BAB 11

PROSEDUR ALGORITMA PEMROGRAMAN

BAB 11
PROSEDUR ALGORITMA PEMROGRAMAN

Capaian Pembelajaran

Setelah mempelajari materi dalam bab ini, Mahasiswa diharapkan mampu menganalisis, menyampaikan pendapat, dan mengoperasikan prosedur algoritma pemrograman pada sebuah kasus operasional bisnis

Pokok Bahasan

1. Sub Program (Modular)
2. Konsep Dasar Prosedur
3. Pemanggilan Prosedur
4. Implementasi Prosedur dalam Algoritma Pemrograman

Evaluasi Pembelajaran

Soal Latihan Prosedur Algoritma Pemrograman

Pre Test
Prosedur Algoritma Pemrograman

1. Apa yang anda Ketahui Sub Program (Modular)?
2. Sebutkan jenis dari Sub program!
3. Jelaskan perbedaan antara Prosedur dan Fungsi!
4. Apa yang anda ketahui dengan Parameter?
5. Sebutkan apa saja jenis dari parameter!
6. Apa yang anda ketahui bagaimana jenis parameter diproses?

Penggunaan Prosedur sering sekali dimanfaatkan oleh pengembang aplikasi untuk memillah milah kode program agar lebih terstruktur. Seluruh kode program tidak dibuat dalam kode utuh (program utama/main program), namun kode program dipilah dan disimpan menjadi beberapa bagian kecil. Walaupun dilakukan pemilahan, kode program ketika dijalankan tidak terjadi perbedaan. Bab ini berisi penjelasan tentang prosedur, yang memiliki beberapa sub bab, diantaranya: Sub Program, Konsep Prosedur, Parameter dalam Prosedur, Pemanggilan prosedur, dan Implementasi prosedur.

11.1. Sub Program (Modular)

Pada sebuah pengembangan program sederhana penuliasan kode program cukup ditulisakan pada satu kode utuh program (main program) saja. Namun, ketika program yang dikembangkan bertujuan untuk menyelesaikan sebuah permasalahan yang rumit, maka baris kode yang digunakan juga akan semakin banyak (kompleks). Penulisan kode program yang kompleks pada satu kode utuh biasanya menyulitkan pengembang pada saat penelusuran kesalahan (*error*) jika kode program dijalankan. Selain itu juga pengembang merasa sulit memahami dan menggunakan kode berulang jika kode program dibuat pada satu kode utuh. Berdasarkan kesulitan yang dihadapi pengembang diatas, maka kode program pada main program perlu dilakukan pemilahan menjadi beberapa bagian program kecil. Walaupun dilakukan pemilahan, fungsi utama program yang dibuat pada program utama ketika dijalankan tidak terjadi perubahan.

Teknik pemillahan kode program yang besar menjadi beberapa bagian program yang lebih kecil agar kode program mudah dipahami dan dapat digunakan kembali disebut dengan teknik pemrograman modular. Sementara itu, bagian program kecil hasil pemilahan dari main program dapat disebut dengan sub program atau dapat disebut dengan modular. Sub program hanya dapat berjalan ketika dipanggil oleh program utama. Pemanggilan sub program dapat dilakukan secara berulang oleh program utama. Namun sebaliknya, jika sub program tidak dipanggil oleh program utama, maka sub program tidak pernah berjalan.

Berikut ini merupakan manfaat dari penggunaan sub program:

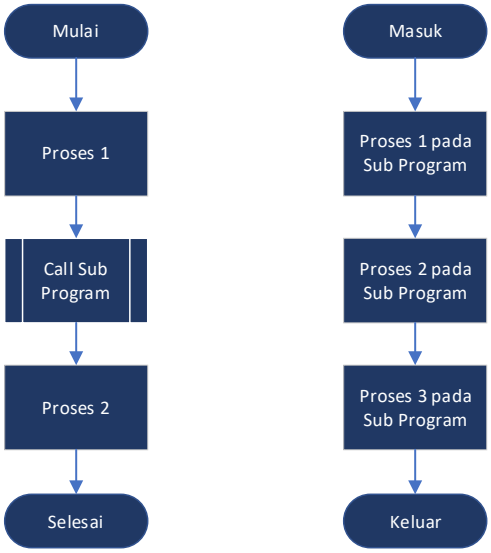
1. Dapat digunakan untuk program utama yang lain
   Jika kode program dipilah menjadi program kecil, maka sub program hasil pemilahan cukup disalin dan dipanggil pada program utama yang lainnya.
2. Kemudahan menemukan kesalahan kode program
   Pengembang mudah menemukan kesalahan dan mengkoreksi kode program karena baris kode dipilah menjadi lebih ringkas pada beberapa file
3. Program lebih fleksibel
   Dengan menggunakan sub program memungkinkan pengembang mudah memodifikasi baris kode yang dibuat
4. Progam menjadi sederhana (mengindari penulisan kode yang berulang)
   Pada program utama biasanya terdapat kode program yang ditulis secara berulang. Penulisan kode berulang dapat dipilah dan disimpan pada sebuah bagian sub program sehingga jika diperlukan pada program utama, maka sub program cukup dipanggil pada baris program utamanya. Hal ini menjadikan pengembang tidak perlu lagi menuliskan kode program beberapa kali, namun cukup dengan sebuah sub program saja dan melakukan pemanggilan sub program.

Dalam sebuah teknik pemrograman, sub program memiliki 2 jenis kharakteristik, yaitu:

1. Prosedur (*Procedure*)
   Prosedur merupakan sub program yang melakukan satu atau lebih proses tanpa mengembalikan nilai dari hasil dari proses yang dijalankan. Namun, pada prosedur diperbolehkan mencetak hasil luaran yang dihasilkan dari proses yang dijalankan. Contohnya, mencetak hasil dari proses aritmatika.
2. Fungsi (*Function*)
   Fungsi merupakan sub program yang melakukan satu atau lebih proses yang mengembalikan sebuah nilai untuk dikirimkan ke program utama *(return)*. Fungsi tidak diperbolehkan melakukan cetak hasil nilai yang didapatkan dari

proses yang sudah dijalankan. Fungsi hanya bertugas sebagai penampungan nilai sementara. Jika nilai tersebut akan dicetak, maka proses mencetak hasil dilakukan pada program utama, bukan pada fungsi itu sendiri. Sub program Function akan dibahas tersendiri pada bab berikutnya.

