VIII. Operadores – Introdução

Operadores Unários

Operadores Binários

Sobrecarga de Operadores

Operadores

- Operadores são métodos com uma sintaxe algébrica, matemática.
  - Ações → Rotinas → Métodos
- Existentes nas LPs, desde o Fortran54.
  - Porém apenas para tipos de dados básicos da linguagem.

Operadores

- Deficiência das LPs tradicionais
  - tipos de dados criados pelo usuário não tem o mesmo poder dos tipos de dados básicos
    - int a,b,c; c = a+b; //SIM
    - Fracao m,n,o; o = m+n; //NÃO
- Transformar expressões obscuras
  - Venda Total, A, B;
  - Total.add_venda(A,B) //obscuro
  - Total = A + B; // mais claro

Definindo Operadores

- Operadores unários
  - incremento (++)
  - decremento (--)
  - menos (-)
- Operadores binários
  - +,-,*,/,>=,+=,[], ...
- Não podem ser sobrecarregados:
  - ponto de acesso a membros (.)
  - resolução de escopo (::)
  - condicional ternário (?:)

Definindo Operadores

- Limitações
  - Não é permitido mudar a cardinalidade de um operador já existente
    - + é binário
    - ! é unário
  - Não é permitido inventar novos operadores
    - ##
    - ^
  - Manter obediência à precedência original do operador

Sobrecarga de Operadores

- Operador unário

```
<tipo_retorno> operator <sinal>()
```

- Operador binário

```
<tipo_retorno> operator <sinal>(<parametro>)
```

Classe Ponto

```
class Ponto {
   int x,y;
public:
   Ponto(int x1=0, int y1=0) { //valor default
      x=x1;  y=y1;
   }
   void operator ++() { x++; y++; }

   void mostra() const {  //não altera objeto
      cout << '(' << x << ',' << y << ')';
   }
};
```

Main

```
int main() {
    Ponto p1, p2(2,2), p3;
    p1.mostra();
    p2.mostra();
    ++p1;
    ++p2;
    p1.mostra();
    p2.mostra();
    p3 = p1;
    p3.mostra();
}
```

Operador de Incremento Prefixado

- Definição
  - void operator ++() { x++; y++; }
- Uso
  - ++p1;
  - p1.operator++(); //forma de método

Operador de Incremento Pós-fixado

- Definição
  - void operator ++(int) { x++; y++; }
- Uso
  - p1++;
  - p1.operator++(1); //forma de método

Operador Binário

- Definição

```
Ponto operator +(Ponto outro) {
    Ponto aux;
    aux.x = this->x + outro.x;
    aux.y = this->y + outro.y;
    return aux;
}
```

- Uso

```
p3 = p1+p2;
p3 = p1.operator+(p2);          //forma de método
```

Exercícios

1) Construir um operador para subtração (–)

   p3 = p1 – p2;

2) Construir um operador (==) que compara dois pontos

   p1 == p2 ??

3) Tornar possível

   p3 = ++p1;

   p3 = p1++;

4) Construir outro operador (+) que soma um inteiro ao ponto

   p2 = p1 + 3;

Exercícios

5) Fazer um operador (-) para inverter x e y.
6) Fazer um operador (!) para negar x e y.
7) Como permitir um inteiro alterar tanto x quanto y?
   - p1 = 3;

EXERCÍCIO 2

Exercício 2 – Classe String

- Seja:

```
const int MAX=80;

class String {
   char buffer[MAX];
   ...
};
```

- Fazer a classe String suportar os seguintes operadores:

```
+
+=
==
>
<
```

Main

```
int main() {
   String s1(“Bom dia!”), s2(“Ate logo”);
   String s3,s4;
   s1++; //tudo maiúscula
   s2--; //tudo minúscula
   s3 = s1 + s2;
   s4 = s1 + s1;
   s4 += s2;
   s5 = s1;
   if( s1 == s5 ) cout << “São iguais”;
   else cout << “São diferentes”;
   if( s1 > s2 )
      cout << s2 << “ vem antes de “ << s1;
}
```

Classe String

- Operador de concatenação (+)
- Operador de concatenação (+=)
- Operadores relacionais (==,!=,>,<,>=,<=)
- Operador de elemento do vetor: []
- Operador de ++: uppercase
- Operador de --: lowercase
- Função de tamanho – size()