Algoritma & Pemrograman

Pertemuan 3: Struktur Kontrol Pemrograman

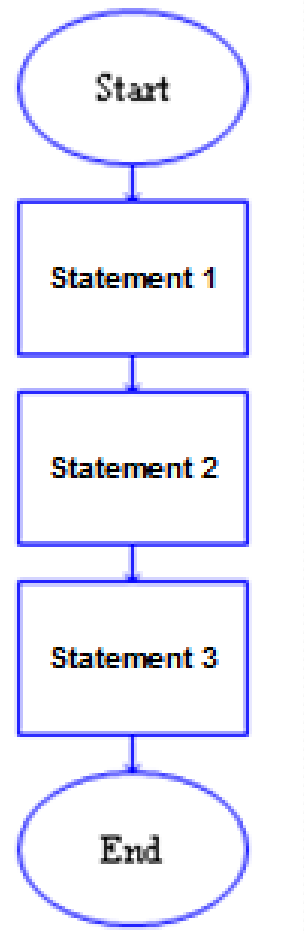
Pendahuluan

- Salah satu aspek terpenting dalam pemrograman adalah mengatur pernyataan/statement mana yang akan dieksekusi berikutnya
- Struktur kontrol memungkinkan seorang programmer untuk menentukan urutan eksekusi statement dalam sebuah program. Struktur ini memungkinkan 2 hal:
  - Melewatkan (tidak mengerjakan) beberapa statement, dan mengerjakan beberapa statement lainnya
  - Mengulangi satu atau lebih statement selama sebuah kondisi masih bernilai benar/true

(1) Kontrol Program Sekuensial

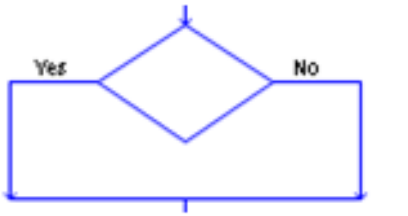
- Sekuensial: berurutan, statement dieksekusi satu per satu
- Sekuensial merupakan bentuk kontrol yang paling mudah dibuat dan ditelusuri
- Pada intinya, letakkan setiap statement pada urutan dimana kita ingin mereka dieksekusi dan program akan mengeksekusi mereka secara berurutan mulai dari bagian Start sampai bagian End

(1) Kontrol Sekuensial



- Peletakan urutan statement berpengaruh terhadap jalannya sebuah program.
  - Contoh: Jika kita ingin mengambil data dari user lalu memprosesnya, maka GET data harus dilakukan terlebih dahulu sebelum kita dapat menggunakannya. Jika urutannya dibalik, maka akan menghasilkan program yang tidak valid.
- Bentuk sekuensial, walaupun sederhana dan mudah, tidak menggambarkan situasi permasalahan “dunia nyata”

(2) Kontrol Seleksi

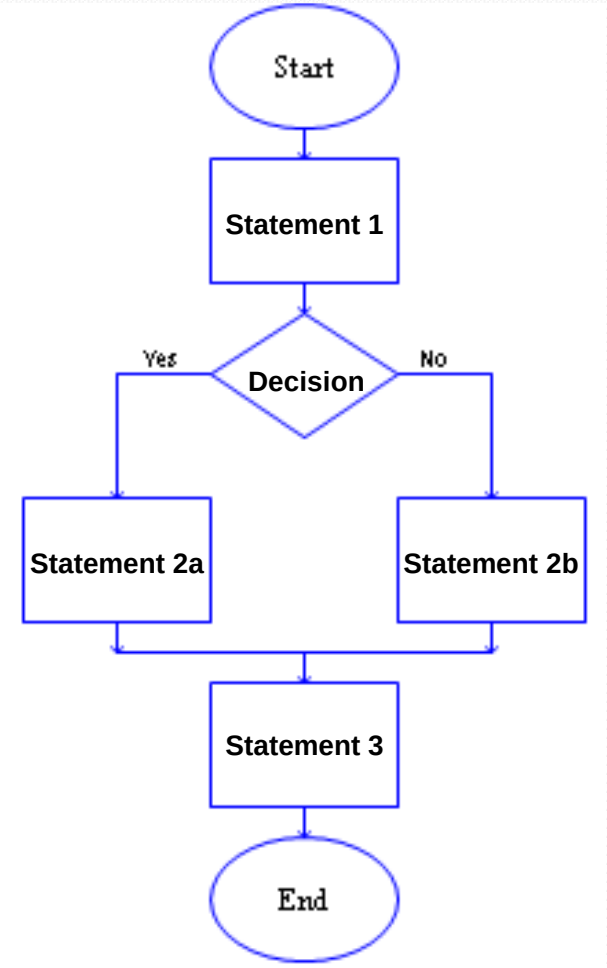


- Terkadang kita perlu membuat sebuah keputusan terhadap sebuah kondisi data pada program kita untuk menentukan apakah sekelompok statement perlu dieksekusi atau tidak
- Contoh: Jika kita ingin menghitung kemiringan dari sebuah garis dengan menggunakan statement:

  `kemiringan ← dy/dx`,

  maka kita harus memastikan bahwa nilai `dx` tidak nol (karena dapat menghasilkan *pembagian dengan nol* yang secara matematis tidak terdefinisikan dan akan menghasilkan error). Sehingga, keputusan yang perlu dibuat adalah "Apakah dx bernilai nol?"

(2) Kontrol Seleksi (lanjutan)



- Kontrol seleksi akan memilih sebuah keputusan terhadap kondisi sebuah data dan melanjutkan eksekusi program ke salah satu dari dua alternatif jalur menuju statement berikutnya
- Pada contoh disamping, perhatikan bahwa statement 2a dan 2b tidak akan dieksekusi pada waktu yang bersamaan, bergantung kepada nilai hasil decision (Yes atau No)

(2) Kontrol Seleksi (lanjutan)

- Sebuah kontrol seleksi memerlukan sebuah “ekspresi” yang dapat dievaluasi nilai kebenarannya menjadi True atau False (Yes atau No).
- Ekspresi sebuah keputusan merupakan kombinasi antara sejumlah nilai (dapat berupa konstanta atau variabel) dan operator
- Komputer hanya dapat memproses instruksi satu per satu, dan ketika sebuah ekspresi keputusan dievaluasi, operasi ekspresi tidak dieksekusi dari kiri ke kanan, tetapi berdasarkan “urutan ranking”
- Urutan operasi yang dilakukan dapat berpengaruh terhadap nilai Yes/No yang dihasilkan dan kita dapat mengatur secara eksplisit urutan tersebut dengan mengelompokkan nilai dan operator dengan menggunakan tanda kurung

Urutan Ranking Eksekusi

1. Hitung semua fungsi/function
2. Hitung semua yang ada dalam tanda kurung
3. Hitung pemangkatan (^, **)
4. Hitung perkalian, dari kiri ke kanan
5. Hitung pembagian, dari kiri ke kanan
6. Hitung penjumlahan, dari kiri ke kanan
7. Hitung pengurangan, dari kiri ke kanan
8. Evaluasi operator relasional (= != /= < <= > >=), dari kiri ke kanan
9. Evaluasi operator logika `not`, dari kiri ke kanan
10. Evaluasi operator logika `and`, dari kiri ke kanan
11. Evaluasi operator logika `xor`, dari kiri ke kanan
12. Evaluasi operator logika `or`, dari kiri ke kanan

