Summary Notes of Sesi 2.1 - Fondasi Pemrograman & Struktur Data – 01 Teorema Structured Control.

Teorema *structured control* merupakan teorema yang merupakan bagian dari konsep pemrograman terstruktur. *Structured control* berisi instruksi atau mekanisme yang terkait dengan kendali alur program. Hal yang paling umum terdapat dalam *structured control* adalah:

- Sequence
- Selection (if dan case)
- Repetition (for dan while)

Pengenalan Sequence

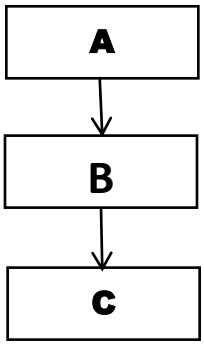
Sequence merupakan mekanisme aliran instruksi (*instruction flows*) dimana tiap instruksi dikerjakan secara berurutan sesuai dengan urutan penulisannya.

Contoh process dengan sequence:

```
Tampil perintah untuk baca nilai A;
Baca A;
B = 2*A;
```

Berdasarkan contoh di atas, maka instruksi akan dieksekusi sesuai urutan dari atas ke bawah.

Contoh diagram dengan sequence:



Artinya:

Proses dijalankan berurutan sesuai urutan proses;
Instruksi A;
Instruksi B;
Instruksi C;

Gambar 2.1. Contoh diagram dengan sequence

Pengenalan Selection

Selection merupakan mekanisme aliran instruksi (*instructions flow*) dimana suatu instruksi akan dikerjakan jika kondisi tertentu dipenuhi.

Terdapat dua model selection:

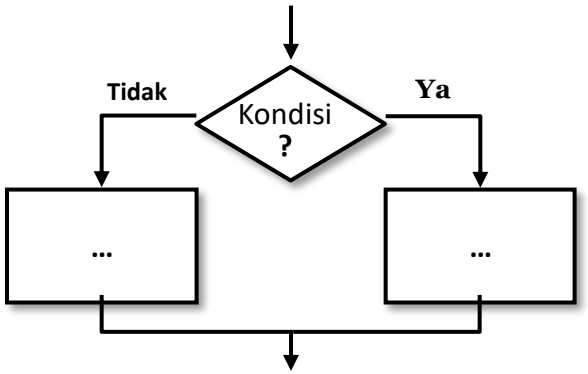
- if – else
- case

if - else

if – else merupakan selection dengan format penulisan sebagai berikut;

```
if (kondisi) {
    ....
}
else {
    ....
}
```

Berikut adalah format diagram untuk selection if-else:



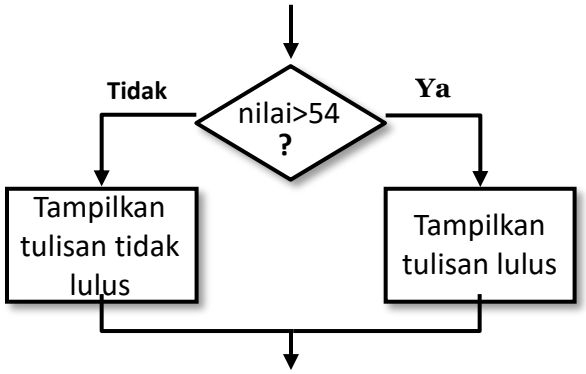
Gambar 2.2. Format diagram selection if-else

Contoh process dengan selection if-else:

```
if(nilai>54) {
    Tampilkan tulisan lulus;
}
else {
    Tampilkan tulisan tidak lulus;
}
```

Berdasarkan contoh di atas, maka jika nilai bernilai lebih besar dari 54 maka instruksi **Tampilkan tulisan lulus** akan dijalankan, sedangkan jika nilai bernilai 54 atau lebih kecil maka instruksi **Tampilkan tulisan lulus** yang akan dijalankan.

Berikut adalah diagram untuk contoh *process* di atas:



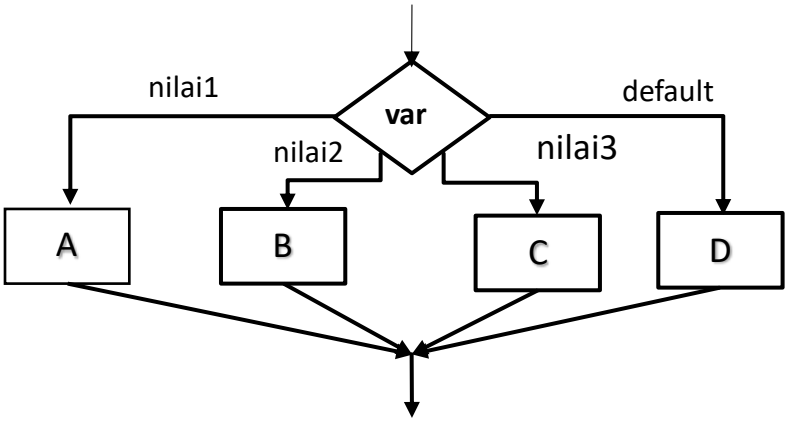
Gambar 2.3. Contoh diagram selection if-else

case

case merupakan *selection* dengan format penulisan sebagai berikut;

```
switch (var) {
    case nilai1: ….;
                 break;
    case nilai2: ….;
                 break;
    case nilai3: ….;
                 break;
    dsb…
    default : ….;
}
```

Berikut adalah format diagram untuk *selection case*:



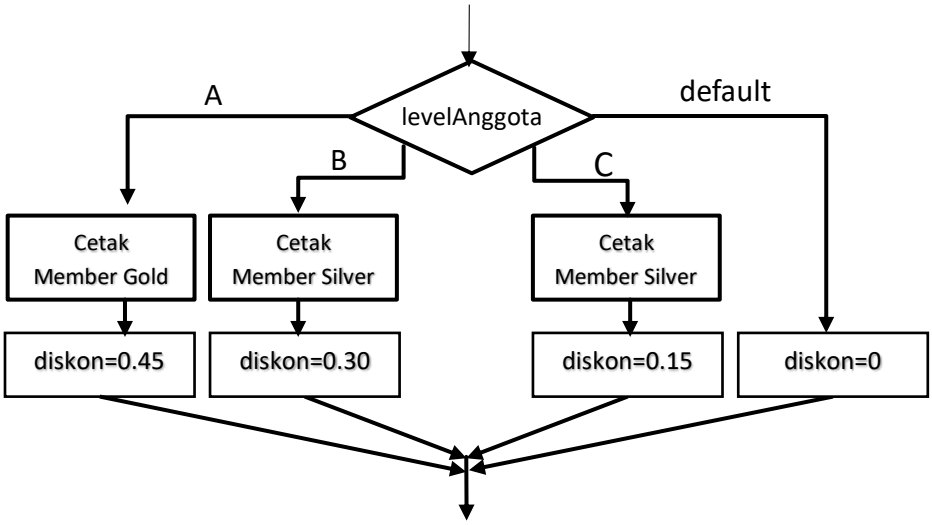
Gambar 2.4. Format diagram selection case

Contoh process dengan selection case:

```
switch (levelAnggota) {
    case 'A': Cetak "Member Gold";
              diskon=0.45;
              break;
    case 'B': Cetak "Member Silver";
              diskon=0.30;
              break;
    case 'C': Cetak "Member Bronze";
              diskon=0.15;
              break;
    default : diskon=0;
}
```

Berdasarkan contoh di atas, maka jika nilai bernilai lebih besar dari 54 maka instruksi **Tampilkan tulisan lulus** akan dijalankan, sedangkan jika nilai bernilai 54 atau lebih kecil maka instruksi **Tampilkan tulisan lulus** yang akan dijalankan.

Berikut adalah diagram untuk contoh *process* di atas:



Gambar 2.5. Contoh diagram selection case

Pengenalan Repetition

Repetition merupakan mekanisme aliran instruksi (*instructions flow*) dimana suatu instruksi dikerjakan berulang–ulang sampai suatu kondisi dicapai..

