BAB 6

STRUKTUR ALGORITMA PEMGROGRAMAN : PERULANGAN

BAB 6
STRUKTUR ALGORITMA PEMROGRAMAN : PERULANGAN

Capaian Pembelajaran

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

Pokok Bahasan

1. Pengertian Struktur Perulangan
2. Perulangan dengan Struktur For
3. Perulangan dengan Struktur While
4. Perulangan dengan Struktur Do-While
5. Nested Loop (Perulangan Bersarang)
6. Implementasi Struktur Percabangan dalam Algoritma Pemrograman

Evaluasi Pembelajaran

Soal Latihan Struktur Perulangan Algoritma Pemrograman

Pre Test
Struktur Perulangan Algoritma Pemrograman

1. Apa yang anda ketahui tentang Konsep Struktur Perulangan dalam Bahasa Pemrograman?
2. Sebutkan yang anda ketahui tentang jenis struktur perulangan yang biasanya digunakan dalam Bahasa Pemrograman?
3. Apa yang dimaksud Inisialisasi Awal dalam perulangan?
4. Apa yang dimaksud dengan Increment dan Decrement?
5. Jelaskan perbedaan penggunaan antara Struktur Perulangan While dan Do-While !

Dalam kehidupan sehari-hari, kita sering melakukan kegiatan yang bersifat berulang, misalnya melakukan gerakan olahraga yang sederhana seperti push up. Kegiatan tersebut dilakukan secara berulang-ulang dan berhenti sampai waktu yang ditentukan ataupun berhenti sesuai dengan keinginan. Dalam pemrograman komputer, pada kondisi yang memerlukan iterasi atau perulangan terdapat metode yang dapat digunakan untuk dapat dilakukan secara sistematis sehingga penulisan sintaks tidak perlu dilakukan sebanyak yang dibutuhkan. Misalnya kita ingin menampilkan statement "Saya Belajar Algoritma dan Pemrograman" sebanyak 10 Kali, mungkin kita masih bisa menuliskan statement tersebut sebanyak itu. Namun jika statement tersebut harus ditampilkan sebanyak 1000 kali, tentu kita tidak mungkin melakukannya dengan cara manual. Bab ini akan mempelajari bagaimana kita dapat membuat sebuah pengkodean dalam pemrograman komputer untuk dapat membuat perintah yang dapat menjalankan statement secara berulang-ulang dengan pengkodingan yang lebih efektif dan sederhana, baik pengkondisian perulangan yang sengaja ditentukan maupun kondisi yang bergantung pada suatu aturan tertentu.

5.1. Pengertian Struktur Perulangan

Struktur perulangan atau disebut loop adalah instruksi kode program yang bertujuan untuk mengulang beberapa baris perintah secara berulang-ulang selama kondisi masih memenuhi.

Secara umum, struktur perulangan terdiri dari empat bagian penting yaitu:

- Inisialisasi

  Inisialisasi merupakan sebuah variabel yang memiliki nilai untuk mendeskripsikan status nilai awal yang menjadi awalan dalam proses perulangan.
- Kondisi perulangan

  Kondisi ini merupakan bentuk ekspresi yang bernilai Boolean yang harus dipenuhi untuk menjalankan perulangan. Kondisi ini bisa dilakukan

dengan cara mengatur sendiri jumlah perulangan yang dibutuhkan atau dapat mengatur sesuai kondisi tertentu sesuai dengan kebutuhan.

- Badan (body) perulangan

  Badan (body) dalam perluangan merupakan sekumpulan statement yang dilakukan proses perulangan selama kondisi memenuhi.

- Terminasi

  Terminasi merupakan statement yang dilakukan setelah proses perulangan dilakukan.

Di dalam algoritma terdapat beberapa jenis struktur perulangan yang dapat digunakan. Jenis perulangan tersebut memiliki karakteristik yang berbeda dalam pemanfaatannya tergantung pada persoalan yang akan diprogram. Perulangan dibagi menjadi dua bagian :

- Counted Loop : (for/for-each) perulangan yang sudah ditentukan banyaknya perulangan.
- Uncounted Loop : (while, do-while) perulangan yang belum jelas berapa kali harus mengulang.

Dari beberapa jenis perulangan diatas, kita perlu melakukan analisa dalam penggunaan jenis perulangan pada setiap kasusnya, sehingga kita memerlukan pemahaman terkait karakteristik penggunaan pada setiap jenis perulangan yang ada beserta penggunaan operator-operator yang dapat dimanfaatkan untuk menunjang dalam proses penyelesaian kasus yang akan kita selesaikan.

5.2. Perulangan dengan Struktur For

Dalam merancang struktur perulangan for, setidaknya kita harus mengetahui tiga komponen yaitu (1) inisialisasi awal perulangan, komponen ini merupakan sebuah variabel untuk mendeskripsikan sebuah nilai awal dalam proses perulangan. Misalnya kita ingin menampilkan angka 1-10, maka variabel awal kita berikan nilai i=1. (2) kondisi untuk dilakukan perulangan, komponen ini merupakan sebuah instruksi proses perulangan yang berfungsi untuk melakukan pengecekan

memenuhi tidaknya sebuah aksi yang akan diproses dan sebagai nilai variabel agar perulangan berhenti. Dalam kasus menampilkan angka 1-10 diatas, maka kondisi yang perlu kita tulis seperti ini i≤10. (3) penggunaan operator untuk melakukan perulangan yang bersifat menambahkan atau menurunkan data yang akan kita tampilkan. Perulangan FOR ini digunakan ketika jumlah yang akan kita lakukan pada perulangan diketahui. Berikut ini adalah bentuk perulangan FOR.

```
for (exp1; exp2; exp3){
    Statement;
}
```

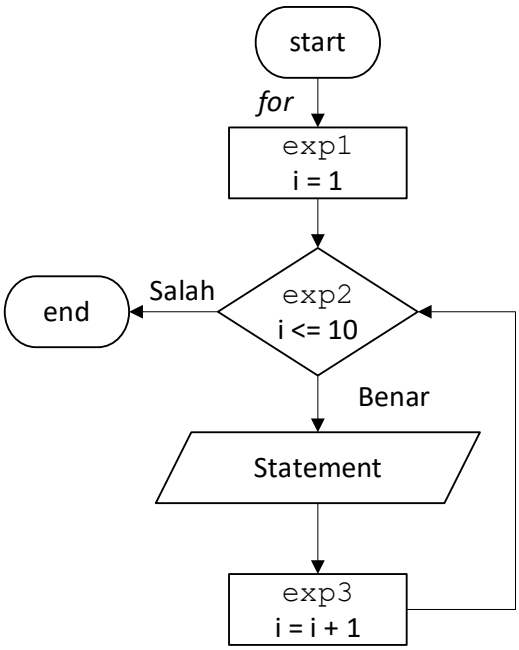
Keterangan :

exp1 : Expresi untuk inisialisasi nilai awal

exp2 : Expresi untuk kondisi

exp3 : Expresi untuk increment (penambahan) atau decrement (pengurangan)

Diagram alir pada Gambar berikut dapat membantu untuk memperjelas proses perulangan dengan menggunakan struktur FOR.