Operator Relasional dan Logika

Operation	Description	Example
=	“sama dengan"	**3 = 4 is No(false)**
!= /=	“tidak sama dengan"	**3 != 4 is Yes(true)** **3 /= 4 is Yes(true)**
<	“kurang dari"	**3 < 4 is Yes(true)**
<=	“kurang dari atau sama dengan"	**3 <= 4 is Yes(true)**
>	“lebih dari"	**3 > 4 is No(false)**
>=	“lebih dari atau sama dengan"	**3 >= 4 is No(false)**
and	Ya(benar) jika keduanya Ya	**(3 < 4) and (10 < 20) is Yes(true)**
or	Ya(benar) jika salah satunya Ya	**(3 < 4) or (10 > 20) is Yes(true)**
xor	Ya(benar) jika nilai "ya/tidak" tidak sama	**Yes xor No is Yes(true)**
not	Balik nilai logikanya, Ya if Tidak; Tidak jika Ya	**not (3 < 4) is No(false)**

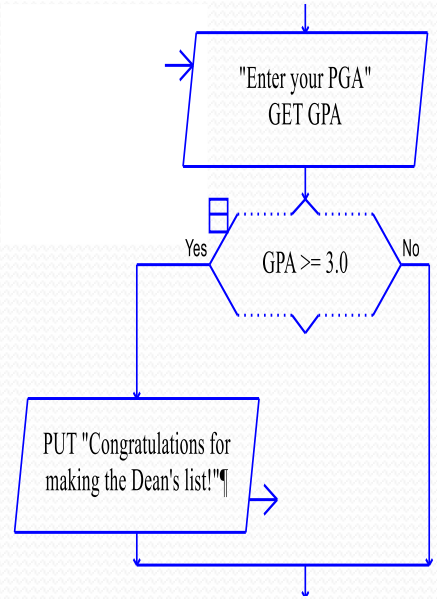
Operator relasional harus selalu membandingkan 2 nilai yang memiliki tipe data yang sama (dapat berupa angka, text, atau nilai “Yes/No”

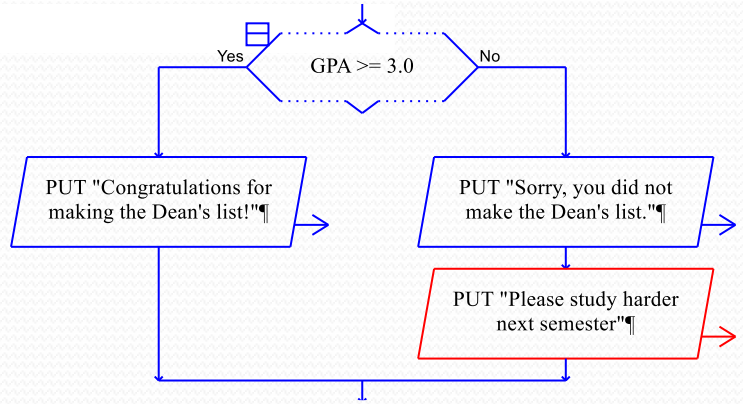
Operator logika harus selalu mengkombinasikan nilai Boolean (True/False) menjadi 1 nilai Boolean

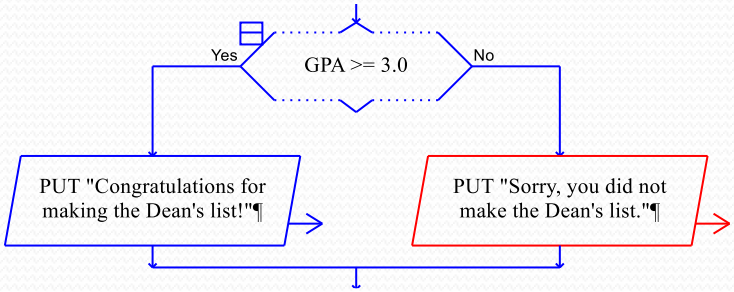
Contoh Ekspresi Valid dan Tidak Valid

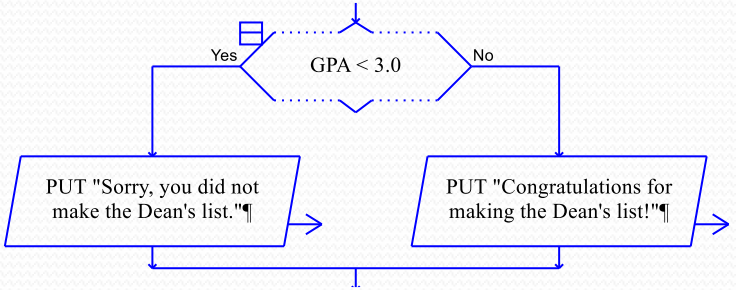
Example	Valid or Invalid?
(3<4) and (10<20)	Valid
(flaps_angle < 30) and (air_speed < 120)	Valid, anggap flaps_angle dan air_speed adalah data numerik.
5 and (10<20)	Invalid – sisi kiri "and" adalah angka, bukan nilai true/false.
5 <= x <= 7	Invalid – karena 5 <= x dievaluasi menjadi nilai true/false value lalu evaluasi true/false <= 7 merupakan perbandingan relasional yang invalid.

