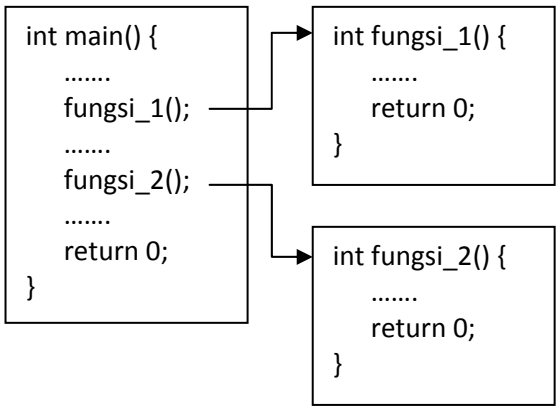
Bab 6
Sub Rutin

A. Pengertian Sub Rutin

Suatu program komputer biasanya merupakan suatu sistem besar yang terdiri dari sub sistem - sub sistem yang mempunyai tugas sendiri-sendiri, saling bekerja sama dan berinteraksi untuk menyelesaikan suatu permasalahan. Dengan adanya pembagian tugas oleh sub sistem – sub sistem ini maka suatu permasalahan dapat diselesaikan dengan lebih cepat dan kesalahan-kesalahan yang mungkin terjadi selama proses penyelesaian masalah dapat dideteksi dan diketahui sedini mungkin, termasuk di sub sistem mana kesalahannya terjadi.

Program komputer (terutama yang komplek) juga sebaiknya "dipecah-pecah" menjadi program-program kecil. Program-program kecil tersebut disebut dengan sub rutin. Sub rutin dibagi menjadi dua macam, yaitu sub rutin yang mengembalikan nilai dan sub rutin yang tidak mengembalikan nilai. Dalam Pascal kedua sub rutin yang mengembalikan nilai disebut dengan function, sedangkan yang tidak mengembalikan nilai disebut dengan procedure. Tetapi untuk C++ dan Java, kedua sub rutin tersebut dijadikan satu tetapi dapat diatur untuk dapat mengembalikan nilai maupun tidak mengembalikan nilai.

Untuk C++, sub rutin tersebut ada dalam suatu function, sedangkan pada Java, sub rutin berbentuk suatu method yang disebut dengan function method.



Baik C++ maupun Java terdapat dua macam fungsi, yaitu user-defined function dan built-in function. User-defined function merupakan fungsi yang didefinisikan sendiri atau dibuat sendiri oleh pemrogram. Sedangkan built-in function adalah fungsi yang sudah ada atau sudah disediakan oleh kompiler dan siap digunakan oleh pemrogram.

B. Sub Rutin Dalam C++ dan Java

Bentuk umum sub rutin (function) pada C++ dan Java sangat mirip. Untuk function yang mengembalikan nilai, setiap function harus didahului oleh tipe data yang sesuai dengan jenis data yang akan dikembalikan, kecuali tipe data array.

Setiap function juga mempunyai daftar parameter dimana setiap parameter dipisahkan dengan tanda koma (,). Parameter-parameter ini digunakan untuk menerima data (nilai) dari program yang memanggilnya.

Kita dapat mendeklarasikan banyak parameter atau tidak sama sekali. Untuk fungsi yang tidak mempunyai parameter merupakan fungsi yang nilainya tetap (tidak berubah). Sedangkan fungsi yang mempunyai parameter nilai fungsinya dinamis (dapat berubah-ubah). Parameter-parameter ini merupakan variabel-variabel yang harus dideklarasikan sendiri-sendiri meskipun tipe datanya sama.

Bentuk umum dari function yang mengembalikan nilai adalah sebagai berikut :

```
tipe_data nama_fungsi(daftar_parameter)
{
    isi dari fungsi
    return<ekspresi>
}
```

Contoh :

```
int contoh(int a, int b) {
    ……..
    return(c);
}
```

Contoh function di atas adalah function yang benar, sedangkan contoh function yang salah adalah :

```
int contoh(int a, b) {
    ……..
    return(c);
}
```

Setiap function mempunyai kode-kode program sendiri-sendiri karena tugas yang harus diselesaikan oleh setiap function juga berbeda-beda. Variabel-variabel yang digunakan dalam function disebut dengan variabel lokal. Oleh karena itu variabel tersebut hanya dapat digunakan didalam function itu sendiri tidak bisa digunakan oleh function lain atau program utama.

Variabel lokal ini berfungsi pada saat function tersebut aktif dan akan hilang (dihapus) jika function sudah tidak aktif lagi atau setelah function selesai melakukan tugasnya (kecuali variabel yang digunakan dalam function adalah variabel global yang dapat digunakan oleh semua function dan program utama).

Untuk dapat digunakan, function biasanya mempunyai parameter-parameter yang digunakan untuk menerima masukan dari program yang memanggilnya. Parameter-parameter ini disebut dengan parameter formal. Parameter formal ini termasuk dalam variabel lokal yang akan berfungsi pada saat function aktif dan akan dihapus pada saat function selesai melakukan tugasnya. Perhatikan contoh function dalam bahasa C++ berikut ini :

```
1. /* function akan menghasilkan nilai 1
      jika c sama dengan s, sebaliknya
      bernilai 0 jika c tidak sama dengan s */
2. int cek(char s, char c) {
3.                 if(s==c) return 1;
4.                 return 0;
5. }
```

Function cek() mempunyai dua buah parameter formal, yaitu s dan c yang mempunyai tipe data yang sama, char. Function cek() merupakan function yang mengembalikan nilai sehingga dia harus mempunyai tipe data dimana tipe data-nya dalam hal ini adalah integer.