Algoritma berdasarkan Diagram Alir Perulangan Struktur FOR diatas :

1. Mulai
2. Inisialisasi nilai awal i =1
3. Cek kondisi apakah nilai i <=10 jika kondisi bernilai benar maka cetak kalimat "Statement"
4. Nilai i dilakukan increment (ditambahkan 1)
5. Cek kondisi kembali apakah nilai i <=10 bila kondisi bernilai benar (true) maka cetak kembali kelimat "Statement"
6. Iterasi/perulangan terus dilakukan hingga konisi bernilai salah (false)
7. Selesai

Berikut ini contoh implementasi penerapan perulangan struktur FOR yang sederhana dalam bentuk algoritma (pseudocode) dan ditranslasikan kedalam Bahasa pemrograman Java.

Contoh 1.

```
Algoritma CetakText
{Mencetak Text sebanyak N kali}

DEKLARASI
n : integer {jumlah pengulangan}
i : integer

DESKRIPSI :
read(i, n)
for i←1 to n do
      write("Algoritma dan Pemrograman")
endfor
```

Translasi Algoritma Cetak Text ke pemrograman Java

```
import java.util.Scanner;
public class CetakText {
    public static void main(String[] args) {
        int i, jumlah;

        Scanner inputan = new Scanner(System.in);
        System.out.print("Masukkan jumlah Text: ");
        jumlah = inputan.nextInt();

        for(i=1; i<=jumlah; i++){
            System.out.println("Algoritma dan Pemrograman");
        }
    }
}
```

Output :

```
C:\Windows\System32\cmd.exe
Masukkan jumlah Text: 5
Algoritma dan Pemrograman
Algoritma dan Pemrograman
Algoritma dan Pemrograman
Algoritma dan Pemrograman
Algoritma dan Pemrograman
```

Proses pada kasus untuk mencetak text ini dilakukan perulangan sebanyak 5 kali (sesuai dengan nilai yang dimasukkan) dimulai dari nilai variabel yang sudah diinisialisasi yaitu i=1 dan dengan kondisi <= 5 (jumlah) sehingga proses memenuhi akan tetap diulang sampai nilai i memiliki nilai 5. Selain itu pada perulangan for ini terdapat operator increment (i++) hal ini menyatakan bahwa perulangan dilakukan secara ascending. Hal tersebut dapat dibuktikan jika variabel i dicetak maka akan bernilai 1, 2, 3, 4, dan 5.

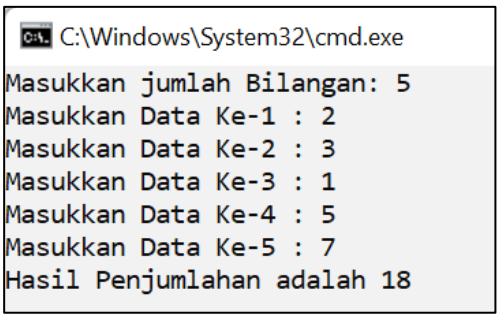
Contoh 2.

```
Algoritma Penjumlahan
{Melakukan operasi penjumlahan dari beberapa inputan nilai}

DEKLARASI
jumlah : integer {jumlah bilangan yang ingin dijumlahkan}
data : integer {bilangan yang dijumlahkan}
i, hasil : integer

DESKRIPSI :
read(i, jumlah)
for i←1 to jumlah do
read(data, hasil)
hasil ← hasil+data
endfor
      write (hasil)
```

Translasi Algoritma Penjumlahan ke pemrograman Java

```
import java.util.Scanner;
public class Penjumlahan {
    public static void main(String[] args) {
        int i, data, jumlah, hasil = 0;

        Scanner inputan = new Scanner(System.in);
        System.out.print("Masukkan jumlah Bilangan: ");
        jumlah = inputan.nextInt();

        for(i=1; i<=jumlah; i++){
            System.out.print("Masukkan Data Ke-" + i + " : ");
            data = inputan.nextInt();
            hasil = hasil + data;
        }
        System.out.println("Hasil Penjumlahan adalah " + hasil);
    }
}
```

Output :



Proses perulangan diatas menunjukkan bahwa perulangan dapat melakukan aksi yang bersifat operasi dalam kasus ini adalah operasi penjumlahan. Operasi ini dilakukan pada saat proses perulangan berlangsung. Hal ini menunjukkan bahwa variabel dapat melakukan penyimpanan data secara dinamis pada setiap proses perulangannya.

Pada beberapa pemrograman khususnya pada pemrograman java, struktur FOR memiliki dua type penulisan yang berbeda. Dua tipe ini mempunyai fungsi yang sama yaitu untuk melakukan fungsi perulangan. Struktur tersebut yaitu FOR-Each. Struktur ini digunakan untuk mencetak sebuah data array yang lebih simple dan praktis jika dibandingkan dengan menggunakan Struktur FOR biasa. Berikut ini bentuk perulangan struktur FOR-Each :

```
for (exp1 : array){
    Statement;
}
```

Keterangan :

exp1 = Expresi untuk inisialisasi nilai awal

array = Type data array.

Berikut ini contoh implementasi penerapan perulangan struktur FOR-Each yang sederhana dalam bentuk algoritma (pseudocode) dan ditranslasikan kedalam Bahasa pemrograman Java.

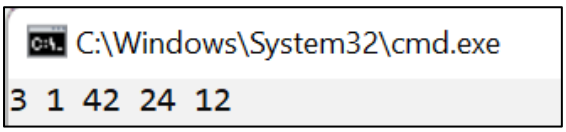
Contoh 3.

```
Algoritma PerulanganForeach
{Menampilkan data berbentuk dengan type array}

DEKLARASI
angka : array [3,1,42,24,12] of integer
i : integer

DESKRIPSI :
read(i, angka)
for i : angka
write(i)
endfor
```

Translasi Algoritma Perulangan Foreach ke pemrograman Java

```
import java.util.Scanner;
public class PerulanganForeach {
    public static void main(String[] args) {
        int angka[] = {3,1,42,24,12};

        for (int x : angka){
            System.out.print(x + " ");
        }
    }
}
```

Output :



Contoh kasus diatas merupakan perulangan for-each (for-each loop) yang merupakan perulangan yang berfungsi untuk mengakses seluruh elemen dari data array. Jika pada perulangan FOR, untuk mengakses data dibutuhkan inisialiasi awal,

kondisi, dan operator increment/decrement, sedangkan pada struktur For Each ini hanya memerlukan inisialisasi awal dan nama varibel dari data array.