Contoh Kontrol Seleksi

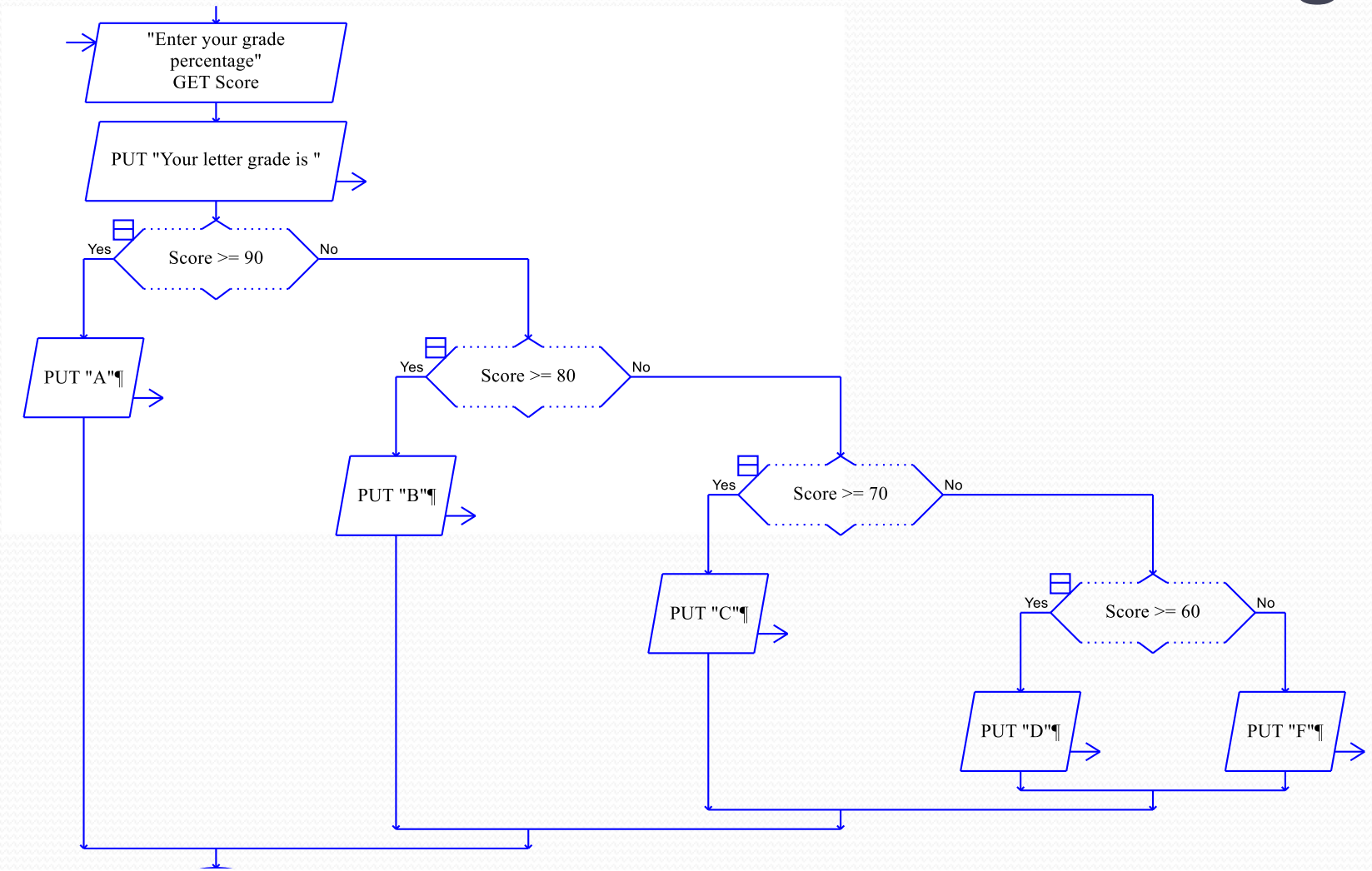






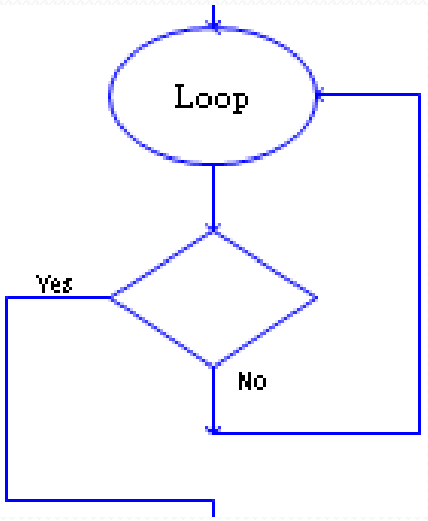


Contoh Kontrol Seleksi Bersarang



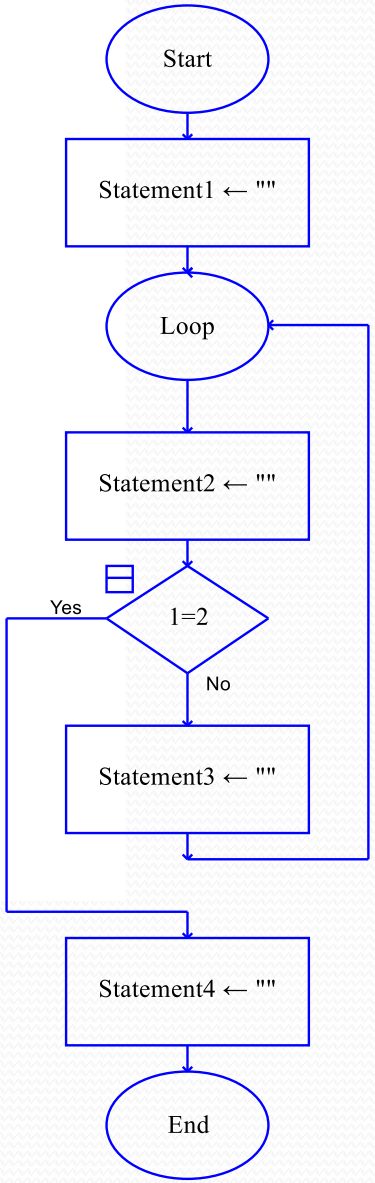
(3) Perulangan/Iterasi/Loop

- Perulangan memungkinkan kita untuk mengulangi satu atau lebih statement sampai sebuah kondisi bernilai true.
- Perulangan digambarkan dengan sebuah elips dan sebuah belah ketupat pada RAPTOR
- Banyaknya perulangan yang dilakukan oleh loop diatur oleh ekspresi keputusan yang dimasukkan dalam simbol belah ketupat
- Selama eksekusi program, ketika simbol belah ketupat dieksekusi, jika ekspresi keputusan bernilai “No”, maka cabang “No” yang diambil dan kembali lagi ke statement Loop dan perulangan (“Yes” dan “No” boleh saling bertukar tempat)
- Statement yang akan diulangi dapat ditaruh di atas atau di bawah simbol belah ketupat



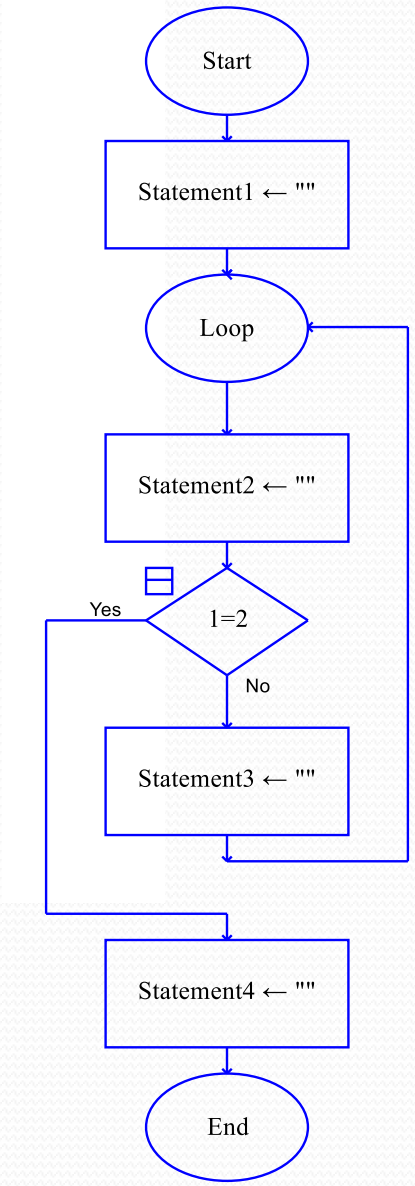
(3) Perulangan

- **Statement1** dieksekusi tepat 1 kali sebelum loop dimulai
- **Statement2** akan selalu dieksekusi setidaknya 1 kali karena ia berada sebelum statement decision
- Jika ekspresi decision bernilai "Yes", maka loop berakhir dan kontrol dilanjutkan ke **statement4**
- Jika ekspresi decision bernilai "No", maka kontrol berlanjut ke **statement3** dan **statement3** yang dieksekusi berikutnya. Lalu kontrol kembali ke atas (ke statement Loop) yang memulai kembali proses perulangan
- Perhatikan bahwa **statement2** dijamin dieksekusi setidaknya 1 kali. Perhatikan juga bahwa **statement3** ada kemungkinan tidak akan pernah dieksekusi sama sekali



