

Programação Orientada a Objetos em C++

Sobrecarga de Operadores

Agostinho Brito

2020

Somando dois Objetos da classe Vetor2d...

- Opção 1: criar o método `Vetor2d soma (Vetor2d v)`, que recebe um vetor e retorna o resultado da soma deste com o executor do método.

```
1  class Vetor2d{
2  public:
3      Vetor2d soma (Vetor2d v) {
4          Vetor2d ret;
5          ret.x = x+v.x;
6          ret.y = y+v.y;
7          return ret;
8      }
9  };
10 ...
11 Vetor2d v1 (3,4), v2 (-1,2), v3;
12 v3 = v1.soma (v2);
```

- E se... ?

```
v3 = v1 + v2;
```

Somando dois Objetos da classe Vetor2d...

- Opção 1: criar o método `Vetor2d soma (Vetor2d v)`, que recebe um vetor e retorna o resultado da soma deste com o executor do método.

```
1  class Vetor2d{
2  public:
3      Vetor2d soma (Vetor2d v) {
4          Vetor2d ret;
5          ret.x = x+v.x;
6          ret.y = y+v.y;
7          return ret;
8      }
9  };
10 ...
11 Vetor2d v1 (3, 4), v2 (-1, 2), v3;
12 v3 = v1.soma (v2);
```

- E se... ?

```
v3 = v1 + v2;
```

Criando sobrecarga de operadores

- Opção 2: usar o recurso de sobrecarga de operadores da linguagem.
- Como funciona: cria-se um método nomeado `operator` <operador>.

```
1  class Vetor2d{
2  public:
3      Vetor2d operator + (Vetor2d v) {
4          Vetor2d ret;
5          ret.x = x+v.x;
6          ret.y = y+v.y;
7          return ret;
8      }
9  };
10 ...
11 Vetor2d v1(3,4), v2(-1,2), v3;
12 v3 = v1 + v2;
```

Utilizando a
sobrecarga

Operadores permitidos e não permitidos

- Os seguintes operadores podem ser sobrecarregados:


`+`, `-`, `*`, `/`, `%`, `^`, `&`, `|`, `~`, `!`, `,`, `=<`, `>`, `<=`, `>=`, `++`, `--`, `<<`,
`>>`, `==`, `!=`, `&&`, `||`, `+=`, `-=`, `/=`, `%=`, `^=`, `&=`, `|=`, `*=`, `<<=`, `>>=`,
`[]`, `()->`, `->*`, `new`, `new []`, `delete`, `delete []`

- Os seguintes operadores **não** podem ser sobrecarregados:

`::` (escopo), `.` (acesso), `.*` (acesso via ponteiro), `?:` (ternário).

- Novos operadores não podem ser criados, tipo `**` ou `<>`.
- O operador de atribuição `=` deve ser implementado com cuidado, levando em conta a cópia ou a movimentação de objetos temporários durante a auto-atribuição.

- O significado da sobrecarga é DEFINIDA pelo programador, não impondo restrições. Ex:
`operator +` pode ser usado para “subtrair” dois objetos.

 Praticando sobrecarga de operadores...

- Como diferenciar os operadores unários de incremento pré-fixado e pós-fixado?

```
1 Vetor2d v;  
2 v++; ++v;  
3 v--; --v;
```

- C++ diferencia um do outro pelo uso de um argumento que é passado (embora não utilizado) para o operador de incremento pós-fixado.

```
1 void operator++() {  
2     cout << "pre-fixado - ex: ++x\n";  
3 }  
4  
5 void operator++(int) {  
6     cout << "pos-fixado - ex: x++\n";  
7 }
```



Obrigado

Sobrecarga do operador ()

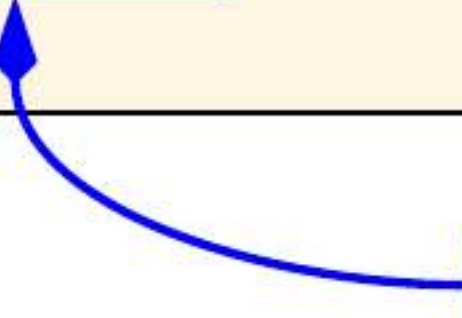
- O operador `()`, quando implementado, permite que o objeto figure como uma função

```
1  class Linear {
2      float a, b;
3  public:
4      Linear(float a_=0, float b_=0) {
5          a = a_; b = b_;
6      }
7      float operator() (float x) {
8          return a*x + b;
9      }
10 };
11 int main() {
12     Linear func(3, 4); // func(x) = 3*x+4
13     float res = func(2);
```

Sobrecarga do operador = (atribuição de cópia)

```
1 Vetor2d& Vetor2d::operator=(const Vetor2d& outro) {  
2     if (this != &outro) {  
3         // copia estrutura do outro para este objeto  
4     }  
5     return *this;  
6 }  
7 int main(void) {  
8     Vetor2d v1, v2;  
9     v1 = v2;  
10 }
```

Atribuição de cópia



Sobrecarga do operador = (atribuição de movimentação de temporários)

```
1 Vetor2d& Vetor2d::operator=(const Vetor2d&& outro) {  
2     if (this != &outro) {  
3         // copia estrutura do outro para este objeto  
4     }  
5     return *this;  
6 }  
7 int main(void) {  
8     Vetor2d v1, v2, v3;  
9     v3 = v1+v2;  
10 }
```

Objeto temporário





Obrigado