5.3. Perulangan dengan Struktur WHILE

Perulangan ini digunakan untuk mengulang suatu proses perulangan yang belum diketahui jumlahnya. Pada perulangan WHILE pengecekan kondisi akan dilakukan terlebih dahulu. Jika kondisi masih bernilai benar (true) maka perulangan akan terus dilakukan. Sebaliknya jika bernilai salah maka akan dihentikan. Kita tahu bahwa program yang efisien adalah program yang memungkinkan pengguna bekerja sedikit mungkin dan komputer bekerja sebanyak mungkin. Struktur while ini juga memiliki tiga komponen yang mengendalikannya yaitu Inisialisasi, Jumlah Iterasi dan Kondisi Berhenti. Berikut ini bentuk perulangan struktur WHILE :

```
exp1;
while(exp2)
{
Statement;
     exp3;
}
```

Keterangan :

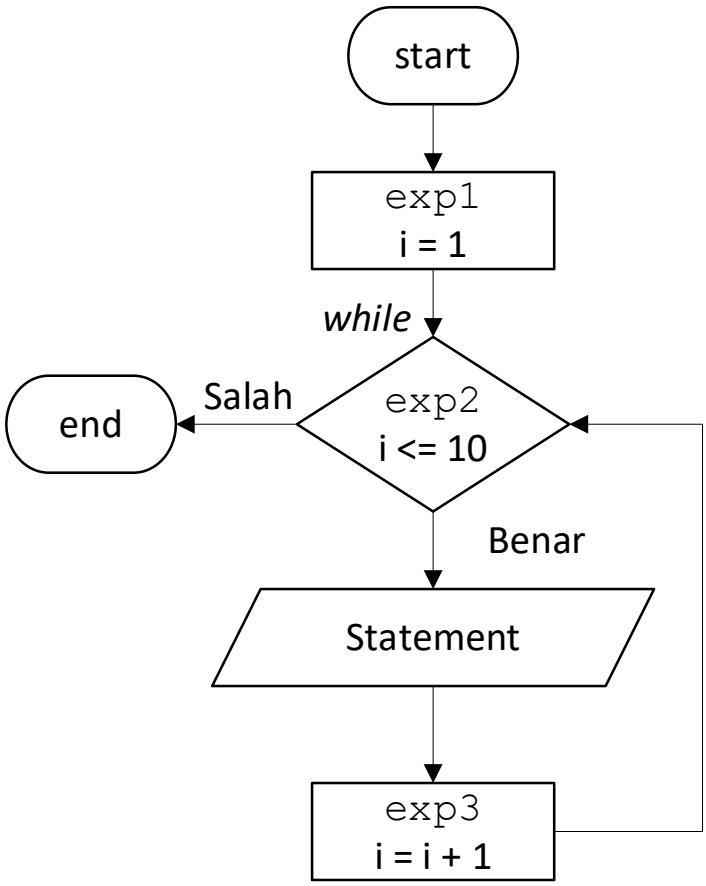
exp1 = Expresi untuk inisialisasi nilai awal

exp2 = Expresi untuk kondisi

exp3 = Expresi untuk increment (penambahan) atau decrement (pengurangan)

Bentuk perulangan pada algoritma diatas menunjukkan bahwa proses perulangan yang dilakukan dengan struktur while. Sebelum memasuki bagian body perulangan, struktur while jika kondisi memenuhi, maka proses perulangan akan selalu dilakukan, namun jika sudah tidak memenuhi, maka proses perulangan selesai/berhenti. Perlu diperhatikan bahwa perulangan perlu diberhentikan, jika perulangan tidak berhenti menandakan bahwa terdapat logika algoritma yang

salah. Karena perulangan akan berhenti jika kondisi bernilai salah (false). Berikut ini diagram alir pada struktur while yang ditunjukkan pada Gambar dibawah ini.



Algoritma berdasarkan Diagram Alir pada struktur While :

1. Mulai
2. Inisialisasi nilai awal i =1
3. Cek kondisi apakah i <= 10 jika ya maka cetak output "Statement"
4. Nilai i ditambahkan 1
5. Cek kondisi kembali apakah i <=10 jika benar maka kembali mencetak dan nilai i ditambahkan 1
6. Iterasi terus dilakukan hingga kondisi bernilai salah (false)

Berikut ini contoh implementasi penerapan perulangan struktur while yang sederhana dalam bentuk algoritma (pseudocode) dan ditranslasikan kedalam Bahasa pemrograman Java.

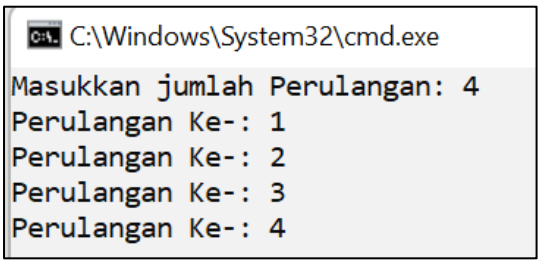
Contoh 4.

```
Algoritma CetakNilaiVariabel
{Menampilkan nilai variabel sebanyak N inputan}

DEKLARASI
jumlah : integer {jumlah perulangan}
i : integer

DESKRIPSI :
read(i, jumlah)
i←1
while i ≤ jumlah do
write(i)
i←i+1
endwhile
```

Translasi Algoritma Cetak Nilai Varibel ke pemrograman Java

```
import java.util.Scanner;
public class CetakNilaiVariabel {
    public static void main(String[] args) {
        int i, jumlah;

        Scanner inputan = new Scanner(System.in);
        System.out.print("Masukkan jumlah Perulangan: ");
        jumlah = inputan.nextInt();
        i=1;
        while (i<=jumlah){
            System.out.println("Perulangan Ke-: " + i);
            i++;
        }
    }
}
```

Output :



Contoh diatas merupakan untuk memastikan variabel inisialiasi memiliki nilai perulangan yang sesuai dengan jumlah kondisi perulangan yang diinginkan. Struktur while pada expresi incremennya (i++) dilakukan setelah statemens. Jika kita bandingkan dengan for proses incremennya berada diatas statemennya. Hal ini menunjukkan bahwa struktur for pada proses perulangannya sudah ditentukan banyaknya perulangan. Sedangkan pada struktur while berada setelah statemennya yang berari proses perulangannya berdasarkan selama kondisi memenuhi.