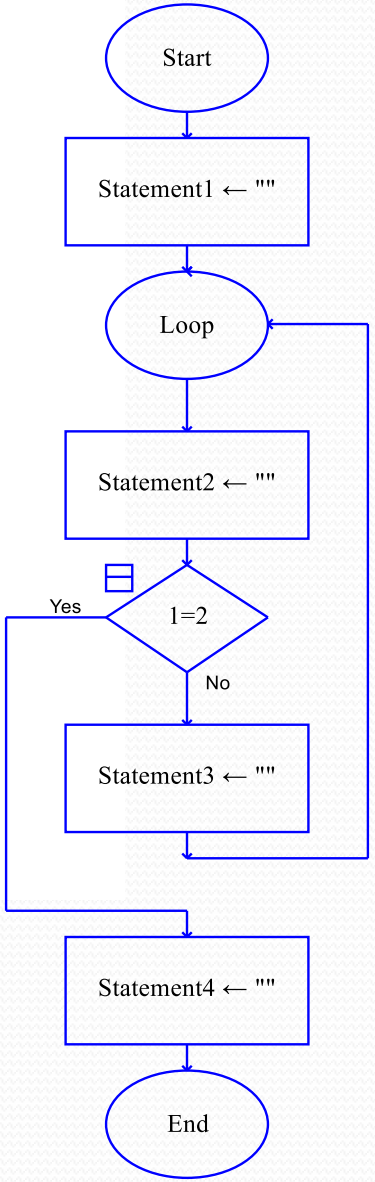
(3) Perulangan (lanjutan)

- Pada contoh di samping, statement2 dapat dihilangkan (maka decision menjadi statement pertama pada loop yang dieksekusi) atau dapat berupa sekelompok statement lainnya. Loop akan tetap dieksekusi
- Sama halnya dengan Statement3 yang dapat dihilangkan atau diganti dengan sekelompok statement lainnya
- Selain itu, semua statement di atas/di bawah statement Decision dapat berupa statement Loop lain!
- Jika ada statement Loop muncul di dalam statement Loop, ini disebut **"nested loop"**



(3) Perulangan (lanjutan)

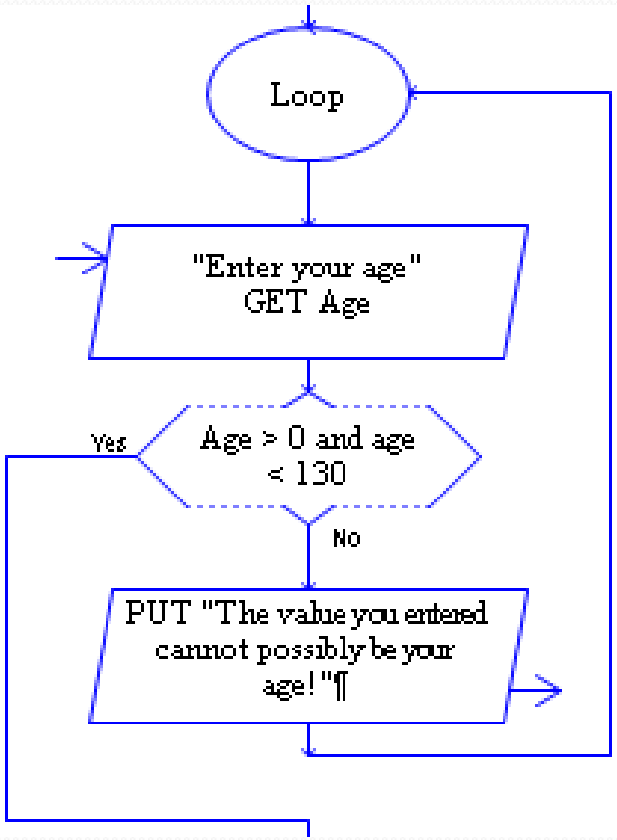
- Dapat juga dimungkinkan statement Decision tidak pernah bernilai “Yes”. Pada kondisi ini, kita akan masuk dalam keadaan “loop tak terbatas (*infinite loop*)” yang tidak akan pernah berhenti. Jika ini terjadi, kita perlu secara manual menghentikannya dengan menekan tombol “STOP” pada bar icon di RAPTOR.
- Kita harus menghindari terjadinya infinite loop dengan tidak menulis statement yang menghasilkan infinite loop tersebut.
- Caranya adalah dengan menuliskan satu (atau beberapa statement) pada Loop untuk mengubah satu atau lebih variabel dalam statement Decision sehingga pada akhirnya akan menghasilkan keputusan “Yes”
- Dengan demikian, eksekusi proses akan keluar dari loop dan melanjutkan ke statement lain hingga mencapai End.

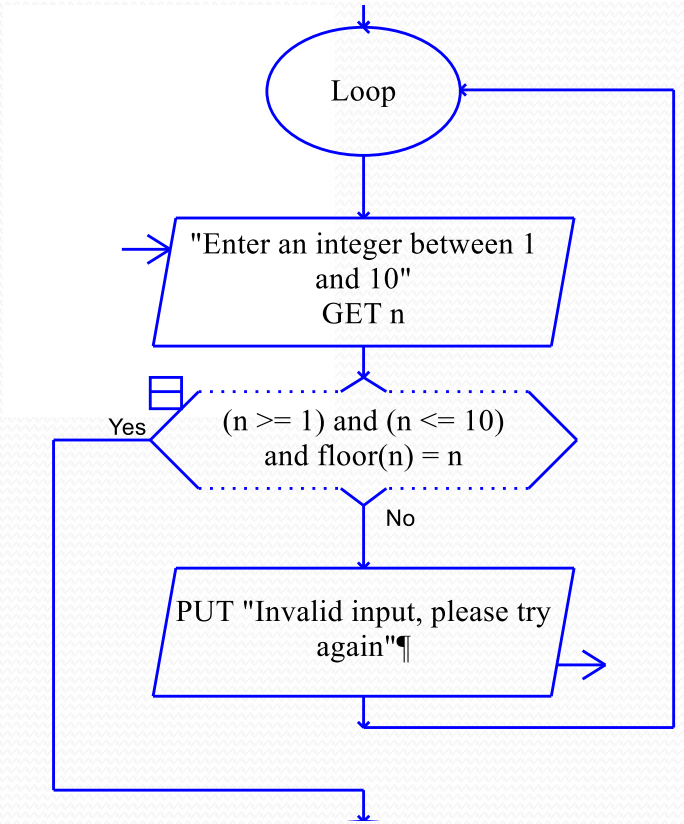


Kegunaan Loop:
(1) Untuk Validasi Input

- Untuk memastikan bahwa user telah memasukkan input data yang sesuai dengan batasan yang dapat diterima sistem, maka loop dapat digunakan untuk melakukan validasi input, sebelum input tersebut digunakan di dalam program
- Program yang melakukan validasi input user dan melakukan pengecekan error lain pada saat program tersebut dijalankan disebut sebagai program yang handal

