TIF1202 – Pemrograman Berorientasi Objek

HO 12 - Fungsi Virtual dan Polymorphism

Opim Salim Sitompul

Department of Information Technology
Universitas Sumatera Utara

Outline

Tujuan

- Setelah menyelesaikan modul ini mahasiswa diharapkan:
  - Memahami konsep fungsi virtual dan polimorfisme
  - Dapat membedakan antara fungsi virtual dan fungsi berlebihan beban
  - Dapat menerapkan aturan-aturan polimorfisme dalam bentuk program

Konsep Dasar

- Objek polimorfisme adalah objek yang mampu memiliki banyak bentuk.
- C++ mengimplementasikan polimorfisme menggunakan fungsi virtual.
- Fungsi virtual dirancang untuk bekerja secara virtual pada anggota kelas dasar dan turunan yang jenisnya belum diketahui.
- Ringkasnya, sebuah fungsi yang dapat mewakili beberapa fungsi atau dapat juga dipandang sebagai “variabel fungsi”.

Deklarasi Fungsi Virtual

- Fungsi virtual dideklarasikan dengan sebuah kata kunci virtual.
- Perhatikan contoh kelas berkut:

```
1 class telepon
2 {
3   protected:
4     char no_telp[32];
5     int volume;
6   public:
7     telepon(char *nomor, int volume);
8     virtual int dial(char *no_keluar);
9 };
```

Deklarasi Fungsi Virtual

- Fungsi dial pada kelas telepon di atas adalah sebuah fungsi virtual.
- Berikut adalah kelas-kelas turunan, *rotary* dan *touch_tone* yang didasarkan pada kelas *telepon*.

```
1  class rotary : public telepon
2  {
3    public:
4      rotary (char *nomor, int volume) :
5      telepon(nomor, volume) { };
6      int dial (char *nomor_keluar);
7  };
```

Deklarasi Fungsi Virtual

```
 8  class touch_tone : public telepon
 9    {
10      public:
11      touch_tone (char *nomor, int volume) :
12        telepon(nomor, volume) { };
13      int dial (char *nomor_keluar);
14    };
```

- Class rotary dan touch_tone adalah dua buah kelas turunan dari kelas dasar telepon.
- Masing-masing kelas turunan ini memiliki fungsi dialsendiri.
- Contoh Program 12.1:

Deklarasi Fungsi Virtual

```
1  //Contoh12_1.cpp
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  class telepon
8  {
9    protected:
10     char no_telp[32];
11     int volume;
12   public:
13     telepon(char *nomor, int volume);
14     virtual int dial(char *no_keluar);
15 };
```

Deklarasi Fungsi Virtual

```
16 telepon::telepon(char *nomor, int volume)
17 {
18   strcpy(telepon::no_telp, nomor);
19   telepon::volume = volume;
20 }
21
22 int telepon::dial(char *no_keluar)
23 {
24   cout << "Menggunakan telp rotary atau touch-
       tone ";
25     << "untuk memanggil: " << no_keluar;
26   cout << " Volume: " << volume << endl;
27
28   return 0;
29 }
```

Deklarasi Fungsi Virtual

```
30  class touch_tone : public telepon
31  {
32    public:
33      touch_tone (char *nomor, int volume)
34                  : telepon(nomor, volume) { };
35      int dial (char *no_keluar);
36  };
37
38  int touch_tone::dial(char *no_keluar)
39  {
40    cout << "Beep beep beep suara touch-tone: "
         << no_keluar;
41    cout << " Volume: " << volume << endl;
42
43    return 0;
44  }
```

Deklarasi Fungsi Virtual

```
1  class rotary : public telepon
2  {
3    public:
4      rotary (char *nomor, int volume) :
5                telepon(nomor, volume) { };
6      int dial (char *nomor_keluar);
7  };
8
9  int rotary::dial(char *no_keluar)
10 {
11   cout << "Klik klik klik suara rotary: " <<
         no_keluar;
12   cout << " Volume: " << volume << endl;
13
14   return 0;
15 }
```

Deklarasi Fungsi Virtual

```
16  int main()
17  {
18    telepon mytelp("821-0123", 6);
19    touch_tone kantor("821-3793", 5);
20    rotary rumah("822-3576", 2);
21
22    mytelp.dial("111-2222");
23    rumah.dial("222-3333");
24    kantor.dial("333-4444");
25
26    return 0;
27  }
```

Polymorphism

- Contoh program 12.1 menggunakan fungsi virtual tetapi tidak menggunakan polimorfisme.
- Contoh berikut memperlihatkan bagaimana polimorfisme berlaku.
- Contoh program 12.2:
- Objek telepon pada fungsi main menggunakan pointer ke sebuah objek bernama seluler.
- Pada waktu eksekusi program, pointer ini menunjuk ke objek *rotary* dan touch_tone.

