        q.push(7);

        cout << "Quem está na frente? " << q.front() << endl;
        cout << "Quem está no fim? " << q.back() << endl;
        cout << "Tamanho da fila: " << q.size() <<endl;
        cout << "Fila vai andar" << endl;
        q.pop();
        cout << "Quem está na frente? " << q.front() << endl;
        cout << "Quem está no fim? " << q.back() << endl;
        cout << "Tamanho da fila: " << q.size() <<endl;
        cout << "Fila vai andar" << endl;
        q.pop();
        cout << "Quem está na frente? " << q.front() << endl;
        cout << "Quem está no fim? " << q.back() << endl;
        cout << "Tamanho da fila: " << q.size() <<endl;
        cout << "Fila vai andar" << endl;
        q.pop();
        cout << "empty=" << q.empty() << "\n";
}
```

Fila de prioridade

É uma fila de prioridade. A parte superior do **priority_queue** é a posição ocupada pelo maior elemento.

Da mesma forma que uma fila, apenas um elemento pode ser removido. E nesse caso, é o de maior prioridade.

Exemplo1

```
#include <iostream>
#include <queue>
using namespace std;

int main(int argc, char *argv[])
{
		priority_queue<int> pq;
		// priority_queue<int, vector<int>, greater<int> > pq; // inverte a
prioridade

		pq.push(20);
		pq.push(15);
		pq.push(50);

		cout << "Elemento do topo: " << pq.top() << endl;

		if(pq.empty())
			cout << "\nFila vazia!!\n";
		else
			cout << "\nFila NAO vazia!!\n";

		cout << "\nTamanho da fila: " << pq.size() << endl;
		cout << "\nMostrando os elementos: ";

		while(!pq.empty())
		{
			cout << pq.top() << " ";
			pq.pop();
		}

		cout << endl;

		return 0;
}
```

Exemplo 2

https://onlinegdb.com/QBjrYmbK9

Fila em Java

Em Java, uma fila pode ser implementada usando a interface Queue.

Essa interface define métodos essenciais para operações de fila, como:

- **offer** (para adicionar elementos),
- **poll** (para remover e retornar o elemento na frente da fila),
- **peek** (para visualizar o elemento na frente da fila sem removê-lo).

Uma das implementações comuns da interface **Queue** é a classe **LinkedList**.

Exemplo

Código: https://onlinegdb.com/Gu1Os5IT4

Fila de prioridades em java

Em Java, uma fila de prioridades pode ser implementada usando a classe **PriorityQueue**.

Ela ordena seus elementos de acordo com a ordem natural ou de acordo com um **Comparator** fornecido no momento da criação da fila.

Para usar a **PriorityQueue** com uma classe personalizada, como Pessoa, você precisa garantir que essa classe implemente a interface Comparable ou forneça um Comparator personalizado.

Implementação

Código: https://onlinegdb.com/Gu1Os5IT4