

Перегрузка операторов

1 июня 2017 г.

Общие сведения

1. Перегрузка операторов нужна затем, чтобы **сделать вызов функций более удобным** (поддержать единый интерфейс).

```
MyArray a;
```

```
MyArray b;
```

```
MyArray c;
```

```
c = a + b;
```

```
if (a != b) {
```

```
    b--;
```

```
}
```

```
// объединение двух массивов в одном
```

```
// проверка на тождественность
```

```
// удаление последнего элемента
```

Общие сведения

2. Не следует пользоваться перегрузкой, если выбранный оператор **плохо согласуется со смыслом** функции.

```
Automobile a;  
Automobile b;  
Automobile c;
```

```
c = a + b;           // ???  
if (a != b) {  
    b--;              // ???  
}
```

Операторы с перегрузкой

Оператор	Назначение	Оператор	Назначение
+	сложение	+	унарный плюс
-	вычитание	-	унарный минус
*	умножение	*	разыменованное указателя
/	деление	&	взятое адреса
=	присваивание	[]	обращение по индексу
+=	присваивание сложения	++	инкремент
-=	присваивание вычитания	--	декремент
==	проверка на равенство	!	логическое не

и т. д.

Операторы без перегрузки

Оператор	Назначение
.	обращение к полю или методу объекта
.*	обращение к указателю на поле или метод объекта
::	разрешение области видимости
: ?	тернарный оператор условия
sizeof	размер в байтах объекта или типа данных
#	директива препроцессора
##	директива препроцессора

Задание

Для каких операций (бинарных или унарных) могут использоваться следующие операторы?

+	%	!
=	/=	!=
=	>	->
-	<<	<=
++	->	==
--	&&	>>
/	&	^

Основные правила перегрузки

1. Символ **бинарной** операции используется для переопределения **бинарной** операции.
2. Символ **унарной** операции используется для переопределения **унарной** операции.

+

- два аргумента

++

- один аргумент

+=

- ~~один аргумент~~ ДВА

Основные правила перегрузки

3. Перегрузка **не меняет приоритет** операторов и их **ассоциативность**.

```
MyClass a;
```

```
MyClass b;
```

```
My Class c = a + b * 3;    // (c = (a + (b * 3)))
```

```
MyClass d = c = a = b;    // (d = (c = (a = b)))
```


Способы перегрузки

метод класса (функция-член)

дружественная функция

глобальная функция

Перегрузка методами класса

1. В метод неявно **передается указатель this**.
2. Метод для **унарной** операции не получает **ни одного** явного аргумента.
3. Метод для **бинарной** операции получает **один** явный аргумент.

унарный -

- MyClass `operator-`();

бинарный -

- MyClass `operator-(const MyClass&);`

Перегрузка бинарных операторов

+	+=	=	!=	%	&&	>>	и др.
a + b	a += b	a = b	a != b	a % 2	a && b	a >> 1	

```
returnType operator op( arg );
```

```
MyClass operator+(const int x);
```

```
bool operator==(const MyClass& obj);
```

- есть один аргумент
- обычно возвращается тип класса или ссылка на тип класса

Арифметические бинарные операторы: пример

```
class Digit{  
    int x_;  
public:  
    Digit( ) : x_(0){ }  
    explicit Digit(const int x) : x_(x) { }  
    ~Digit( ){ }  
    Digit operator+(const Digit newDigit){  
        Digit tmp;  
        tmp.x_ = x_ + newDigit.x_;  
        return tmp;  
    }  
};
```

Операторы сравнения: пример

```
class Digit{  
    int x_;  
public:  
    Digit() : x_(0) { }  
    explicit Digit(const int x) : x_(x) { }  
    ~Digit( ){ }  
    bool operator>(const Digit newDigit){  
        return x_ > newDigit.x_;  
    }  
};
```

Оператор присваивания

- должен быть нестатическим методом класса
- не наследуется
- возвращается сам объект (*this)

```
Rectangle& Rectangle::operator=(const Rectangle& rc){  
    if (this == &rc){  
        return *this;  
    }  
    side1 = rc.side1;  
    side2 = rc.side2;  
    return *this;  
}
```

Оператор копирующего присваивания (=)

- если в классе есть конструктор копирования, обязательно нужно перегрузить оператор =

```
MyArray& MyArray ::operator=(const MyArray& obj){  
    if (this == &obj){  
        return *this;  
    }  
    size = obj.size;  
    delete [] arr;  
    arr = new double[size];  
    for (size_t i = 0; i < size; ++i){ arr[i] = obj.arr[i]; }  
    return *this;  
}
```

Оператор перемещающего присваивания (=)

- если в классе есть конструктор перемещения, обязательно нужно перегрузить оператор =

```
MyArray& MyArray ::operator=(MyArray&& obj){  
    if (this == &obj){  
        return *this;  
    }  
    size = obj.size;  
    delete [] arr;  
    arr = obj.arr;  
    obj.size = 0;  
    obj.arr = nullptr;  
    return *this;  
}
```


Перегрузка унарных операторов

!	&	~	*	+	-	++	--
!p	&p	a = ~b	*p	+10	-7	++a	a--

```
returnType operator op( );
```

```
MyClass operator-();
```

```
MyClass operator++();
```

- нет аргументов
- указывается возвращаемый тип

Унарные операторы: пример

```
class Digit{  
    int x_;  
public:  
    Digit() : x_(0) { }  
    explicit Digit(const int x) : x_(x) { }  
    ~Digit( ){ }  
    Digit operator-(){  
        Digit tmp;  
        tmp.x_ = -x_;  
        return tmp;  
    }  
};
```

Инкремент и декремент

- префиксная и постфиксная формы реализуются отдельно

```
void f( ){  
    Rectangle r(1, 1);  
    ++(++(++(++r)));  
  
    Rectangle s(1, 1);  
    s++;  
}
```

Префиксная форма

- нет аргументов
- возвращается сам объект (*this)

```
Rectangle& Rectangle::operator++( ){  
    ++side1;  
    ++side2;  
    return *this;  
}  
  
void f( ){  
    Rectangle r(1, 1);  
    ++(++(++(++r)));  
}
```

Постфиксная форма

- передается один аргумент типа `int`, который не используется и равен 0
- возвращается временный объект

```
Rectangle Rectangle::operator++(int){  
    Rectangle tmp = *this;  
    ++side1;  
    ++side2;      // ++*this;  
    return tmp;  
}
```

Вопросы?