TIF1202 – Pemrograman Berorientasi Objek

HO 06 - Operator Berlebihan Beban

Opim Salim Sitompul

Department of Information Technology
Universitas Sumatera Utara

Outline

Tujuan

- Setelah menyelesaikan modul ini mahasiswa diharapkan:
  - Memahami konsep operator berlebihan beban (*operator overloading*)
  - Memahami bagaimana C++ menentukan operasi mana yang diinginkan
  - Mengimplementasi operator berlebihan beban unary dan binary.
  - Membangun program dengan konsep operator berlebihan beban.

Pendahuluan

- Ketika definisikan kelas, kita dapat membuat jenis data (*data type*) sendiri.
  - Jenis data menentukan sebuah himpunan nilai yang dapat disimpan oleh data beserta sebuah himpunan operasi yang dapat dilakukan program pada data tersebut.
  - Contohnya pada jenis data dasar:
  - Untuk variabel berjenis **int** dengan ukaran 4 bytes, range bilangan yang dapat disimpan adalah -2,147,483,648 hingga 2,147,483,647.
  - Program dapat melakukan operasi-operasi penjumlahan, pengurangan, perkalian, pembagian, dan beberapa operasi bitwise pada variabel berjenis **int** tersebut.

Pendahuluan

- Ketika mendefinisikan kelas, hal yang sama sebenarnya juga berlaku.
- Selain mendefinisikan anggota variabel, dalam sebuah kelas juga didefinisikan anggota-anggota fungsi yang akan dioperasikan terhadap anggota variabel.
- Adakalanya sangat menyenangkan apabila operasi-operasi yang dilakukan oleh fungsi tersebut dapat dinyatakan dalam bentuk operator, sehingga operasi yang dilakukan terasa lebih alamiah seperti halnya menggunakan operator pada jenis-jenis data dasar.

Pendahuluan

- Contoh:
  - Kelas Tanggal mempunyai anggota variabel hari, bulan dan tahun.
  - Anggota fungsi untuk menambahkan 30 hari ke tanggal atau mengurangkan 15 hari dari tanggal, tentu akan lebih mudah dipahami apabila dinyatakan dengan operator + atau -, misanya:
  - tgl_tagihan = tgl_pesan + 30;
  - pemberitahuan_1 = tgl_tagihan – 15;

Pendahuluan

- Agar operator + atau - dapat digunakan, program harus mendefinisikan operasi yang harus dilakukan apabila menemukan operator-operator ini pada kelas Tanggal.
- Dengan demikian, program akan melaksanakan sekumpulan operasi (mis. +, -, dll) baik ketika menemukan operator yang digunakan pada jenis data int, float, dll, maupun pada anggota variabel kelas Tanggal.

Pendahuluan

- Proses penugasan dua atau lebih operasi ke operator yang sama untuk jenis-jenis data yang berbeda inilah yang disebut operator overloading.
- Tergantung pada bagaimana operator digunakan, kompiler akan menentukan operasi mana yang akan dilakukan.
- Untuk melakukan operator berlebihan beban digunakan kata kunci operator.

Unary Operator Tanpa Argumen

- Penggunaan kata kunci operator untuk melebihkan beban unary operator ++
- Contoh Program 6.1:

Unary Operator Tanpa Argumen

```
1  //Contoh6_1.cpp
2  #include <iostream>
3  using namespace std;
4
5  class Hitung
6  {
7    private:
8      unsigned int cacah;
9    public:
10     Hitung() { cacah = 0; }
11     int get_Hitungan() { return cacah; }
12     void operator++ () { cacah += 1; }
13     ~Hitung() { }
14 };
```

Unary Operator Tanpa Argumen

```
15  int main()
16  {
17    Hitung c1, c2;
18
19    cout << "\nc1 = " << c1.get_Hitungan();
20    cout << "\nc2 = " << c2.get_Hitungan();
21    ++c1;
22    ++c2;
23    cout << "\nc1 = " << c1.get_Hitungan();
24    cout << "\nc2 = " << c2.get_Hitungan();
25    cout << endl;
26
27    return 0;
28  }
```