Pada gambar 11.1 merupakan bentuk *flowchart* pada sebuah program yang menggunakan sub program. Simbol *flowchart* pada sebuah sub program memiliki bentuk persegi panjang yang memiliki tambahan garis vertikal di sisi kanan dan kiri.



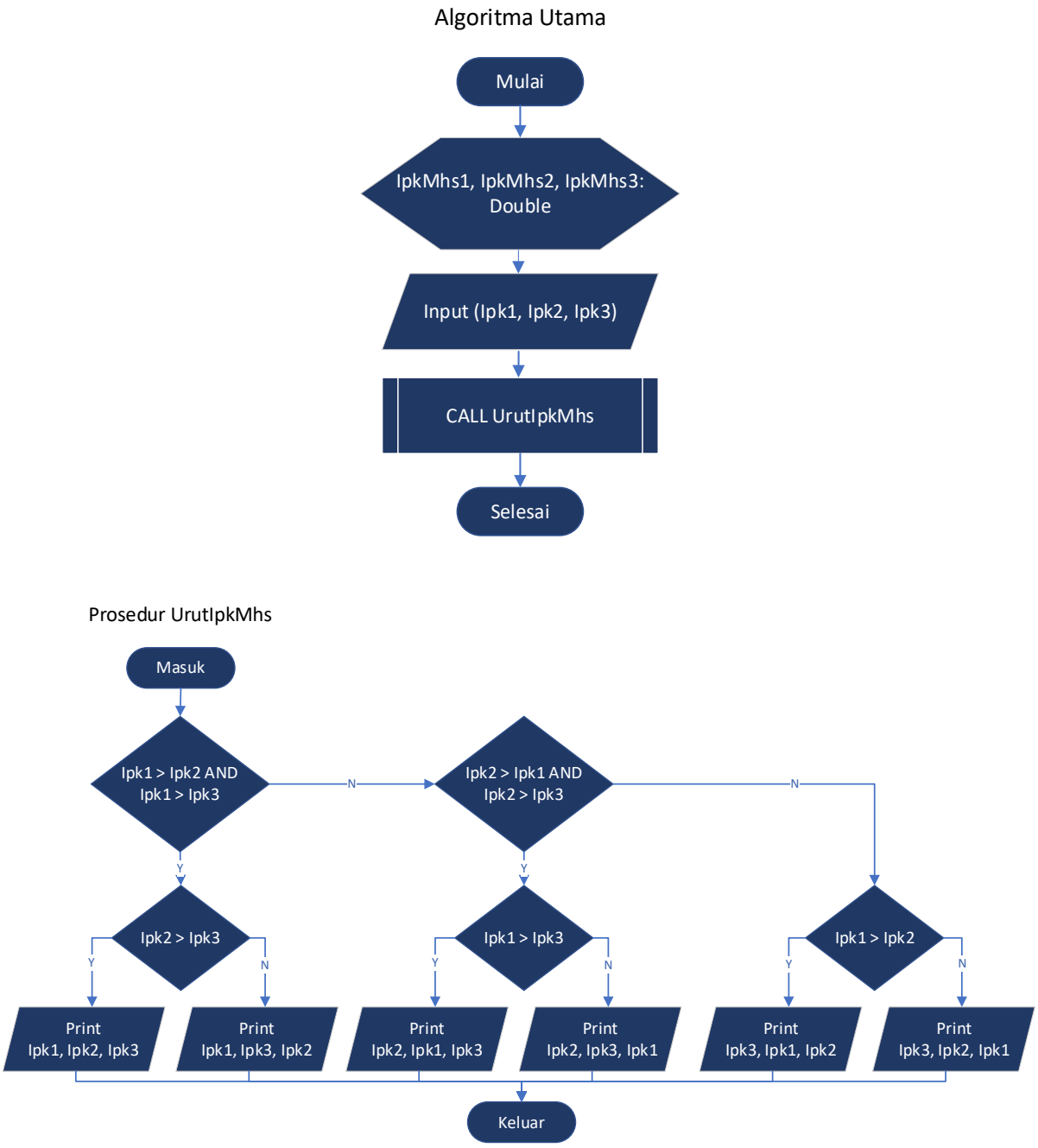
Gambar 11.1 Bentuk Flowchart untuk Sub Program

11.2. Konsep Prosedur

Sub Program dengan jenis prosedur merupakan bagian dari kode program yang terpisah yang melakukan tugas spesifik dimana hasilnya dapat diketahui setelah semua proses didalamnya selesai dikerjakan. Prosedur perlu dipanggil dalam program utama agar hasil akhir dari sebuah prosedur dapat diketahui. Prosedur dapat dipanggil secara berulang kali di program utama. Pengguaan prosedur biasanya ditandai dengan penggunaan `void`. Sebagai contoh, prosedur untuk mengurutkan nilai dari IPK dari tiga Mahasiswa mulai yang terbesar ke terkecil. Sebelum prosedur dijalankan nilai IPK Mahasiswa belum terurut, namun

setekah prosedur dijalankan, maka nilai IPK dari tiga mahasiswa sudah terurut dari terbesar ke terkecil.