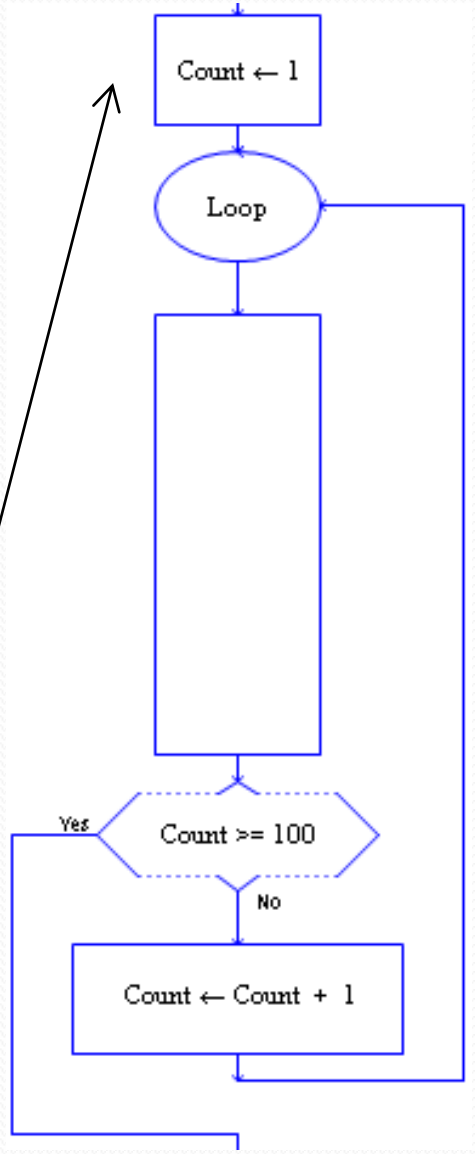
Contoh Validasi Input dengan Loop





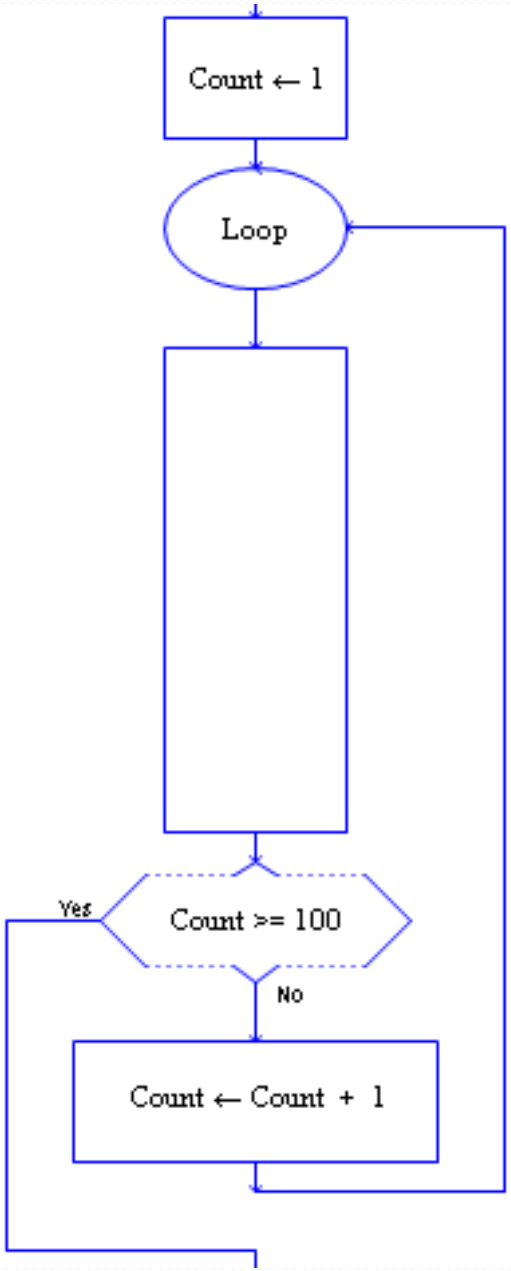
Kegunaan Loop:
(2) Mengeksekusi blok kode sebanyak beberapa kali (counting Loop)

- Membutuhkan sebuah variabel yang "naik 1 nilai" pada setiap eksekusi loop
- Pada Counting Loop ada sebuah variabel "counter" yang:
  - Diinisialisasi (diberikan nilai awal) sebelum loop dimulai
  - Dimodifikasi di dalam loop
  - Digunakan dalam ekspresi statement decision untuk menghentikan loop

