Programação Orientada a Objetos em C++

Gabaritos (Templates)

Agostinho Brito

2020

Agenda

O que são gabaritos

Motivação

- Uma das funcionalidades mais procuradas na programação é o reuso de código.
- Considere os dois exemplos a seguir:

```
float media(float a, float b){
  return (a+b)/2;
}
```

```
int media(int a, int a){
  return (a+b)/2;
}
```

- Trata-se EXATAMENTE do MESMO ALGORITMO, onde a única mudança existente é nos tipos de dados de ambas as funções.
- Mas, considerando tipos diferentes de dados, seria necessária a implementação em C++ de uma sobrecarga para cada um.
- Solução: usar **gabaritos**.

Motivação

- Uma das funcionalidades mais procuradas na programação é o reuso de código.
- Considere os dois exemplos a seguir:

```
float media(float a, float b){
  return (a+b)/2;
}
```

```
int media(int a, int a){
  return (a+b)/2;
}
```

- Trata-se EXATAMENTE do MESMO ALGORITMO, onde a única mudança existente é nos tipos de dados de ambas as funções.
- Mas, considerando tipos diferentes de dados, seria necessária a implementação em C++ de uma sobrecarga para cada um.
- Solução: usar **gabaritos**.

O que são gabaritos

- Gabaritos são trechos de código escritos em C++ usando programação genérica.
- Em suma, constrói-se o algoritmo abstraindo o tipo na forma de um identificador, que poderá ser substituído em tempo de compilação.
- Assim, torna-se possível criar várias versões do mesmo algoritmo, uma para cada tipo necessário, usando exatamente o mesmo código.
- Em C++, pode-se criar **gabaritos de funções** e **gabaritos de classes**.
- Os gabaritos são criados com a palavra reservada `template`.

Gabaritos de Funções

Gabaritos de Funções

- Um gabarito de função é uma implementação genérica de uma função em C++, funcionando como no exemplo seguinte:

```
1 template <typename T>
2 T media(T a, T b){
3   return (a+b)/2;
4 }
```

- A linha `template <typename T>` indica que a função **IMEDIATAMENTE** a seguir será uma implementação genérica.
- As construções das linhas seguintes à função não são influenciadas pela declaração do gabarito.
- Todas as ocorrências do símbolo `T` serão substituídas pelo tipo fornecido durante a utilização.
- A palavra reservada `typename` pode também ser substituída por `class`.

Gabaritos de funções

- A função genérica é instanciada quando sua construção é invocada, fornecendo o tipo que será substituído, conforme o exemplo a seguir:

```
1 int a=1, b=2, c=3;
2 float x=4, y=5, z=6;
3 c = media<int>(a, b);
4 c = media(a, b);
5 z = media<float>(x, y);
6 z = media(x, y);
```

- Nestes casos, o compilador cria sobrecargas da função `media` para atender a cada um dos tipos fornecidos.
- Na prática, a função `media<int>()` é usada como se houvesse sido declarada da seguinte forma:

```
1 int media(int a, int b){
2   return (a+b)/2;
3 }
```

Gabaritos de Funções

- Fornecer o tipo após o nome da função pode ser dispensável, se os tipos passados como argumentos forem suficientes para que o compilador identifique o gabarito necessário.

```
z = media(a, x); // ERRO qual a sobrecarga???
// int media(int a, int b);
// float media(float a, float b);
```

Praticando gabaritos de funções...

Obrigado

Gabaritos de Classes

Gabaritos de Classes

- Um gabarito de classe, assim como um gabarito de função, também é uma implementação de um algoritmo genérico.
- A definição de um gabarito de classe é feita da forma

```
1 template <typename T>
2 class Alo{
3 private:
4   T x, y;
5 public:
6   Alo(T param);
7   T getX();
8 };
```

- Novamente, a declaração `template <typename T>` influencia apenas a classe que segue.

- A classe genérica é instanciada quando sua construção é invocada, fornecendo o tipo que será substituído, como no exemplo a seguir:

```
1 Alo<int> aloInt;
2 Alo<float> aloFloat;
3 int a = aloInt.getX();
4 float x = aloFloat.getX();
```

- Nestes casos, o compilador cria diferentes classes `Alo`, uma para cada tipo fornecido.

Praticando gabaritos de classes...

Gabaritos de Classes

- Em se tratando da implementação de uma classe, é importante notar que a declaração e a definição da classe geralmente são implementadas em momentos distintos.
- Logo, para cada implementação de um método, é necessário criar uma nova declaração de template para que a construção seguinte possa ser influenciada por este.

```
1  template <typename T>
2  class Alo{
3  private:
4    T x, y;
5  public:
6    Alo(T x_, T y_);
7    T getX();
8  };
9
10 template <typename T>
11 Alo<T>::Alo(T x_, T y_){
12   x = x_; y = y_;
13 }
```

Praticando gabaritos de classes...

Algumas considerações sobre o uso de Gabaritos

- Gabaritos podem ser criados para mais de um tipo ao mesmo tempo.

```
1 template <typename T, typename U>
2 class Alo{
3   T x;
4   ...
5   void doIt(U param);
6 };
7 Alo<int, float> alo;
```

- É permitido também uso de parâmetros *non-type*.

```
1 template <typename T, int N=10>
2 class Vetor{
3 private:
4   T x[N];
5   ...
6 };
7 Vetor<int, 30> v;
```

Algumas considerações sobre o uso de Gabaritos

- Apesar da versatilidade, os gabaritos são estruturas que precisam ser definidas **EM TEMPO DE COMPILAÇÃO**, mas **NÃO EM TEMPO DE EXECUÇÃO**.
- Os compiladores C++ não permitem que o código fonte de um gabarito seja compilado na forma de um módulo separado, para posterior linkedição com os demais módulos.
- Toda as estruturas de classes e funções precisam estar prontas para uso, incluindo os tipos de variáveis que serão utilizadas, antes de a execução do programa ter início.
- Logo, bibliotecas de programação baseadas em gabaritos devem ser providas na forma de código fonte (`.hpp`, `.h`). problema???

Algumas considerações sobre o uso de Gabaritos

- Apesar da versatilidade, os gabaritos são estruturas que precisam ser definidas **EM TEMPO DE COMPILAÇÃO**, mas **NÃO EM TEMPO DE EXECUÇÃO**.
- Os compiladores C++ não permitem que o código fonte de um gabarito seja compilado na forma de um módulo separado, para posterior linkedição com os demais módulos.
- Toda as estruturas de classes e funções precisam estar prontas para uso, incluindo os tipos de variáveis que serão utilizadas, antes de a execução do programa ter início.
- Logo, bibliotecas de programação baseadas em gabaritos devem ser providas na forma de código fonte (`.hpp`, `.h`). problema???

Algumas considerações sobre o uso de Gabaritos

- Apesar da versatilidade, os gabaritos são estruturas que precisam ser definidas **EM TEMPO DE COMPILAÇÃO**, mas **NÃO EM TEMPO DE EXECUÇÃO**.
- Os compiladores C++ não permitem que o código fonte de um gabarito seja compilado na forma de um módulo separado, para posterior linkedição com os demais módulos.
- Toda as estruturas de classes e funções precisam estar prontas para uso, incluindo os tipos de variáveis que serão utilizadas, antes de a execução do programa ter início.
- Logo, bibliotecas de programação baseadas em gabaritos devem ser providas na forma de código fonte (`.hpp`, `.h`). 👈 problema???

🎁

C++ provê **STL**, a Biblioteca Padrão de Gabaritos!

Obrigado