Contoh penggunaan prosedur pada kasus mengurutkan nilai IPK dari tiga orang mahasiswa yang disajikan dalam bentuk flowchart



Gambar 11.2 Contoh Flowchart Prosedur UrutIpkMhs

Berikut ini adalah pseudocode untuk kasus mengurutkan nilai IPK dari tiga orang mahasiswa yang diinputkan

```
Algoritma: UrutIpkMhs {Melakukan Pengurutan 3 nilai IPK Mahasiswa yang diiputkan}
Deklarasi
ipkmhs1, ipkmhs2, ipkmhs3 : Double {tiga buah ipk yang akan diurutkan}

PROSEDURE PengurutanMhs()
Deklarasi
Tidak ada
Deskripsi
if ipkmhs1>ipkmhs2 AND ipkmhs1>ipkmhs3 then
      if ipkmhs2>ipkmhs3 then Write(ipkmhs1,ipkmhs2,ipkmhs3)
            else Write(ipkmhs1, ipkmhs3, ipkmhs2)
else if ipkmhs2>ipkmhs1 AND ipkmhs2>ipkmhs3 then
      if ipkmhs1>ipkmhs3 then Write(ipkmhs2,ipkmhs1,ipkmhs3)
            else Write(ipkmhs2, ipkmhs3, ipkmhs1)
else if ipkmhs1>ipkmhs2 then Write(ipkmhs3,ipkmhs1,ipkmhs2)
      else Write(ipkmhs3, ipkmhs2, ipkmhs1)

ALGORITMA UTAMA
Deklarasi
Tidak ada
Deskripsi
Input(ipkmhs1)
Input(ipkmhs2)
Input(Ipkmhs3)
Procedure pengurutanMhs()
END
```

Sementara itu, berikut merupakan hasil pengkodean yang dilakukan dengan bahasa java pada penerapan prosedur kasus mengurutkan nilai IPK dari tiga orang mahasiswa yang diinputkan

```
import java.io.*;
class UrutIpkMhs{
```

```
      public static Double ipkmhs1, ipkmhs2, ipkmhs3;
      static void pengurutanMhs() {

    if (ipkmhs1>ipkmhs2 && ipkmhs1>ipkmhs3){
      if (ipkmhs2>ipkmhs3){
            System.out.println(ipkmhs1 + "," + ipkmhs2 + "," + ipkmhs3);
      }else{
            System.out.println(ipkmhs1 + "," + ipkmhs3 + "," + ipkmhs2);
      }
    }else if (ipkmhs2>ipkmhs1 && ipkmhs2>ipkmhs3) {
      if (ipkmhs1>ipkmhs3){
            System.out.println(ipkmhs2 + "," + ipkmhs1 + "," + ipkmhs3);
      }else{
            System.out.println(ipkmhs2 + "," + ipkmhs3 + "," + ipkmhs1);
      }
    }else if (ipkmhs1>ipkmhs2) {
          System.out.println(ipkmhs3 + "," + ipkmhs1 + "," + ipkmhs2);
      }else{
            System.out.println(ipkmhs3 + "," + ipkmhs2 + "," + ipkmhs1);
      }
  }
      public static void main (String args[]) throws IOException{
      BufferedReader br = new BufferedReader (new InputStreamReader
(System.in));

      System.out.println("======Mengurutkan Ipk Mahasiswa======= ");
      System.out.print("Masukkan Nilai IPK Mahasiswa Pertama: ");
      ipkmhs1 = Double.parseDouble(br.readLine());
      System.out.print("Masukkan Nilai IPK Mahasiswa Kedua:  ");
      ipkmhs2 = Double.parseDouble(br.readLine());
      System.out.print("Masukkan Nilai IPK Mahasiswa Ketiga:  ");
      ipkmhs3 = Double.parseDouble(br.readLine());

      pengurutanMhs();
      }
}
```

11.3. Parameter Prosedur

Ketika sebuah prosedur dipanggil, maka pada dasarnya dapat melakukan proses pertukaran data antara program utama dan sub program. Proses pertukaran ini dapat dilakukan dengan penggunaan parameter. Pada prosedur terdapat parameter yaitu sebagai berikut:

1. Parameter aktual

   Parameter aktual merupakan parameter yang disertakan pada saat prosedur dipanggil untuk dijalankan pada program utama, atau dapat sering jenis ini sebagai argumen

   Contoh:

```
import java.io.*;
class PenjumlahanNilai{

public static void main (String args[]) throws IOException{

  BufferedReader br = new BufferedReader (new InputStreamReader
(System.in));

System.out.println("======Penjumlahan Nilai======= ");
  System.out.print("Masukkan Nilai Pertama: ");
  nilai1 = Double.parseDouble(br.readLine());
  System.out.print("Masukkan Nilai Kedua:  ");
  Nilai2 = Double.parseDouble(br.readLine());

  jumlahNilai(nilai1, nilai2);
  }
}
```

Pada contoh diatas merupakan proses pemanggilan prosedur pada kasus penjumlahan dua buah nilai. jumlahNilai() merupakan penerapan dari sebuah prosedur. Sementara itu, nilai1 dan nilai2 merupakan paramater aktual atau argumen pada prosedur jumlahNilai().

2. Parameter formal

   Parameter formal merupakan paramater yang dituliskan pada saat melakukan pendefinisian sebuah prosedur. Parameter ini ditulis pada bagian kode sub program untuk menerima nilai dari parameter aktual pada program utama. Terdapat tiga jenis parameter formal yang dapat digunakan pada prosedur, diantaranya:

   a. Parameter masukan (input)

      Parameter masukan merupakan parameter yang menerima nilai dari prameter aktual untuk dilakukan pemrosesan pada prosedur

      Contoh:

```
import java.io.*;
class PenjumlahNilai{

class PenjumlahanNilai{
     static void jumlahNilai(int bilangan1, int bilangan2){
           int hasiljumlah;
           hasiljumlah = bilangan1 + bilangan2;
           System.out.println("Hasil   Penjumlahan   adalah   "   +
hasiljumlah);
     }
}
```

Pada contoh yang ditampilkan diatas, bilangan1 dan bilangan2 merupakan parameter masukan yang menerima nilai dari parameter aktual dari program utama. Penamaan parameter pada parameter masukan tidak harus sama dengan parameter aktual yang terdapat pada program utama.

b. Parameter keluaran (output)

Parameter keluaran merupakan parameter yang digunakan untuk menampung hasil dari pemrosesan dari sebuah prosedur, selanjutnya hasilnya dikirimkan ke program utama program.

