Programação Orientada a Objetos em C++

Funções Amigas

Agostinho Brito

2020

Analisando a sobrecarga `operator*()`

```
1  class Vetor2d{
2    public:
3      Vetor2d operator* (float a){
4        Vetor2d ret;
5        ret.x = x*a;
6        ret.y = y*a;
7        return ret;
8      }
9  };
10 ...
11 Vetor2d v1(3,4), v2;
12 v2 = v1 * 4;
```

- Mas, e quanto a...

```
v2 = 4 * v1; // ???
```

Analisando a sobrecarga `operator*()`

- `v1*4` equivale a `v1.operator*(4)`.
- Mas `4*v1` equivaleria a que? **`4.operator*(v1)`**???
- Entretanto...

1

A operação de multiplicação de escalar por vetor produz o mesmo resultado da multiplicação vetor por escalar, podendo compartilhar, inclusive, uma mesma implementação.

2

Tipicamente, essa poderia ter acesso às propriedades privadas da classe, evitando chamadas desnecessárias de funções `get/set`.

- Essa funcionalidade pode ser provida com o uso de funções amigas.

Analisando a sobrecarga `operator*()`

- `v1*4` equivale a `v1.operator*(4)`.
- Mas `4*v1` equivaleria a que? **`4.operator*(v1)`**???
- Entretanto...

1

A operação de multiplicação de escalar por vetor produz o mesmo resultado da multiplicação vetor por escalar, podendo compartilhar, inclusive, uma mesma implementação.

2

Tipicamente, essa poderia ter acesso às propriedades privadas da classe, evitando chamadas desnecessárias de funções `get/set`.

- Essa funcionalidade pode ser provida com o uso de funções amigas.

Analisando a sobrecarga `operator*()`

- `v1*4` equivale a `v1.operator*(4)`.
- Mas `4*v1` equivaleria a que? **`4.operator*(v1)`**???
- Entretanto...

1

A operação de multiplicação de escalar por vetor produz o mesmo resultado da multiplicação vetor por escalar, podendo compartilhar, inclusive, uma mesma implementação.

2

Tipicamente, essa poderia ter acesso às propriedades privadas da classe, evitando chamadas desnecessárias de funções `get/set`.

- Essa funcionalidade pode ser provida com o uso de funções amigas.

Criando amizades...

- Diz-se que uma função é amiga de uma classe quando é colocado antes de seu tipo de retorno o especificador `friend`.
- Uma vez a função é declarada como amiga, esta terá acesso às propriedades e métodos privados e protegidos que a classe possui.

```
friend Vetor2d operator*(float a, Vetor2d v);
```

- Neste caso, a função amiga `operator*(float, Vetor2d)` poderá suprir a funcionalidade que procuramos para realizar a operação `v2 = 4 * v1`.
- Logo...

```
v2 = operator*(4,v1);
```

e

```
v2 = 4*v1;
```

são equivalentes.

Criando amizades...

```
1  class Vetor2d{
2    public:
3      friend Vetor2d operator* (float a, Vetor2d v);
4  };
5
6  Vetor2d operator*(float a, Vetor2d v){
7    Vetor2d ret;
8    ret.x = a*v.x;
9    ret.y = a*v.y;
10   return ret;
11 }
12 ...
13 Vetor2d v1(3,4), v2;
14 v2 = 4 * v1; // ou v2 = operator*(4,v1);
```

Praticando funções amigas...

Obrigado