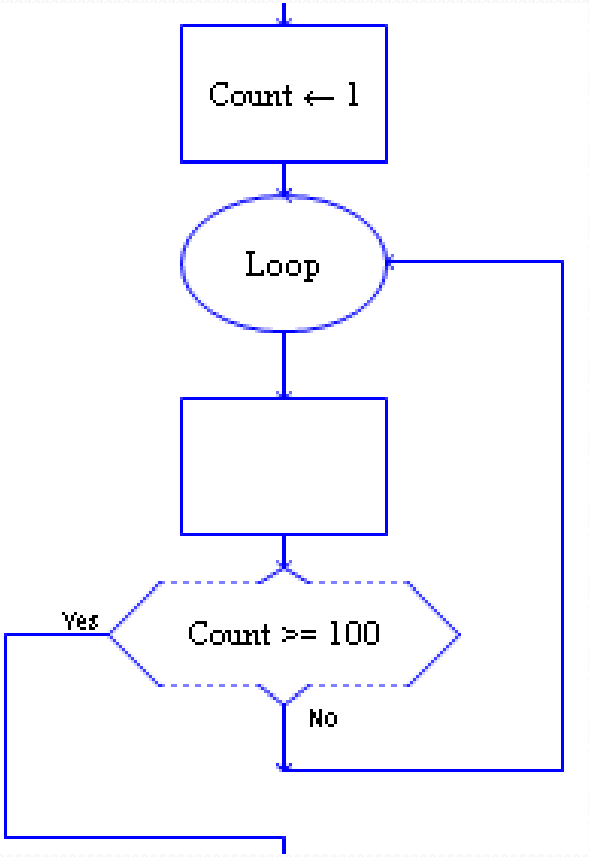


Counting Loop

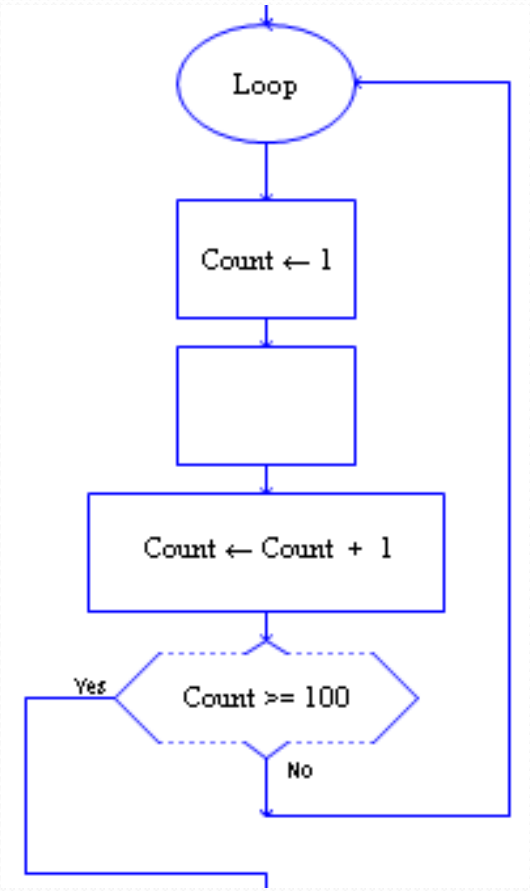
- Contoh di sebelah kanan adalah counter loop yang dieksekusi tepat 100 kali:
  - Variabel counter-nya adalah “Count”
  - Counter harus diinisialisasi sebelum loop dimulai. Pada umumnya, inisialisasi dimulai dari nilai 1 (atau dapat juga dari nilai 0)
  - Ekspresi decision yang mengatur loop perlu melakukan pengecekan “lebih besar atau sama dengan”. Ini lebih aman dari sekedar “sama dengan”
  - Loop yang diatur oleh counter biasanya menaikkan nilai variabel counter dengan 1 nilai lebih besar pada setiap eksekusi loop.



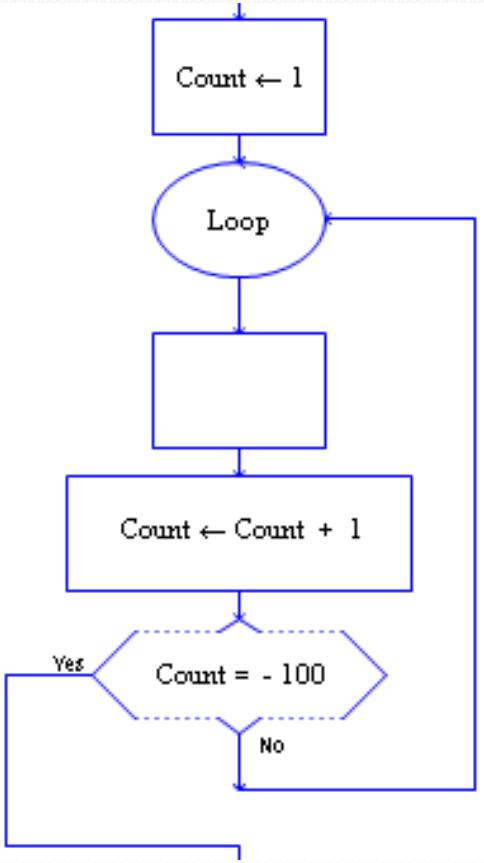
Contoh Counting Loop yang menghasilkan *loop tak berhingga*



(a)

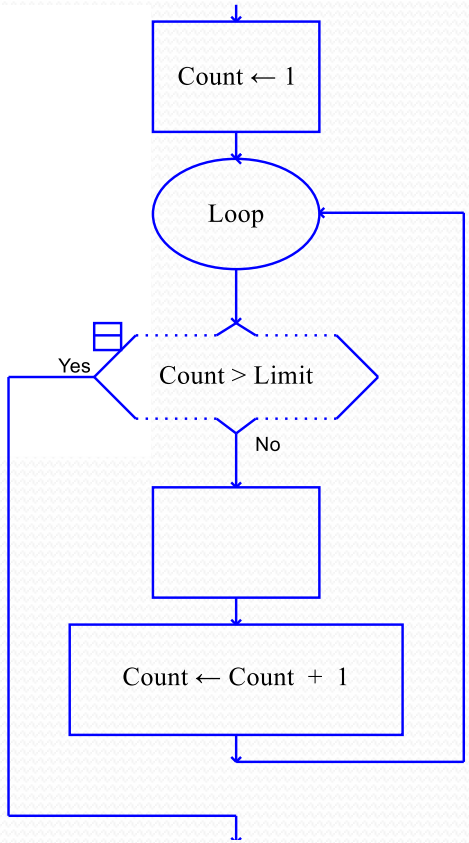


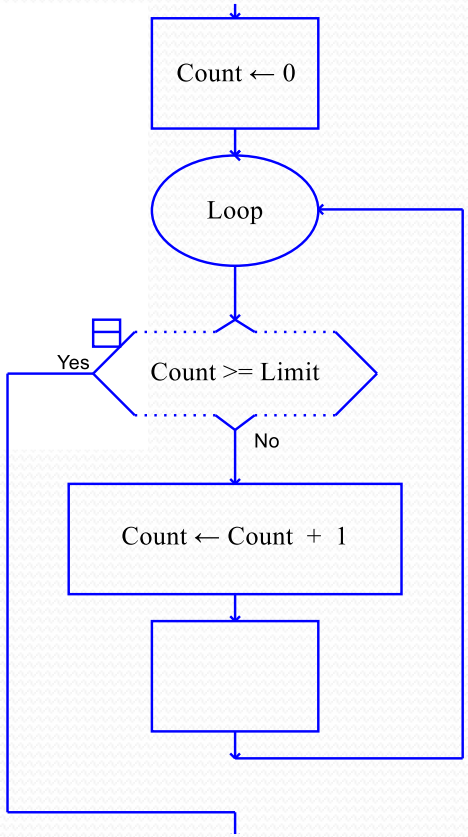
(b)

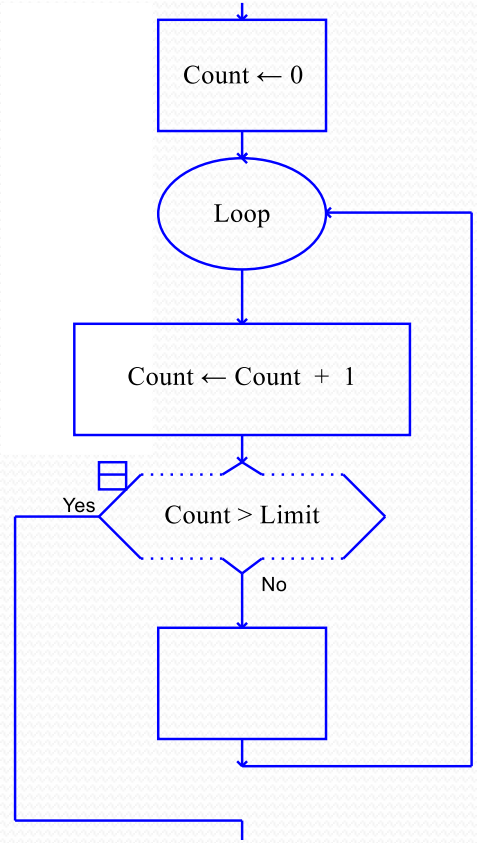


(c)