Contoh:

```
class ParamOutput{
     public static String cetak;
     static void tampilHasil(String tampil){
           tampil = "Mahasiswa Sistem Informasi";
           System.out.println(tampil);
     }
     public static void main (String args[]) {

           tampilHasil(cetak);
     }
}
```

Pada contoh yang ditampilkan diatas, cetak merupakan parameter keluaran yang dikirimkan ke prosedur tampilHasil() dan memiliki fungsi untuk menampung hasil dari pemrosesan prosedur tersebut. Hal yang perlu diperhatikan adalah pada parameter output harus berupa nama dan tidak perlu didefinisikan nilainya. Nilai dari parameter output akan dihasilkan oleh prosedur dan dapat dimanfaatkan pada instruksi kode berikutnya.

c. Parameter masukan dan keluaran (input / output)

Parameter yang menerima nilai dari parameter aktual untuk diproses dalam prosedur kemudian nilai yang dihasilkan dari pemrosesan tersebut ditampung ke dalam parameter output. parameter output yang dihasilkan oleh prosedur dapat dimanfaatkan pada instruksi kode berikutnya.

Contoh:

```
import java.io.*;
class PenjumlahanNilai{
     public static int a, b, hasil;
```

```
     static void jumlahNilai(int bilangan1, int bilangan2, int
hasiljumlah){
           hasiljumlah = bilangan1 + bilangan2;
           System.out.println("Hasil Penjumlahan adalah " +
hasiljumlah);
     }
     public static void main (String args[]) throws IOException{
           BufferedReader br = new BufferedReader (new
InputStreamReader (System.in));
           System.out.println("======Jumlah Nilai======= ");
           System.out.print("Masukkan Nilai Pertama: ");
           a = Integer.parseInt(br.readLine());
           System.out.print("Masukkan Nilai Kedua: ");
           b = Integer.parseInt(br.readLine());
           jumlahNilai(a,b,hasil);
     }
}
```

Pada contoh yang ditampilkan diatas, bilangan1, bilangan2, dan hasiljumlah merupakan parameter formal yang terdapat prosedur jumlahNilai(). bilangan1 dan bilangan2 merupakan parameter formal yang menerima nilai dari parameter aktual untuk dilakukan pemrosesan pada prosedur jumlahNilai(). Sedangkan untuk hasiljumlah merupakan parameter formal yang menampung hasil nilai dari pemrosesan prosedur.

11.4. Pemanggilan Parameter pada Prosedur

Banyaknya parameter tidak dibatasi jumlahnya, namun dalam impementasi prosedur yang baik, jumlah parameter dalam sebuah prosedur sebaiknya paling banyak adalah tujuh buah. Jika parameter terlalu banyak, maka sebaiknya ada pemisahan lagi menjadi beberapa prosedur (Rosa, 2022). Pemanggilan nama prosedur dapat dilakukan dengan menuliskan nama prosedur yang terdefinisikan. Pemanggilan terhadap prosedur dapat juga menambahkan nilai parameter aktual (argumen) untuk diproses pada prosedur yang menggunakan parameter formal. Urutan nilai parameter aktual (argumen) harus sama dengan dengan urutan pada parameter formal, karena type pada parameter aktual (argumen) dan parameter formal harus sama. Berikut ini merupakan contoh gambaran dari sebuah pseudocode untuk pemanggilan sebuah prosedur.

Contoh:

```
Algoritma: Prosedur hitungKali {Penerapan algoritma dari sebuah perkalian
pada dua buah bilangan}

PROCEDURE hitungKali (input bil1, bil2: integer)
Deklarasi
hasil : integer
Deskripsi
hasil ← bil1*bil2)
PRINT (hasil)

ALGORITMA UTAMA
Deklarasi
bilangan1, bilangan2: integer
Procedure hitungJumlah (input bil1, bil2: integer)
Deskripsi
Input(bilangan1, bilangan2)
hitungKali (bilangan1, bilangan2)

END
```

Pada pseudocode diatas, variabel bilangan1 dan bilangan2 merupakan sebuah parameter aktual dari sebuah prosedur hitungJumlah(). Pada saat pemanggilan prosedur, dapat dilakukan dengan cara menuliskan nama prosedur yang sudah didefinisikan dan menambahkan nilai parameter aktual untuk dikirimkan ke parameter formal. Tipe data parameter formal yang dikirimkan harus memiliki tipe yang sama dengan parameter formal. Jika jumlah parameter aktual lebih dari satu, maka urutan pada parameter aktual yang dikirimkan harus sama dengan urutan pada parameter formal yang menerimanya. Pada contoh diatas bil1 dan bil2 merupakan parameter formal untuk prosedur hitungKali().

Sementara itu, berikut dibawah ini merupakan hasil pengkodean bahasa java pada contoh diatas (prosedur .hitungKali()).

Contoh:

```
import java.io.*;
class Perkalian{
      public static int bilangan1, bilangan2;
      static void hitungKali(int bil1, int bil2){
            int hasil;
            hasil = bil1 * bil2;
            System.out.println("Hasil Perkalian adalah " + hasil);
      }
      public static void main (String args[]) throws IOException{
            BufferedReader    br    =    new    BufferedReader    (new
InputStreamReader (System.in));
            System.out.println("======Perkalian Nilai======= ");
            System.out.print("Masukkan Bilangan Pertama: ");
```

```
            bilangan1 = Integer.parseInt(br.readLine());
            System.out.print("Masukkan Bilangan Kedua: ");
            bilangan2 = Integer.parseInt(br.readLine());

            hitungKali(bilangan1, bilangan2);
      }
}
```

Parameter dapat diproses dengan menggunakan dua cara, yaitu pemrosesan parameter dengan melewatkan berdasarkan nilai (*pass by value*) dan pemrosesan parameter dengan melewatkan berdasarkan acuan (*pass by refferernce).*

11.4.1. Pemrosesan parameter berdasarkan nilai

Jenis ini merupakan pemrosesan parameter melalui penggunaan nilai parameter aktual untuk memanggil prosedur dengan cara menyalin nilainya ke parameter formal. Nilai dari parameter formal yang didapatkan, selanjutnya hanya dikenali dan dapat diproses di dalam prosedur yang bersangkutan saja. Jika terjadi perubahan nilai yang dihasilkan dari prosedur, maka tidak mempengarui nilai dari parameter aktual yang dikirimkan dari program utama. Pada contoh berikut ini merupakan contoh *pseudocode* pemrosesan parameter berdasarkan nilai.

