

Aula extra 1

STL

Estruturas de Dados 2018/1
Prof. Diego Furtado Silva

Standard Template Library

Quando usamos C++, podemos utilizar as implementações “padrão” de algumas estruturas de dados.

Nesse cenário, as ED (junto a algumas outras coisinhas) são chamadas de **contêineres**.

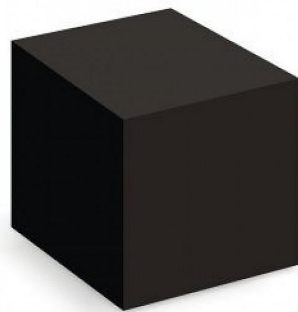
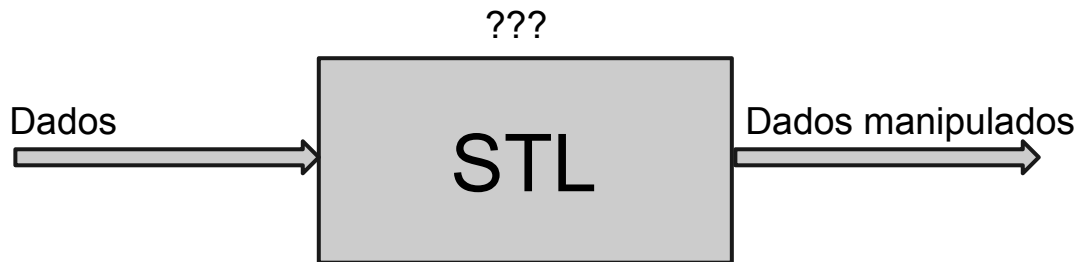
Standard Template Library

Além dos contêineres, há também iteradores, algoritmos e alguns operadores sobrecarregados.

Nesta aula, vamos focar em alguns contêineres e algoritmos que podem ser úteis para vocês desenvolverem os trabalhos.

Standard Template Library

Podemos usar como uma caixa preta



Mas, ao longo do curso, vamos aprender (quase) tudo o que há na STL.

STL – Vantagens e desvantagens

- **Eficiência na implementação** quando se sabe o que é e como usar determinados algoritmos e EDs, mas não temos eles já implementados e testados
- Bem **documentado**
- **Confiar na eficiência** das implementações
- Difícil **debugar**
- Não dá para fazer **pequenas alterações**

Listas

STL - List

Lists são implementações (claro) de listas

- Possuem alocação dinâmica, ou seja, não precisamos saber *a priori* o número de elementos

Na prática, utiliza-se muito a classe **Vector** para fazer as mesmas coisas. Segundo a documentação, as *lists* costuma ter melhor desempenho em inserção, remoção e trocas de posição.

STL - List

Vamos por partes

```
#include<list>  
using namespace std; // wut?
```


STL - List

Para declarar, podemos definir um tipo ou usar qualquer tipo já predefinido do C. Ainda, podemos usar um novo tipo, uma outra ED e assim por diante.

```
list<int> l; //poderia ser outro tipo
```

STL - List

Podemos inserir e remover (sem retornar) em qualquer extremidade

```
l.push_back(1); //insere 1 no fim
```

```
l.push_front(2); //insere 2 no início
```

```
l.pop_back(); //remove aquele 1 do fim
```

```
l.pop_front(); //remove aquele 2 do início
```

STL - List

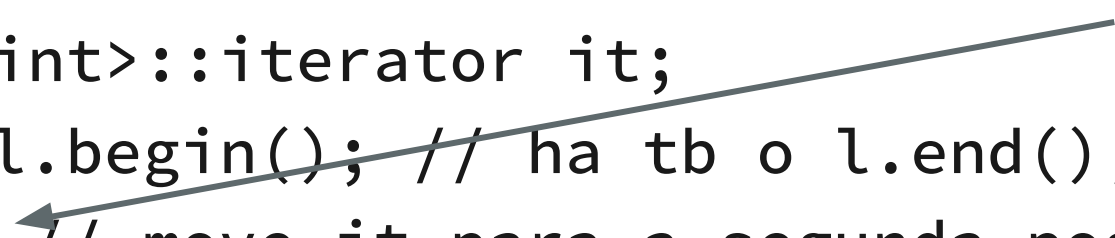
Mas, também podemos inserir em qualquer posição, com a ajuda de um iterador. Vejam “iterador” como o ponteiro auxiliar para que sempre usamos para percorrer a lista.

```
list<int>::iterator it;  
it = l.begin(); // ha tb o l.end();  
it++; // move it para a segunda posicao  
l.insert(it,3); // insere 3 na segunda pos
```