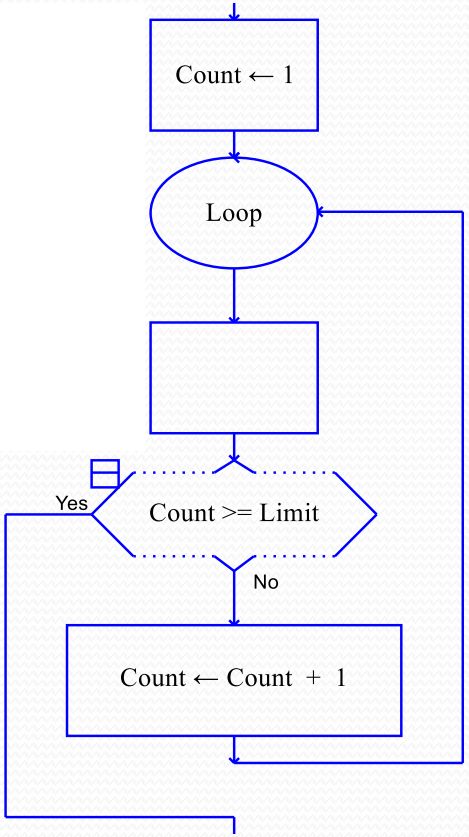
Variasi Counter Loop

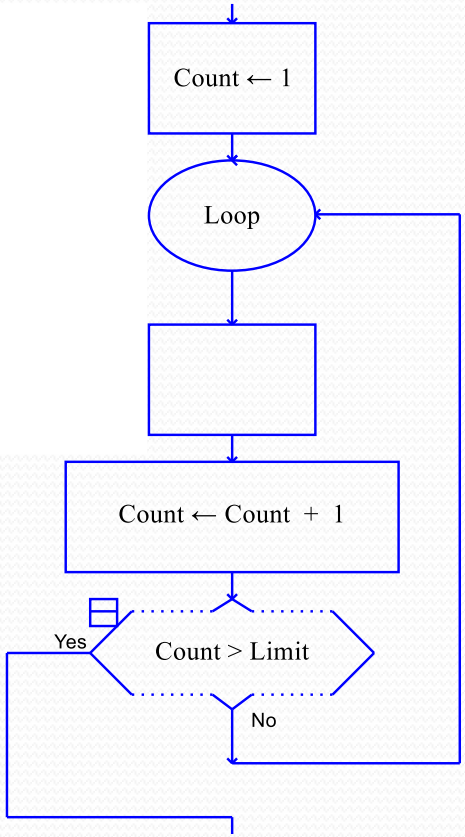


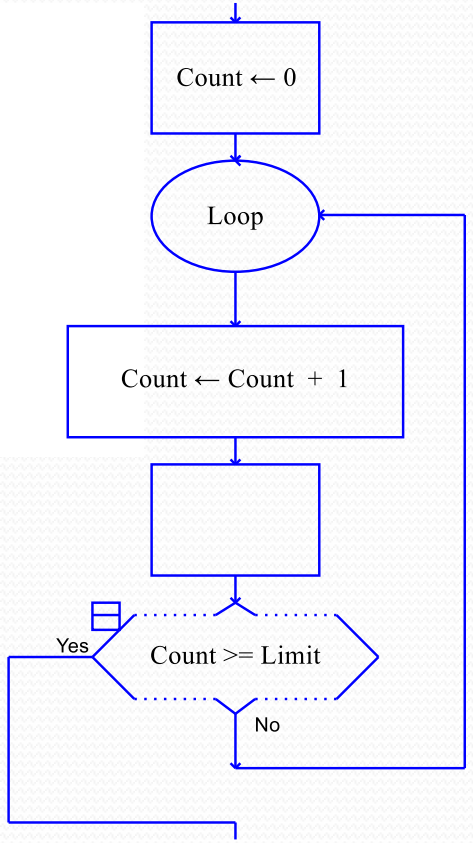




Variasi Counter Loop (2)



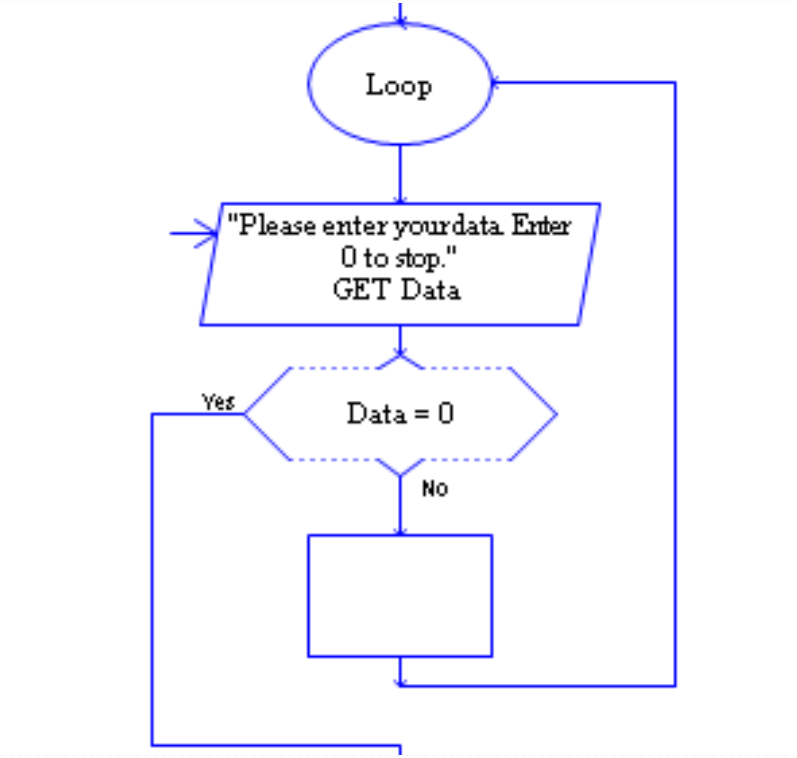




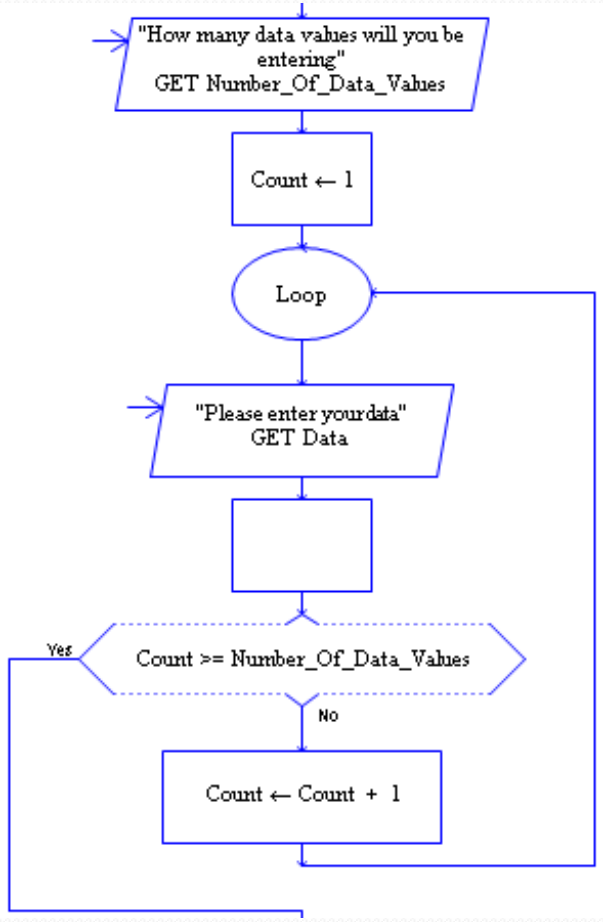
Kegunaan Loop:
(3) Untuk Memasukkan serangkaian input