Function cek() ini bertugas untuk memeriksa apakah variabel c sama dengan variabel s. Jika sama maka function cek() akan bernilai 1, sebaliknya jika tidak maka akan bernilai 0. Variabel s dan c ini merupakan variabel lokal yang hanya dapat digunakan dalam function cek() saja, tidak bisa digunakan oleh function atau program lain.

Permasalahan yang sama untuk Java adalah sebagai berikut :

```
1. class ricek {
2.     public int cek(char s, char c) {
3.             if(s==c) return 1;
4.             return 0;
5.      }
6. }
```

Telah disebutkan sebelumnya bahwa sub rutin dalam Java berbentuk class dimana didalam class tersebut dimungkinkan untuk mempunyai satu atau lebih function method. Class ricek() ini dapat digunakan oleh class lain, dalam hal ini untuk mengecek suatu karakter karena class ricek() mempunyai function method cek().

Berikut ini adalah kode lengkap dari kedua program di atas.
Untuk C++ :

```
1.  #include <iostream>
2.  using namespace std;
3.  int cek(char s, char c) {
4.      if(s==c) return 1;
5.      return 0;
6.  }
7.  int main(void) {
8.      char a,b;
9.      a = 'a';
10.     b = 'a';
11.     cout << cek(a,b) << endl;
12.     return 0;
13. }
```

Keluaran program adalah :

Dari kode program C++ di atas terlihat bahwa function cek() dipanggil melalui argumen pada program utama (main()) dimana argumen tersebut mempunyai variabel masukan untuk function cek() yaitu variabel a dan b (baris ke-11).

Variabel a dan b ini kemudian disalin oleh parameter formal function cek() yaitu s dan c. Parameter formal ini kemudian diproses lebih lanjut yaitu pengecekan apakah parameter formal s sama dengan parameter formal c. Jika sama maka function cek() akan bernilai 1, sebaliknya jika tidak sama maka function cek() akan bernilai 0. Nilai dari function cek() ini kemudian langsung dicetak ke layar oleh program utama (baris ke-11).

Kode program untuk bahasa Java dari permasalahan yang sama adalah :

```
1.  class ricek {
2.      public int cek(char s, char c) {
3.      if(s==c) return 1;
4.      return 0;
5.      }
6.  }
7.  class ricekApp {
8.      public static void main
       (String[ ] args) {
9.      ricek ck = new ricek();
10.     char a='a',b='b';
11.     System.out.println(ck.cek(a,b));
12.     }
13. }
```

Keluaran program adalah :

Dalam Java, class ricek() yang mempunyai function method cek() tidak bisa langsung kita gunakan (kita panggil), tetapi harus dibuat / diciptakan dahulu obyek dari class ricek() seperti yang terlihat pada baris ke-9 dimana obyek baru dari class ricek() dalam class ricekApp() diberi nama ck. Dengan obyek ck inilah kita dapat mengakses method yang dipunyai oleh class ricek() karena secara otomatis obyek ck juga mempunyai method cek() yang dimiliki oleh class ricek() (baris ke-11).

Contoh di atas merupakan contoh function dimana function tersebut mempunyai parameter, dalam hal ini parameter s dan c yang bertipe char. Function cek() tersebut nilainya akan berubah-ubah sesuai dengan nilai masukannya. Untuk contoh di atas nilai function dari cek() cuma ada dua, yaitu 0 atau 1.

Berikut ini adalah contoh function yang tidak mempunyai parameter sama sekali sehingga nilai function-nya tidak akan berubah-ubah seperti halnya pada contoh sebelumnya.

Contoh dalam bahasa C++ :

```
1.  #include <iostream>
2.  using namespace std;
3.  int fpb(){
4.      int a=24,b=18,hasil;
5.      int r = a % b;
6.      if (r==0) hasil = b;
7.      else {
8.      while(r!=0) {
9.              a = b;
10.             b = r;
11.             r = a % b;
12.             hasil = b;
13.       }
14.     }
15.     return(hasil);
16. }
17. void main() {
18.     cout << "FPB-nya = ";
19.     cout << fpb() <<endl;
20. }
```

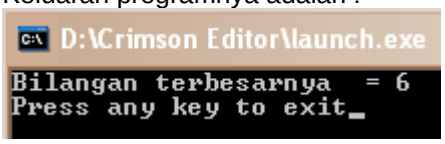
Keluaran programnya adalah :

Untuk bahasa Java :

```
1.  class hitung {
2.      public int fpb(){
3.      int a=78,b=24,hasil=0;
4.      int r = a % b;
5.      if (r==0) hasil = b;
6.        else {
7.              while(r!=0) {
8.              a = b;
9.              b = r;
10.             r = a % b;
11.             hasil = b;
12.       }
13.     }
14.     return hasil;
```

```
15.      }
16.    }
17.    class fpbApp {
18.      public static void main
       (String[ ] args) {
19.        hitung sekutu = new hitung();
20.        System.out.println("Bilangan
         terbesarnya  = " + sekutu.fpb());
21.      }
22. }
```

Keluaran programnya adalah :



Nilai dari function fpb() di atas untuk bahasa C++ pasti 6 karena nilai dari a dan b telah ditetapkan besarnya, yaitu 24 dan 18. Sedangkan untuk bahasa Java nilai method fpb() dari class hitung() juga pasti tetap, yaitu 6 karena nilai a dan b juga telah ditentukan (a=78 dan b=24).

C. Function yang Mengembalikan Nilai

Yang dimaksud dengan function yang mengembalikan nilai adalah suatu sub rutin yang bila dipanggil oleh suatu program (argumen) maka argumen tersebut akan memperoleh nilai balikan dari function tersebut. Atau dengan kata lain, suatu function yang mempunyai nilai.