Terdapat dua model *repetition*:

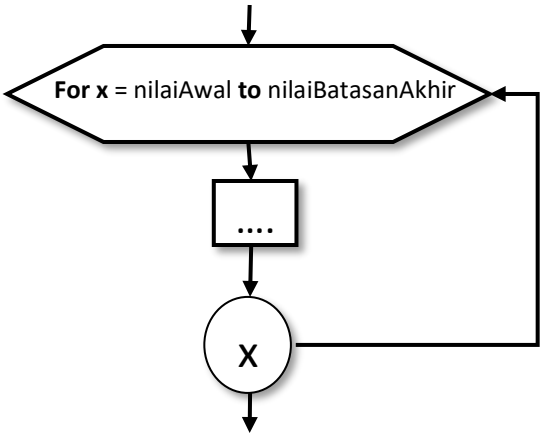
- for
- while

for

for merupakan repetition dengan format penulisan sebagai berikut;

```
for (nilaiAwal; nilaiBatasanAkhir; multiplikasi) {
    ....
}
```

Berikut adalah format diagram untuk repetition for:



Gambar 2.6. Format diagram repetition for

Contoh process dengan repetition for:

```
for (x=0; x<3; x++) {
    tampilkan tulisan "A=";
    baca A;
    total=total+A;
}
Tampilkan tulisan "keseluruhan=" total;
```

Berdasarkan contoh di atas, misal nilai awal total=0, maka;

Pertama-tama **x=0**

Akan tampil tulisan **A=**

Baca A, menunggu pengguna memasukkan nilai untuk A, misal pengguna mengetik 3 sehingga menjadi A=3

total=total+A, maka total=0+3=3

x++, adalah x=x+1=0+1, x=1, karena masih belum batasan akhir maka masuk ke blok repetition

Akan tampil tulisan **A=**

Baca A, menunggu pengguna memasukkan nilai untuk A, misal pengguna mengetik 4 sehingga menjadi A=4

total=total+A, maka total=3+4=7

x++, adalah x=x+1=1+1, x=2, karena masih belum batasan akhir maka masuk ke blok repetition

Akan tampil tulisan **A=**

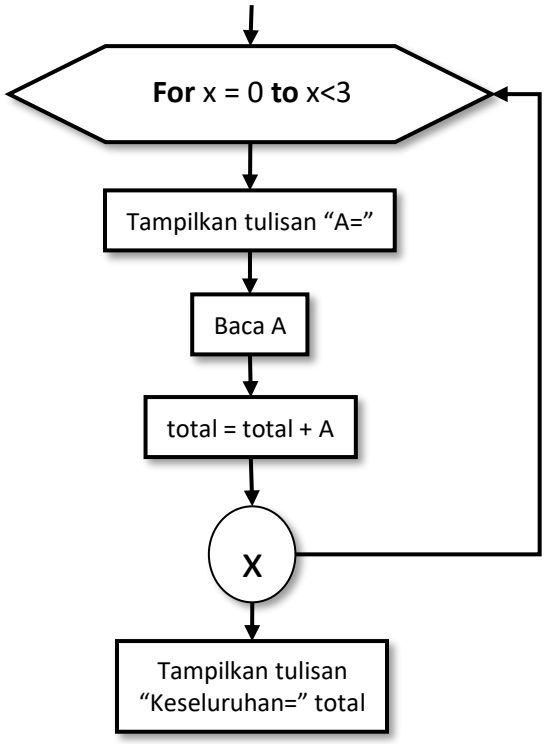
Baca A, menunggu pengguna memasukkan nilai untuk A, misal pengguna mengetik 2 sehingga menjadi A=2

total=total+A, maka total=7+2=9

x++, adalah x=x+1=2+1, x=3, sudah pada batasan akhir maka tidak masuk ke blok repetition

Akan tampil tulisan **keseluruhan=9**

Berikut adalah diagram untuk contoh *process* di atas:



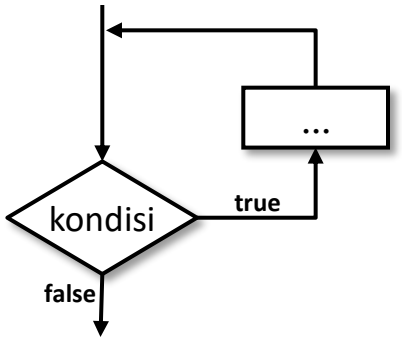
Gambar 2.7. Contoh diagram repetition for

while

while merupakan repetition dengan format penulisan sebagai berikut;

```
while (kondisi) {
    ....
}
```

Berikut adalah format diagram untuk repetition while:



Gambar 2.8. Format diagram repetition while

Contoh process dengan repetition while:

```
x=0;
while (x<3) {
    tampilkan tulisan “A=”;
    baca A;
    total=total+A;
    x++;
}
tampilkan tulisan “keseluruhan=” total;
```

Berdasarkan contoh di atas, misal nilai awal total=0, maka;

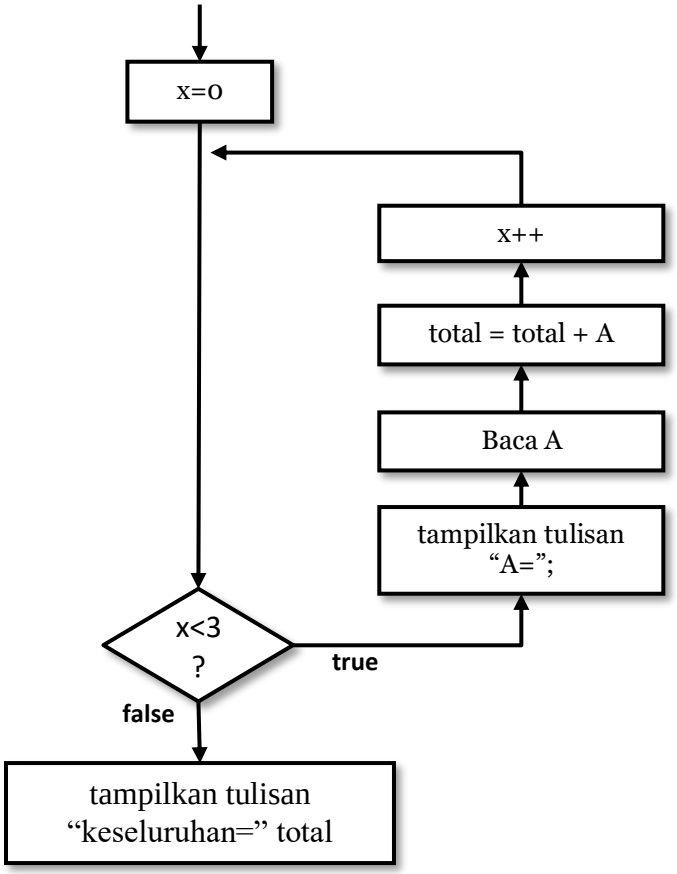
Pertama-tama **x=0**
karena x masih lebih kecil dari 3 maka masuk ke blok repetition
- Akan tampil tulisan **A=**
- **Baca A**, menunggu pengguna memasukkan nilai untuk A, misal pengguna mengetik 3 sehingga menjadi A=3
- **total=total+A**, maka total=0+3=3
- **x++**, adalah x=x+1=0+1, x=1

karena x=1 dan masih lebih kecil dari 3 maka masuk ke blok repetition
- Akan tampil tulisan **A=**
- **Baca A**, menunggu pengguna memasukkan nilai untuk A, misal pengguna mengetik 4 sehingga menjadi A=4
- **total=total+A**, maka total=3+4=7
- **x++**, adalah x=x+1=1+1, x=2

karena x=2 dan masih lebih kecil dari 3 maka masuk ke blok repetition
- Akan tampil tulisan **A=**

Baca A, menunggu pengguna memasukkan nilai untuk A, misal pengguna mengetik 2 sehingga menjadi A=2
total=total+A, maka total=7+2=9
x++, adalah x=x+1=2+1, x=3
karena x=3 dan sudah tidak lebih kecil dari 3 maka tidak masuk ke blok repetition
Akan tampil tulisan **keseluruhan=9**

Berikut adalah diagram untuk contoh *process* di atas:



Gambar 2.9. Contoh diagram repetition while

Berdasarkan pembahasan-pembahasan dan contoh-contoh yang terkait dengan teorema structured control, maka dapat kita simpulkan bahwa dalam teorema terdapat mekanisme aliran (*flow*) proses dalam pemrograman. Mekanisme aliran (flow) proses pada teorema structured control tersebut adalah:

- Mekanisme aliran berurutan → sequence
- Mekanisme aliran percabangan → selection (dengan **if-else** dan **case**)
- Mekanisme aliran perulangan (*loop*) → Repetition (dengan **for** dan **while**)

Summary Notes of Sesi 2.2 - Fondasi Pemrograman & Struktur Data – 02 Flow Chart, Sequence & Statement.