Unary Operator Dengan Argumen

- Perhatikan bagaimana operator berlebihan beban ++ menerima sebuah argument berjenis class Hitung dan mengembalikan nilai dari kelas yang sama.
- Contoh program 6.2 masih melakukan overloading pada operator pre-increment, tetapi selain itu dapat pula memberikan nilai inkremennya ke variabel lain.
- Contoh Program 6.2:

Unary Operator Dengan Argumen

```
1  //Contoh6_2.cpp
2  #include <iostream>
3
4  using namespace std;
5
6  class Hitung
7  {
8    private:
9      unsigned int cacah;
10   public:
11     Hitung() { cacah = 0; }
12     int get_Hitungan() { return cacah; }
```

Unary Operator Dengan Argumen

```
13        Hitung operator ++ ()
14        {
15          cacah++;
16          Hitung temp;
17          temp.cacah = this->cacah;
18
19          return temp;
20        }
21        ~Hitung() { }
22    };
```

Unary Operator Dengan Argumen

```
23  int main()
24  {
25    Hitung c1, c2;
26    cout << "\nc1 = " << c1.get_Hitungan();
27    cout << "\nc2 = " << c2.get_Hitungan();
28    ++c1;
29    c2 = ++c1;
30    cout << "\nc1 = " << c1.get_Hitungan();
31    cout << "\nc2 = " << (++c2).get_Hitungan();
32    cout << endl;
33    return 0;
34  }
```

Objek Sementara Tanpa Nama

- Pada contoh program 6.2 terdapat sebuah object bernama temp berjenis Hitung yang bersifat sementara dan bertujuan untuk menampung nilai yang dikembalikan oleh operator ++.
- Pada contoh program berikut, objek sementara itu dapat dihilangkan sehingga dapat menghemat kode program.
- Contoh Program 6.3:

Objek Sementara Tanpa Nama

```
1  //Contoh6_3.cpp
2  #include <iostream>
3  using namespace std;
4  class Hitung
5  {
6    protected:
7        unsigned int cacah;
8    public:
9      Hitung() { cacah = 0; }
10     Hitung(int c) { cacah = c; }
11     int get_Hitungan() { return cacah; }
12     Hitung operator ++ ()
13     {
14       cacah++;
15       return Hitung(cacah);
16     }
```

Objek Sementara Tanpa Nama

```
17    ~Hitung() { }
18  };
19
20  int main()
21  {
22    Hitung c1, c2;
23
24    cout << "\nc1 = " << c1.get_Hitungan();
25    cout << "\nc2 = " << c2.get_Hitungan();
26    ++c1;
27    c2 = ++c1;
28    cout << "\nc1 = " << c1.get_Hitungan();
29    cout << "\nc2 = " << (++c2).get_Hitungan();
30    cout << endl;
31    return 0;
32  }
```

Postfix dan Prefix Unary Increment Operator

- Untuk melengkapi definisi overloading operator prefix dan postfix increment, pada program berikut diperlihatkan bagaimana mendefinisikan operator postfix dan prefix inkremen tersebut.
- Contoh Program 6.4:

Postfix dan Prefix Unary Increment Operator

```
1  //Contoh6_4.cpp
2  #include <iostream>
3  using namespace std;
4  class Hitung
5  {
6    private:
7      unsigned int cacah;
8    public:
9      Hitung() {cacah = 0;};
10     int get_Hitungan() {return cacah;}
11     void operator ++(){cacah++;}//postfix ++
12     void operator ++(int){++cacah;}//prefix ++
13     ~Hitung() { }
14 };
```

Postfix dan Prefix Unary Increment Operator

```
1  int main()
2  {
3    Hitung c1, c2;
4
5    cout << "\nc1 = " << c1.get_Hitungan();
6    cout << "\nc2 = " << c2.get_Hitungan();
7    c1++;
8    c2++;
9    ++c2;
10   cout << "\nc1 = " << c1.get_Hitungan();
11   cout << "\nc2 = " << c2.get_Hitungan();
12   cout << endl;
13
14   return 0;
15 }
```

Operator Aritmatika

- Berikut ini diperlihatkan bagaimana binary operator + dilebihkan beban untuk menjumlahkan class berjenis Jarak.
- Contoh Program 6.5:

Overloading Binary Operator

```
1  //Contoh6_5.cpp
2  #include <iostream>
3
4  using namespace std;
5
6  class Jarak
7  {
8    private:
9      int kaki;
10     float inches;
11   public:
12     Jarak() { kaki = 0;  inches = 0.0; };
13     Jarak(int ft, float in)
14                { kaki = ft;  inches = in; };
15     void get_Jarak();
```