Karena mempunyai nilai inilah maka suatu function yang mengembalikan nilai harus mempunyai tipe data sesuai dengan nilai yang dihasilkannya. Perhatikan baris ke-2 pada function cek() pada contoh sebelumnya. Function cek() menghasilkan nilai integer (lihat baris ke-3 dan ke-4) yaitu 0 atau 1 (return 0 dan return 1), sehingga tipe data-nya juga harus integer.

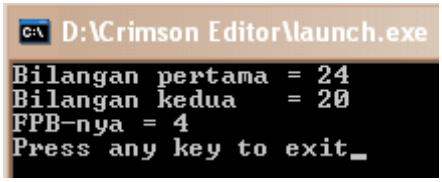
Dengan demikian dapat disimpulkan bahwa ciri dari function yang mengembalikan nilai adalah :

1. Function tersebut mempunyai tipe data.
2. Diakhiri dengan klausa return.

Berikut contoh program C++ yang menggunakan function dimana function-nya dapat mengembalikan nilai.

```
1.  #include <iostream>
2.  using namespace std;
3.  int fpb(int a, int b) {
4.        int hasil;
5.        int r = a % b;
6.        if (r==0) hasil = b;
7.        else {
8.              while(r!=0) {
9.              a = b;  b = r;
10.             r = a % b;
11.             hasil = b;
12.             }
13.       }
14.       return(hasil);
15.   }
16.   void main() {
17.       int m,n;
18.       do {
19.             cout << "Bilangan pertama
    = ";
20.             cin >> m;
21.             cout << "Bilangan kedua   =
    ";
22.             cin >> n;
23.       } while (m < n);
24.       cout << "FPB-nya = " << fpb(m,n)
      <<endl;
25. }
```

Keluaran programnya :



Baris ke-3 sampai dengan ke-16 merupakan sub rutin (function) yang bernama fpb(). Sedangkan baris ke-17 sampai dengan ke-26 merupakan program utamanya. Program utama ini akan memanggil function fpb() melalui suatu argumen (lihat baris ke-25). Function fpb() bertugas untuk melakukan pencarian faktor persekutuan besar dari dua buah bilangan yang dimasukkan di program utama (lihat baris ke-19 sampai dengan ke-24). Setelah selesai melakukan tugasnya, maka function fpb() akan mempunyai nilai yang langsung ditampilkan pada program utama.

Function fpb() mempunyai tipe data integer dan mempunyai dua buah parameter

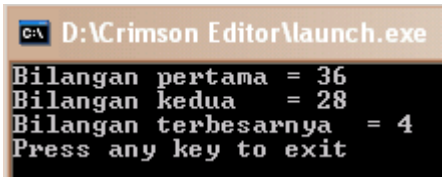
formal yang bertipe data integer juga, yaitu a dan b (baris ke-3). Function tersebut juga mempunyai variabel hasil yang bertipe data integer (baris ke-4). Function fpb() ini nilainya akan sama dengan variabel hasil (baris ke-15).

Variabel a, b, dan hasil merupakan variabel lokal dimana ketiga variabel ini hanya berfungsi pada function fpb() saja. Variabel a dan b bertugas untuk menerima data yang dikirim oleh program lain yang memanggilnya sedangkan variabel hasil digunakan untuk menyimpan data hasil pencarian faktor persekutuan besar (baris ke-6 dan baris ke-12).

Untuk permasalahan yang sama dengan menggunakan bahasa Java adalah sebagai berikut :

```
1.  import java.util.Scanner;
2.  import java.io.*;
3.  class hitung {
4.      public int fpb(int a, int b) {
5.        int hasil=0;
6.        int r = a % b;
7.        if (r==0) hasil = b;
8.        else {
9.              while(r!=0) {
10.             a = b;
11.             b = r;
12.             r = a % b;
13.             hasil = b;
14.       }
15.       }
16.       return hasil;
17.       }
18.   }
19.   class sekutuBesar {
20.       public static void
      main(String[ ] args) {
21.         hitung sekutu = new hitung();
22.         int m,n;
23.         Scanner input =
          new Scanner(System.in);
24.         do {
25.               System.out.print("Bilangan
          pertama = ");
26.               m = input.nextInt();
27.               System.out.print("Bilangan
          kedua   = ");
28.               n = input.nextInt();
29.         } while(m < n);
30.         System.out.println("Bilangan
            terbesarnya  = " +
            sekutu.fpb(m,n));
31.       }
32. }
```

Keluaran programnya :



Pada program Java di atas terlihat bahwa pencarian faktor persekutuan besar-nya dilakukan oleh Class hitung(). Class hitung() ini mempunyai method fpb(), yang merupakan function fpb() pada C++, dan bertugas untuk melakukan pencarian faktor persekutuan besar. Class sekutuBesar() kemudian membuat obyek baru dari Class hitung() dengan nama sekutu (lihat baris ke-21). Atau dengan kata lain Class hitung() “dipanggil” oleh program utamanya yaitu Class sekutuBesar().

Program Java tersebut jika di-compile akan menghasilkan dua buah class, yaitu hitung.class dan sekutuBesar.class dimana sekutuBesar.class memanggil hitung.class untuk melakukan perhitungan faktor persekutuan besar dari dua buah bilangan.

D. Function yang Tidak Mengembalikan Nilai

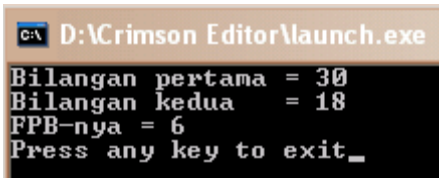
Untuk sub rutin (function) yang tidak mengembalikan nilai bentuknya sangat mirip dengan function yang mengembalikan nilai. Perbedaannya adalah penggunaan kata kunci atau klausa void pada function yang tidak mengembalikan nilai baik pada bahasa C++ maupun bahasa Java.

