TIF1202 – Pemrograman Berorientasi Objek

HO 07 - Fungsi Berlebihan Beban

Opim Salim Sitompul

Department of Information Technology
Universitas Sumatera Utara

Outline

Tujuan

- Setelah menyelesaikan modul ini mahasiswa diharapkan:
  - Memahami konsep fungsi berlebihan beban (function overloading)
  - Dapat menentukan apabila dua fungsi berbeda memiliki nama yang sama, bagaimana C++ menentukan fungsi mana yang dipanggil
  - Mengimplementasikan fungsi berlebihan beban untuk menambah keterbacaan program
  - Menerapkan penggunaan *default parameter*

Pendahuluan

- Fungsi berlebihan beban adalah proses mendefinisikan dua atau lebih fungsi, dengan nama yang sama, yang hanya berbeda pada parameter-parameternya, baik jumlah parameter atau jenis data parameter itu.
- Ketika program dikompilasi, kompiler akan menetukan fungsi mana yang dipanggil, berdasarkan bagaimana program itu menggunakan fungsi-fungsi tersebut.
- Contoh Program 7.1 dan 7.2 berikut:

Pendahuluan

```
1  //Contoh7_1.cpp
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  class Greetings
8  {
9    public:
10     void show_Greetings() { cout << "Selamat␣
           datang␣di␣Fasilkom-TI␣USU" << endl;}
11     void show_Greetings(string sapaan) { cout
           << sapaan << endl; }
12 };
```

Pendahuluan

```
13  int main()
14  {
15    Greetings myMsg;
16
17    myMsg.show_Greetings();
18    myMsg.show_Greetings("Semoga sukses!");
19
20    return 0;
21  }
```

Berlebihan beban pada jenis argumen

- Fungsi jlh_Array() memanfaatkan fungsi berlebihan beban pada array berjenis int dan array berjenis float.

Berlebihan beban pada jenis argumen

```
1  //Contoh7_2.cpp
2  #include <iostream>
3  #include <iomanip>
4  #include <cstdlib>
5  #include <ctime>
6  #define N 100
7  using namespace std;
8
9  class Array
10 {
11   private:
12     int array_integer[N];
13     float array_float[N];
14   public:
15     Array(int);
16     Array(double);
```

Berlebihan beban pada jenis argumen

```
17        long jlh_Array(int);
18        float jlh_Array(double);
19        void print_array(int);
20        void print_array(double);
21        ~Array() { }
22    };
23
24    Array::Array(int type)
25    {
26      srand(time(NULL));
27
28      for(int i=0; i<N; i++)
29        array_integer[i] = rand() % N;
30    }
```

Berlebihan beban pada jenis argumen

```
31  Array::Array(double type)
32  {
33    for(int i=0; i<N; i++)
34      array_float[i] = (float) (rand() % N);
35  }
36
37  long Array::jlh_Array(int type)
38  {
39    long jlh = 0L;
40    for(int i=0; i < N; i++)
41      jlh += array_integer[i];
42
43    return jlh;
44  }
```

Berlebihan beban pada jenis argumen

```
45  float Array::jlh_Array(double type)
46  {
47    float jlh = 0.0;
48
49    for(int i=0; i < N; i++)
50      jlh += array_float[i];
51    return jlh;
52  }
53
54  void Array::print_array(int type)
55  {
56    cout << "Array Integer: " << endl;
57    for(int i=0; i < N; i++)
58      cout << array_integer[i] << " ";
59    cout << endl;
60  }
```

Berlebihan beban pada jenis argumen

```
61  void Array::print_array(double type)
62  {
63    cout << "Array Float: " << endl;
64
65    for(int i=0; i < N; i++)
66      cout << setprecision(2)
67                << setiosflags(ios::fixed)<<
                     array_float[i]
68                << " ";
69    cout << endl;
70  }
71
72  int main()
73  {
74    Array myArr1(0), myArr2(0.0);
```