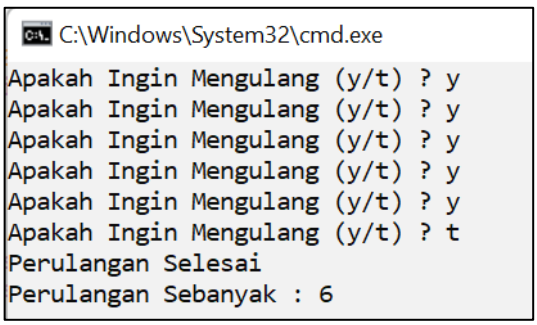
Contoh 5.

```
Algoritma PilihUlang
{Menentukan apakah proses ingin diulang apa diselesaikan dan menampilkan
jumlah perulangannya}

DEKLARASI
ulang : char {ulang = 'y'}
i : integer

DESKRIPSI :
read(i)
i←0
while ulang = 'y' do
read (ulang)
i←i+1
endwhile
write(i)
```

Translasi Algoritma Pilih Ulang ke pemrograman Java

```
import java.util.Scanner;
public class PilihUlang {
    public static void main(String[] args) {
        int i;
        char ulang = 'y';

        i=0;
        while (ulang == 'y'){
            Scanner inputan = new Scanner(System.in);
            System.out.print("Apakah Ingin Mengulang (y/t) ? ");
            ulang = inputan.next().charAt(0);
            i++;
        }
        System.out.println("Perulangan Selesai");
        System.out.println("Perulangan Sebanyak : " + i);
    }
}
```

Output :



Pada contoh kasus ini jumlah perulangan dapat ditentukan sendiri. Kondisi yang digunakan adalah dengan menggunakan variabel dengan type data char. Variabel ini sebagai parameter apakah perulangannya ingin dilanjutkan atau diselesaikan. Pada sumber kode diatas bahwa inisialiasi dimulai dari 0 (i = 0) sedangkan output yang dihasilkan menghasilkan 6 kali perulangan. Hal ini dikarenakan adanya proses increment dimana nilai yang awalnya 0 akan dilakukan

penambahan (0=0+1) sehingga outputnya menjadi 1. Jika diawal kita buat inisial i=1 maka ketika kita cetak nilai variabel tersebut maka meskipun perulangannya hanya satu kali namun nilai dari variabel i ini akan bernilai 2

5.4. Perulnagan degan Struktur Do-While

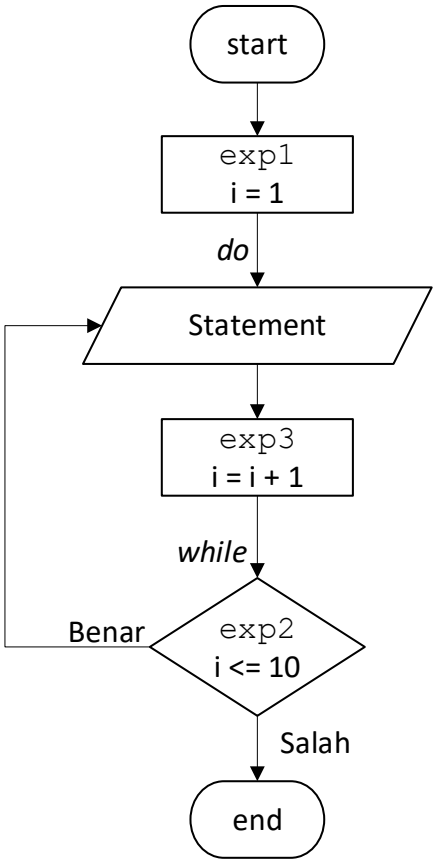
Perulangan ini hampir sama dengan perulangan dengan struktur while. Perbedaan dari keduanya adalah jika pada perulangan while kondisi sebagai syarat perulangan yang akan dilakukan dilakukan pengecekan pada awal sebelum perulangan dilakukan, jika bernilai benar (true) baru kemudian perulangan dijalankan. Sedangkan pada perulangan Do-While perulangan dilakukan terlebih dahulu minimal sekali baru kemudian setelah itu melakukan pengecekan sesuai dengan kondisinya. Berikut bentuk perulangan struktur DO-WHILE :

```
exp1;
do{
     Statement;
     exp3;
}while(exp2)
```

Keterangan :
exp1 = Expresi untuk inisialisasi nilai awal
exp2 = Expresi untuk kondisi
exp3 = Expresi untuk increment (penambahan) atau decrement (pengurangan)

Sama seperti perulangan while, dibagian start biasanya terdapat perintah inisialisasi variabel counter, misalnya i = 0. Kemudian di dalam bagian do ditulis kode program yang akan di ulang, tidak lupa sebuah perintah untuk menaikkan nilai variabel counter, misalnya dengan perintah i++. Di bagian paling bawah, terdapat perintah while (condition). Di sinilah kondisi perulangan akan diperiksa. Selama kondisi ini menghasilkan nilai true, maka perulangan akan lanjut ke iterasi

berikutnya. Diagram alir pada gambar berikut ini menggambarkan proses perulangan dengan do-while.



Algoritma berdasarkan Diagram Alir Struktur Perulangan Do-While :

1. Mulai
2. Inisialisasi nilai awal i =1
3. Cetak output "Statement"
4. Nilai i ditambahkan 1
5. Cek kondisi apakah nilai i <= 10, jika kondisi bernilai benar maka cetak output "Statement"
6. Kemudian selanjutnya nilai i kembali ditambahkan 1
7. Cek kondisi apakah nilai i masih <=10 jika ya maka akan dicetak output Kembali
8. Iterasi terus dilakukan hingga kondisi bernilai salah
9. Selesai.

Berikut ini contoh implementasi penerapan perulangan struktur do-while yang sederhana dalam bentuk algoritma (pseudocode) dan ditranslasikan kedalam Bahasa pemrograman Java.

Contoh 6.

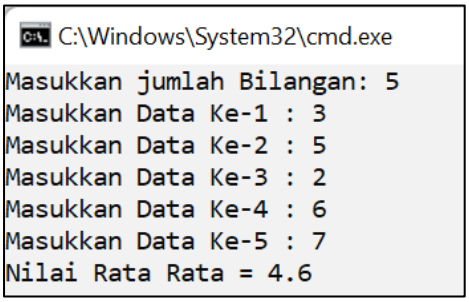
```
Algoritma NilaiRataRata
{Menghitung nilai rata-rata dari banyakknya bilangan yang dinputan}

DEKLARASI
jumlah : integer {jumlah perulangan}
i, data : integer
ratarata, hasil : float

DESKRIPSI :
read(i)
i←1
do
read(data,hasil,jumlah)
hasil←hasil+data
i←i+1
while i ≤ jumlah
endwhile
ratarata ← hasil/jumlah
write(ratarata)
```

Translasi Algoritma Nilai Rata-Rata ke pemrograman Java

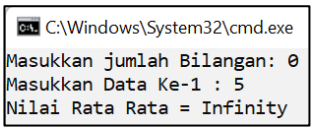
```
import java.util.Scanner;
public class NilaiRataRata {
    public static void main(String[] args) {
        int i, data, jumlah;
        float ratarata, hasil = 0;
        Scanner inputan = new Scanner(System.in);
        System.out.print("Masukkan jumlah Bilangan: ");
        jumlah = inputan.nextInt();
        i=1;
        do{
```

```
            System.out.print("Masukkan Data Ke-" + i + " : ");
            data = inputan.nextInt();
            hasil = hasil + data;
            i++;
        }while(i<=jumlah);
        ratarata = hasil/jumlah;
        System.out.println("Nilai Rata Rata = " + ratarata);
    }
}
```

Output :



Pada kasus ini pada dasarnya dapat dilakukan dengan menggunakan struktur for, while maupun do-while. Pada penggunaan struktur do-while statemen "Masukkan Data Ke-n" pasti akan memenuhi meskipun hanya 1 kali, dikarenakan pada statement kondisi mulai memprosesnya ketika sudah 1 kali perulangan dilakukan. Berikut ini jika memasukkan inputan bilangan dengan jumlah sama dengan 0.



