Перегрузка операторов

Visual Studio 2015

Самая актуальная документация по Visual Studio 2017: Документация по Visual Studio 2017 (http://docs.microsoft.com/visualstudio).

Ключевое слово `operator` объявляет функцию, которая указывает, что означает `operator-symbol` при применении к экземпляру класса. Это дает оператору более одного значения — "перегружает" его. Компилятор различает разные значения оператора, проверяя типы его операндов.

Синтаксис

```
type operator operator-symbol ( parameter-list )
```

Заметки

Функцию большинства встроенных операторов можно переопределить глобально или для отдельных классов. Перегруженные операторы реализуются в виде функции.

Перегруженный оператор имеет имя `operator``x`, где `x` означает оператор из следующей таблицы. Например, для перегрузки оператора сложения необходимо определить функцию `operator+`. Аналогично, для сложения/присвоения `+=` необходимо определить функцию `operator+=`.

Переопределяемые операторы

Оператор	Имя	Тип
`,`	Comma	Binary
`!`	Логическое НЕ	Унарный
`!=`	Неравенство	Binary
`%`	Модуль	Binary
`%=`	Назначение модуля	Binary
`&`	Побитовое И	Binary
`&`	Взятие адреса	Унарный
`&&`	Логическое И	Binary

Оператор	Имя	Тип
&=	Назначение побитового И	Binary
( )	Вызов функции	—
( )	Оператор приведения типа	Унарный
*	Умножение	Binary
*	Разыменование указателя	Унарный
*=	Присваивание умножения	Binary
+	Сложение	Binary
+	Унарный плюс	Унарный
++	Инкремент[1]	Унарный
+=	Присваивание сложения	Binary
–	Вычитание	Binary
–	Унарное отрицание	Унарный
––	Декремент[1]	Унарный
–=	Присваивание вычитания	Binary
–>	Выбор члена	Binary
–>*	Выбор указателя на член	Binary
/	Деление	Binary
/=	Присваивание деления	Binary
<	Меньше	Binary
<<	Сдвиг влево	Binary
<<=	Сдвиг влево и присваивание	Binary
<=	Меньше или равно	Binary
=	Назначение	Binary
==	Равенство	Binary
>	Больше	Binary

Оператор	Имя	Тип
>=	Больше или равно	Binary
>>	Сдвиг вправо	Binary
>>=	Сдвиг вправо и присваивание	Binary
[ ]	Нижний индекс массива	—
^	Исключающее ИЛИ	Binary
^=	Исключающее ИЛИ/присваивание	Binary
		Побитовое включающее ИЛИ	Binary
|=	Назначение побитового включающего ИЛИ	Binary
||	Логическое ИЛИ	Binary
~	Дополнение до единицы	Унарный
delete	Delete	—
new	New	—
conversion operators	операторы преобразования	Унарный

1 Существуют две версии унарных операторов инкремента или декремента: префиксная и постфиксная.

Дополнительные сведения см. в разделе Общие правила перегрузки операторов (https://msdn.microsoft.com/ru-ru/library/4x88tzx0.aspx). Ограничения для разных категорий перегруженных операторов описываются в следующих разделах.

- Унарные операторы (https://msdn.microsoft.com/ru-ru/library/f672kxz8.aspx)
- Бинарные операторы (https://msdn.microsoft.com/ru-ru/library/czs2584d.aspx)
- Назначение (https://msdn.microsoft.com/ru-ru/library/7ac5szsk.aspx)
- Вызов функции (https://msdn.microsoft.com/ru-ru/library/df74sak1.aspx)
- Индексация ; (https://msdn.microsoft.com/ru-ru/library/1bhdy722.aspx)
- Обращение к членам класса (https://msdn.microsoft.com/ru-ru/library/ds81sa81.aspx)
- Инкремент и декремент (https://msdn.microsoft.com/ru-ru/library/f6s9k9ta.aspx).
- Преобразования (https://msdn.microsoft.com/ru-ru/library/wwywka61.aspx)

Операторы, перечисленные в следующей таблице, не могут быть перегружены. В этой таблице содержатся символы препроцессора `#` и `##`.

Непереопределяемые операторы

Operator	Name
.	Выбор члена
.*	Выбор указателя на член
::	Разрешение области
? :	Условие
#	Препроцессор: преобразование в строку
##	Препроцессор: конкатенация

Хотя перегруженные операторы обычно называются компилятором неявным образом при их появлении в коде, их можно вызывать и явным образом — точно так же, как и любую функцию-член или функцию, не являющуюся членом.

```
Point pt;
pt.operator+( 3 );  // Call addition operator to add 3 to pt.
```

Пример

В следующем примере выполняется перегрузка оператора + , после чего он складывает два комплексных числа и возвращает результат.

```
// operator_overloading.cpp
// compile with: /EHsc
#include <iostream>
using namespace std;

struct Complex {
   Complex( double r, double i ) : re(r), im(i) {}
   Complex operator+( Complex &other );
   void Display( ) {   cout << re << ", " << im << endl; }
private:
   double re, im;
};

// Operator overloaded using a member function
Complex Complex::operator+( Complex &other ) {
   return Complex( re + other.re, im + other.im );
}

int main() {
   Complex a = Complex( 1.2, 3.4 );
   Complex b = Complex( 5.6, 7.8 );
   Complex c = Complex( 0.0, 0.0 );

   c = a + b;
   c.Display();
}
```

Вывод

```
6.8, 11.2
```

В данном разделе

1. Общие правила перегрузки операторов (https://msdn.microsoft.com/ru-ru/library/4x88tzx0.aspx)
2. Перегрузка унарных операторов (https://msdn.microsoft.com/ru-ru/library/f672kxz8.aspx)
3. Бинарные операторы (https://msdn.microsoft.com/ru-ru/library/czs2584d.aspx)
4. Назначение (https://msdn.microsoft.com/ru-ru/library/7ac5szsk.aspx)
5. Вызов функции (https://msdn.microsoft.com/ru-ru/library/df74sak1.aspx)
6. Индексация (https://msdn.microsoft.com/ru-ru/library/1bhdy722.aspx)
7. Доступ к членам (https://msdn.microsoft.com/ru-ru/library/ds81sa81.aspx)

См. также

Операторы C++ (https://msdn.microsoft.com/ru-ru/library/x04xhy0h.aspx)
Ключевые слова в C++ (https://msdn.microsoft.com/ru-ru/library/2e6a4at9.aspx)