Programação Orientada a Objetos em C++

A Biblioteca Padrão de Gabaritos

Agostinho Brito

2020

Agenda

Apresentação

A Biblioteca Padrão de Gabaritos

- Os gabaritos (*templates*) são recursos do C++ usados para desenvolver programação genérica.
- A *Standard Template Library* - STL - vem sendo desenvolvida desde a década de 80, cujas ideias iniciais foram propostas por Alexander Stepanov.
- Em 1984, o comitê ANSI/ISO aprovou a proposta inicial para inclusão da biblioteca ao conjunto de funcionalidades padrão dos compiladores C++.
- Ela provê uma quantidade enorme de funcionalidades, MUITO MAIORES que as providas pelas bibliotecas do C padrão.
- Todas as funcionalidades da são providas sob o namespace `std`.

A Biblioteca Padrão de Gabaritos

Containers são estruturas capazes de armazenar dados em alguma estrutura, como listas, filhas, pilhas, deques, conjuntos, mapeamentos, etc.

Iteradores são a generalização de um ponteiro. Usando um objeto do tipo iterador, é possível caminhar pelos containers para acessar seus dados, muito embora possam acessar tipos primitivos também.

Algoritmos são disponibilizados para implementar processos de busca e classificação de dados nos containers, geralmente operando através dos iteradores.

Functores são classes que implementam a sobrecarga de `operator()` para prover diversos tipos de funções diferentes, tais como somar, procurar ou ordenar elementos.

Containers

Containers

- STL provê alguns tipos de estruturas de dados mais comuns usadas em programação:

Containers sequenciais como `vector`, `list` e `deque`.

Adaptadores de containers como `stack` ou `queue`.

Containers associativos como `set` ou `map`, ordenados por uma chave.

Containers associativos desordenados como `unordered_set`, dispersados por uma chave.

- Cada classe provê acesso a iteradores, elementos, informações como o tamanho do container, se está vazio, além de modificadores de conteúdo.

A classe `vector`

- `vector` é um dos containers mais simples. Ele provê um conjunto de elementos contínuos cujo acesso é aleatório, realizado usando uma sobrecarga de `operator[]`.
- `vector` provê um bloco dinamicamente alocado para armazenar internamente seus elementos, de sorte que precisa ser realocado para crescer em tamanho.
- Geralmente, containters `vector` alocam mais elementos que o necessário para evitar realocações constantes e gasto desnecessário de tempo.
- Como declarar um objeto da classe `vector`.

```
1 #include <vector>
2 std::vector<int> v; // um vetor vazio de inteiros
3 std::vector<int> w(5, 30); // um vetor com 5 inteiros de valor 30
4 std::vector<int> x(v); // uma copia de v
5 int y[] = {4, 2, -5, 19};
6 std::vector<int> z(y, y + sizeof(y)/sizeof(int)); // copia de um array
```

Praticando `vector`...

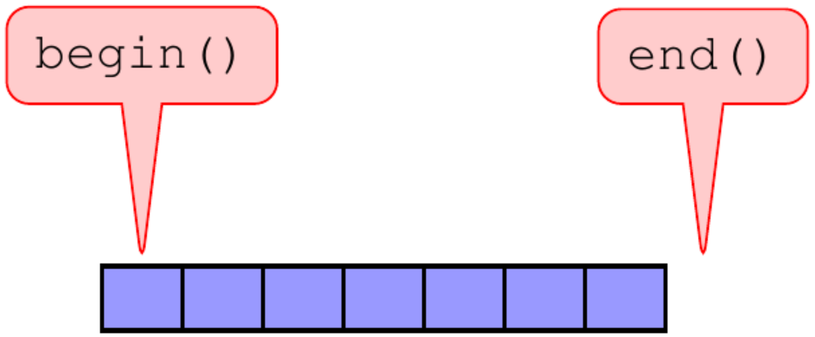
Iteradores

Iteradores

- Iteradores são objetos usados para acessar elementos de um container sequencial.
- Podem ser diretos, quando seu incremento avança na sequência de elementos ou reversos, quando seu incremento retrocede na sequência.
- Como declarar

```
1 #include <vector>
2 int x[] = {4, 2, -5, 19};
3 std::vector<int> v(x, x + sizeof(x)/sizeof(int));
4 std::vector<int>::iterator it;
```

- Os containers sequenciais possuem dois métodos `begin()` e `end()` que retornam iteradores para o início e para o pós-ultimo elemento.



Iteradores

- Invocando os iteradores para os elementos do container sequencial

```
1 std::vector<int> v;
2 std::vector<int>::iterator it;
3 it = v.begin();
4 for(it = v.begin(); it!=v.end(); it++){
5   std::cout << *it << std::endl;
6 }
```

- O laço continua iterando sobre os elementos enquanto o fim não é encontrado.
- A sequência de valores reside, portanto, na faixa `[v.begin(),v.end())`.
- O acesso aos elementos pelo iterador é feito pelo operador de dereferenciação.

Praticando `iterator`...

Algoritmos

Algoritmos

- Os algoritmos são declarados no header `algorithm`.
- Há implementações de algoritmos de busca, ordenação, preenchimento, contagem de elementos, cópia, junção de containers, etc.
- Geralmente, operam em um container através de iteradores que são repassados.

```
std::vector<int> v;
std::vector<int>::iterator it;
it = std::find(v.begin(), v.end(), -5);
```

Praticando `algoritmos`...

Functores

Functores

- Um functor é uma classe projetada para realizar cálculos de funções. A classe deve prover, pelo menos um método `operator()` para que seus objetos possam ser invocados como funções.

```
1 class square {
2   public:
3     float operator()(float &x) const { return x * x; }
4 };
```

```
1 template <typename T>
2 class square {
3   public:
4     T operator()(T &x) const { return x * x; }
5 };
```

Functores na STL

- Os functores podem ser processados por algumas funções providas pela STL, como `transform()` e `copy()`.

```
1  class square {
2   public:
3    float operator()(float &x) const { return x * x; }
4  };
5
6  std::transform(x.begin(), x.end(), y.begin(), square()));
```

Praticando `functores`...

Obrigado