

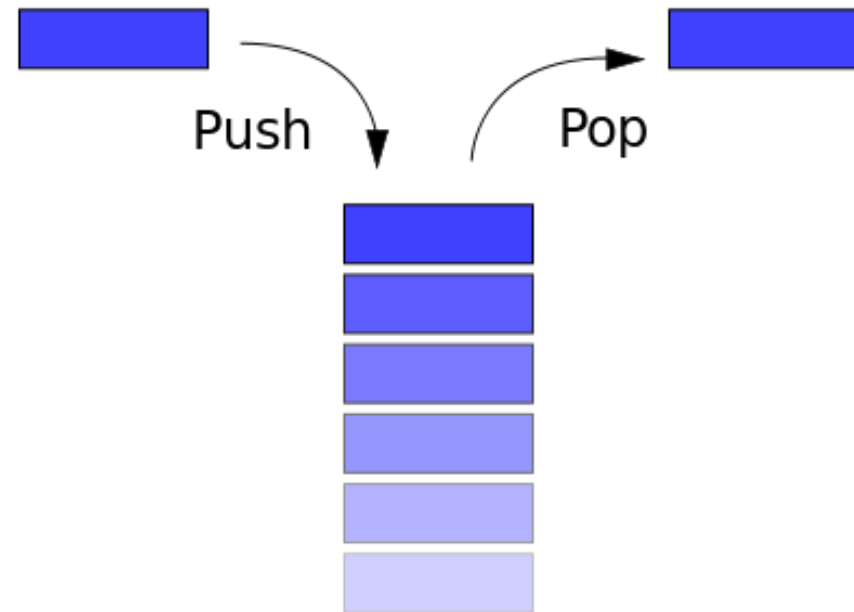
Programação



Pilha

Pilha

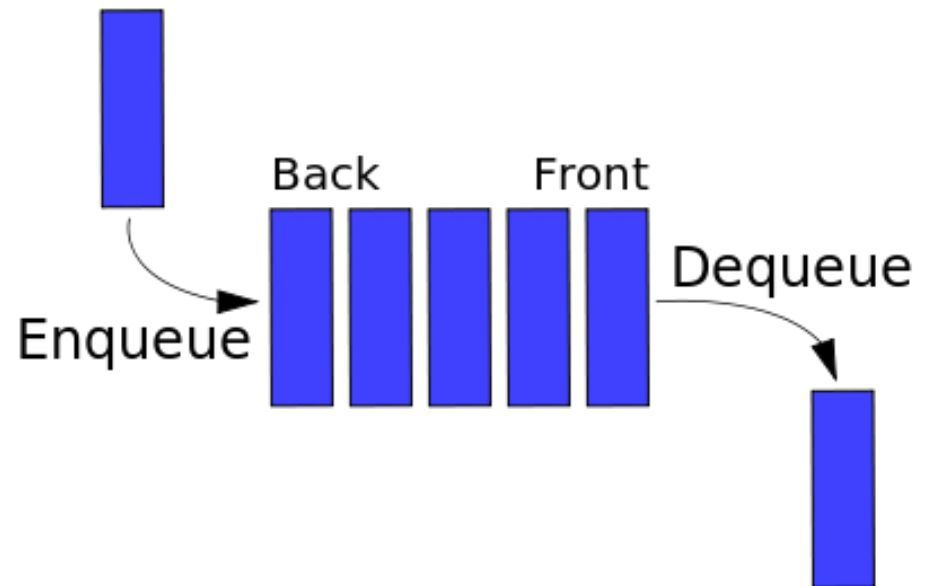
- Uma **pilha** é uma estrutura de dados que implementa a política *last-in, first-out* (LIFO)
- Os elementos são inseridos e removidos no **topo**.



Fila

Fila

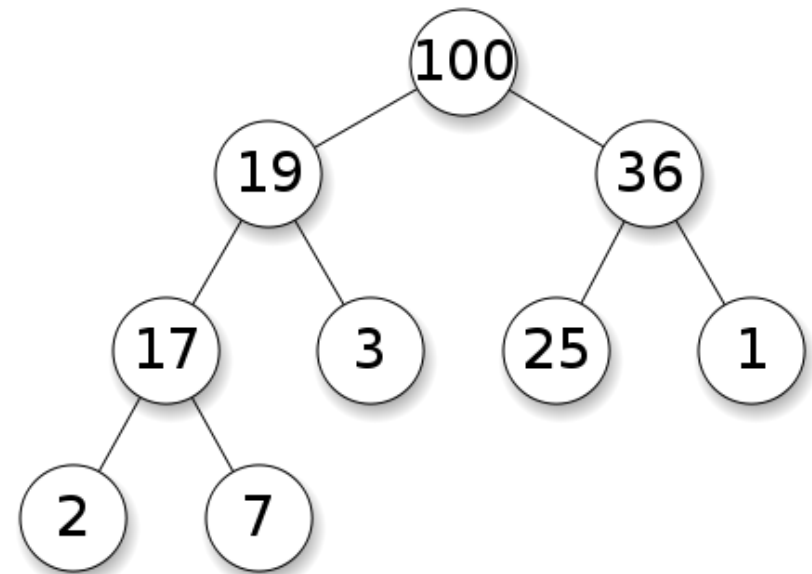
- Uma **fila** é uma estrutura de dados que implementa a política *first-in, first-out* (FIFO)
- Os elementos são inseridos no **final** e removidos na **frente**.



