

Maratona de Programação com STL

MARCOS CASTRO

STL ??

STL

- Standard Template Library
- Biblioteca padrão do C++
- Possui algoritmos, containers, funções...
- Exemplo de algoritmo: Busca binária (binary search)
- Exemplo de container: vector (sequence container)
- Exemplo de função (rotina): `binary_search()`
- STL é uma das vantagens de C++ sobre C

Várias estruturas de dados

```
#include <iostream>
#include <vector> // vetor
#include <list> // lista
#include <queue> // fila
#include <deque> // filas double-ended
#include <map> // mapa (dicionário)
#include <set> // conjunto
#include <stack> // pilha

using namespace std;
```

Estruturas genéricas

```
int main(int argc, char *argv[])
{
    stack<int> s1;
    stack<double> s2;
    stack<long long int> s3;
    stack<meu_tipo> s4;

    return 0;
}
```

#include <vector>

`vector<int> vec;`

- `vec.push_back(1);` // insere o elemento 1 (insere no final)
- `vec.pop_back();` // deleta o último elemento
- `vec.size();` // retorna o tamanho do vetor
- `vec.empty();` // o vetor está vazio ??
- `vec.at(0);` // retorna o elemento da posição passada como parâmetro
- `vec.front();` // acessa o primeiro elemento do vetor
- `vec.back();` // acessa o último elemento do vetor
- `vec.clear();` // remove todos os elementos do vetor
- <http://www.cplusplus.com/reference/vector/>

#include <stack>

`stack<string> pilha;`

- `pilha.empty();` // a pilha está vazia ??
- `pilha.size();` // retorna o tamanho da pilha
- `pilha.top();` // acessa o próximo elemento (o elemento do topo)
- `pilha.push("STL");` // insere na pilha
- `pilha.pop();` remove o elemento do topo
- <http://www.cplusplus.com/reference/stack/>

#include <set>

Um set não possui elementos repetidos.

Os elementos ficam ordenados crescentemente.

```
set<double> my_set;
```

- `my_set.insert(3.14);` // insere elemento no conjunto
- `my_set.empty();` // o conjunto está vazio ??
- `my_set.count(3.14);` // quantos 3.14 existem ??
- `my_set.clear();` // você sabe...

#include <set>

- `my_set.erase(it);` // remove elementos em um intervalo
 - `set<double>::iterator it = my_set.begin();` // “it” apontando para o primeiro elemento
 - `my_set.erase(it);` // remove somente o primeiro elemento
 - `my_set.erase(it, my_set.end());` // remove todos os elementos
 - `my_set.erase(my_set.begin(), my_set.end());` // remove todos os elementos também!
- <http://www.cplusplus.com/reference/set/>

#include <set>

Um multiset permite elementos repetidos.

```
multiset<int> myms;  
  
myms.insert(3);  
myms.insert(3);  
myms.insert(2);  
  
multiset<int>::iterator it;  
  
// imprime: 2 3 3  
for(it = myms.begin(); it != myms.end(); it++)  
    cout << *it << " ";
```

#include <queue>

queue<int> fila;

- fila.empty(); // você sabe...
- fila.push(10); // essa também você sabe...
- fila.size(); // adivinha ??
- fila.front(); // retorna o primeiro da fila
- fila.back(); // retorna o último da fila
- fila.pop(); // remove elemento (primeiro da fila)
- <http://www.cplusplus.com/reference/queue/>

#include <list>

`list<int> my_list;`

- `my_list.empty();` // o que será ??
- `my_list.size();` // os nomes são os mesmos :)
- `my_list.front();` // acessa o primeiro elemento
- `my_list.back();` // acessa o último elemento
- `my_list.push_front();` // insere o elemento no início
- `my_list.push_back();` // insere o elemento no final
- `my_list.pop_back();` // remove o último elemento
- `my_list.pop_front();` // remove o primeiro elemento

#include <list>

`list<int> my_list;`

- `my_list.insert(my_list.begin(), 10);` // insere a partir de uma posição
- `my_list.clear();` // deixe seu comentário aqui
- `my_list.splice();` // insere elementos de uma lista em outra
 - Supor que `my_list` possua os elementos `1, 2, 3` e `my_list2` possua os elementos `4, 5, 6`
 - Quero inserir os elementos de `my_list2` em `my_list` no início
 - `list<int>::iterator it = my_list.begin();` // “it” aponta para o primeiro elemento de `my_list`
 - `my_list.splice(it, my_list2);` // insere os elementos de `my_list2` no início de `my_list`
 - `my_list` agora possui os elementos: `4, 5, 6, 1, 2, 3`
- <http://www.cplusplus.com/reference/list/>

#include <map>

map<string, int> mapa; // container associativo <chave, valor>

- mapa["C++"] = 10;
- mapa["C"] = 10;
- mapa["python"] = 10;
- mapa["perl"] = 9;
- mapa["java script"] = 9;
- mapa["haskell"] = 8;
- mapa["php"] = 8;
- mapa["java"] = 1;