Seperti yang sudah dibahas pada sesi sebelumnya bahwa mekanisme aliran (*flow*) proses pada teorema *structured control* tersebut adalah:

- Mekanisme aliran berurutan → *sequence*
- Mekanisme aliran percabangan → *selection* (dengan ***if-else*** dan ***case***)
- Mekanisme aliran perulangan (*loop*) → *Repetition* (dengan ***for*** dan ***while***)

Flow chart merupakan *tools diagram* yang digunakan untuk menggambarkan aliran proses pada program. Kalau kita kelompokkan, simbol-simbol pada *flow chart* terdiri dari kelompok simbol untuk mekanisme *sequence*, *selection*, dan *repetition* (teorema *structured control*).

Sequence dan *Statement*

Sequence merupakan mekanisme aliran instruksi (*instruction flows*) dimana tiap instruksi dikerjakan secara berurutan sesuai dengan urutan penulisannya. Dalam *sequence* terdapat beberapa instruksi yang posisinya berurutan untuk dikerjakan atau dieksekusi dalam program.

Statement merupakan **unit sintaksis dari instruksi (imperatif) pada bahasa pemrograman yang menyatakan tindakan yang harus dilakukan**, atau bisa dikatakan *statement* merupakan instruksi pada bahasa pemrograman yang harus dilaksanakan oleh program.

Susunan dari beberapa *statement* merupakan suatu urutan, sehingga akan mengikuti mekanisme *sequence*.

Statement Terminator

Statement terminator berfungsi sebagai permulaan atau akhir, dalam suatu alur program selalu ada permulaan dan akhir. Dalam *flow chart* Untuk menandakan awal dan akhir suatu flow digunakan simbol sebagai berikut:

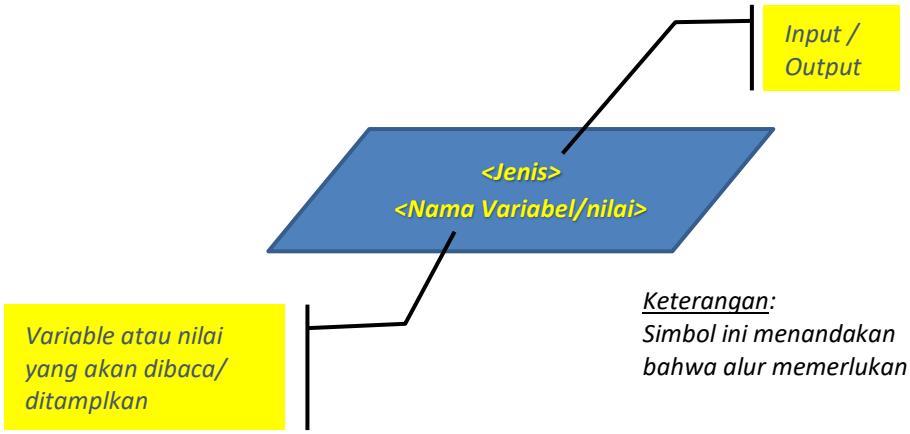
Tabel 2.1. Tabel simbol terminator

Simbol	Keterangan
	Simbol ini menandakan suatu flow dimulai
	Simbol ini menandakan suatu flow berhenti

Statement Input atau Output

Statement input atau output berfungsi sebagai *interface* (perantara) program dengan pengguna. Dengan *statement* ini bisa terjadi komunikasi dan interaksi antara program dengan pengguna,

selain itu dengan *statement* ini peran pengguna juga bisa menentukan alur instruksi dalam program. Dalam *flow chart* untuk menandakan instruksi input atau output digunakan simbol sebagai berikut:

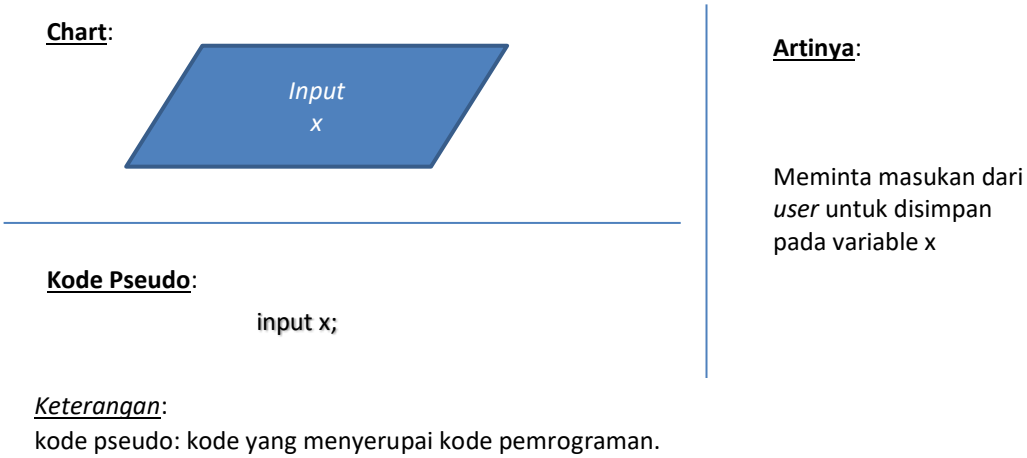


Gambar 2.10. Simbol input atau output

Dalam flowchart simbol tersebut digunakan sebagai representasi bahwa proses memerlukan masukan (input) dari pengguna atau proses menampilkan/mengeluarkan (*output*) untuk pengguna. Dalam Bahasa pemrograman akan diterjemahkan dengan:

- Kode instruksi untuk membaca masukan dari user.
- Kode instruksi untuk menampilkan suatu teks, nilai atau nilai variable

Contoh input:



Gambar 2.11. Simbol input, kode pseudo untuk input dan artinya

Contoh output:

Chart:

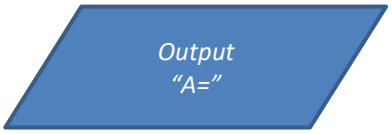
Kode Pseudo:

```
output x;
```

Artinya:

Menampilkan nilai yang tersimpan pada variable x

Chart:



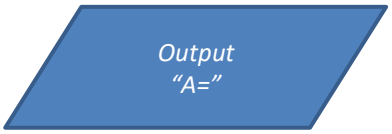
Kode Pseudo:

```
output "A=";
```

Artinya:

Menampilkan tulisan (teks), yaitu teks: **A=**

Chart:



Kode Pseudo:

```
output "A=";
```

Artinya:

Menampilkan tulisan (teks), yaitu teks: **A=**

Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.12. Contoh simbol output dan kode pseudo untuk output

Statement Process

Statement process berfungsi untuk menyatakan dilakukan suatu proses. Proses yang dilakukan diantaranya proses atau operasi matematika, operasi string, komparasi (pembandingan), deklarasi atau inisialisasi. Dalam *flow chart* untuk menandakan instruksi proses atau operasi digunakan simbol sebagai berikut:

Keterangan:
Simbol ini menandakan bahwa dilakukan suatu proses atau operasi sesuai yang tertulis pada chart

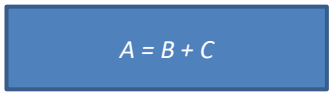
Gambar 2.13. Simbol process

Dalam flowchart simbol tersebut digunakan sebagai representasi bahwa dilakukan proses atau operasi sesuai yang tertulis didalam simbol tersebut. Dalam Bahasa pemrograman akan diterjemahkan dengan:

- Kode instruksi operasi Matematika.
- Kode instruksi operasi String.
- Kode instruksi operasi Komparasi.
- Kode instruksi operasi deklarasi atau inisialisasi.