Untuk lebih jelasnya, perhatikan program C++ berikut ini untuk permasalahan yang sama, yaitu mencari faktor persekutuan besar.

```
1.  #include <iostream>
2.  using namespace std;
3.  void fpb(int a, int b) {
```

```
4.      int hasil;
5.      int r = a % b;
6.      if (r==0) hasil = b;
7.      else {
8.      while(r!=0) {
9.              a = b;
10.             b = r;
11.             r = a % b;
12.             hasil = b;
13.     }
14.     }
15.     cout << "FPB-nya = " << hasil
    <<endl;
16.   }
17.   void main() {
18.     int m,n;
19.     do {
20.     cout << "Bilangan pertama = ";
21.     cin >> m;
22.     cout << "Bilangan kedua   = ";
23.     cin >> n;
24.     } while (m < n);
25.     fpb(m,n);
26. }
```

Keluaran programnya :



Program di atas, function fpb() (baris ke-3 sampai dengan baris ke-16) tidak mempunyai tipe data dan klausa return diakhir program, sehingga dapat kita simpulkan bahwa function fpb() bukan merupakan suatu function yang mengembalikan nilai.

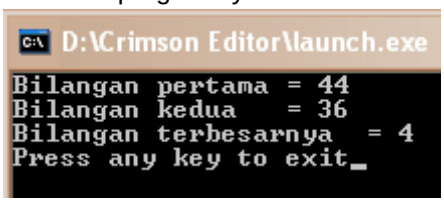
Sebaliknya, function fpb() diawali dengan klausa void sehingga function tersebut merupakan suatu function yang tidak mengembalikan nilai.

Hasil perhitungan faktor persekutuan besar dari dua buah bilangan tidak dikembalikan ke program utama yang memanggilnya tetapi ditampilkan sendiri oleh function tersebut (baris ke-15). Dengan argumen pada program utama yang memanggil fuction fpb() (baris ke-25) tidak menghasilkan suatu nilai seperti halnya pada contoh sebelumnya (function yang mengembalikan nilai).

Untuk contoh function yang tidak mengembalikan nilai pada bahasa Java dengan permasalahan yang sama adalah sebagai berikut :

```
1.  import java.util.Scanner;
2.  import java.io.*;
3.  class hitung {
4.      public void fpb(int a, int b) {
5.              int hasil=0;
6.              int r = a % b;
7.              if (r==0) hasil = b;
8.              else {
9.              while(r!=0) {
10.             a = b;
11.             b = r;
12.             r = a % b;
13.             hasil = b;
14.             }
15.         }
16.         System.out.println("Bilangan
        terbesarnya  = " + hasil);
17.     }
18. }
19. class sekutuBesar {
20.     public static void main
      (String[ ] args) {
21.     hitung sekutu = new hitung();
22.     int m,n;
23.     Scanner input =
        new Scanner(System.in);
24.     do {
25.             System.out.print("Bilangan
        pertama = ");
26.             m = input.nextInt();
27.             System.out.print("Bilangan
        kedua   = ");
28.             n = input.nextInt();
29.     } while(m < n);
30.     sekutu.fpb(m,n);
31.   }
32. }
```

Keluaran programnya :

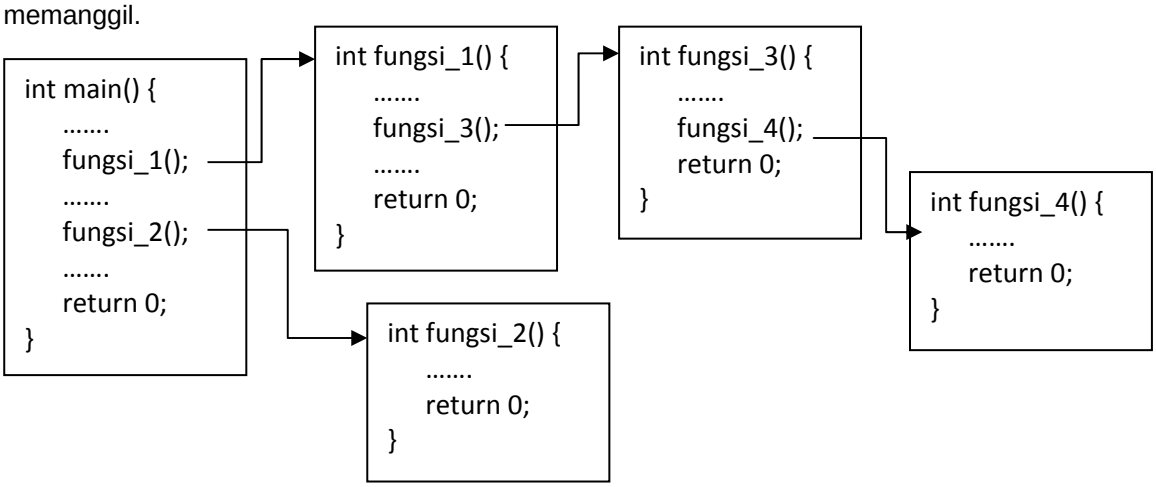


Sama seperti pada bahasa C++, class yang mempunyai function method untuk yang tidak mengembalikan nilai pada bahasa Java juga tidak mempunyai tipe data, tetapi diawali dengan klausa void (baris ke-4).

Hasil perhitungan faktor persekutuan besar oleh class hitung() juga tidak ditampilkan oleh class sekutuBesar() yang memanggilnya (baris ke-30), tetapi oleh class hitung() itu sendiri (baris ke-16).

E. Function Call Function

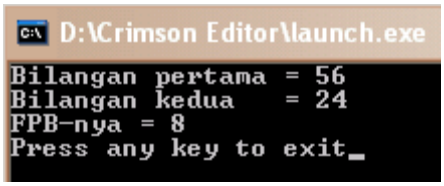
Sub rutin dalam suatu program tidak hanya dapat dipanggil oleh program utama saja tetapi antar sub rutin juga dapat saling memanggil.