Overloading Binary Operator

```
16      void show_Jarak()
17      {
18          cout << kaki << "\'-"
19          << inches << '\"';
20          }
21      Jarak operator + (Jarak);
22      ~Jarak() { }
23  };
24
25  void Jarak::get_Jarak()
26  {
27      cout << "\nBerikan kaki dan inchi: ";
28      cin >> kaki >> inches;
29  }
```

Overloading Binary Operator

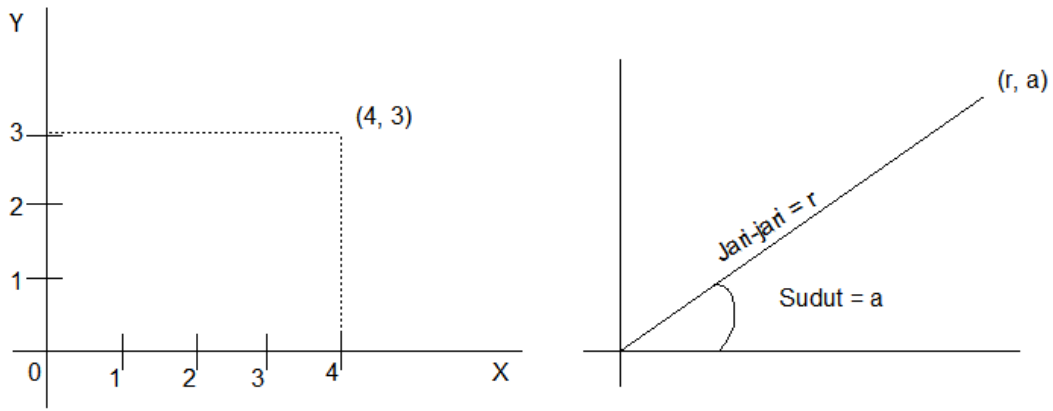
```
30  Jarak Jarak::operator + (Jarak j)
31  {
32    int f = kaki + j.kaki;
33    float i = inches + j.inches;
34    if (i >= 12.0)
35    {
36      i -= 12.0;
37      f++;
38    }
39    return Jarak(f, i);
40  }
```

Overloading Binary Operator

```
41  int main()
42  {
43    Jarak j1, j3, j4;
44
45    j1.get_Jarak();
46    Jarak j2(11, 6.25);
47    j3 = j1 + j2;
48    j4 = j1 + j2 + j3;
49    cout << "\nJarak 1 = "; j1.show_Jarak();
50    cout << "\nJarak 2 = "; j2.show_Jarak();
51    cout << "\nJarak 3 = "; j3.show_Jarak();
52    cout << "\nJarak 4 = "; j4.show_Jarak();
53    cout << endl;
54
55    return 0;
56  }
```

Menambahkan Koordinat Polar

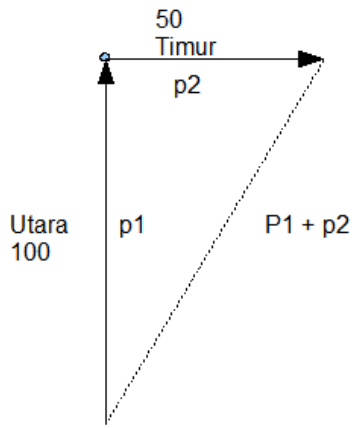
- Berikut adalah representasi koordinat rectangular dan koordinat polar.