Berlebihan beban pada jenis argumen

```
75      myArr1.print_array(0);
76      cout << "Jumlah array integer = "
77           << myArr1.jlh_Array(0) << endl;
78      myArr2.print_array(0.0);
79      cout << "Jumlah array float = " <<
           setprecision(2)
80           << setiosflags(ios::fixed)
81           << myArr2.jlh_Array(0.0)<< endl;
82
83      return 0;
84  }
```

Berlebihan beban pada jumlah argumen

- Contoh 7.3 dan 7.4 berikut memperlihatkan bagaimana sebuah fungsi dapat memiliki 0 atau lebih parameter.
- Fungsi Tanggal() pada Contoh 7.3 berturut-turut menerima 0, 1, dan 3 buah argumen.
- Fungsi mygetLine() pada Contoh 7.4 menerima 0 dan 1 argumen.

Berlebihan beban pada jumlah argumen

```
1  //Contoh7_3.cpp
2  #include <iostream>
3  #include <string>
4  #include <cstring>
5  #include <ctime>
6
7  using namespace std;
8
9  class Tanggal
10 {
11   private:
12     int hari;
13     int bulan;
14     int tahun;
15   public:
16     Tanggal();
```

Berlebihan beban pada jumlah argumen

```
17    Tanggal(string tglstr) {cout << "String␣
          tanggal:␣" << tglstr << endl;}
18    Tanggal(int, int, int);
19    ~Tanggal() {}
20  };
21
22  Tanggal::Tanggal()
23  {
24    time_t hari_ini;
25
26    time(&hari_ini);
27    cout << "Tanggal␣hari␣ini:␣" << ctime(&
          hari_ini);
28  }
```

Berlebihan beban pada jumlah argumen

```
29  Tanggal::Tanggal(int hari, int bulan, int
        tahun)
30  {
31    char bln[15];
32
33    switch(bulan)
34    {
35      case 1: strcpy (bln, "␣Januari␣"); break;
36      case 2: strcpy (bln, "␣Pebruari␣"); break;
37      case 3: strcpy (bln, "␣Maret␣"); break;
38      case 4: strcpy (bln, "␣April␣"); break;
39      case 5: strcpy (bln, "␣Mei␣"); break;
40      case 6: strcpy (bln, "␣Juni␣"); break;
```

Berlebihan beban pada jumlah argumen

```
41        case 7: strcpy (bln, "␣Juli␣");break;
42        case 8: strcpy (bln, "␣Agustus␣");break;
43        case 9: strcpy (bln, "␣September␣");break;
44        case 10: strcpy (bln, "␣Oktober␣");break;
45        case 11: strcpy (bln, "␣Nopember␣");break;
46        case 12: strcpy (bln, "␣Desember␣");break;
47        default: strcpy (bln, "␣Invalid␣Date␣");
48      }
49      cout << hari << bln << tahun << endl;;
50  }
```

Berlebihan beban pada jumlah argumen

```
51  int main()
52  {
53    Tanggal t1, t2("15 April 2020"),
54      t3(5, 5, 2017);
55
56    return 0;
57  }
```

Berlebihan beban pada jumlah argumen

```
1  //Contoh7_4.cpp
2  #include <iostream>
3
4  using namespace std;
5
6  class Teks
7  {
8    private:
9      char teks[256];
10   public:
11     void mygetLine();
12     void mygetLine(char);
13     char *displayTeks() { return teks;}
14 };
```

Berlebihan beban pada jumlah argumen

```
15  void Teks::mygetLine()
16  {
17    int i;
18    char hrf;
19
20    for(i = 0; i < sizeof(teks); i++)
21      if ((hrf = cin.get()) == '\n')
22        break;
23      else
24        teks[i] = hrf;
25    teks[i] = '\0';
26  }
```