Dari eksperimen output yang dihasilkan, secara kondisi jika jumlah bilangan yang dimasukkan sama dengan 0, maka harusnya tidak memerlukan aksi apapun. Namun dalam penggunaan strukutur do-while aksi masih dilakukan minimal satu kali meskipun kondisi tidak memenuhi.

5.5. Nested Loop (Perulangan Bersarang)

Perulangan bersarang (nested loop) merupakan sebutan dari perulangan didalam perulangan, yang membentuk beberapa pernyataan perulangan yang harus diproses. Nested Loop atau perulangan bersarang biasanya digunakan pada sebuah statement yang memerlukan suatu proses yang panjang. Semua jenis perulangan bisa dibuat dalam bentuk perulangan bersarang, termasuk perulangan for, while dan do-while. Dalam bahasa inggris, perulangan bersarang ini dikenal dengan sebutan nested loop. Di dalam perulangan bersarang terdapat istilah outer loop dan inner loop. Sesuai dengan namanya, outer loop adalah sebutan untuk perulangan dibagian luar, sedangkan inner loop sebutan untuk perulangan dibagian dalam. Bentuk perulangan struktur FOR Nested Loop :

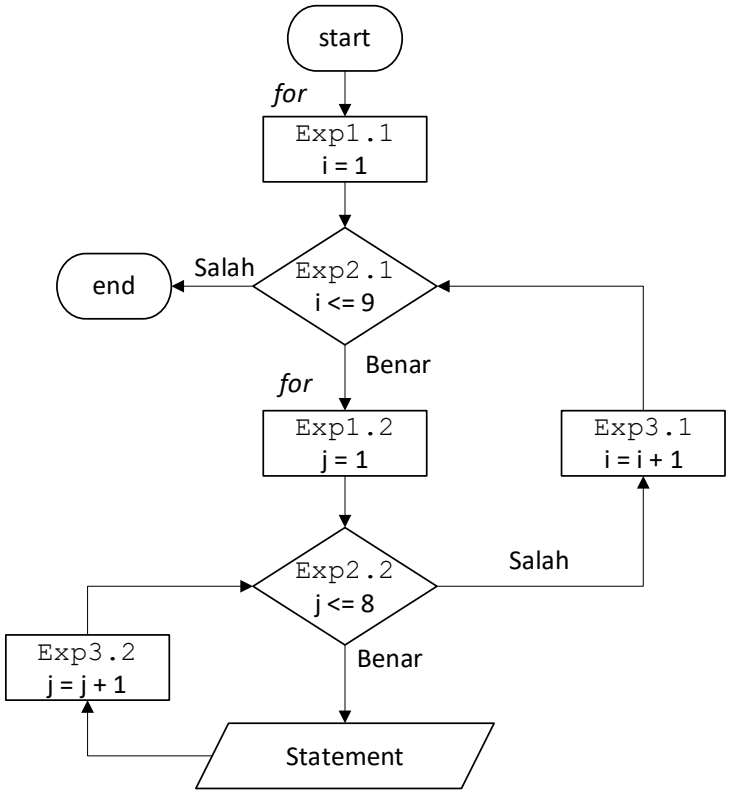
```
for (exp1.1; exp2.1; exp3.1)
{
    for (exp1.2; exp2.2; exp3.2)
    {
        Statement;
    }
}
```

Keterangan :
exp1.1 = Expresi untuk inisialisasi nilai awal (i)
exp1.2 = Expresi untuk inisialisasi nilai awal (j)
exp2.1 = Expresi untuk kondisi 1
exp2.2 = Expresi untuk kondisi 2
exp3.1 = Expresi untuk increment/decrement (i)
exp3.2 = Expresi untuk increment/decrement (j)

Pada bentuk For (Nested Loop) di atas, outer loop adalah perulangan di struktur for paling atas (exp1.1; exp2.1; exp3.1), sedangkan inner loop adalah

perulangan di struktur for yang ada didalamnya (exp1.2; exp2.2; exp3.2). Kode program di dalam outer loop akan dijalankan sejumlah kondisi perulangan di outer saja. Sedangkan kode program yang ada di dalam perulangan inner loop akan dijalankan sebanyak perulangan outer * inner. Tidak ada batasan seberapa banyak kedalaman dari sebuah perulangan bersarang. Bisa saja membuat perulangan di dalam perulangan di dalam perulangan dan seterusnya. Tentu saja semakin banyak perulangan yang bersarang, kode programnya juga akan semakin kompleks.

Dalam membuat perulangan bersarang kita juga harus sangat teliti dalam penggunaan tanda kurung kurawal "{ }" untuk menandakan awal dan akhir sebuah blok kode program. Tidak jarang hasilnya jadi berantakan karena salah menulis posisi penutup kurung kurawal. Berikut ini gambaran proses nested loop dalam bentuk diagram alir yang ditunjukkan pada Gambar berikut ini.



Algoritma berdasarkan Diagram Alir Nested Loop :

1. Mulai

2. Inisialisasi nilai awal i =1
3. Cek kondisi apakah kondisi i <= 9, jika kondisi bernilai benar maka melanjutkan proses perulangan selanjutnya.
4. Inisialisasi nilai awal j = 1
5. Cek kondisi apakah kondisi j <= 8, jika kondisi bernilai benar maka cetak output "Statemen" dan nilai j ditambahkan 1 selama kondisi terpenuhi. Jika kondisi bernilai salah maka melakukan proses perulangan yang sebelumnya.
6. Nilai i ditambahkan 1
7. Cek kembali kondisi apakah nilai i <= 9, Iterasi terus dilakukan hingga kondisi bernilai salah
8. Selesai.

Berikut ini contoh implementasi penerapan perulangan nested loop (for bersarang) yang sederhana dalam bentuk algoritma (pseudocode) dan ditranslasikan kedalam Bahasa pemrograman Java.