Gambar 1: Koordinat rectangular dan koordinat polar

Menambahkan Koordinat Polar

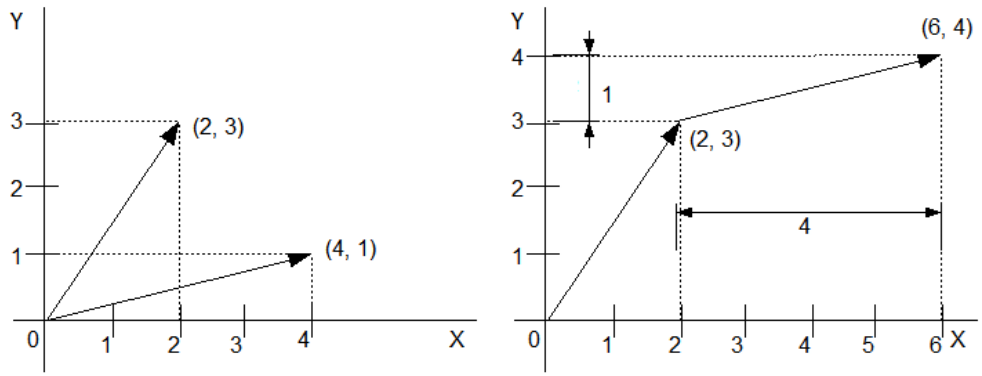
- Untuk menjumlahkan dua buah koordinat polar, perhatikan gambar berikut:



Gambar 2: Menjumlahkan dua buah koordinat polar

Menambahkan Koordinat Polar

- Untuk menjumlahkan dua buah koordinat polar, perhatikan gambar berikut:



Gambar 3: Konversi koordinat polar ke koordinat rectangular

Menambahkan Koordinat Polar

- Dengan menggunakan rumus konversi berikut:
- Rectangular ke Polar:

$$X = r\cos(a)$$
$$Y = r\sin(a)$$

- Polar ke Rectangular:

$$a = atan(x/y)$$
$$r = \sqrt{x^2 + y^2}$$

- Contoh program berikut memperlihatkan bagaimana proses penjumlahan yang rumit tersebut dapat disederhanakan penggunaannya melalui operator overloading.
- Contoh Program 6.6:

Menambahkan Koordinat Polar

```
1  //Contoh6_6.cpp
2  #include <iostream>
3  #include <cmath>
4  using namespace std;
5
6  class Polar
7  {
8    private:
9      double jari2;
10     double sudut;
11     double getx() { return jari2*cos(sudut); }
12     double gety() { return jari2*sin(sudut); }
13   public:
14     Polar() { jari2 = 0.0;  sudut = 0.0; };
15     Polar(double r, double a)
16       { jari2 = r;  sudut = a; };
```

Menambahkan Koordinat Polar

```
17    void display(){cout << "(" << jari2 << ",␣"
          << sudut << ")";}
18      Polar operator + (Polar);
19      ~Polar() { }
20  };
21
22  Polar Polar::operator + (Polar p2)
23  {
24    double x = getx() + p2.getx();
25    double y = gety() + p2.gety();
26    double r = sqrt(x*x + y*y);
27    double a = atan(x/y);
28
29    return Polar(r, a);
30  }
```

Menambahkan Koordinat Polar

```
31  int main()
32  {
33    Polar p1(10.0, 0.0), p2(10.0, 1.570796325),
          p3;
34
35    p3 = p1 + p2;
36    cout << "\np1 = "; p1.display();
37    cout << "\np2 = "; p2.display();
38    cout << "\np3 = "; p3.display();
39    cout << endl;
40
41    return 0;
42  }
```

Menyambung String

- C/C++ tidak membenarkan kita untuk menyambung dua buah string menggunakan operator +. Untuk maksud ini, C/C++ menyediakan fungsi-fungsi untuk melakukan operasi pada string seperti strcpy() untuk menyalin string, strrcat() untuk menyambung (concatenate) dua buah string, dan strcmp() untuk membandingkan string.
- Untuk memudahkan operasi penyambungan string, berikut ini diperlihatkan bagaimana operaor + dapat dilebihkan beban untuk melaksanakan tugas tersebut.
- Contoh Program 6.7:

Menyambung String

```
1  //Contoh6_7.cpp
2  #include <iostream>
3  #include <cstring>
4  using namespace std;
5  const int SZ = 80;
6  class String
7  {
8    private:
9      char str[SZ];
10   public:
11     String() { strcpy(str, ""); }
12     String(char s[]) { strcpy(str, s); }
13     void display() { cout << str; }
14     String operator + (String);
15     ~String() { }
16 };
```

Menyambung String

```
17  String String::operator + (String ss)
18  {
19    String temp;
20
21    if (strlen(str) + strlen(ss.str) < SZ)
22    {
23      strcpy(temp.str, str);
24      strcat(temp.str, ss.str);
25    }
26    else
27      cout <<"\nString overflow";
28
29    return temp;
30  }
```