Contoh proses:

Chart:



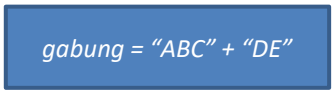
Kode Pseudo:

```
A=B+C;
```

Artinya:

Operasi matematika, nilai isi variabel B tambah nilai isi variabel C dan hasilnya disimpan di variabel A

Chart:



Kode Pseudo:

```
gabung="ABC" + "DE";
```

Artinya:

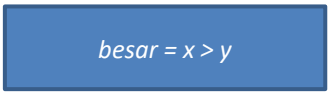
Operasi String, teks ABC digabung dengan teks DE dan hasilnya disimpan di variabel gabung

Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.14. Contoh simbol proses dan kode pseudo untuk operasi matematika dan operasi string

Contoh proses:

Chart:



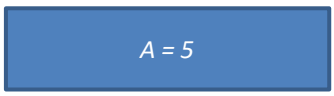
Kode Pseudo:

```
besar = x > y;
```

Artinya:

Operasi Komparasi, nilai isi variable x dibandingkan terhadap nilai isi variable y, hasilnya adalah nilai true atau false yang akan disimpan di variable besar

Chart:



Kode Pseudo:

```
A=5;
```

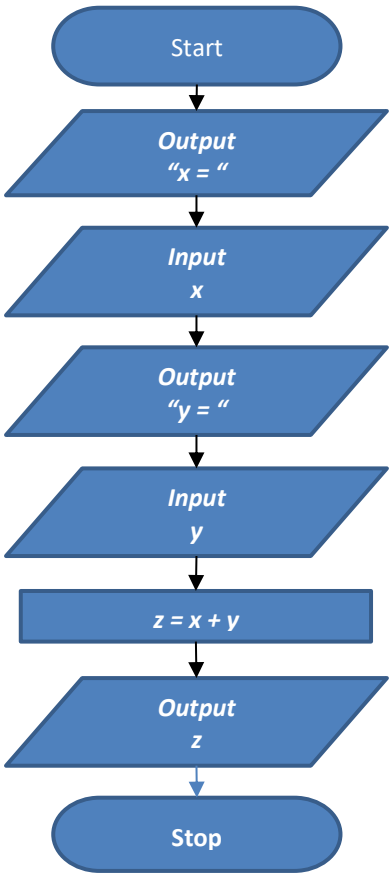
Artinya:

Operasi Inisialisasi, isi variable A diinisialisasi atau diisi dengan nilai 5

Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.15. Contoh simbol proses dan kode pseudo untuk operasi komparasi dan operasi inisialisasi/deklarasi

Contoh flowchart dengan statement terminator, input, output dan proses:



Gambar 2.16. Contoh flowchart dengan statement terminator, input, output dan proses

Contoh dan Flashback ke Langkah–langkah Pengembangan Program

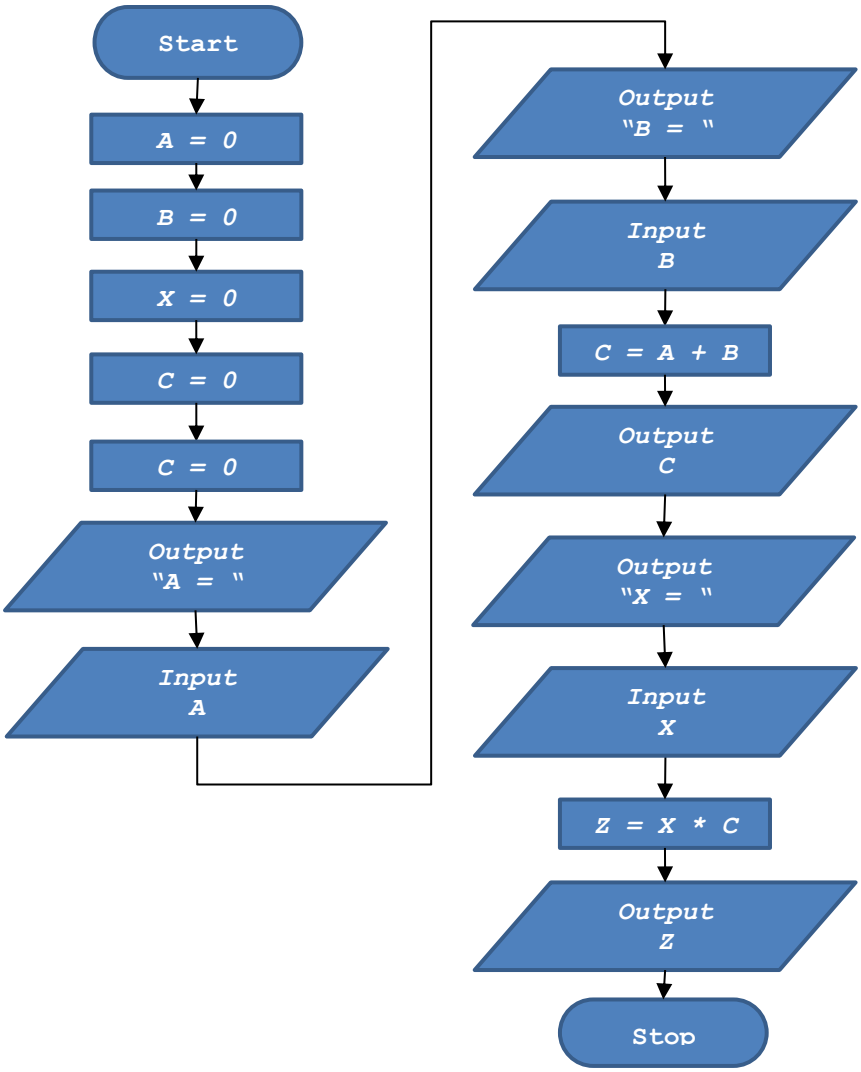
1. Definisikan masalah
2. Rancang outline pemecahan masalah
3. **Buat algoritma berdasarkan outline pemecahan masalah**
4. Test algoritma
5. Coding
6. Execute
7. Dokumentasi dan pemeliharaan

Sekarang kita sudah bisa lanjutkan langkah-langkah pengembangan program ke **langkah nomor 3**, karena kita sudah mempelajari cara penyajian algoritma, baik itu secara flowchart maupun secara kode pseudo.

Berdasarkan hasil terakhir terkait kasus pada sesi sebelumnya kita sudah membuat sampai langkah 2 yaitu outline pemecahan masalah sebagai berikut:

```
Deklarasi variable A
Deklarasi variable B
Deklarasi variable X
Deklarasi variable C
Deklarasi variable Z
Tampilkan tampilan untuk memasukkan nilai A
Baca nilai A
Tampilkan tampilan untuk memasukkan nilai A
Baca nilai B
C  = A + C
Tampilkan nilai C
Tampilkan tampilan untuk memasukkan nilai X
Baca nilai X
Z = X * C
Tampilkan nilai Z
```

Maka algoritmanya dalam flowchart adalah:



Gambar 2.17. Flowchart berdasarkan outline pemecahan masalah kasus perhitungan di atas

Summary Notes of Sesi 2.3 - Fondasi Pemrograman & Struktur Data – 03 Flow Chart, Selection Statement.