Berikut adalah contoh sub rutin yang memanggil sub rutin lainnya.

```
1.  #include <iostream>
2.  using namespace std;
3.  void fpb(int a, int b){
4.        int hasil;
5.        int r = a % b;
6.        if (r==0) hasil = b;
7.        else {
8.        while(r!=0) {
9.                  a = b;
10.                 b = r;
11.                 r = a % b;
12.                 hasil = b;
13.       }
14.       }
15.       cout << "FPB-nya = " << hasil
      <<endl;
16.     }
17.     void input_data(){
18.       int m,n;
19.       do {
20.         cout << "Bilangan pertama = ";
21.         cin >> m;
22.         cout << "Bilangan kedua   = ";
23.         cin >> n;
24.       } while (m < n);
25.       fpb(m,n);
26.     }
27.     void main() {
28.       input_data();
29. }
```

Keluaran programnya adalah :



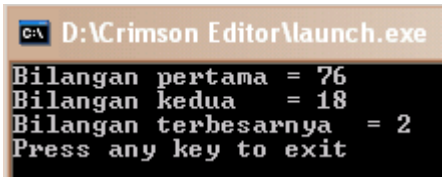
Program di atas mempunyai dua buah function, yaitu function fpb() dan function input_data(). Pertama kali function yang dipanggil oleh program utama adalah function input_data() (baris ke-28). Kemudian function

input_data() melakukan pemanggilan function lain yaitu function fpb() (baris ke-25) setelah user memasukkan data untuk bilangan pertama dan bilangan kedua.

Sedangkan kode program dalam bahasa Java untuk permasalahan yang sama adalah :

```
1.  import java.util.Scanner;
2.  import java.io.*;
3.  class hitung {
4.      public void fpb(int a, int b){
5.      int hasil=0;
6.      int r = a % b;
7.      if (r==0) hasil = b;
8.      else {
9.              while(r!=0) {
10.             a = b;
11.             b = r;
12.             r = a % b;
13.             hasil = b;
14.         }
15.       }
16.       System.out.println("Bilangan
        terbesarnya  = " + hasil);
17.     }
18.   }
19.   class input_data {
20.     public void data_input() {
21.       hitung sekutu = new hitung();
22.       int m,n;
23.       Scanner input =
        new Scanner(System.in);
24.       do {
25.         System.out.print("Bilangan
        pertama = ");
26.         m = input.nextInt();
27.         System.out.print("Bilangan
        kedua   = ");
28.         n = input.nextInt();
29.       } while(m < n);
30.       sekutu.fpb(m,n);
31.     }
32.   }
33.   class sekutuBesar {
34.     public static void main(String[ ]
      args) {
35.       input_data masukan =
        new input_data();
36.       masukan.data_input();
37.     }
38. }
```

Keluaran programnya adalah :



F. Call by Value dan Call by References

Ada dua cara bagaimana suatu argumen dalam suatu program dapat memanggil sub rutin (function), yaitu call by value dan call by reference.

Yang dimaksud dengan call by value adalah metode yang menyalin data (nilai) dari argumen yang memanggil function ke parameter dari function tersebut. Akibatnya jika ada perubahan nilai pada parameter function tidak akan berpengaruh pada nilai aslinya (nilai yang ada pada argumen program yang memanggil function tersebut).

Sebaliknya untuk call by reference yang disalin bulan nilainya tetapi alamat memori yang menyimpan nilai tersebut sehingga jika terjadi perubahan-perubahan nilai pada parameter function, maka secara otomatis nilai argumennya juga akan ikut berubah.

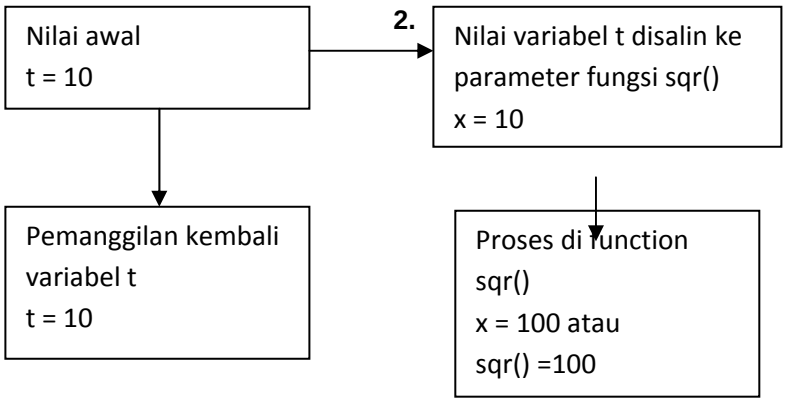
Untuk lebih jelasnya, perhatikan contoh call by value dengan C++ berikut ini :

```
1.  #include <iostream>
2.  using namespace std;
3.  int sqr(int x) {
4.      x = x*x;
5.      return(x);
6.  }
7.  int main(void) {
8.      int t=10;
9.      cout << sqr(t) << ", " << t <<endl;
10.     return 0;
11. }
```

Keluaran programnya adalah :

Nilai argumen sqr() pada program utama yaitu t (bernilai 10) disalin ke parameter x function sqr(). Didalam function sqr(), nilai x dirubah (x=x*x) sehingga function sqr() bernilai 100. Nilai function sqr() ini langsung ditampilkan oleh program yang memanggilnya (baris ke-9). Akan tetapi nilai t, yang juga

ditampilkan oleh program utama (bariske-9), tetap 10. Keluaran program ini adalah 100, 10 dimana 100 adalah nilai dari function fpb() dan 10 adalah nilai variabel t.