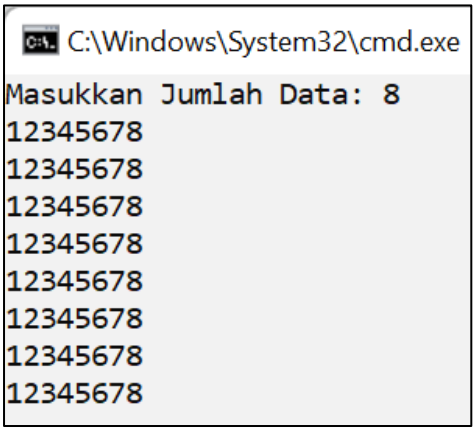
Contoh 7.

```
Algoritma CetakVariabelInisial
{Mencetak variabel Inisial (innerloop) sejumlah nilai inputan}

DEKLARASI
i, j, jumlah : integer

DESKRIPSI :
read(i, jumlah)
i←1
for i to jumlah do
read(j, jumlah)
j←1
      for j to jumlah do
write(j)
endfor
write(" ")
endfor
```

Translasi Algoritma Cetak Variabel Inisial ke pemrograman Java

```
import java.util.Scanner;
public class CetakVariabelInisial {
    public static void main(String[] args) {
        int i;
        int j;
        Scanner inputan = new Scanner(System.in);
        System.out.print("Masukkan Jumlah Data: ");
        int jumlah = inputan.nextInt();

        for(i=1; i<=jumlah; i++){
            for(j=1; j<=jumlah; j++){
                System.out.print(j);
            }
            System.out.println();
        }
    }
}
```

Output :



Proses pada algoritma ini adalah dengan diawali melakukan pengecekan kondisi pada struktur for luar (outer loop). Jika memenuhi maka selanjutnya memproses struktur for dibagian dalam (inner loop). Struktur for bagian dalam ini melakukan proses perulangan sampai tidak memenuhi. Setelah proses di inner

loop selesai, maka selanjutnya akan melakukan proses pengecekan terhadap perulangan for dibagian luar (outer loop) lagi. Proses ini akan berlangsung sampai kedua struktur for semua terpenuhi.

Contoh 8.

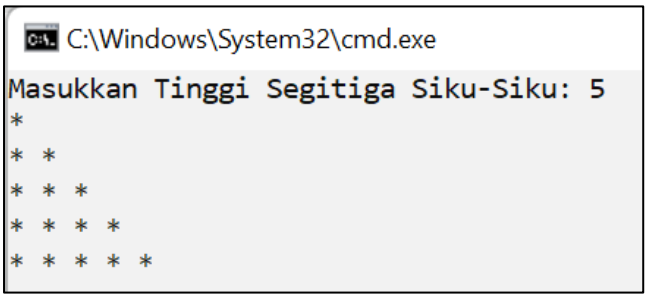
```
Algoritma BintangSegitigaSiku
{Membuat keluaran bintang berbentuk segitiga siku-siku sesuai nilai inputan}

DEKLARASI
i, j, jumlah : integer

DESKRIPSI :
read(i, jumlah)
i←1
for i to jumlah do
read(j)
j←1
      for j to i do
write("* ")
endfor
write(" ")
endfor
```

Translasi Algoritma Bintang Segitiga Siku ke pemrograman Java

```
import java.util.Scanner;
public class BintangSegitigaSiku {
    public static void main(String[] args) {
        int i;
        int j;
        Scanner inputan = new Scanner(System.in);
        System.out.print("Masukkan Tinggi Segitiga Siku-Siku: ");
        int jumlah = inputan.nextInt();

        for(i=1; i<=jumlah; i++){
            for(j=1; j<=i; j++){
                System.out.print("* ");
```

```
            }
            System.out.println();
        }
    }
}
```

Output :



Proses pada kasus ini sama dengan proses pada Contoh 7. Namun pada proses algoritma ini pada bagian dalam for (inner loop) memberikan kondisi (j < i), dimana variabel i merupakan variabel inisialiasi dari perulangan for luar (outer loop). Sehingga perulangan for bagian dalam melakukan perulangan sebanyak nilai i yang diperolah dari perulangan sebelumnya yang sudah dilakukan dari proses for bagian luar (outer loop)

5.6. Implementasi Struktur Perulangan dalam Algoritma Pemrograman

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

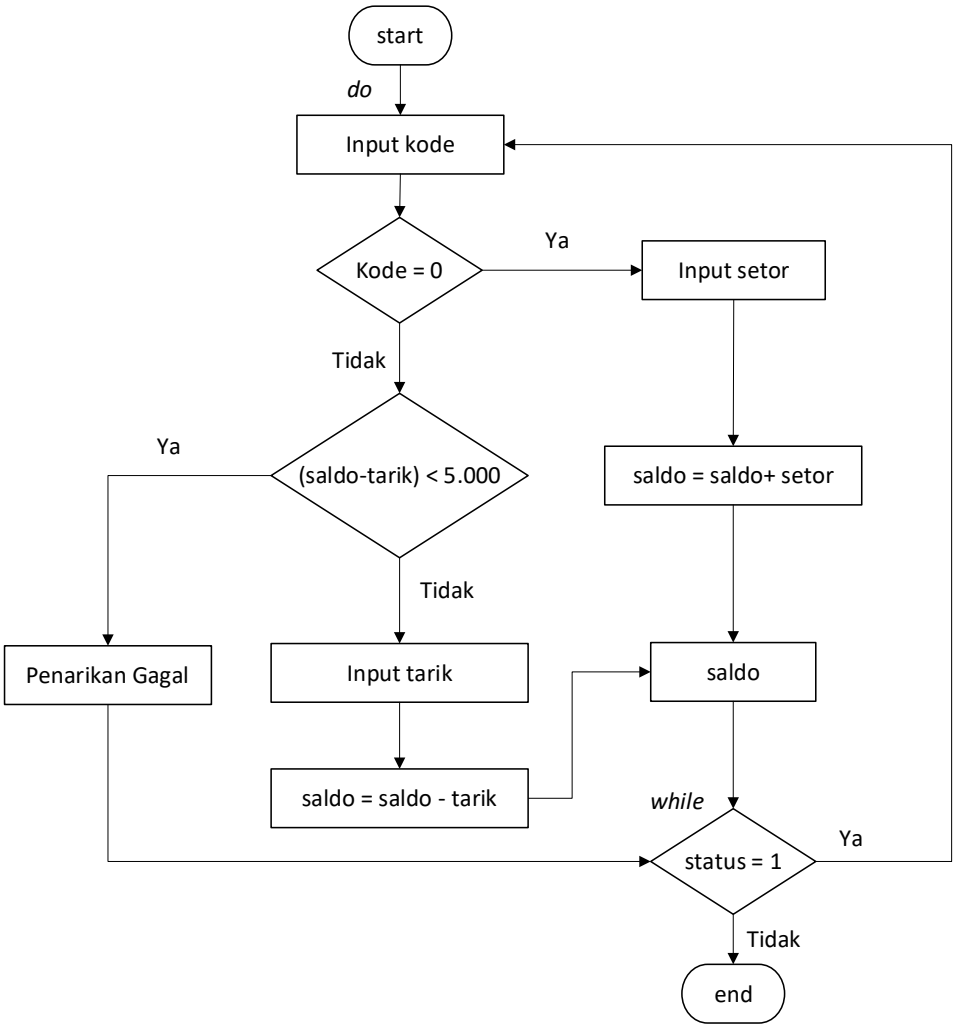
Contoh Implementasi 1: Setor dan Penarikan Tabungan Rekening