Menyambung String

```
31  int main()
32  {
33    String s1 = "\nFasilkom-TI␣USU␣";
34    String s2 = "Medan,␣Sumatera␣Utara";
35    String s3;
36
37    s3 = s1 + s2;
38    s3.display();
39    cout << endl;
40
41    return 0;
42  }
```

Operator Perbandingan

- Selain operator aritmatika, operator lain seperti operator relasional juga dapat dilebihkan beban.
- Pada contoh berikut diperlihatkan bagaimana operator < dapat dilebihkan beban untuk membandingkan dua buah objek berjenis Jarak.
- Contoh Program 6.8:

Operator Perbandingan

```
1  //Contoh6_8.cpp
2  #include <iostream>
3  using namespace std;
4  enum boolean {salah, benar};
5  class Jarak
6  {
7    private:
8      int kaki;
9      float inches;
10   public:
11     Jarak() { kaki = 0;  inches = 0.0; };
12     Jarak(int ft,float in){kaki=ft;inches=in;}
13     void get_Jarak();
```

Operator Perbandingan

```
14      void show_Jarak(){cout << kaki
15        << "\'-" << inches << '\"';}
16      boolean operator < (Jarak);
17      ~Jarak() { }
18  };
19
20  void Jarak::get_Jarak()
21  {
22    cout << "\nBerikan kaki: ";
23    cin >> kaki;
24    cout << "\nBerikan inches: ";
25    cin >> inches;
26  }
```

Operator Perbandingan

```
27  boolean Jarak::operator < (Jarak j)
28  {
29     float f1 = kaki + inches/12.0F;
30     float f2 = j.kaki + j.inches/12.0F;
31
32     return (f1 < f2) ? benar : salah;
33  }
```

Operator Perbandingan

```
34  int main()
35  {
36    Jarak j1;
37    j1.get_Jarak();
38
39    Jarak j2(6, 2.5F);
40    cout << "\nJarak 1 = "; j1.show_Jarak();
41    cout << "\nJarak 2 = "; j2.show_Jarak();
42    if (j1 < j2)
43      cout << "\nJarak 1 < Jarak 2";
44    else
45      cout << "\nJarak 1 > Jarak 2";
46    cout << endl;
47
48    return 0;
49  }
```

Operator Perbandingan

- Perhatikan pula bagaimana operator == digunakan untuk membandingkan dua buah string.
- Contoh Program 6.9:

Operator Perbandingan

```
1  //Contoh6_9.cpp
2  #include <iostream>
3  #include <cstring>
4  using namespace std;
5  const int SZ = 80;
6  enum boolean {salah, benar};
7
8  class String
9  {
10   private:
11     char str[SZ];
12   public:
13     String() { strcpy(str, ""); }
14     String(char s[]) { strcpy(str, s); }
15     void getstr() { cin.get(str, SZ); }
```

Operator Perbandingan

```
16      void display() { cout << str; }
17      boolean operator == (String);
18      ~String() { }
19  };
20
21  boolean String::operator == (String ss)
22  {
23    return (strcmp(str, ss.str) == 0) ? benar :
          salah;
24  }
25
26  int main()
27  {
28    String s1 = "ya";
29    String s2 = "tidak";
30    String s3;
```

Operator Perbandingan

```
31      cout << "\nJawab dengan 'ya' atau 'tidak: ";
32      s3.getstr();
33      if (s3 == s1)
34        cout << "Jawaban Anda ya\n";
35      else if (s3 == s2)
36        cout << "Jawaban Anda tidak\n";
37      else
38        cout << "Anda tidak mematuhi instruksi"
39          << " yang diberikan\n";
40
41      return 0;
42  }
```

Operator yang tidak boleh di-overload

- Ada beberapa operator tertentu yang tidak dapat di-overload. Operator-operator tersebut adalah:
  - Operator dot (.) untuk mengakses anggota
  - Operator sizeof
  - Scope resolution operator (::)
  - Operator ternary aritmatika if (?:)
  - Operator (.*) mengakses anggota melalui pointer ke fungsi
- Dua operator, yaitu = dan & secara default telah di-overload dalam C++. Jadi, untuk menyalin objek yang berasal dari kelas yang sama, dapat langsung digunakan operator =.