Seperti yang sudah dibahas pada materi sesi sebelumnya bahwa mekanisme aliran (*flow*) proses pada teorema *structured control* tersebut adalah:

- Mekanisme aliran berurutan → *sequence*
- Mekanisme aliran percabangan → *selection* (dengan ***if-else*** dan ***case***)
- Mekanisme aliran perulangan (*loop*) → *Repetition* (dengan ***for*** dan ***while***)

Pada materi bahasan kali ini kita akan bahas tentang mekanisme aliran percabangan yaitu *selection*. *Selection* secara umum terdiri dari dua jenis, yaitu; *if-else* dan *case*

Selection statement

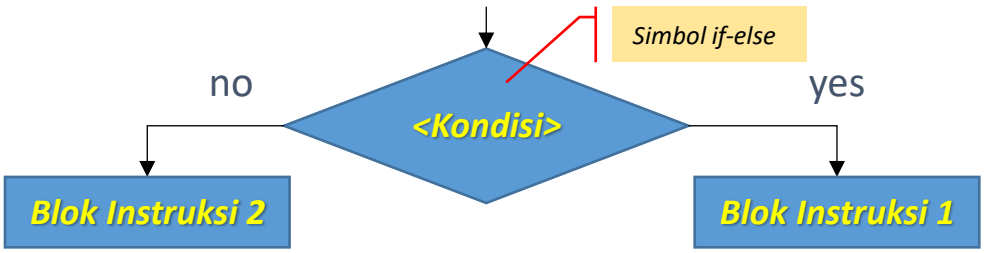
Selection merupakan mekanisme aliran instruksi (*instruction flows*) dimana instruksi yang dikerjakan sesuai kondisi yang disyaratkan sebelumnya. Dalam *selection* terdapat dua jenis atau mekanisme kendali alur, yaitu:

- *if-else*
- *case*

Selection if-else

Merupakan mekanisme kendali alur instruksi dimana arah alur instruksi tergantung pada kondisi. Terdapat dua kondisi yang menentukan alur instruksi yaitu; ***true*** (bisa dikatakan kondisi **terpenuhi**) atau ***false*** (bisa dikatakan kondisi ***tidak terpenuhi***). Misal; jika kondisi adalah true (terpenuhi) maka blok instruksi 1 yang dijalankan, jika kondisi adalah false (tidak terpenuhi) maka blok instruksi 2 yang dijalankan.

Statemen *selection if-else* berfungsi sebagai kendali alur instruksi untuk mengendalikan alur instruksi yang hanya mensyaratkan dua kondisi, yaitu; *true* atau *false*. Dalam *flow chart* untuk menandakan instruksi selection if-else digunakan simbol sebagai berikut:



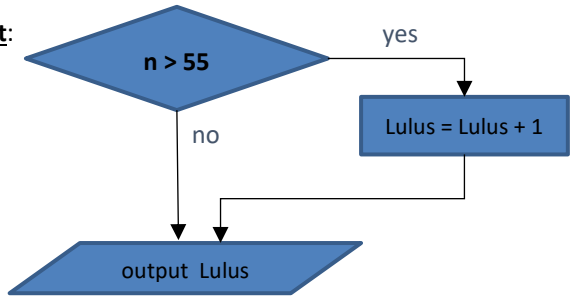
Gambar 2.18. Simbol selection if-else

Simbol flowchart tersebut digunakan sebagai representasi bahwa proses selanjutnya yang adalah tergantung dari nilai kondisi. Hasil pengujian kondisi biasanya berupa nilai true (benar/terpenuhi) atau false (salah/tidak terpenuhi). Jika hasil pengujian kondisi adalah true maka alur proses selanjutnya adalah alur dengan label yes, jika hasil pengujian kondisi adalah false maka alur proses selanjutnya adalah alur dengan label no. Dalam Bahasa pemrograman akan diterjemahkan dengan:

- Kode instruksi if( )
- Kode instruksi if( )-else
- Kode instruksi if( )-else if( )-else

Contoh if( ):

Chart:



Kode Pseudo:

```
if(n>55) {
    Lulus = Lulus + 1;
}
ouput Lulus;
```

Artinya:

Jika nilai variable n lebih besar dari 55 maka jalankan operasi matematika **Lulus=Lulus+1**, setelah itu tampilkan nilai yang tersimpan pada variable **Lulus**.

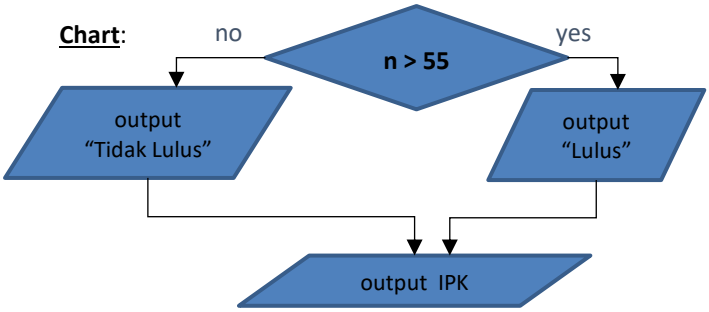
Jika nilai variable n tidak lebih besar dari 55 (lebih kecil atau sama dengan 55) maka langsung tampilkan nilai yang tersimpan pada variable **Lulus**..

Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.19. Contoh simbol if() dan kode pseudo

Contoh if()-else:

Chart:



Kode Pseudo:

```
if(n>55) {
    output Lulus;
}
else {
    ouput Tidak Lulus;
}
Output IPK;
```

Artinya:

Jika nilai variable n lebih besar dari 55 maka tampilkan tulisan **Lulus**, setelah itu tampilkan nilai yang tersimpan pada variable **IPK**.

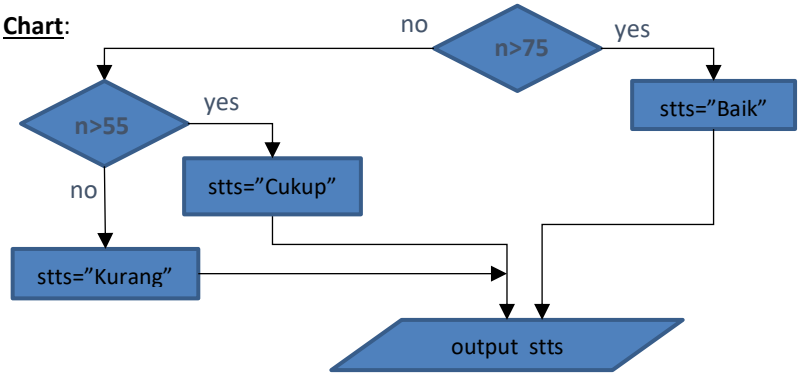
Jika nilai variable n tidak lebih besar dari 55 (lebih kecil atau sama dengan 55) maka tampilkan tulisan **Tidak Lulus**, setelah itu tampilkan nilai yang tersimpan pada variable **IPK**.

Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.20. Contoh simbol if()-else dan kode pseudo

Contoh if()-else if()-else:

Chart:



Kode Pseudo:

```
if(n>75) {
    stts = "Baik";
}
else if(n>55){
    stts = "Cukup";
}
else {
    stts = "Kurang";
}
output stts;
```

Artinya:

Jika nilai variable n lebih besar dari 75 maka isi variable **stts** dengan nilai **"Baik"**, setelah itu tampilkan nilai yang tersimpan pada variable **stts**.

Jika nilai variable n tidak lebih besar dari 75 (lebih kecil atau sama dengan 75) maka isi variable **stts** dengan nilai **"Cukup"**, setelah itu tampilkan nilai yang tersimpan pada variable **stts**.

Jika nilai variable n tidak lebih besar dari 55 (lebih kecil atau sama dengan 55) maka isi variable **stts** dengan nilai **"Kurang"**, setelah itu tampilkan nilai yang tersimpan pada variable **stts**.

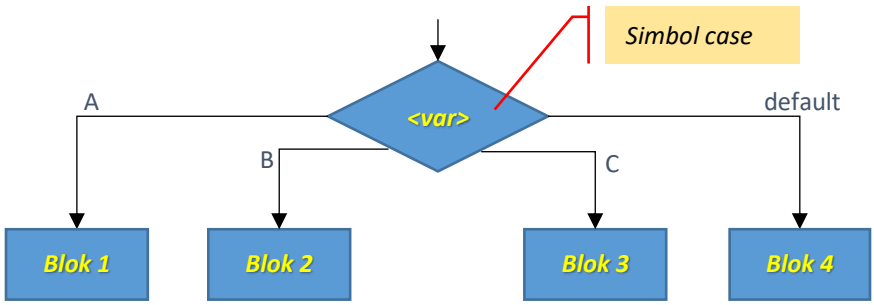
Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.21. Contoh simbol if()-else if()-else dan kode pseudo

Selection case

Merupakan mekanisme kendali alur instruksi dimana arah alur instruksi tergantung pada kondisi. Berbeda dengan *selection if-else* yang hanya terdapat dua kondisi dalam menentukan alur instruksi, dengan *selection case* jumlah kondisi yang disyaratkan bisa lebih dari dua (disesuaikan kebutuhan). Pada *selection case* dilakukan pengujian isi suatu variabel terhadap beberapa nilai yang ditetapkan, jika ditemukan kecocokan isi variabel tersebut dengan suatu nilai maka kondisi tersebut dinyatakan terpenuhi dan alur instruksi akan masuk ke blok instruksi yang mensyaratkan nilai tersebut. Jika tidak ditemukan kecocokan sisi variabel tersebut dengan setiap nilai yang disyaratkan maka alur instruksi akan masuk ke blok instruksi default.

