Т.2.7.5. Статический полиморфизм. Перегрузка операций.

Перегрузка операций

Программист может задать интерпретацию операций, когда они применяются к объектам определенного класса. Помимо арифметических, логических и операций отношения можно переопределить вызов функций (), индексацию [], косвенное обращение ->, а также присваивание и инициализацию.

Обычно в программах используются объекты, являющиеся конкретным представлением абстрактных понятий. Например, в C++ тип данных **int** вместе с операциями **+, -, *, / и т.д.** реализует математическое понятие целого.

К сожалению, в языках программирования непосредственно представляется только малое число понятий. Так, понятия комплексных чисел, алгебры матриц, логических сигналов и строк в C++ не имеют непосредственного выражения. Возможность задать представление сложных объектов вместе с набором операций, выполняемых над такими объектами, реализуют в C++ классы.

Перегрузка операций

Имеется два способа описания функции, соответствующей переопределяемой операции:

- если функция задается как обычная функция-элемент класса, то первым операндом операции является объект класса;
- если первый операнд переопределяемой операции не является объектом некоторого класса, либо требуется передавать в качестве операнда не указатель, а сам объект (значение), то соответствующая функция должна быть определена как дружественная классу с полным списком аргументов.

Перегрузка операций

тип operator @ (список_параметров-операндов)
{
... тело функции ...
}

где **@** — знак перегружаемой операции (-, +, * и т. д.),
тип — тип возвращаемого значения.

Тип возвращаемого значения должен быть отличным от void, если необходимо использовать перегруженную операцию внутри другого выражения.

Перегружать можно большинство операций

new delete

+ - * / % ^ & | ~ ! = < > += -= *= /= %=

^= &= |= << >> >>= <<= == != <= >= && || ++ --

-> () []

Операции, не допускающие перегрузки:

- **. прямой выбор члена объекта класса;**
- **.* обращение к члену через указатель на него;**
- **?: условная операция;**
- **:: операция указания области видимости;**
- **sizeof операция вычисления размера в байтах;**
- **# препроцессорная операция.**
- **##**

Правила перегрузки операций

- Язык C++ не допускает определения для операций нового лексического символа, кроме уже определенных в языке. Например, нельзя определить в качестве знака операции @.
- Не допускается перегрузка операций для встроенных типов данных. Нельзя, например, переопределить операцию сложения целых чисел: **int operator +(int i, int j);**
- Нельзя переопределить приоритет операции.
- Нельзя изменить синтаксис операции в выражении. Например, если некоторая операция определена как унарная, то ее нельзя определить как бинарную. Если для операции используется префиксная форма записи, то ее нельзя переопределить в постфиксную. Например, !a нельзя переопределить как a!
- Перегружать можно только операции, для которых хотя бы один аргумент представляет тип данных, определенный пользователем. Функция-операция должна быть определена либо как функция-член класса, либо как внешняя функция, но дружественная классу.

Пример

Функция- член класса	Дружественная функция
class String { ... public: String operator + (String &); ... };	class String { ... public: friend String operator +(String &, String &); ... };

Пример1

```
class Complex // нам знакомый класс
{int real;      //приватные поля
 int image;
 public: // общедоступные методы
 void show(); //печать числа
 Complex();   //конструктор пустой, сами конструкторы тут не расписаны
 Complex(int re, int im);   //конструктор для инициализации
 Complex operator+ (Complex x); // операция сложения
 };
Complex Complex::operator+(Complex x)   //перегружаем +
 {  Complex result;
    result.real=real + x.real;
    result.image=image+x.image;
    return result;
 }
```

Пример1

```
int main()
{
   Complex a(3,6),b(6,7); //инициализация объектов
   a.show();
   b.show();
   Complex c=a+b;  //применяем новое сложение
   c.show();
   return 0;
}
```

Пример2

```
class Complex
{int real;
 int image;
 public:
 void show();
 Complex();
 Complex(int re, int im);
 friend Complex operator+ (Complex& ob, Complex& ob1)   // описываем дружественную функцию
 {Complex result;
  result.real=ob.real + ob1.real;  //учим складывать по-новому
  result.image=ob.image+ob1.image;
  return result;}
     };
```

Пример2

```
int main()
{
   Complex a(3,6),b(6,7);
   a.show();
   b.show();
   Complex c;
   c=a+b;
   c.show();
   return 0;
}
```

Перегрузка унарной операции

- Если унарная операция перегружается как функция-член, то она не должна иметь аргументов, так как в этом случае ей передается неявный аргумент-указатель на текущий объект.
- Если унарная операция перегружается дружественной функцией, то она должна иметь один аргумент – объект, для которого она выполняется.

Перегрузка унарной операции

Функция- член класса	**Дружественная функция**

class A
{...
public:
  A operator !();
 ...
};
	class A
	{...
	public:
	  friend A operator !(A);
	 ...
	};