Pada sebuah bank nasabah dapat menyetor dan mengambil uangnya, namun jumlah saldo minimum yang harus disisakan pada tabungan nasabah sebesar Rp. 5.000. Hal ini berarti jika saldo nasabah sebesar Rp. 5.000, maka nasabah tersebut tidak dapat mengambil uangnya. Kode transaksi untuk menyetor adalah 0 dan kode transaksi untuk mengambil adalah 1. Buatlah Flowchart dan Algoritma (*Pseudocode*) dan translasikan ke pemrograman Java yang mensimulasikan

dilakukan nasabah tersebut. Algoritma dapat menerima masukan berupa kode transaksi dan jumlah uang yang disetor/diambil. Rancanglah algoritma tersebut sehingga memungkinkan nasabah dapat melakukan transaksi berulangkali sampai saldo yang tersisa Rp. 5.000 atau jumlah yang yang diambil lebih besar dari saldonya.

a. *Flowchart* Setor dan Penarikan Tabungan Rekening.

Berikut ini adalah Flowchart untuk kasus pemberian diskon pada Maskapai Penerbangan.



b. *Pseudocode* Setor dan Penarikan Tabungan Rekening

Berikut ini adalah *Pseudocode* untuk kasus setor dan penarikan tabungan rekening.

```
Algoritma Setor dan Penarikan Tabungan Rekening
{Dapat melakukan transaksi penyetoran dan penarikan dan minimal sisa saldo
yang dapat diambil adalah 5.000 Rupiah}

DEKLARASI
limit : int {Rp. 5.000}
saldo, setor, tarik : int
status : int {1:ya/0:tidak}
kode : int {0:penyetoran; 1:penarikan}

DESKRIPSI :
do
read(limit,saldo,setor, tarik, kode)
if kode = 0 then
   saldo←saldo+setor
   write(saldo)
else
   if saldo-tarik < limit then
   write('Sisa saldo minimal 5.000')
   else
   saldo←saldo-tarik
   write(saldo)
   read (status)
   endif
endif
while status = 1
endwhile
write('terima kasih')
```

c. Bahasa pemrograman java Setor dan Penarikan Tabungan Rekening.

Berikut ini adalah hasil implementasi pada bahasa pemrograman java untuk kasus Setor dan Penarikan Tabungan Rekening.

```
import java.util.Scanner;
public class RekeningBank {
    public static void main(String[] args) {
    int limit = 5000;
    int saldo = 0;
    int setor, tarik, kode, status;

    do{
        System.out.println("Kode 0 : Untuk Penyetoran");
        System.out.println("Kode 1 : Untuk Penarikan");
        Scanner inputan = new Scanner(System.in);
        System.out.print("Pilih Kode Transaksi : ");
        kode = inputan.nextInt();
        if (kode == 0){
            System.out.print("Masukkan nominal yang akan disetorkan : ");
            setor = inputan.nextInt();
            if ((saldo+setor) < limit){
                System.out.println("Penyetoran  Gagal,  Setoran  Minimal
5000 !");
            }else{
                saldo = saldo + setor;
```

```
                System.out.println("Penyetoran  Berhasil,  jumlah  Saldo
anda = " + saldo);
            }
        }else if(kode == 1){
            System.out.print("Masukkan nominal yang akan diambil : ");
            tarik = inputan.nextInt();
            if ((saldo - tarik) < limit){
                //saldo = saldo - tarik;
                System.out.println("Penarikan    Gagal,    Saldo    tidak
mencukupi ");
                if(saldo < 0)
                  System.out.println("Sisa Saldo anda = 0");
                else{
                  System.out.println("Sisa Saldo anda = " + saldo);
                }
            }else{
                saldo = saldo - tarik;
                System.out.println("Penarikan Berhasil, Sisa Saldo anda
= " + saldo);
            }

        }else{
            System.out.println("Kode yang dimasukkan kurang tepat!");
        }
        System.out.print("Lanjut transaksi lain ? Jika Ya, Tekan (1) /
Tidak Tekan (0) : ");
        status = inputan.nextInt();
    }while(status == 1);
        System.out.println("Terima Kasih");
    }
}
```

Output :

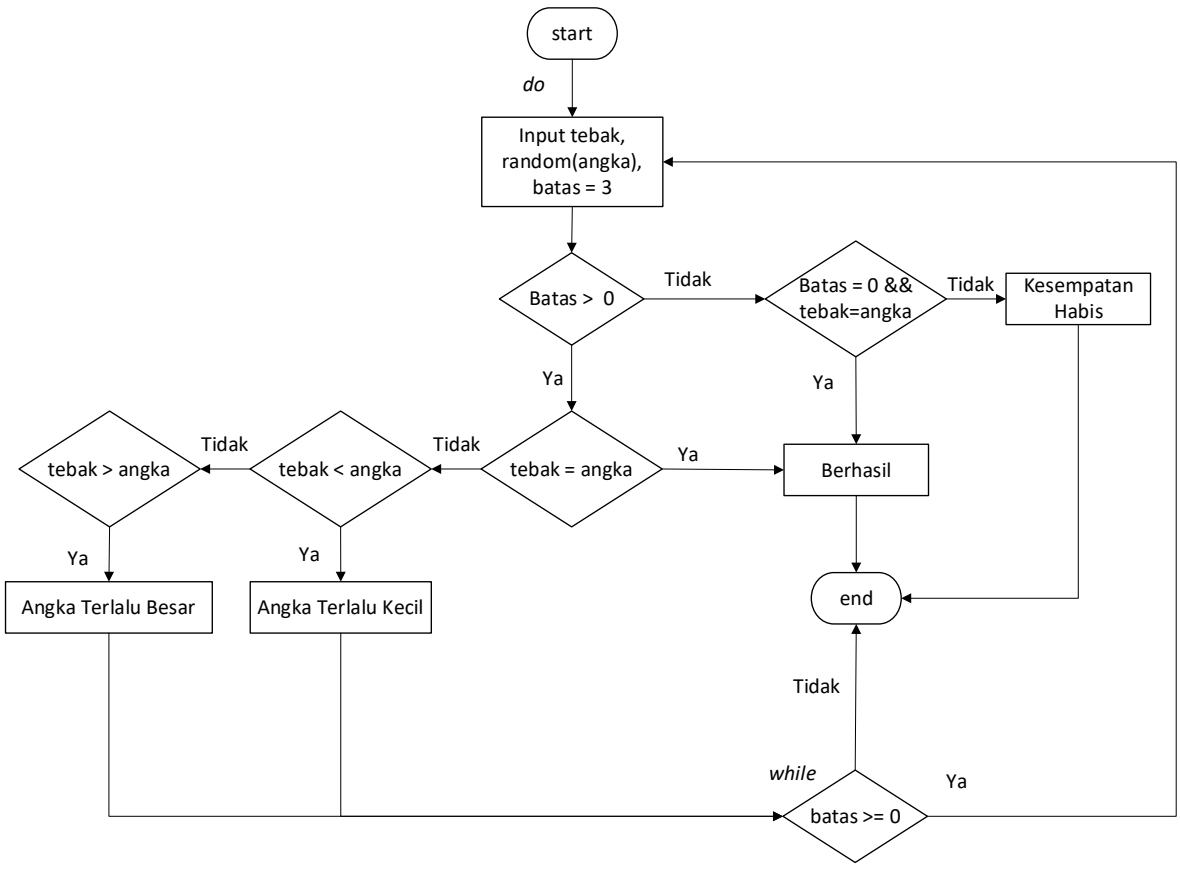
```
C:\Windows\System32\cmd.exe
Kode 0 : Untuk Penyetoran
Kode 1 : Untuk Penarikan
Pilih Kode Transaksi : 0
Masukkan nominal yang akan disetorkan : 15000
Penyetoran Berhasil, jumlah Saldo anda = 15000
Lanjut transaksi lain ? Jika Ya, Tekan (1) / Tidak Tekan (0) : 1
Kode 0 : Untuk Penyetoran
Kode 1 : Untuk Penarikan
Pilih Kode Transaksi : 1
Masukkan nominal yang akan diambil : 5000
Penarikan Berhasil, Sisa Saldo anda = 10000
Lanjut transaksi lain ? Jika Ya, Tekan (1) / Tidak Tekan (0) : 0
Terima Kasih
```