Untuk permasalahan yang sama dalam bahasa Java adalah sebagai berikut :

```
1.  import java.io.*;
2.  class kuadrat {
3.        public int sqr(int x) {
4.        x = x*x;
5.        return(x);
6.        }
7.  }
8.  class power {
9.        public static void main
        (String[ ] args) {
10.       kuadrat a = new kuadrat();
11.       int t=10;
12.       System.out.println(a.sqr(t) +
          ", " + t);
13.       }
14. }
```

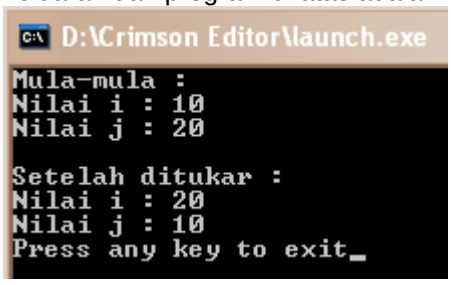
Keluaran programnya adalah :

Untuk call by reference, tipe data yang digunakan adalah tipe data pointer karena yang disalin adalah alamat dari memori dimana data disimpan (pembahasan mengenai pointer ini ada di bab tersendiri). Untuk bahasa Java tidak menggunakan call by reference karena tidak ada pointer dalam bahasa Java.

Contoh call by reference (dengan menggunakan bahasa C++) adalah sebagai berikut :

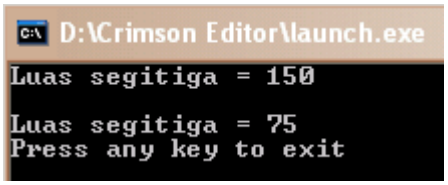
```
1.  #include <iostream>
2.
3.  void tukar(int *x, int *y) {
4.        int temp;
5.        temp = *x;
6.        *x = *y;
7.        *y = temp;
8.  }
9.  int main(void) {
10.       int i, j;
11.       i = 10;
12.       j = 20;
13.       cout << "Mula-mula : " << endl;
14.       cout << "Nilai i : " << i << endl;
15.       cout << "Nilai j : " << j << endl;
16.       tukar(&i, &j);
17.       cout << endl << "Setelah ditukar : "
        << endl;
18.       cout << "Nilai i : " << i << endl;
19.       cout << "Nilai j : " << j << endl;
20.       return 0;
21. }
```

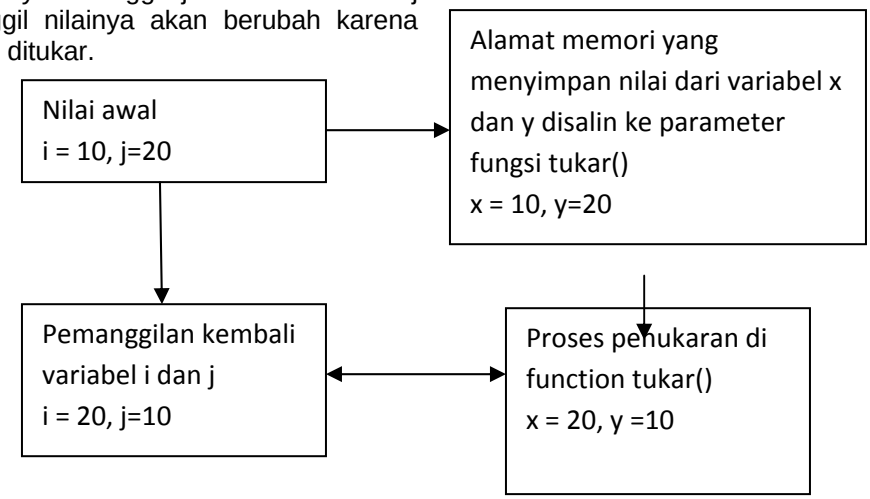
Tanda * pada variabel di function tukar() merupakan variabel pointer. Adapun keluaran dari program di atas adalah :



Baris ke-14 dan 15, digunakan untuk menampilkan nilai dari variabel i dan j yang nilainya masing-masing 10 dan 20. Setelah dilakukan penukaran dengan memanggil function tukar() pada baris ke-16, program kemudian memanggil kembali variabel i dan j (baris ke-18 dan ke-19). Ternyata variabel i dan j telah tertukar.

Hal ini terjadi karena pada function tukar() menyalin alamat memori yang menyimpan variabel i dan j dan kemudian menukar isinya sehingga jika variabel i dan j jika dipanggil nilainya akan berubah karena isinya telah ditukar.





G. Parameter dengan Nilai Default

Nilai parameter dalam suatu function C++ dapat kita beri nilai default, yaitu suatu nilai yang diberikan secara default oleh function jika parameter dari argumen yang memanggil function tersebut tidak kita berikan.

Untuk lebih jelasnya perhatikan contoh berikut ini :

```
1.  #include <iostream>
2.  using namespace std;
3.  double luas_segi3(int alas, int
    tinggi=10){
4.      return(0.5 * alas * tinggi);
5.    }
6.    void main() {
7.      int a=15,t=20;
8.      cout << "Luas segitiga = ";
9.      cout << luas_segi3(a,t) <<endl;
10.     cout << endl;
11.     cout << "Luas segitiga = ";
12.     cout << luas_segi3(a) <<endl;
13. }
```

Keluaran programnya adalah :

Program di atas mempunyai satu buah function yaitu function luas_segi3() dimana function tersebut mempunyai dua buah parameter formal yaitu alas dan tinggi. Parameter alas tidak mempunyai nilai default, tetapi untuk parameter tinggi mempunyai nilai default, yaitu 10.