Berlebihan beban pada jumlah argumen

```
27  void Teks::mygetLine(char akhir_str)
28  {
29    int i;
30    char hrf;
31
32    for(i = 0; i < sizeof(teks); i++)
33      if ((hrf = cin.get()) == '\n')
34        break;
35      else if (hrf == akhir_str)
36        break;
37      else
38        teks[i] = hrf;
39    teks[i] = '\0';
40  }
```

Berlebihan beban pada jumlah argumen

```
41  int main()
42  {
43    Teks myTeks;
44
45    cout << "Tuliskan satu baris teks: ";
46    myTeks.mygetLine();
47    cout << "Anda menulis teks berikut: "
48         << myTeks.displayTeks() << endl;
49    cout << "Tuliskan satu baris teks akhiri
        dengan X: ";
50    myTeks.mygetLine('X');
51    cout << "Anda menulis teks berikut: "
52         << myTeks.displayTeks() << endl;
53
54    return 0;
55  }
```

Default Parameter

- Menggunakan default parameter dapat menghemat penulisan fungsi berlebihan beban karena definisi fungsi cukup diberikan sekali.
- Program memanggil fungsi tertentu dan apabila satu atau lebih nilai parameter tidak diberikan, fungsi akan menggunakan nilai default yang diberikan.
- Nilai default yang digunakan dimulai dari kiri ke kanan dari urutan parameter.
- Contoh Program 7.5:

Default Parameter

```
1  //Contoh7_5.cpp
2  #include <iostream>
3
4  using namespace std;
5
6  class Nilai
7  {
8    public:
9      void show_nilai(int a = 1, int b = 2, int
           c = 3)
10     {
11       cout << a << "␣" << b << "␣" << c <<
             endl;
12     }
13 };
```

Default Parameter

```
14  int main()
15  {
16    Nilai myNilai;
17
18    myNilai.show_nilai();
19    myNilai.show_nilai(2012);
20    myNilai.show_nilai(2012, 2013);
21    myNilai.show_nilai(2012, 2013, 2014);
22
23    return 0;
24  }
```

Default Parameter

- Fungsi show_Nilai() memiliki 3 buah argumen yang masing-masing diberi nilai default 1, 2, dan 3.
- Pada pemanggilan pertama tidak ada diberikan sebuahpun parameter, sehingga program akan menampilkan ketiga nilai default.
- Pada tiga pemanggilan berikutnya berturut-turut diberikan 1, 2 dan 3 parameter.

Default Parameter

- Program berikut memperlihatkan bagaimana default parameter dapat membantu penghematan penulisan kode program sehingga membantu meningkatkan keterbacaannya.
- Program ini menggabungkan 2 fungsi mygetLine() pada Contoh Program 7.5 dengan cara memberikan nilai default untuk parameter ketiga.
- Contoh Program 7.6:

Default Parameter

```
1  //Contoh7_6.cpp
2  #include <iostream>
3  #include <string>
4
5  using namespace std;
6
7  class Teks
8  {
9    private:
10     char teks[256];
11   public:
12     void Teks::mygetLine(char);
13     string displayTeks() { return teks; }
14 };
```

Default Parameter

```
15  void Teks::mygetLine(char akhir_str = '\n')
16  {
17    char hrf;
18    int i;
19
20    for(i = 0; i < sizeof(teks); i++)
21      if ((hrf = cin.get()) == akhir_str)
22        break;
23      else
24        teks[i] = hrf;
25    teks[i] = '\0';
26  }
```

Default Parameter

```
27  int main()
28  {
29    Teks myTeks;
30
31    cout << "Tuliskan satu baris teks: ";
32    myTeks.mygetLine();
33    cout << "Anda menulis teks berikut: "
34         << myTeks.displayTeks() << endl;
35    cout << "Tuliskan satu baris teks akhiri
        dengan X: ";
36    myTeks.mygetLine('X');
37    cout << "Anda menulis teks berikut: "
38         << myTeks.displayTeks() << endl;
39
40    return 0;
41  }
```