Contoh:

```
Algoritma: ParamByValue {Algoritma melakukan perhitungan perkalian dengan
menggunakan dua nilai yang ditentukan}

PROSEDUR hitungNilai(x: integer, y: integer)
Deklarasi
Tidak ada
Deskripsi
x ← x*y)
PRINT ("Nilai x di dalam prosedur: ", x)

PROGRAM UTAMA
Deklarasi
x, y: integer
Deskripsi
x ← 4;
y ← 5;
PRINT ("Nilai x sebelum prosedur: ", x)
Procedure hitungNilai(x,y)
PRINT ("Nilai x setelah prosedur: ", x)

END
```

Berdasarkan algoritma diatas pada program utama nilai x diisi dengan 4 dan nilai y diisi dengan 5, maka hasil keluarannya adalah sebagai berikut:

```
Nilai x sebelum prosedur: 4
Nilai x didalam prosedur: 20
Nilai x setelah prosedur: 4
```

Sementara itu, berikut ini merupakan hasil dari algoritma *pseudocode* jika dijalankan pada bahasa pemrograman java.

```
class ParamByValue{
      public static int x, y;
      static void hitungNilai(int x, int y){
            x = x*y;
            System.out.println("Nilai x dalam prosedur: " + x);
      }
      public static void main (String args[]){
            x = 4;
            y = 5;
            System.out.println("Nilai x sebelum prosedur dijalankan: "+ x );
            tampilHasil(x, y);
            System.out.println("Nilai x setelah prosedur dijalankan: "+ x );
      }
}
```

Pada hasil yang ditampilkan pada bahasa pemrograman java diatas menunjukkan bahwa nilai dari x sebelum dijalankan prosedur memiliki nilai 4, sedangkan nilai x pada saat prosedur dijalankan menghasilkan nilai 20. Walaupun nilai x dieksekusi di dalam prosedur dan terdapat perubahan nilai, maka nilai tidak mempengaruhi nilai pada variabel nilai x pada parameter aktual yang dikirimkan dari program utama.

11.4.2. Pemrosesan parameter berdasarkan acuan

Jenis ini merupakan pemrosesan parameter melalui penggunaan acuan dari parameter aktual (bukan nilai dari argumen) untuk menjalankan prosedur dengan memasukkan ke parameter formal. Dengan kata lain, parameter acuan ini mengacu pada variabel yang dijadikan sebagai parameter aktual dalam menjalankan prosedur. Nilai pada parameter formal yang dijalankan pada prosedur akan sama dengan nilai dari variabel yang ditunjuk sebagai acuan. Sehingga, jika di dalam prosedur nilainya terjadi perubahan, maka akan berpengaruh terhadap nilai aktual

(argumen) pada variabel yang diacu. Pada contoh berikut ini merupakan contoh *pseudocode* pemrosesan parameter berdasarkan acuan.

```
Algoritma: ParamByRefference {Algoritma melakukan perhitungan penjumlahn
dengan menggunakan dua nilai yang ditentukan}

PROSEDUR hitungNilai(arr: Array[0,1] of integer)
Deklarasi
Tidak ada
Deskripsi
arr[0] = arr[0] + arr[1]
PRINT ("Nilai indeks pertama di dalam prosedur: ", arr[0])

PROGRAM UTAMA
Deklarasi
a: Array[0,1] of integer
Deskripsi
a ← {12, 15};
PRINT ("Nilai indeks pertama sebelum prosedur: ", a[0])
Procedure hitungNilai(a)
PRINT ("Nilai indeks pertama setelah prosedur: ", a[0])

END
```

Berdasarkan algoritma diatas pada program utama nilai a merupakan sebuah array diisi dengan nilai 12 dan 15. Jika algoritma diatas dijalankan maka menghasilkan:

```
Nilai indeks pertama sebelum prosedur: 12
Nilai indeks pertama di dalam prosedur: 27
Nilai indeks pertama setelah prosedur:27
```

Sementara itu, berikut ini merupakan hasil dari algoritma *pseudocode* jika dijalankan pada bahasa pemrograman java.

```
class ParamByRefference {
	public static int[] a;
	static void tampilHasil(int arr[]){
		arr[0] = arr[0] + arr[1];
		System.out.println("Nilai indeks pertama di dalam prosedur " +
arr[0]);
	}
	public static void main (String args[]){
		int[] a = {12, 15};
		System.out.println("Nilai indeks ke1 sebelum prosedur " + a[0]);
		tampilHasil(a);
		System.out.println("Nilai indeks ke1 setelah prosedur " + a[0]);
	}
}
```

Pada hasil yang ditampilkan pada bahasa pemrograman java diatas menunjukkan bahwa nilai indeks pertama sebelum dijalankan prosedur memiliki nilai 12, sedangkan nilai indeks pertama pada saat dan setelah prosedur dijalankan masing masing menghasilkan nilai yang sama yaitu 27. Nilai indeks array pertama pada program diatas terjadi perubahan karena parameter dijalankan dengan mengacu variabel yang dijadikan sebagai parameter aktual. Sehingga nilai dari acuan mengalami perubahan di dalam prosedur, maka nilai parameter aktual juga megalami perubahan.

Di dalam bahasa pemrograman java, tidak menyebutkan apakah pemrosesan parameter dilewatkan berdasarkan nilai atau referensinya. Bahasa java melewatkan parameter bergantung pada tipe argumen yang digunakan untuk melakukan pemanggilan prosedur. Apabila argumen yang diberikan memiliki tipe sederhana/primitif (integer, char, varchar, boolan, dll), maka bahasa java akan menggunakan cara *pass by value*. Namun apabila argumen yang diberikan berupa tipe objek atau array, maka java akan menggunakan metode *pass by refference*.