Program utama dari program tersebut telah menentukan bahwa nilai dari variabel a yang nantinya digunakan sebagai variabel masukan untuk parameter alas dari function luas_segi3() adalah 15. Sedangkan variabel t yang nantinya digunakan sebagai variabel masukan untuk parameter tinggi dari function luas_segi3() adalah 20. Sehingga ketika argumen program utama yang memanggil function luas_segi3() pada baris ke-9 dijalankan, maka nilai dari parameter alas dan tinggi pada function luas_segi3() masing-masing adalah 15 dan 20. Dengan demikian function luas_segi3() akan bernilai 150.

Akan tetapi untuk argumen baris ke-12 pada program utama terlihat bahwa argumen tersebut tidak menyertakan variabel t, tetapi hanya menyertakan variabel a. Sehingga ketika argumen tersebut dijalankan maka hanya parameter alas pada function luas_segi3() saja yang menerima masukan dari argumen tersebut. Sedangkan untuk parameter tinggi tidak memperoleh masukan. Akan tetapi karena parameter tinggi mempunyai nilai default 10, maka nilai variabel tinggi yang digunakan untuk menghitung luas segitiga adalah 10. Dengan demikian nilai dari function luas_segi3() adalah 75.

H. Overloading

Overloading adalah function-function yang ada dalam suatu program dimana function-function tersebut mempunyai nama yang sama tetapi parameter formal-nya berbeda-beda antara yang satu dengan yang lainnya.

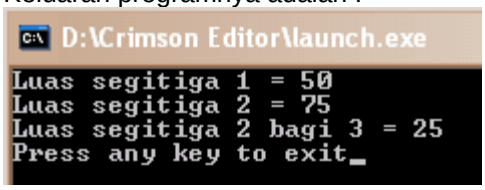
Ada tiga jenis overloading, yaitu :

1. Function-function tersebut mempunyai jumlah parameter formal yang berbeda tetapi tipe data-nya sama.
2. Function-function tersebut mempunyai jumlah parameter formal yang sama tetapi tipe data yang berbeda.
3. Function-function tersebut mempunyai jumlah parameter formal yang berbeda dan tipe data dari parameter formal tersebut juga berbeda.

Berikut ini diberikan contoh untuk function-function yang mempunyai jumlah parameter formal yang berbeda tetapi tipe data-nya sama.
Untuk bahasa C++ :

```
1.  #include <iostream>
2.  using namespace std;
3.  void luas_segi3(int alas){
4.        int tinggi=10;
5.        cout << "Luas segitiga 1 = "
6.        cout << (0.5 * alas * tinggi) << endl;
7.     }
8.     void luas_segi3(int alas,int tinggi){
9.        cout << "Luas segitiga 2 = "
10.       cout << (0.5 * alas * tinggi) << endl;
11.   }
12.   void luas_segi3(int alas, int tinggi,
      int bagi){
13.       cout << "Luas segitiga 2 bagi "
14.       cout << bagi << " = "
15.       cout << (0.5 * alas * tinggi)/bagi
16.       cout << endl;
17.   }
18.   void main() {
19.     luas_segi3(10);
20.     luas_segi3(10,15);
21.     luas_segi3(10,15,3);
22. }
```

Keluaran programnya adalah :



Program di atas mempunyai tiga buah function yang mempunyai nama sama yaitu luas_segi3(). Function yang pertama (baris ke-3) hanya mempunyai satu buah parameter formal yaitu alas dengan tipe data integer. Sedangkan function yang kedua (baris ke-8) mempunyai dua buah parameter formal alas dan tinggi dimana keduanya bertipe data integer. Function yang terakhir (baris ke-12) mempunyai tiga buah parameter formal yaitu alas, tinggi, dan bagi yang semuanya bertipe data yang sama, yaitu integer.

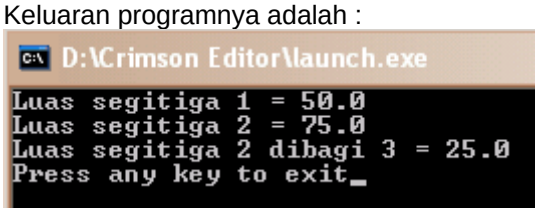
Pada program utama, terdapat tiga buah argumen dimana argumen pertama (baris ke-19) hanya mempunyai satu variabel masukan untuk function, argumen kedua (baris ke-20) mempunyai dua buah variabel masukan, dan argumen terakhir (baris ke-21) mempunyai tiga buah variabel masukan.

C++ secara otomatis akan mengarahkan argumen pertama kepada function yang pertama, argumen kedua kepada function yang kedua, dan argumen ketiga kepada function yang ketiga, sesuai dengan variabel masukan yang dipunyai oleh masing-masing argumen.
Sedangkan untuk bahasa Java :

```
1.  class hitung {
2.        public void luas_segi3(int alas){
3.                int tinggi=10;
4.                double luas=0.5*alas*tinggi;
```

```
5.              System.out.println("Luas segitiga
        1 = " + luas);
6.      }
7.      public void luas_segi3(int alas, int tinggi){
8.              double luas=0.5*alas*tinggi;
9.              System.out.println("Luas segitiga
        2 = " + luas);
10.     }
11.     public void luas_segi3(int alas, int tinggi,int bagi){
12.             double luas=(0.5*alas*tinggi)/bagi;
13.             System.out.println("Luas segitiga
        2 dibagi " + bagi + " = " + luas);
14.     }
15.   }
16.   class overload1 {
17.     public static void main (String[ ] args) {
18.             hitung sekutu = new hitung();
19.             sekutu.luas_segi3(10);
20.             sekutu.luas_segi3(10,15);
21.             sekutu.luas_segi3(10,15,3);
22.     }
23. }
```

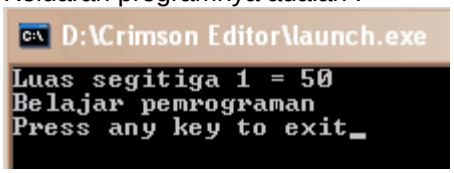
Keluaran programnya adalah :