Contoh Implementasi 2: Permainan Tebak Angka

Pada sebuah permainan tebak angka, pemain mendapatkan tiga kesempatan untuk menebak angka antara range 1 – 10. Range angka tersebut dilakukan secara acak oleh sistem. Jika angka tebakan pemain sesuai dengan angka acak, maka pemain tersebut berhasil. Namun jika tebakan pemain salah, maka pemain akan mendapatkan petunjuk supaya angka tebakan selanjutnya disesuaikan dengan petunjuknya yaitu apakah angka yang ditebak pemain sebelumnya terlalu kecil atau terlalu besar.

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

a. *Flowchart* Permainan Tebak Angka



b. *Pseudocode* Permainan Tebak Angka

```
Algoritma Permainan Tebak Angka
{Menebak angka 1-10 dengan tiga kali kesempatan}

DEKLARASI
angka, tebak : int
batas : int {3 kali kesempatan}

DESKRIPSI :
read(angka, tebak, batas)
do{
if batas > 0 then
   if batas = angka then
         write ('Tebakan Berhasil')
   else if tebak < angka
         write ('Tebakan Terlalu Kecil')
   else if tebak > angka
         write ('Tebakan Terlalu Besar')
   endif
   endif
   endif
   write (batas)
else if batas = 0 and tebak = angka
   write ('Tebakan Berhasil')
endif
else
   write ('Anda Gagal, Kesempatan Habis')
endif
endif
   batas←batas - 1
}while batas >= 0
```

c. Bahasa pemrograman java Permainan Tebak Angka

```
import java.util.Scanner;
public class PermainanTebakAngka {
    public static void main(String[] args) {
    int angka = (int)(Math.random() *10);
    int tebak;
    int batas = 2;
        System.out.println("Permainan Tebak Angka");
        System.out.println("Tebaklah angka 1-10 dan kamu hanya memiliki
3 kesempatan");
    do{
        Scanner inputan = new Scanner (System.in);
        System.out.print("Masukkan Angka yang anda tebak = ");
        tebak = inputan.nextInt();

        if (batas > 0){
            if (tebak == angka){
                System.out.println("Selamat,  angka  yang  anda  masukkan
Benar !");
                break;
            }
            else if (tebak < angka){
```

```
                System.out.println("Maaf,  angka  yang  anda  masukkan
terlalu kecil.");
            }
            else if (tebak > angka){
                System.out.println("Maaf,  angka  yang  anda  masukkan
terlalu besar.");
            }
            System.out.println("Anda Memiliki " + batas + " Kesempatan
Lagi");
            System.out.println("----------------------------------------
----------");
        }else if (batas == 0 && tebak == angka){
            System.out.println("Selamat, angka yang anda masukkan Benar
!");
        }else{
            System.out.println("----------------------------------------
----------");
            System.out.println("Anda  Salah  !  Kesempatan  Telah  Habis.
Silahkan Coba Lagi.");
            System.out.println("Angka yang dimaksud adalah " + angka);
            System.out.println("----------------------------------------
----------");
        }
        batas--;
      }while (batas >= 0);
    }
}
```

Output :

```
C:\Windows\System32\cmd.exe
Permainan Tebak Angka
Tebaklah angka 1-10 dan kamu hanya memiliki 3 kesempatan
Masukkan Angka yang anda tebak = 5
Maaf, angka yang anda masukkan terlalu kecil.
Anda Memiliki 2 Kesempatan Lagi
--------------------------------------------------
Masukkan Angka yang anda tebak = 6
Maaf, angka yang anda masukkan terlalu kecil.
Anda Memiliki 1 Kesempatan Lagi
--------------------------------------------------
Masukkan Angka yang anda tebak = 8
Selamat, angka yang anda masukkan Benar !
```

POST TEST

Soal Latihan Struktur Algoritma Perulangan

1. Buatlah *flowchart*, algortima (*pseudocode*) dan translasikan kedalam Bahasa pemrograman java untuk menghitung mundur pelncuran roket dimulai dari inputan yang dimasukkan sampai angka 1 dengan struktur for.
2. Buatlah *flowchart*, algortima (*pseudocode*) dan translasikan kedalam Bahasa pemrograman java untuk membalikkan inputan berupa string dengan struktur for.
3. Buatlah *flowchart*, algortima (*pseudocode*) dan translasikan kedalam Bahasa pemrograman java dengan menggunakan while untuk menghasilkan deret bilangan Fibonacci sesuai dengan nilai inputan.
4. Buatlah *flowchart*, algortima (*pseudocode*) dan translasikan kedalam Bahasa pemrograman java untuk menampilkan pola segitiga piramida dengan bintang menggunakan for dengan ketinggian sesuai dengan nilai inputan.
5. Buatlah *flowchart*, algortima (*pseudocode*) dan translasikan kedalam Bahasa pemrograman java untuk menampilkan pola segitiga siku terbalik dengan bintang menggunakan for dengan ketinggian sesuai dengan nilai inputan