Bahasa Pemrograman

Kuliah 11 :: Operator Overloading, Friend

Julio Adisantoso

Operator Overloading

- Programmer dapat menggunakan beberapa simbol operator untuk mendefinisikan fungsi anggota tertentu dari suatu class.
- Memungkinkan sebuah obyek lebih lengkap menyediakan fasilitas operator dari suatu ekspresi, misalkan digunakan notasi +, ++, =, dsb.

Mengapa perlu Operator Overloading?

```
int i, j, k;
float m, n, p;

k = i + j;
p = m + n;

Complex a,b,c;
c = a + b;
```

Compiler meng-overload operator "+" untuk melakukan penjumlahan bilangan bulat (int) dan riil (float), menghasilkan bilangan bulat dari ekspresi i+j, dan bilangan riil dari ekspresi m+n.

Persoalannya, bagaimana kalau penjumlahan bilangan komplek yang diimplementasikan dengan class Complex? Bagaimana bentuk fungsinya agar ekspresi c=a+b dapat dilakukan?

Sintaks Operator Overloading

operator@*(argument-list)*

Contoh:
operator+
operator-
operator*
operator/

- **operator** adalah sebuah fungsi
- @ adalah salah satu dari simbol operator C++, misalnya +, -, =, dsb

Contoh : class myString

```
class myString {
   char *st;
   int len;
   public:
      void set(char *s);
      void set(myString s);
      void cat(char *s);
      char* getString() {
        return st; }
      int getLength() {
        return len; }
      // ...
      myString operator+(myString s);
      // ...
};
```

```
int main() {
  myString s1,s2;
  myString s3,s4,s5;
  s1.set("bogor");
  s2.set("ipb");

  s3=s1+s2;

  return 0;
}
```

```
s3=s1.operator+(s2);
```


Implementasi fungsi anggota

```
void myString::set(char *s) {
   int n=strlen(s);
   if (n==0) return;
   st=new char[n+1];
   strcpy(st,s);
   len=n; }

void myString::set(myString s) {
   int n=s.getLength();
   if (n==0) return;
   st=new char[n+1];
   strcpy(st,s.getString());
   len=n; }
```

Implementasi fungsi anggota

```
void myString::cat(char *s) {
   int n=strlen(s);
   if (n==0) return;
   char *pTemp = new char[n+len+1];
   strcpy(pTemp,st);
   strcat(pTemp,s);
   st=pTemp;
   len+=n; }

myString myString::operator+(myString s) {
   myString baru;
   baru.set(st);
   baru.cat(s.getString());
   return baru; }
```

Slide 6

Implementasi operator overloading

- Sebagai fungsi anggota (seperti contoh dalam myString)

`s3=s1+s2;` ⇨ **`s3=s1.operator+(s2);`**

- Bukan sebagai fungsi anggota (di luar class myString)

`s3=s1+s2;` ⇨ **`s3=operator+(s1,s2);`**

bentuk fungsi menjadi:

```
myString operator+(myString s1, myString s2);
```

- Sebagai friend function (akan dijelaskan berikutnya)

Slide 7

Latihan

Buatlah implementasi fungsi agar dapat melakukan beberapa ekspresi berikut (s1, s2, .. adalah objek dari class myString) :

1. s4=s1+"hujan";
2. s5=s1*3; // string pada s1 diduplikasi 3x
3. if (s1==s2) cout << "sama";

Slide 8

FRIEND

Slide 9

Friend

- Deklarasi friend menjadikan dua class sebagai pasangan. Jadi, jika suatu obyek dideklarasikan sebagai friend, maka obyek tersebut dapat mengakses anggota class yang bukan public.
- Akses bersifat searah, artinya jika B sebagai friend A maka B dapat mengakses anggota A tetapi A tidak dapat mengakses B.
- Fungsi friend dari suatu class didefinisikan di luar lingkup class.
- Manfaat: menyediakan akses yang lebih efisien bagi anggota data yang dipanggil oleh suatu fungsi, dan mengakomodasi fungsi operator agar dapat mengakses anggota data yang bersifat private.

Slide 10

Contoh : stream

- cin>> dan cout<< adalah *operator overloading* dalam pustaka baku C++ dari iostream.h
- cin dan cout adalah objek dari class istream dan ostream.
- Kita dapat menambahkan fungsi friend yang meng-overload operator << dan juga >>.
- Contoh untuk <<:

```
friend ostream& operator<< (ostream &os, const myString &s);
```

Slide 11

class myString (1)

```
class myString {
   friend ostream& operator<<
      (ostream &os, const myString &s) {
          os<<s.st<<"("<<s.len<<")"<<endl;
          return os; }

   char *st;
   int len;
   public:
      void set(char *s);
      void set(myString s);
      void cat(char *s);
      char* getString() { return st; }
      int getLength() { return len; }

      myString operator+(myString s);
};
```

```
...
s3=s1+s2;
cout << s3;
...
```

Slide 12

class myString (2)

```
class myString {
   friend ostream& operator<<
      (ostream &os, const myString &s) {
          os<<s.st<<"("<<s.len<<")"<<endl;
          return os; }
   friend istream& operator>>
      (istream &in, myString &s) {
          char t[100];  // maksimum 100 kar
          in>>t;
          s.set(t); return in; }

   ...
   ...
};
```

```
cin >> s1;
cin >> s2;
...
s3=s1+s2;
cout << s3;
...
```

Slide 13