Polymorphism

```
1  //Contoh12_2.cpp
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  class telepon
8  {
9    protected:
10     char no_telp[32];
11     int volume;
12   public:
13     telepon(char *nomor, int volume);
14     virtual int dial(char *no_keluar);
15 };
```

Polymorphism

```
16  telepon::telepon(char *nomor, int volume)
17  {
18    strcpy(telepon::no_telp, nomor);
19    telepon::volume = volume;
20  }
21
22  int telepon::dial(char *no_keluar)
23  {
24    cout << "Menggunakan telepon rotary atau
          touch-tone";
25       << " untuk memanggil: " << no_keluar;
26    cout << " Volume: " << volume << endl;
27
28    return 0;
29  }
```

Polymorphism

```
30  class touch_tone : public telepon
31  {
32    public:
33      touch_tone (char *nomor, int volume) :
            telepon(nomor, volume) { };
34      int dial (char *no_keluar);
35  };
36
37  int touch_tone::dial(char *no_keluar)
38  {
39    cout << "Beep beep beep suara touch-tone: "
          << no_keluar;
40    cout << " Volume: " << volume << endl;
41
42    return 0;
43  }
```

Polymorphism

```
44  class rotary : public telepon
45  {
46    public:
47      rotary (char *nomor, int volume) : telepon
            (nomor, volume) { };
48      int dial (char *nomor_keluar);
49  };
50
51  int rotary::dial(char *no_keluar)
52  {
53    cout << "Klick klik klik suara rotary: " <<
          no_keluar;
54    cout << " Volume: " << volume << endl;
55
56    return 0;
57  }
```

Polymorphism

```
1  int main()
2  {
3    telepon my_telp("821-4000", 3);
4    touch_tone kantor("821-3793", 5);
5    rotary rumah("822-3576", 2);
6
7    telepon *seluler;
8
9    seluler = &my_telp;
10   seluler->dial("111-2222");
11   seluler = &rumah;
12   seluler->dial("222-3333");
13   seluler = &kantor;
14   seluler->dial("333-4444");
15   return 0;
16 }
```

Non Polymorphism

- Bandingkan program pada Contoh Program12.3 berikut yang bukan merupakan polimorfisme.
- Objek seluler yang digunakan tidak pernah berubah bentuk dan tetap berupa objek telepon.
- Contoh Program 12.3:

Non Polymorphism

```
1  //Contoh12_3.cpp
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  class telepon
8  {
9    protected:
10     char no_telp[32];
11     int volume;
12   public:
13     telepon(char *nomor, int volume);
14     virtual int dial(char *no_keluar);
15 };
```

Non Polymorphism

```
16 telepon::telepon(char *nomor, int volume)
17 {
18   strcpy(telepon::no_telp, nomor);
19   telepon::volume = volume;
20 }
21
22 int telepon::dial(char *no_keluar)
23 {
24   cout << "Menggunakan telepon rotary atau
       touch-tone ";
25     << " untuk memanggil: " << no_keluar;
26
27   cout << " Volume: " << volume << endl;
28
29   return 0;
30 }
```

Non Polymorphism

```
31  {
32    public:
33      touch_tone (char *nomor, int volume)
34            : telepon(nomor, volume) { };
35      int dial (char *no_keluar);
36  };
37
38  int touch_tone::dial(char *no_keluar)
39  {
40    cout << "Beep␣beep␣beep␣suara␣touch-tone:␣"
         << no_keluar;
41    cout << "␣Volume:␣" << volume << endl;
42
43    return 0;
44  }
```

Non Polymorphism

```
46  class rotary : public telepon
47  {
48    public:
49      rotary (char *nomor, int volume)
50                  : telepon(nomor, volume) { };
51      int dial (char *nomor_keluar);
52  };
53
54  int rotary::dial(char *no_keluar)
55  {
56    cout << "Klick klik klik suara rotary: " <<
          no_keluar;
57    cout << " Volume: " << volume << endl;
58
59    return 0;
60  }
```

