Перегрузка операторов (продолжение)

6 июня 2017 г.

Повторение

1. Какое значение имеет параметр int при перегрузке постфиксной формы декремента?
2. Можно ли ввести свои собственные операторы для работы с классом?
3. Можно ли перегрузить операторы для стандартных типов данных (int, double, char…)?
4. Какие операторы всегда следует реализовывать как методы класса?
5. Что должны возвращать операторы:

/ != = == [ ] ( ) ++ char

Какое вызовы оператора [ ] верны?

```
void f(){
    Point myPoint(10, 7);
1)  myPoint[0] = 8;
2)  myPoint.operator[ ](0, 8);
3)  this->operator[0](8);
4)  myPoint.operator[ ](0) = 8;
5)  this.operator[8];
6)  operator[0](this, 8);
}
```

Выберите верные объявления

```
class MyClass{
    /* … */
};
```

1) `MyClass& int( ) const;`
2) `int( ) const;`
3) `operator int( ) const;`
4) `MyClass operator int( ) const;`
5) `void operator int( ) const;`
6) `operator int ( ) = delete;`
7) `operator int( ) = default;`

Какие методы можно объявить с ключевым словом default?

```
class MyClass{
public:
    1) MyClass();
    2) MyClass(const int);
    3) MyClass(const int, const double);
    4) MyClass(const MyClass&);
    5) MyClass& operator=(const MyClass&);
    6) bool operator ==(const MyClass&);
    7) ~MyClass();
    8) MyClass operator +(const MyClass&);
};
```

Перегрузка глобальной функцией

1. Функция для унарной операции получает один явный аргумент.
2. Функция для бинарной операции получает два явных аргумента.

унарный -

- ClassA operator-(const ClassA&);

бинарный -

- ClassA operator-(const ClassA&, const ClassA&);

Перегрузка глобальной функцией

3. Объявляется и реализуется вне класса.
4. Перед словом operator не надо писать имя класса (~~*MyClass::*~~operator+).
5. Обращение к полям класса осуществляется через аксессоры.

```
return-type operator op( arg );
```

Арифметические бинарные операторы: пример

```
class Digit{
      int x_;
public:
      explicit Digit(const int x) : x_(x) { }
      int getDigit( ) const { return x_; }
      void setDigit(const int x) { x_ = x; }
};

Digit  operator+(const Digit& a, const Digit& b){
      Digit tmp;
      tmp.setDigit(a.getDigit() + b.getDigit());
      return tmp;
}
```

Бинарные операторы: пример

```
void f ( ){
    Digit a(5);
    Digit b(7);
    Digit c(0);

    c = a + b;

    Digit d = operator+(a, b);
}
```

Унарные операторы: пример

```
class Digit{
      int x_;
public:
      explicit Digit(const int x) : x_(x) { }
      int getDigit( ) const { return x_; }
      void setDigit( const int x ) { x_ = x; }
};

Digit  operator-(const Digit& a){
      Digit tmp = a;
      tmp.setDigit( -a.getDigit() );
      return tmp;
}
```

```
Digit a(10);
Digit b = -a;
b = operator-(a);
```

Операторы >> и <<

```
std::ostream& operator<<( std::ostream&, const Digit&);

std::istream& operator>>( std::istream&, Digit&);
```

- cin – объект класса istream (оператор >>)
  cout – объект класса ostream (оператор <<)
- всегда не член класса
- возвращают ссылку на поток
- принимают два аргумента (ссылка на поток и пользовательский объект)

Операторы << и >>: пример

```
class Digit{ /* … */ };

std::ostream& operator<<(std::ostream& os, const Digit& a){
        os << a.getDigit( );
        return os;
}

std::istream& operator>>(std::istream& is, Digit& a){
        int tmp;
        is >> tmp;
        a.setDigit(tmp);
        return is;
}
```

```
std::cout << a << b;
std::cin >> a >> b;

((std::cout << a) << b);
((std::cin >> a) >> b);
```

Вопросы?