Sedangkan contoh untuk function-function yang mempunyai jumlah parameter formal yang sama tetapi tipe data-nya berbeda adalah seperti berikut.
Untuk bahasa C++ :

```
1.  #include <iostream>
2.  using namespace std;
3.  void luas_segi3(int alas){
4.      int tinggi=10;
5.      cout << "Luas segitiga 1 = " << (0.5 * alas * tinggi) << endl;
6.    }
7.    void luas_segi3(char* alas){
8.      cout << alas << endl;
9.    }
10.   void main() {
11.     luas_segi3(10);
12.     luas_segi3("Belajar pemrograman");
13. }
```

Keluaran programnya adalah :



Program di atas mempunyai dua buah function yaitu luas_segi3() dimana setiap function mempunyai satu buah parameter formal tetapi tipe datanya berbeda. Function yang pertama parameter formalnya mempunyai tipe data integer, dan function kedua mempunyai parameter formal dengan tipe data pointer char.

Sama seperti sebelumnya, C++ juga secara otomatis akan mengarahkan argumen yang memanggil function pada program tersebut kepada function yang berkesesuaian. Argumen pertama (baris ke-11) akan diarahkan kepada function pertama pula (baris ke-3) karena tipe data dari variabel masukan sama dengan parameter formal dari function yang pertama, yaitu integer.

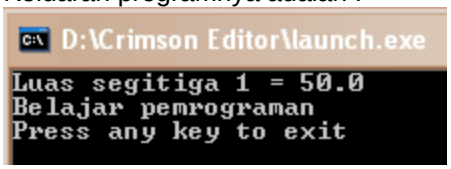
Sedangkan untuk argumen kedua (baris ke-12) akan diarahkan kepada function kedua (baris ke-7) karena tipe data dari variabel masukan sama dengan parameter formal dari function yang kedua, yaitu char.

Untuk bahasa Java dengan permasalahan yang sama adalah sebagai berikut :

```
1.  class hitung {
2.      public void luas_segi3(int alas){
3.              int tinggi=10;
4.              double luas=0.5*alas*tinggi;
5.              System.out.println
        ("Luas segitiga 1 = " + luas);
6.      }
7.      public void luas_segi3(String alas){
8.              System.out.println(alas);
9.      }
10.   }
11.   class overload2 {
12.     public static void main
        (String[ ] args) {
13.     hitung sekutu = new hitung();
```

```
14.     sekutu.luas_segi3(10);
15.     sekutu.luas_segi3("Belajar
        pemrograman");
16.     }
17. }
```

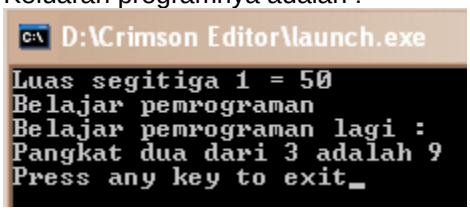
Keluaran programnya adalah :



Contoh untuk function-function yang mempunyai jumlah parameter formal yang berbeda dan tipe data yang berbeda pula adalah seperti berikut.
Untuk bahasa C++ :

```
1.  #include <iostream>
2.  using namespace std;
3.  void luas_segi3(int alas){
4.      int tinggi=10;
5.      cout << "Luas segitiga 1 = "
6.      cout << (0.5 * alas * tinggi) << endl;
7.    }
8.    void luas_segi3(char* alas){
9.      cout << alas << endl;
10.   }
11.   void luas_segi3(char* alas,int data){
12.     cout << alas << " : " << endl;
13.     cout << "Pangkat dua dari "
14.     cout << data << " adalah "
15.     cout << (data*data) << endl;
16.   }
17.   void main() {
18.     luas_segi3(10);
19.     luas_segi3("Belajar
    pemrograman");
20.     luas_segi3("Belajar pemrograman
        lagi",3);
21.   }
```

Keluaran programnya adalah :



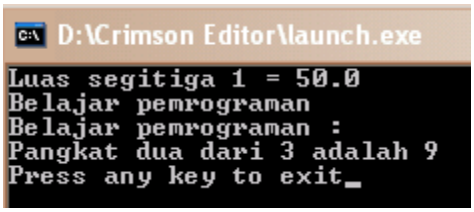
Program di atas mempunyai tiga buah function yang mempunyai nama yang sama, yaitu luas_segi3(). Masing-masing function mempunyai jumlah parameter formal yang berbeda dan tipe data dari parameter formal tersebut juga berbeda.

Sama dengan sebelumnya, C++ secara otomatis juga akan mengarahkan argumen yang memanggil function-function tersebut sesuai dengan jumlah variabel masukan dan tipe datanya.

Untuk bahasa Java dengan permasalahan yang sama adalah sebagai berikut :

```
1.  class hitung {
2.      public void luas_segi3(int alas){
3.              int tinggi=10;
4.              double luas=0.5*alas*tinggi;
5.              System.out.println
        ("Luas segitiga 1 = " + luas);
6.      }
7.      public void luas_segi3(String
    alas){
8.              System.out.println(alas);
9.      }
10.     public void luas_segi3(String alas,
      int data){
11.             System.out.println(alas + " :
    ");
12.             System.out.println("Pangkat
    dua
        dari " + data + " adalah " +
        (data*data));
13.     }
14.   }
15.   class overload3 {
16.     public static void main
      (String[ ] args) {
17.             hitung sekutu = new
    hitung();
18.             sekutu.luas_segi3(10);
19.             sekutu.luas_segi3("Belajar
        pemrograman");
20.             sekutu.luas_segi3("Belajar
        pemrograman",3);
21.     }
22. }
```

Keluaran programnya adalah :



I. Contoh-contoh

J.

K.