STL - List

Mas, também podemos inserir em qualquer posição, com a ajuda de um iterador. Vejam “iterador” como o ponteiro auxiliar para que sempre usamos para percorrer a lista.

```
list<int>::iterator it;
it = l.begin(); // ha tb o l.end();
it++; // move it para a segunda posicao
l.insert(it,3); // insere 3 na segunda pos
```

A diagram consisting of a long arrow pointing from the text 'advance(it, nPos);' on the right side of the code block to the 'it++;' line. The arrow starts at the 'advance' function and points to the 'it++' statement, indicating that this line of code is equivalent to the commented-out 'advance' function call.

STL - List

Outras muitas utilidades

`l.front();` // retorna o primeiro

`l.back();` // retorna o último

`l.erase(it);` // remove por posicao/iterador

`l.clear();` // limpa a lista

`l.size();` // retorna o número de elementos

`l.empty();` // retorna se está vazia

STL - Vector

Vectors são mais fáceis de usar

- Acesso direto, sem iteradores
- Seria o equivalente à implementação estática, mas o tamanho cresce conforme necessidade
- Métodos bastante parecidos

Filas

STL - Queue

Queue são implementações (claro) de filas, com as operações

- empty (retrona se vazia)
- size (retorna o número de elementos na fila)
- front (retorna o elemento da frente da fila - próximo)
- push (enfileira elemento)
- pop (desenfileira elemento)

STL - Queue

Declaração e uso similares à list

```
#include<queue>
```

```
using namespace std;
```

```
...
```

```
queue<int> q; //poderia ser outro tipo
```

Pilhas

STL - Stack

Stacks são implementações (claro) de pilhas, com as operações

- empty (retrona se vazia)
- size (retorna o número de elementos na pilha)
- top (retorna o elemento no topo da pilha)
- push (empilha elemento)
- pop (desempilha elemento)

STL - Stack

Declaração e uso similares à list

```
#include<stack>
```

```
using namespace std;
```

```
...
```

```
stack<int> s; //poderia ser outro tipo
```

Outras utilidades

STL - Outras utilidades

- **Sets:** Lembrem do TAD conjuntos? Então.
- **Maps:** liga um valor de um tipo a um valor de outro (ou mesmo) tipo. Implementação disso vai ser matéria da P3.
- **Pair:** como uma struct com duas variáveis. Ajuda em muita coisa. Por exemplo, priority queues.

STL - PriorityQueue

É uma implementação de *heaps* para filas de prioridade. Por enquanto, veja como uma lista ordenada pelo valor da chave. Um jeito bom de fazer isso vai ser matéria da P3 ;)

```
#include<queue>
```

```
...
```

```
priority_queue<int> pq;
```

```
priority_queue< pair<int, string> > pq2;
```

STL - Algorithm

Essa classe tem um zilhão (!) de algoritmos interessantes. Ex:

- **sort**: Ordena seu vetor de forma eficiente.
- **min_element/max_element**: Retorna mínimo/máximo de dois elementos ou de um vetor (pode ser Vector).
- **find**: procura elemento no vetor (há muitas variações)

STL

Para saber mais (e tem muuuuuito mais), procure em:

<http://www.cplusplus.com/reference/stl/>

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

Ou

<http://pt.cppreference.com/w/cpp/container>

<http://pt.cppreference.com/w/cpp/algorithm>

O Ministério da Saúde adverte:
Utilizar STL em vez de implementações próprias das
ED requeridas em cada trabalho pode ser prejudicial à
sua nota.

Em caso de suspeita de dengue, um médico deverá
ser consultado.