11.5. Contoh Implementasi Prosedur

Penerapan prosedur dapat digunakan untuk menyelesaikan beberapa aktifitas dalam kehidupan sehari hari. Beberapa contohnya adalah sebagai berikut:

Contoh Implementasi 1: Diskon Koperasi Mahasiswa

Koperasi di sebuah kampus menjual beberapa kebutuhan peralatan mahasiswa. pada bulan september koperasi memberikan potongan bagi mahasiswa yang telah menjadi member maupun belum menjadi member. Ketentuan diskon tersebut adalah sebagai berikut:

- Jika mahasiswa memiliki kartu member
  - ✓ Jika berbelanja lebih sama dengan dari 100.000 dapat diskon 5.000
  - ✓ Jika berbelanja lebih sama dengan dari 50.000 dapat diskon 2.000
  - ✓ Kurang dari 50.000 tidak mendapatkan diskon
- Jika mahasiswa tidak memiliki member
  - ✓ Jika berbelanja lebih dari sama dengan 50.000 dapat diskon 1.000

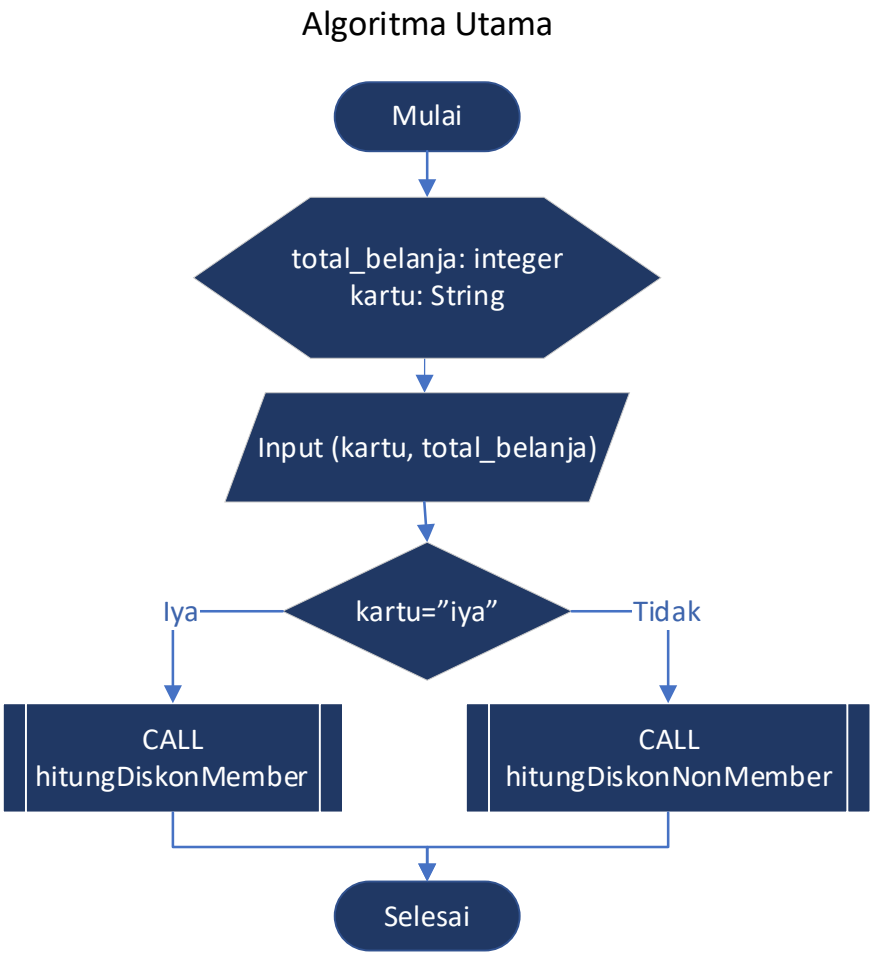
✓ Kurang dari 50.000 tidak mendapatkan diskon

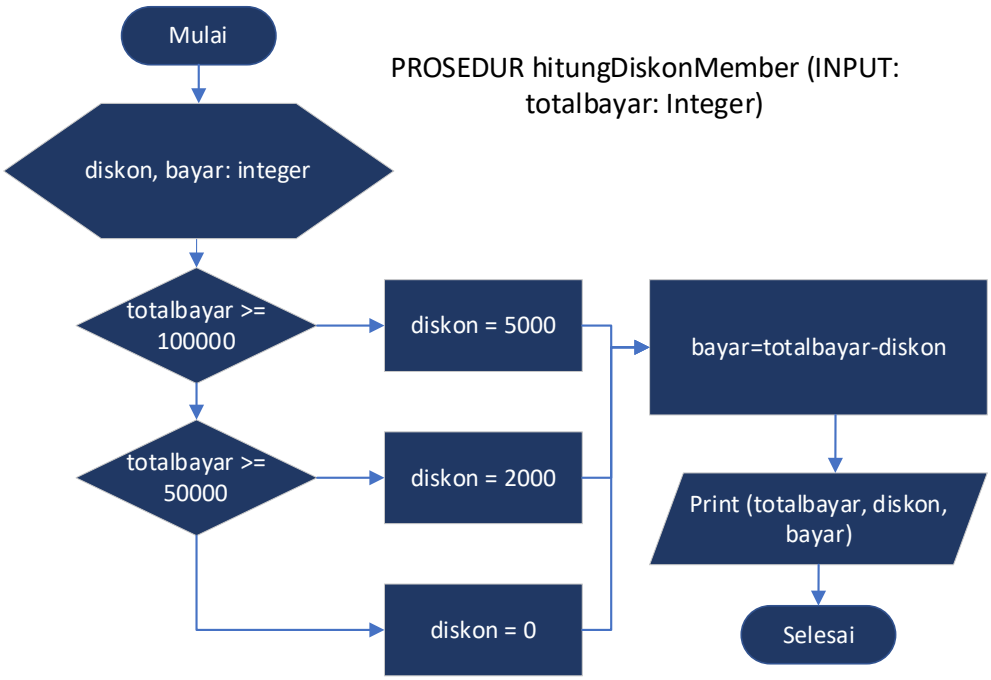
Proses perhitungan terhadap diskon dapat disimpan di dalam prosedur berupa perhitungan nilai, kemudian prosedur dapat dipanggil berdasarkan parameter jumlah belanja yang dikeluarkan.