Statemen *selection case* berfungsi sebagai kendali alur instruksi untuk mengendalikan alur instruksi yang mensyaratkan beberapa kondisi (sesuai nilai syarat yang ditetapkan). Dalam *flow chart* untuk menandakan instruksi selection case digunakan simbol sebagai berikut:



Gambar 2.22. Simbol selection case

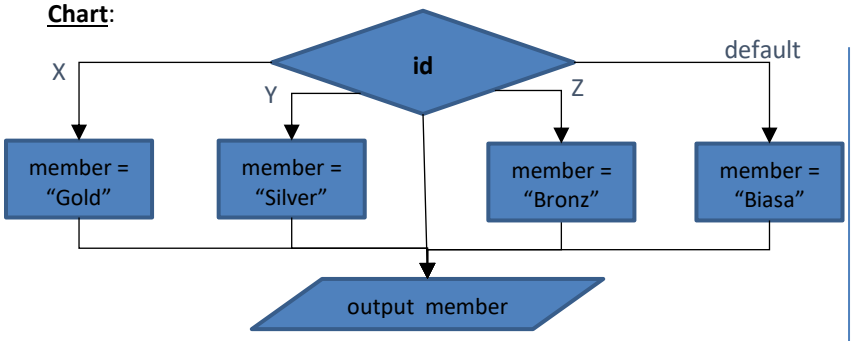
Dalam flowchart simbol tersebut digunakan sebagai representasi bahwa proses selanjutnya yang adalah tergantung dari nilai kondisi. Berikut adalah mekanisme alur pada selection case:

- Jika isi variabel **var bernilai A**, maka kondisi yang terpenuhi adalah **alur dengan label A** sehingga instruksi berikutnya adalah **blok 1**.
- Jika isi variabel **var bernilai B**, maka kondisi yang terpenuhi adalah **alur dengan label B** sehingga instruksi berikutnya adalah **blok 2**.
- Jika isi variabel **var bernilai C**, maka kondisi yang terpenuhi adalah **alur dengan label C** sehingga instruksi berikutnya adalah **blok 3**.
- Jika isi variabel **var bernilai selain A, B,** dan **C**, maka tidak ada kondisi yang terpenuhi dengan demikian yang ditempuh adalah **alur dengan label default** sehingga instruksi berikutnya adalah **blok 4**.

Dalam Bahasa pemrograman *selection case* ini biasa diterjemahkan dengan kode instruksi *switch-case* (catatan: ada beberapa bahasa pemrograman penulisannya *select-case*).

Contoh *selection-case*:

Chart:



Artinya:

Jika isi variabel **id** bernilai **X**, maka isi variable member dengan "Gold".

Jika isi variabel **id** bernilai **Y**, maka isi variable member dengan "Silver".

Jika isi variabel **id** bernilai **Z**, maka isi variable member dengan "Bronz".

Jika isi variabel id bernilai selain X, Y, dan Z, maka isi variable member dengan "Biasa".

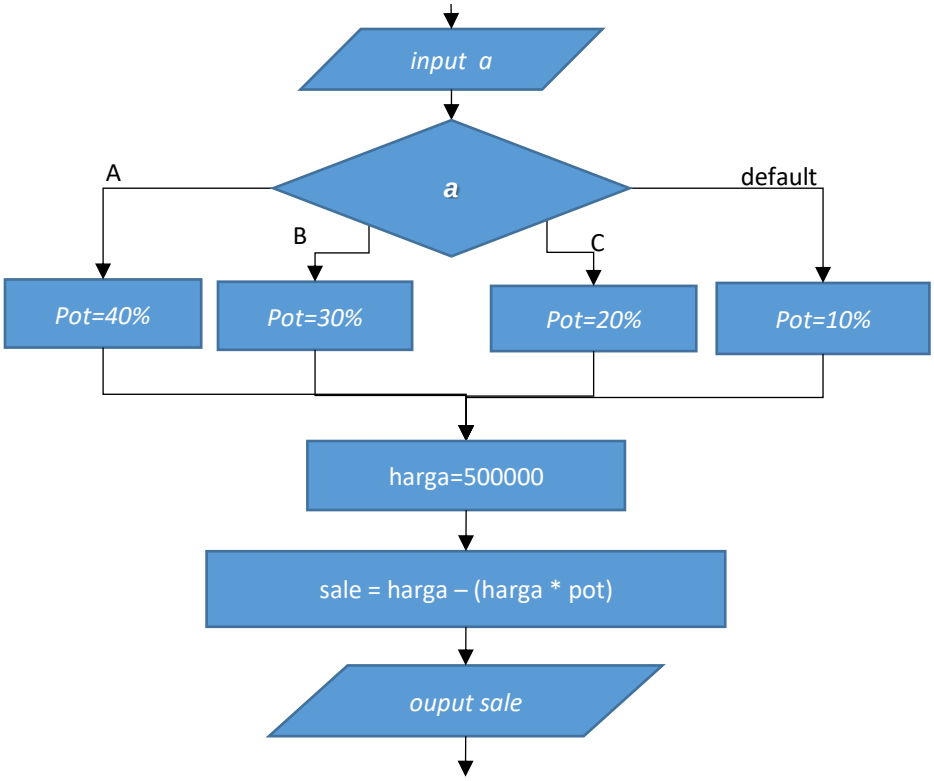
Kode Pseudo:

```
switch(var) {
    case 'X': member="Gold"; break;
    case 'Y': member="Silver"; break;
    case 'Z': member="Bronz"; break;
    default: member="Biasa";
}
ouput member;
```

Gambar 2.23. Contoh simbol select-case dan kode pseudo

Contoh *selection-case*:

Flow Chart:



Kode Pseudo:

```
input a;
switch(a) {
    case 'A': pot=40%; break;
    case 'B': pot=30%; break;
    case 'C': pot=20%; break;
    default: pot=10%;
}
harga=500000;
sale=harga-(harga*pot);
output sale;
```

Gambar 2.23. Contoh flow chart select-case dan kode pseudo

Summary Notes of Sesi 2.4 - Fondasi Pemrograman & Struktur Data – 04 Flow Chart, Repetition Statement.

Seperti yang sudah dibahas pada materi sesi sebelumnya bahwa mekanisme aliran (*flow*) proses pada teorema *structured control* tersebut adalah:

- Mekanisme aliran berurutan → *sequence*
- Mekanisme aliran percabangan → *selection* (dengan ***if-else*** dan ***case***)
- Mekanisme aliran perulangan (*loop*) → *Repetition* (dengan ***for*** dan ***while***)

Pada materi bahasan kali ini kita akan bahas tentang mekanisme aliran perulangan yaitu *repetition*. *Repetition* secara umum terdiri dari dua jenis, yaitu; *for* dan *while*

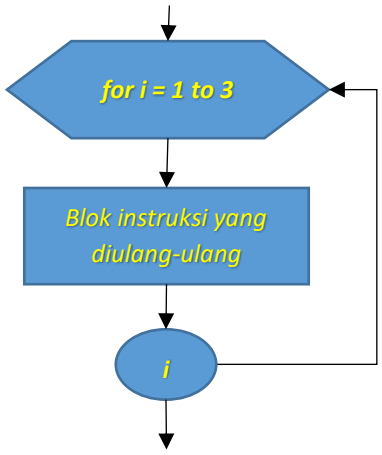
Repetition statement

Repetition merupakan mekanisme aliran instruksi (*instruction flows*) dimana instruksi yang dikerjakan diulang-ulang. Dalam *repetition* terdapat dua jenis atau mekanisme kendali alur, yaitu:

- *for*
- *while*

Repetition for

Merupakan mekanisme kendali alur instruksi dimana instruksi-instruksi yang berada di dalam perulangan tersebut akan diulang-ulang sampai kententuan yang ditetapkan tercapai atau dipenuhi. Dalam *flow chart* untuk menandakan instruksi *repetition for* digunakan simbol sebagai berikut:



Gambar 2.24. Simbol repetition for

Simbol *flow chart* tersebut digunakan sebagai representasi bahwa blok instruksi yang berada di dalam *repetition for* akan diulang-ulang (keterangan: pada gambar terletak **di antara simbol for dan simbol lingkaran dengan huruf i**). **Faktor yang menentukan banyaknya perulangan yang dilakukan adalah variabel i**, perulangan akan dilakukan mulai dari variabel i bernilai 1 sampai dengan 3 dan setiap satu kali perulangan nilai variabel i akan selalu ditambah 1. Perulangan akan berhenti jika nilai variabel i sudah lebih dari 3.