Fila de Prioridades

Fila de Prioridade

- Uma fila de prioridades (ou *heap*) é como uma **fila**
- Porém, os elementos possuem uma *prioridade*;
- Elementos são inseridos no início e são removidos da fila de acordo com sua prioridade;
- A *maior* prioridade é considerada.



Adaptadores de Contêineres

Adaptadores de Contêineres

- **Adaptadores de contêineres** são classes que usam um contêiner encapsulado como estrutura subjacente;
- Na STL:
 - Uma **pilha** pode ser implementada sobre um *vector*, *deque* ou *list*;
 - Uma **fila** pode ser implementada sobre um *deque* ou *list*;
 - Uma **fila de prioridades** pode ser implementada sobre um *vector* ou um *deque*.

stack, queue e priority_queue

stack, queue e priority_queue

- Os contêineres pilha, fila e fila de prioridades contêm praticamente os mesmos métodos:
 - **empty**: testa se o contêiner está vazio;
 - **size**: retorna a quantidade de elementos do contêiner;
 - **push**: insere um elemento;
 - **pop**: remove um elemento;
 - **top** (exceto fila): acessa o elemento do topo;
 - **front** (somente fila): acessa o próximo elemento;
 - **back** (somente fila): acessa o último elemento.

stack

- O adaptador de contêiner *stack* implementa a estrutura de dados pilha
 - Por padrão, utiliza um *deque* como estrutura subjacente;
 - Os elementos são inseridos e removidos no final da estrutura.

stack

```
#include <iostream>
#include <stack> // definição de stack
#include <vector>
#include <list>
using namespace std;
int main(){
    // pilha com deque subjacente padrão
    stack< int > intDequeStack;
    // pilha com vector subjacente padrão
    stack< int, vector<int> > intVectorStack;
    // pilha com deque subjacente padrão
    stack< int, list<int> > intListStack;
```

stack

// insere o valor na pilha

```
intDequeStack.push(1);
```

// imprime o tamanho e o elemento do topo

```
cout << intDequeStack.size() << ' ' << intDequeStack.top();
```

// remove o elemento da pilha

```
intDequeStack.pop();
```

```
return 0;
```

```
}
```

queue

- O adaptador de contêiner *queue* implementa a estrutura de dados fila
 - Por padrão, utiliza um *deque* como estrutura subjacente;
 - Os elementos são inseridos no final e removidos no início da estrutura.

queue

```
#include <iostream>
#include <queue> // definição da classe queue
#include <list>
using namespace std;
int main(){
    queue< int, list<int> > intListQueue; // fila usando list
    queue< double > values; // fila usando deque
    // insere elementos nos valores de fila
    values.push( 3.2 );
    values.push( 9.8 );
    values.push( 5.4 );
```


queue

```
// remove elementos da fila
while ( !values.empty()) {
    cout << values.front() << ' '; // visualiza elemento da frente da fila
    values.pop(); // remove o elemento
}

return 0;
}
```

priority_queue

- O adaptador de contêiner `priority_queue` implementa a estrutura de dados fila de prioridades
 - Por padrão, utiliza um *vector* como estrutura subjacente, organizado como um *heap*;
 - Elementos são inseridos e removidos no final da estrutura.
- Por padrão, quanto maior o valor do elemento, maior sua prioridade.

priority_queue

```
#include <iostream>
#include <queue> // definição do adaptador priority_queue
#include <deque>
using namespace std;
int main(){
    priority_queue< double, deque<double> > dequePriorityQueue; // usa
    deque
    priority_queue< double > priorities; // usa vector
    // insere elementos em prioridades
    priorities.push( 3.2 );
    priorities.push( 9.8 );
    priorities.push( 5.4 );
```

priority_queue

```
cout << priorities.size(); // imprime a quantidade de elementos
// remove elemento de priority_queue
while ( !priorities.empty() )
{
    cout << priorities.top() << ' '; // visualiza elemento superior
    priorities.pop(); // remove elemento superior
}

return 0;
}
```

Problemas Seleccionados

Problemas Seleccionados - Geral

- <http://www.urionlinejudge.com.br/judge/pt/problems/view/1340>

Problemas Seleccionados - Pilha

- <http://www.urionlinejudge.com.br/judge/pt/problems/view/1068>
- <http://www.urionlinejudge.com.br/judge/pt/problems/view/1069>
- <http://www.urionlinejudge.com.br/judge/en/problems/view/1451>

Problemas Seleccionados - Fila

- <http://www.urionlinejudge.com.br/judge/pt/problems/view/1425>
- <http://www.urionlinejudge.com.br/judge/pt/problems/view/1110>



Perguntas?