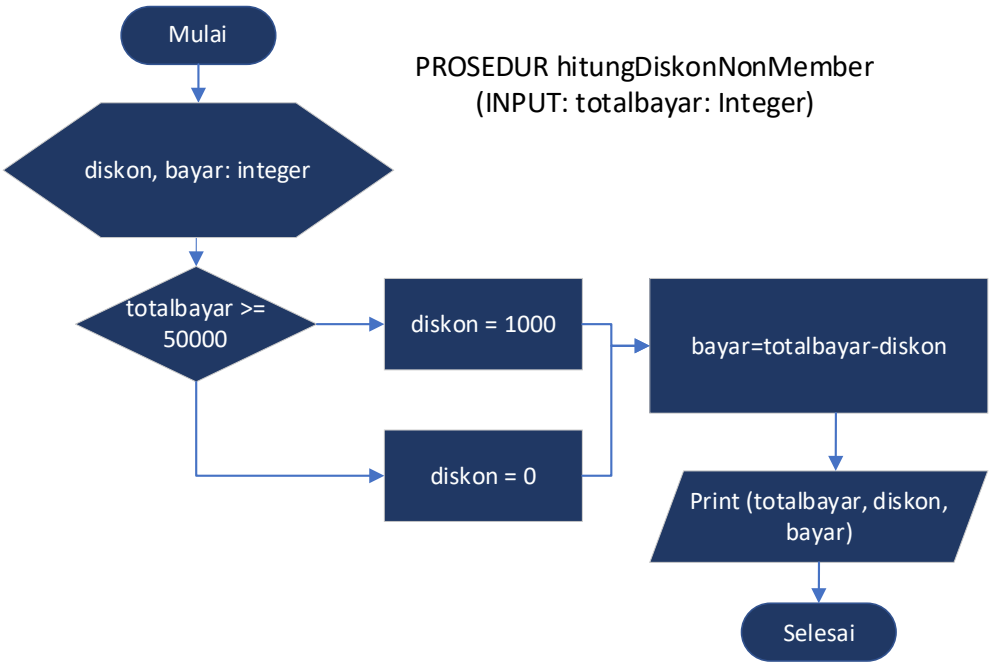
Berdasarkan kasus diatas, maka dapat ditunjukkan hasil penerapan algoritma berupa *flowchart*, *pseudocode*, dan bahasa pemrograman java.

a. *Flowchart* contoh kasus Diskon Koperasi Mahasiswa

Berikut ini adalah Flowchart untuk kasus pemberian diskon pada koperasi mahasiswa.







b. *Pseudocode* contoh kasus Diskon Koperasi Mahasiswa

Berikut ini adalah *Pseudocode* untuk kasus pemberian diskon pada koperasi mahasiswa.

```
Algoritma: Diskon Koperasi Mahasiswa {Algoritma melakukan perhitungan
diskon bagi mahasiswa yang memiliki member maupun non member. }

PROSEDUR hitungDiskonMember(totalbayar: integer)
Deklarasi
diskon, bayar: integer
Deskripsi
if (totalbayar >= 100000) then diskon ← 5000
   else if (totalbayar >= 50000) then diskon ← 2000
   else diskon ← 0
bayar ← totalbayar - diskon
PRINT "Total Belanja: Rp ", totalbayar)
PRINT "Diskon: Rp ", diskon)
PRINT "Total Bayar: Rp ", bayar)

PROSEDUR hitungDiskonNonMember(totalbayar: integer)
Deklarasi
diskon, bayar: integer
Deskripsi
if (totalbayar >= 50000) then diskon ← 1000
   else diskon ← 0
bayar ← totalbayar - diskon
PRINT "Total Belanja: Rp ", totalbayar)
PRINT "Diskon: Rp ", diskon)
PRINT "Total Bayar: Rp ", bayar)

PROGRAM UTAMA
Deklarasi
total_belanja: integer
kartu: String
Deskripsi
Input(kartu)
Input(total_belanja)

if (kartu ← "iya") then hitungDiskonMember(total_belanja)
else hitungDiskonNonMember(total_belanja)

END
```

c. Bahasa pemrograman java contoh kasus Diskon Koperasi Mahasiswa

Berikut ini adalah hasil implementasi pada bahasa pemrograan java untuk kasus pemberian diskon pada koperasi mahasiswa.

```
import java.util.Scanner;
class Kasusprosedur{
   static int total_belanja;
   static String kartu;
   static void hitungDiskonMember(int totalbayar){
         int diskon, bayar;
         if (totalbayar >= 100000) {
                diskon = 5000;
            } else if (totalbayar >= 50000) {
                diskon = 2000;
            } else {
```