Non Polymorphism

```
61  int main()
62  {
63    touch_tone kantor("821-3793", 5);
64    rotary rumah("822-3576", 2);
65
66    telepon seluler("555-5555", 3);
67
68    rumah.dial("222-3333");
69    kantor.dial("333-4444");
70    seluler = rumah;
71    seluler.dial("555-1234");
72    seluler = kantor;
73    seluler.dial("555-5678");
74
75    return 0;
76  }
```

Non Polymorphism vs Overloading

- Polimorfisme sering dikacaukan dengan overloading.
- Pada contoh program berikut, fungsi dial di-overload di dalam kelas *rotary* dan *touch_tone*.
- Hal ini karena definisi kelas dasar tidak menggunakan kata kunci virtual.
- Contoh Program 12.4:

Polymorphism vs Overloading

```
1  //Contoh12_4.cpp
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  class telepon
8  {
9    protected:
10     char no_telp[32];
11     int volume;
12   public:
13     telepon(char *nomor, int volume);
14     int dial(char *no_keluar);
15 };
```

Polymorphism vs Overloading

```
16 telepon::telepon(char *nomor, int volume)
17 {
18   strcpy(telepon::no_telp, nomor);
19   telepon::volume = volume;
20 }
21
22 int telepon::dial(char *no_keluar)
23 {
24   cout << "Menggunakan telepon rotary atau
       touch-tone ";
25       << " untuk memanggil: " << no_keluar;
26   cout << " Volume: " << volume << endl;
27
28   return 0;
29 }
```

Polymorphism vs Overloading

```
30  class touch_tone : public telepon
31  {
32    public:
33      touch_tone (char *nomor, int volume)
34                  : telepon(nomor, volume) { }
35      int dial (char *no_keluar);
36  };
37
38  int touch_tone::dial(char *no_keluar)
39  {
40    cout << "Beep beep beep suara touch-tone: "
         << no_keluar;
41    cout << " Volume: " << volume << endl;
42
43    return 0;
44  }
```

Polymorphism vs Overloading

```
45  class rotary : public telepon
46  {
47    public:
48      rotary (char *nomor, int volume)
49                    : telepon(nomor, volume) {}
50      int dial (char *nomor_keluar);
51  };
52
53  int rotary::dial(char *no_keluar)
54  {
55    cout << "Klick klik klik suara rotary: " <<
          no_keluar;
56    cout << " Volume: " << volume << endl;
57
58    return 0;
59  }
```

Polymorphism vs Overloading

```
60  int main()
61  {
62    touch_tone kantor("821-3793", 5);
63    rotary rumah("822-3576", 2);
64
65    telepon *seluler=new telepon("555-5555",3);
66    rumah.dial("222-3333");
67    kantor.dial("333-4444");
68
69    seluler->dial("555-1234");
70    seluler = &kantor;
71    seluler->dial("555-5678");
72    seluler = &rumah;
73    seluler->dial("555-5679");
74    return 0;
75  }
```

Menggunakan Anggota Fungsi Kelas Dasar

- Dalam kasus berikut ini kelas turunan tidak mengganti fungsi virtual, melainkan menggunakan fungsi kelas dasar.
- Perhatikan kelas mesin berikut dimana fungsi get_bbm didefinisikan berupa fungsi virtual.

```
1  class mesin
2  {
3    private:
4      char nama[64];
5      int jenis_bbm;
6    public:
7      mesin(char *nama, int jenis_bbm);
8      virtual void get_bbm(int liter);
9  };
```

- Contoh Program 12.5:

Menggunakan Anggota Fungsi Kelas Dasar

```
1  //Contoh12_5.cpp
2  #include <iostream>
3  #include <string.h>
4
5  class mesin
6  {
7    private:
8      char nama[64];
9      int jenis_bbm;
10   public:
11     mesin(char *nama, int jenis_bbm);
12     virtual void get_bbm(int liter);
13 };
```

Menggunakan Anggota Fungsi Kelas Dasar

```
14  mesin::mesin(char *nama, int jenis_bbm)
15  {
16    strcpy(mesin::nama, nama);
17    mesin::jenis_bbm = jenis_bbm;
18  }
19
20  void mesin::get_bbm(int liter)
21  {
22    cout << "Tolong isi: " << liter
23         << " premimum bersubsidi." << endl;
24  }
```