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

int main(int argc, char *argv[])
{
    map<string, int> mapa;

    mapa["C++"] = 10;
    cout << mapa["C++"] << endl;

    return 0;
}
```

#include <map>

As funções seguem o mesmo padrão:

- `mapa.size();`
- `mapa.insert(make_pair("C++", 10));` // outra forma de inserir
- `mapa.insert(pair<string, int>("C", 10));` // outra forma de inserir
- `mapa.clear();`
- <http://www.cplusplus.com/reference/map/>

Priority queue (fila de prioridades)

O que será impresso ??

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

int main(int argc, char *argv[])
{
    priority_queue<int> pq;

    pq.push(30);
    pq.push(40);

    cout << pq.top() << endl;

    return 0;
}
```

Priority queue (fila de prioridades)

E agora ??

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

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

    pq.push(30);
    pq.push(40);

    cout << pq.top() << endl;

    return 0;
}
```

esse espaço é importante...



Priority queue (fila de prioridades)

Usando “greater” você tem uma fila de prioridade mínima!

Usando com **classe**:

```
class Pessoa
{
public:
    int idade;
    double peso;
};
```

```
struct compara
{
    bool operator()(const Pessoa& p1, const Pessoa& p2)
    {
        return (p1.idade + p1.peso) > (p2.idade + p2.peso);
    }
};
```

min ou max ??



```
int main()
{
    priority_queue<Pessoa, vector<Pessoa>, compara > pq;

    return 0;
}
```

Priority queue (fila de prioridades)

Sempre que tiver dúvida, consulte a referência:

- http://www.cplusplus.com/reference/queue/priority_queue/

#include <algorithm>

A STL possui várias funções (rotinas) que podem **otimizar** o seu **tempo**...

Caso você queira ordenar crescentemente um vetor: **sort()**

```
vector<int> v(3); // vetor de 3 posições

v[0] = 10;
v[1] = 5;
v[2] = 7;

sort(v.begin(), v.end()); // a mágica
```

#include <algorithm>

Caso você queira achar um elemento: **find()**

```
vector<int> v;  
  
v.push_back(3);  
v.push_back(2);  
  
if(find(v.begin(), v.end(), 3) == v.end())  
    cout << "nao encontrou\n";  
else  
    cout << "encontrou\n";
```

#include <algorithm>

Gerando permutações: `next_permutation()`

```
vector<int> v;  
  
v.push_back(1);  
v.push_back(2);  
v.push_back(3);  
  
do  
{  
    cout << v[0] << " " << v[1] << " " << v[2] << endl;  
}  
while(next_permutation(v.begin(), v.end()));
```

#include <algorithm>

Referência:

- <http://www.cplusplus.com/reference/algorithm/>

Exemplo de problema: Agente 004

Acesse: www.urionlinejudge.com.br

SEARCH TOOLBAR

Problem ▾

SEARCH

#	Name	Solved	Level
1806	✓ Agent 004	32.42%	4

Agente 004

Que tal utilizar STL na resolução desse problema?

Os próximos slides contém dicas de código com STL para resolvê-lo.

Tente resolver antes de passar para os próximos slides!

Agente 004

O problema envolve grafos.

- Como codificar usando STL?

O problema fornece as rotas conhecidas por Bino (grafo do Bino) e as rotas conhecidas somente pelos criminosos (grafo dos criminosos).

Lembrando que os criminosos conhecem **todas** as rotas possíveis!

O problema quer a quantidade de criminosos que Bino irá eliminar.

O grafo é bidirecional!

Em que a STL pode nos ajudar?

Agente 004

Representando um grafo:

```
vector<list<pair<int, int> > > grafo_bino(N);  
vector<list<pair<int, int> > > grafo_criminosos(N);
```

Adicionando no grafo:

```
grafo_bino[a - 1].push_back(make_pair(b - 1, v));  
grafo_bino[b - 1].push_back(make_pair(a - 1, v));
```

Agente 004

Set usado como fila de prioridades:

```
// set usado como fila de prioridade mínima  
set<pair<int, int> > fila;
```

O set ordena pelo primeiro parâmetro do pair e depois pelo segundo.

Inserindo na fila:

```
fila.insert(make_pair(dist_bino[orig], orig));
```

Agente 004

Armazenando o lugar onde está cada criminoso:

```
// mapa de criminosos e lugares onde eles estão
map<int, int> lugares_criminosos;

for(int i = 0; i < B; i++)
    cin >> lugares_criminosos[i];
```

Agente 004

A solução abaixo utiliza STL, o código está comentado:

https://github.com/marcoscastro/maratona_unifesp/blob/master/URI-1806/URI-1806.cpp



Dúvidas?



mcastrosouza@live.com