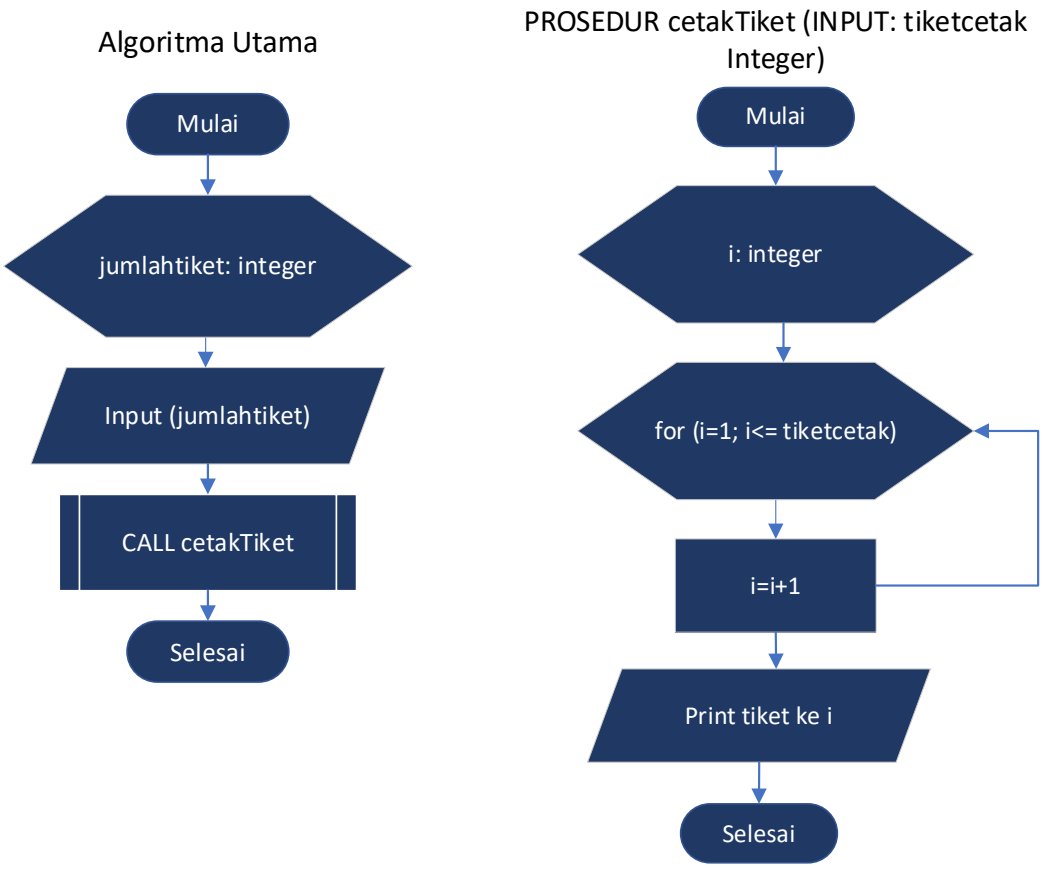
```
                diskon = 0;
            }
         bayar = total_belanja - diskon;
         System.out.println("-----------------------------------");
         System.out.println("Total Belanja: Rp " + totalbayar);
         System.out.println("Diskon (Rp): Rp " + diskon);
         System.out.println("Total Bayar: Rp " + bayar);
   }
   static void hitungDiskonNonMember(int totalbayar){
         int diskon, bayar;

         if (totalbayar >= 50000) {
            diskon = 1000;
        } else {
            diskon = 0;
        }
         bayar = total_belanja - diskon;
         System.out.println("-----------------------------------");
         System.out.println("Total Belanja: Rp " + totalbayar);
         System.out.println("Diskon (Rp): Rp " + diskon);
         System.out.println("Total Bayar: Rp " + bayar);

   }

   public static void main(String args[]){
   Scanner inputan = new Scanner(System.in);
   System.out.print("Apakah Punya Kartu Member (iya/tidak) ? ");
         kartu = inputan.nextLine();
         System.out.print("Total Belanja: ");
         total_belanja = inputan.nextInt();
       if (kartu.equalsIgnoreCase("iya")) {
         hitungDiskonMember(total_belanja);
       } else {
           hitungDiskonNonMember(total_belanja);
       }
   }
}
```

Contoh Implementasi 2: Mencetak tiket masuk kawasan wisata

Salah satu tempat wisata di indonesia ingin menerapkan sistem berupa mencetak tiket masuk kawasan wisata berdasarkan jumlah jumlah orang yang akan masuk. Proses mencetak tiket dapat dibuat berupa prosedur dimana parameter yang digunakan berupa jumlah tiket/orang yang menerima tiket tersebut.

Berdasarkan kasus diatas, maka dapat ditunjukkan hasil penerapan algoritma berupa *flowchart*, *pseudocode*, dan bahasa pemrograman java.

a. *Flowchart* contoh kasus mencetak tiket kawasan wisata



b. *Pseudocode* contoh kasus mencetak tiket kawasan wisata

```
Algoritma: Cetak Tiket Masuk Kawasan Wisata {Algoritma melakukan cetak
tiket berdasarkan jumlah tiket/orang yang dimasukkan oleh user. }

PROSEDUR cetakTiket(tiketcetak: integer)
Deklarasi
i: integer d
Deskripsi
for i←1 to tiketcetak do
   PRINT "Tiket Cetak ke: ", i
endfor

PROGRAM UTAMA
Deklarasi
jumlahtiket: integer
Deskripsi
Read(jumlahtiket)
cetakTiket(jumlahtiket)

END
```

c. Bahasa pemrograman java contoh kasus mencetak tiket kawasan wisata

```
import java.util.Scanner;
class Kasusprosedur{
   static int jumlahtiket;
   static void cetaktiket(int tiketcetak){
         int i;
         for(i=1; i<=tiketcetak; i++){
               System.out.println("Tiket Cetak ke: " +i);
         }
   }
   public static void main(String args[]){
   Scanner inputan = new Scanner(System.in);
   System.out.print("Jumlah Tiket yang Ingin di Cetak ? " );
    jumlahtiket = inputan.nextInt();
    cetaktiket(jumlahtiket);
   }
}
```

POST TEST

Soal Latihan tentang Materi Prosedur

1. Sebuah koperasi menerapkan gaji bagi pegawainya dengan beberapa ketentuan. Gaji pokok seorang pegawai memiliki ketergantungan waktu awal masuk kerja. Uang makan yang diberikan kepada pegawai sebesar Rp.30.000 per hari. Selain uang makan pegawai menerima uang transport sebesar 40.000 per hari. Bagi pegawai yang lembur diberikan uang sebanyak 100.000 per jam, pad saat 2 jam pertama. Jika pegawai melakukan lembur diatas 3 jam, maka uang lembur menjadi 200.000 per jam. Jika disimpulkan, maka total gaji yang diterima pegawai koperasi adalah sebagi berikut:

   Total Gaji = Gaji pokok + Tunjangan Jabatan + uang makan + uang transport dan uang lembur.

   Sementara itu tunjangan jabatan bagi pegawai berdaasarkan jabatannya adalah sebagai berikut:

Staff	: 250.000
Sekeretais	: 500.000
Kepala Koperasi	: 750.000
Manager	: 1.000.000

   Berdasarkan studi kasus diatas, maka tentukan lah penulisan algorima berupa flowchart, pseudocode, dan panerapan implmentasi pada java (gunakan konsep prosedur.

2. Buatlah flowchart, pseudocode, dan panerapan implementasi pada java (gunakan konsep prosedur) pada deret bilangan Fibonacci sesuai dengan nilai inputan yang dimasukkan