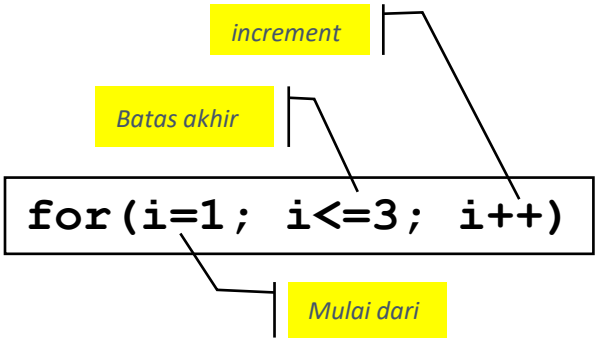
Untuk penulisan variabel di dalam simbol *for* bisa beragam variabel yang digunakan, dengan demikian maka blok instruksi yang diulang adalah instruksi yang berada di antara simbol;

- ***for i*** dan **lingkaran i**, atau
- ***for x*** dan **lingkaran x**, atau
- ***for z*** dan **lingkaran z**,
- dan sebagainya secara berpasangan.

Dalam Bahasa pemrograman penulisan repetition for akan diterjemahkan dengan instruksi perulangan ***for()*** seperti berikut:

```
for(i=1; i<=3; i++)  {
      … instruksi-instruksi yang akan diulang-
          ulang
}
Instruksi berikutnya setelah perulangan;
```

Penjelasannya:

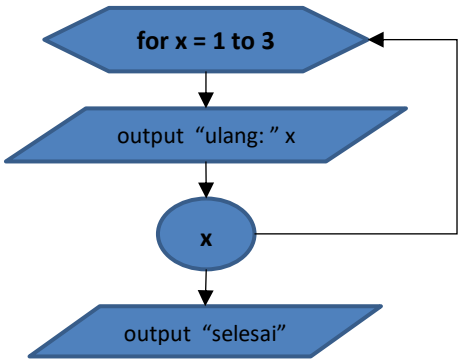


Gambar 2.25. Penulisan repetition for dan penjelasannya

Berdasarkan gambar di atas, maka perulangan akan dilakukan mulai dari variabel i bernilai 1 dan selagi variabel i masih bernilai lebih kecil atau sama dengan 3, serta setiap satu kali perulangan nilai variabel i akan ditambah 1.

Contoh for( ):

Chart:



Kode Pseudo:

```
for(x=1; x<=3; x++) {
    output "ulang" + x;
}
output "selesai";
```

Artinya:

Untuk **x** dengan nilai dimulai dari **1 sampai 3** lakukan perulangan terhadap instruksi-instruksi berikut:

output "ulang";

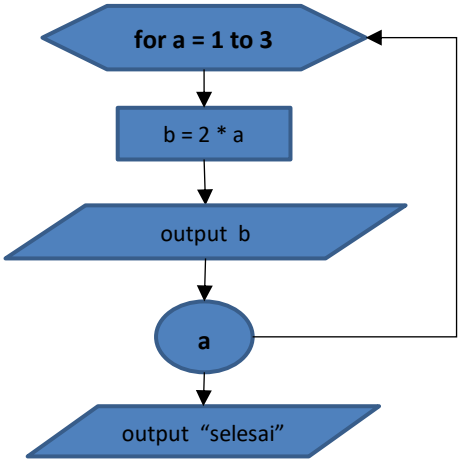
Hasilnya:

ulang 1
ulang 2
ulang 3
selesai

Keterangan: kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.26. Contoh 1 simbol for() dan kode pseudo

Chart:



Kode Pseudo:

```
for(x=1; x<=3; x++) {
    b=2*a;
    output b;
}
output "selesai";
```

Artinya:

Untuk **a** dengan nilai dimulai dari **1 sampai 3** lakukan perulangan terhadap instruksi-instruksi berikut:

b=2*a;
output b;

Hasilnya:

2
4
6
selesai

Keterangan: kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.27. Contoh 2 simbol for() dan kode pseudo

Repetition while

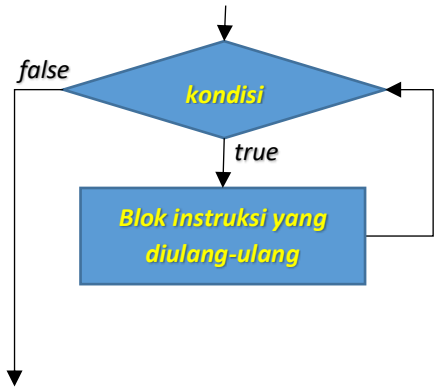
Merupakan mekanisme kendali alur instruksi dimana instruksi-instruksi yang berada di dalam perulangan tersebut akan diulang-ulang sampai kondisi yang ditetapkan tercapai atau dipenuhi. Terdapat dua jenis *repetition while*, yaitu;

- *while()*
- *do-while()*

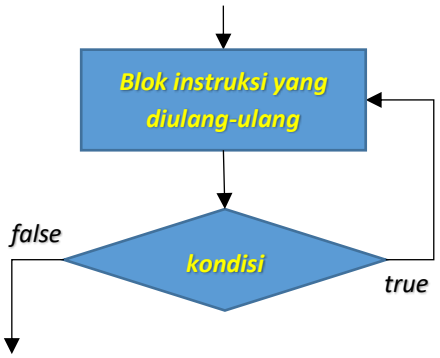
Faktor yang membedakan antara ***while()*** dan ***do-while()*** adalah faktor pengujian kondisi. Untuk repetition jenis ***while()*** kondisi akan diuji terlebih dahulu (di depan) sebelum alur memasuki blok instruksi yang akan diulang, sedangkan pada repetition jenis ***do-while()*** kondisi akan diuji di akhir atau setelah alur memasuki blok instruksi yang akan diulang. Jadi untuk repetition jenis **do-while()**, blok instruksi yang akan diulang minimal dijalankan 1 kali walaupun kondisi di dalam *while* tidak terpenuhi (perhatikan gambar *flow chart* di bawah).

Hal yang harus menjadi perhatian utama dalam penggunaan *repetition while* adalah jangan pernah lupa untuk menyertakan instruksi yang bisa membuat kondisi pada while tidak terpenuhi di dalam blok instruksi yang akan diulang.

Dalam *flow chart* untuk menandakan instruksi repetition for digunakan simbol sebagai berikut:



*Gambar 2.28. Simbol repetition **while()***

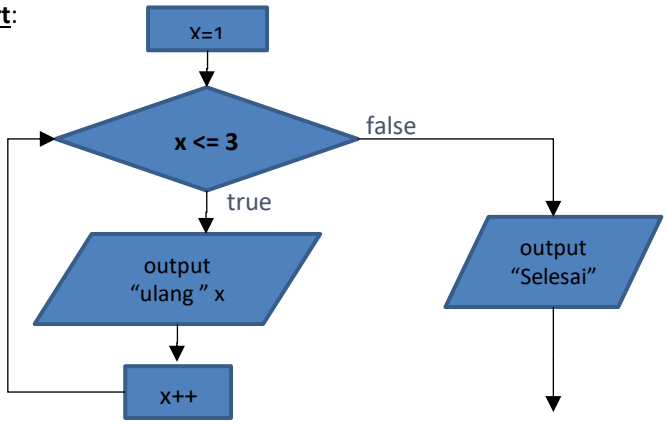


*Gambar 2.29. Simbol repetition **do-while()***

Simbol *flow chart* tersebut digunakan sebagai representasi bahwa blok instruksi yang berada di dalam *repetition while* akan diulang-ulang (keterangan: blok instruksi yang berada pada alur ***true***). Faktor yang menentukan banyaknya perulangan yang dilakukan adalah terpenuhinya kondisi yang ditetapkan pada *while*, artinya selagi kondisi pada *while* masih terpenuhi maka perulangan terus dilakukan. **Perulangan akan berhenti jika kondisi pada *while* sudah tidak terpenuhi**.