Menggunakan Anggota Fungsi Kelas Dasar

```
25  class mesin_jet : public mesin
26  {
27    public:
28      mesin_jet(char *nama, int jenis_bbm)
29                  : mesin(nama, jenis_bbm) { };
30      void get_bbm(int liter)
31      {
32        cout << "Tolong isi: " << liter
33                    << " bbm avtur jet JP4"  <<
                         endl;
34      }
35  };
```

Menggunakan Anggota Fungsi Kelas Dasar

```
36  class mesin_truk : public mesin
37  {
38    public:
39      mesin_truk(char *nama, int jenis_bbm)
40                  : mesin(nama, jenis_bbm) { };
41      void get_bbm(int liter)
42      {
43        cout << "Tolong isi: " << liter
44                    << " bbm solar non subsidi"
                          << endl;
45      }
46  };
```

Menggunakan Anggota Fungsi Kelas Dasar

```
47  int main()
48  {
49    mesin rolls("Rolls Royce", 1);
50    mesin_jet F100("Pesawat jet", 2);
51    mesin_truk dyna("Truk sampah", 3);
52    mesin harley("Harley Davidson", 1);
53
54    rolls.get_bbm(20);
55    F100.get_bbm(100);
56    dyna.get_bbm(40);
57    harley.get_bbm(5);
58
59    return 0;
60  }
```

Aturan Fungsi Polimorphism

- Fungsi anggota kelas dasar harus didahului kata kunci **virtual**.
- Jenis data pengembalian atau jenis data parameter fungsi kelas turunan yang namanya sesuai dengan fungsi virtual harus sama.
- Contoh Program 12.6:
- Fungsi *get_bbm()* pada kelas *mesin_jet* memiliki jenis parameter (**long**) yang berbeda dengan jenis data parameter pada fungsi kelas dasar (**int**).
- Akibatnya pada waktu program dieksekusi, tidak terjadi polimorfisme.

Aturan Fungsi Polimorphism

```
1  //Contoh12_6.cpp
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  class mesin
8  {
9    protected:
10     char nama[64];
11     int jenis_bbm;
12   public:
13     mesin(char *nama, int jenis_bbm);
14     virtual void get_bbm(int liter);
15 };
```

Aturan Fungsi Polimorphism

```
16  mesin::mesin(char *nama, int jenis_bbm)
17  {
18    strcpy(mesin::nama, nama);
19    mesin::jenis_bbm = jenis_bbm;
20  }
21
22  void mesin::get_bbm(int liter)
23  {
24    cout << "Tolong isi: " << liter
25         << " liter premimum bersubsidi "
26         << "untuk " << nama << endl;
27  }
```

Aturan Fungsi Polimorphism

```
28  class mesin_bmw : public mesin
29  {
30    public:
31      mesin_bmw(char *nama, int jenis_bbm)
32                    : mesin(nama, jenis_bbm) { };
33      void get_bbm(int liter)
34      {
35        cout << "Tolong isi: " << liter
36                        << " liter bbm non subsidi
                            octane 92 "
37                        << "untuk " << nama <<
                            endl;
38      }
39  };
```

Aturan Fungsi Polimorphism

```
40  class mesin_truk : public mesin
41  {
42    public:
43      mesin_truk(char *nama, int jenis_bbm)
44                    : mesin(nama, jenis_bbm) { };
45      void get_bbm(int liter)
46      {
47        cout << "Tolong isi: " << liter
48                       << " bbm solar non subsidi"
                              << " untuk "
49                       << nama << endl;
50      }
51  };
```

Aturan Fungsi Polimorphism

```
52  int main()
53  {
54    mesin rolls("Rolls Royce", 1);
55    mesin_bmw bmw320i("BMW Serie 320i", 2);
56    mesin_truk dyna("Truk sampah", 3);
57
58    mesin *mesinPtr;
59    mesinPtr = &rolls;
60    mesinPtr->get_bbm(20);
61    mesinPtr = &bmw320i;
62    mesinPtr->get_bbm(100);
63    mesinPtr = &dyna;
64    mesinPtr->get_bbm(50);
65
66    return 0;
67  }
```