Contoh ***while()***:

Chart:



Kode Pseudo:

```
x=1;
while(x<=3) {
    output "ulang" + x;
    x++;
}
output "selesai";
```

Artinya:

Untuk x selagi memiliki nilai lebih kecil atau sama dengan 3 lakukan perulangan terhadap instruksi-instruksi berikut:

output "ulang";
x++;

Hasilnya:

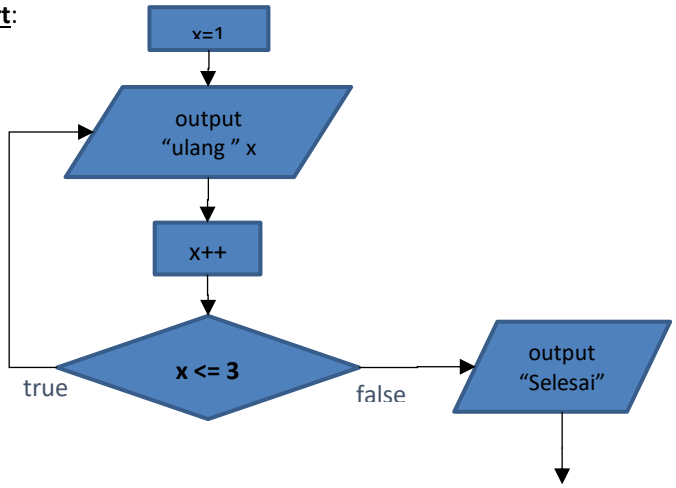
ulang
ulang
ulang
selesai

Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.30. Contoh simbol while() dan kode pseudo

Contoh **do-while()**:

Chart:



Artinya:

Untuk x selagi memiliki nilai lebih kecil atau sama dengan 3 lakukan perulangan terhadap instruksi-instruksi berikut:

output "ulang";
x++;

Hasilnya:

ulang 1
ulang 2
ulang 3
selesai

Kode Pseudo:

```
x=1;
do {
    output "ulang" + x;
    x++;
} while(x<=3)
output "selesai";
```

Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

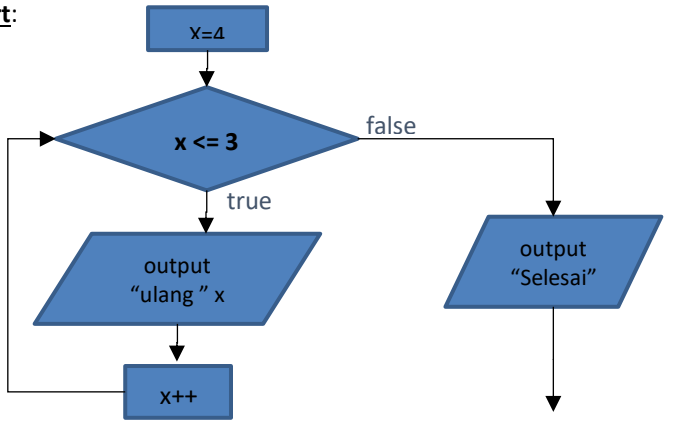
Gambar 2.31. Contoh simbol do-while() dan kode pseudo

Hal yang harus menjadi perhatian utama dalam penggunaan *repetition while* adalah jangan pernah lupa untuk **menyertakan instruksi yang bisa membuat kondisi pada while tidak terpenuhi** di dalam blok instruksi yang akan diulang. Instruksi yang bisa menyebabkan kondisi pada *while* tidak terpenuhi bisa beragam, dan hal ini tergantung kondisi yang ditetapkan pada *while*. Misal mengacu pada gambar 2.30 dan gambar 2.31, **kondisi yang ditetapkan pada *while* adalah x<=3, kondisi ini bisa tidak terpenuhi jika nilai x berubah hingga menjadi lebih besar dari 3, dalam hal ini instruksi tersebut adalah x++**.

Untuk melihat perbedaan *repetition while()* dengan *repetition do-while()* mari kita lihat contoh berikut:

Contoh ***while()*** kalau kondisi pada *while* sudah tidak terpenuhi sejak awal:

Chart:



Kode Pseudo:

```
x=4;
while(x<=3) {
    output "ulang" + x;
    x++;
}
output "selesai";
```

Artinya:

Untuk x selagi memiliki nilai lebih kecil atau sama dengan 3 lakukan perulangan terhadap instruksi-instruksi berikut:

output "ulang";
x++;

Namun **karena x sejak awal sudah bernilai 4** (lebih besar dari 3, maka instruksi tersebut tidak dieksekusi.

Hasilnya:

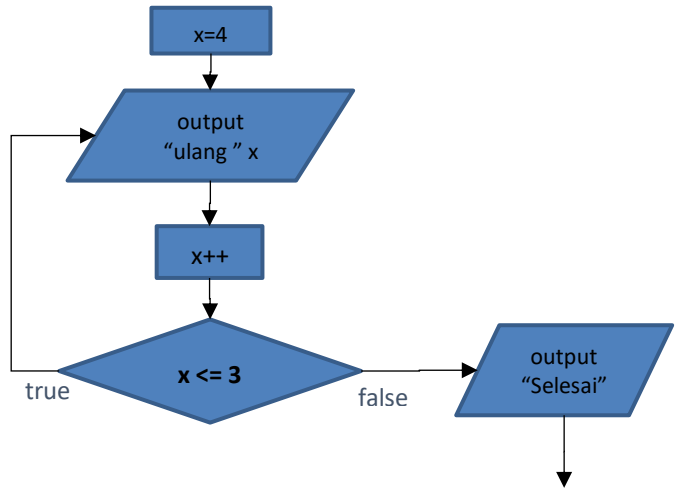
selesai

Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.32. Contoh while() jika kondisi pada while tidak terpenuhi sejak awal

Contoh ***do-while()*** kalau kondisi pada *while* sudah tidak terpenuhi sejak awal:

Chart:



Artinya:

Untuk x selagi memiliki nilai lebih kecil atau sama dengan 3 lakukan perulangan terhadap instruksi-instruksi berikut:

output “ulang”;
x++;

Hasilnya:

ulang 4
selesai

Kode Pseudo:

```
x=1;
do {
    output “ulang” + x;
    x++;
} while(x<=3)
output “selesai”;
```

Keterangan:
kode pseudo: kode yang menyerupai kode pemrograman.

Gambar 2.33. Contoh simbol do-while() jika kondisi pada while tidak terpenuhi sejak awal

Berdasarkan kedua contoh di atas (gambar 2.32 dan gambar 2.33) dapat kita lihat, bahwa hasil pada **gambar 2.32 hanya berupa tulisan selesai**, sedangkan pada **gambar 2.33 adalah tulisan ulang 4 dan selesai**.

Hal tersebut dapat terjadi karena pada gambar 2.32 (*repetition while()*) pemeriksaan kondisi dilakukan di awal sedangkan nilai x adalah 4 sehingga kondisi pada *while* sudah tidak terpenuhi, maka instruksi yang berada di dalam perulangan (yaitu; *output* “ulang”; dan x++;) tidak dieksekusi sama sekali, dan alur langsung menuju instruksi *output* “selesai”.

Sedangkan pada gambar 2.33 (*repetition do-while()*) pemeriksaan kondisi dilakukan di akhir, walaupun nilai x adalah 4 namun instruksi yang berada di dalam perulangan (yaitu; *output* “ulang”; dan x++;) tetap dieksekusi, setelah itu baru dilakukan pemeriksaan kondisi pada *while*, karena terdeteksi kondisi tidak terpenuhi maka selanjutnya tidak ada perulangan, dan alur langsung menuju instruksi *output* “selesai”.