- Terkadang kita membutuhkan user untuk memasukkan serangkaian input untuk diproses (Misal: memasukkan beberapa nilai tugas mhs untuk kemudian dihitung rata-ratanya)
- Ada 2 cara untuk melakukan hal ini:
  - **Cara 1:** Meminta user memasukkan sebuah "karakter khusus" sebagai penanda bahwa user telah selesai memasukkan serangkaian data
  - **Cara 2:** Menanyakan kepada user, di awal program, akan memasukkan berapa banyak data. Dengan demikian, kita sudah mengetahui di awal, perlu melakukan loop sebanyak berapa kali

Loop untuk Memasukkan serangkaian input



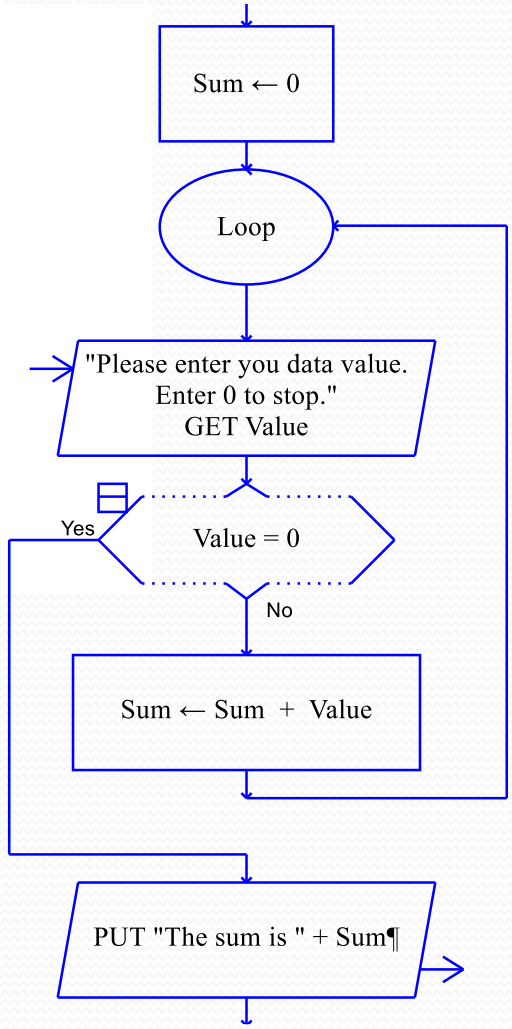
Cara 1



Cara 2

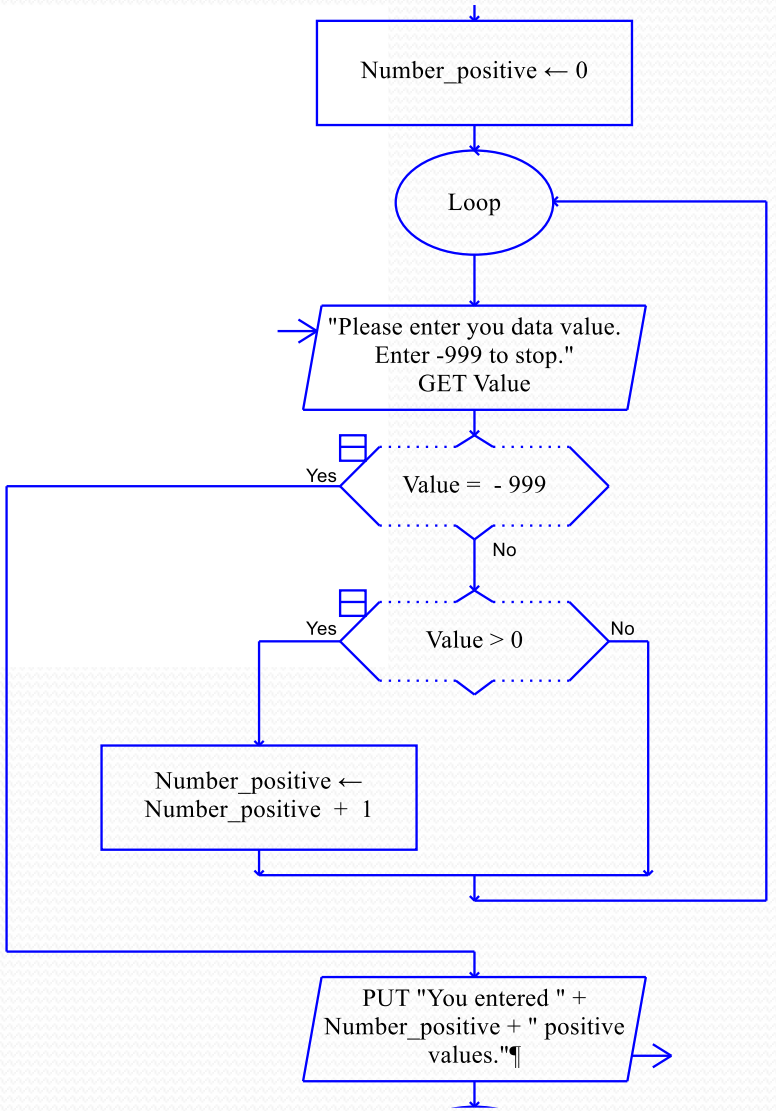
Kegunaan Loop:
(4) Untuk Menghitung jumlah dari serangkaian data input

- Untuk menghitung jumlah dari serangkaian data input, tambahkan 2 statement berikut ke dalam loop:
  - Statement inisialisasi, sebelum loop dimulai, yang mengeset variabel jumlah menjadi 0. Contoh: Sum ← 0
  - Statement penugasan, di dalam loop, yang menambahkan setiap nilai individu dari data input ke variabel jumlah. Contoh: Sum ← Sum + Value
- Pahami maksud dari statement penugasan, **Sum ← Sum + Value**. Ini artinya, hitung ekspresi di sebelah kanan tanda panah dengan menjumlahkan nilai variabel Sum saat ini dengan nilai variabel Value. Lalu simpan hasilnya di variabel Sum.



Kegunaan Loop:
(5) Untuk menghitung berapa kali sebuah kejadian muncul

- Apa yang dikerjakan oleh flowchart di samping?



Ringkasan: Kapan harus menggunakan Selection atau Loop?

- Apakah kita butuh untuk melakukan sesuatu atau tidak melakukan sesuatu? → **Selection**
- Apakah kita butuh melakukan satu hal atau hal lainnya (tapi tidak keduanya)? → **Selection**
- Apakah kita butuh melakukan satu dari banyak hal yang lain? → **Cascade Selection (Seleksi bersarang)**
- Apakah kita butuh melakukan suatu hal yang sama berulang-ulang? → **Loop**
- Apakah kita tahu berapa kali kita perlu melakukan suatu hal yang sama? → **Counting loop**

Hal-hal yang perlu diperhatikan pada Selection

- Apakah ekspresi decision mengakibatkan statement yang benar yang perlu dieksekusi?
- Akankah ekspresi decision mengakibatkan statement yang benar justru tidak dieksekusi (dengan kata lain, dilewati)?

Hal-hal yang perlu diperhatikan pada Loop

- Statement mana yang perlu dikerjakan berulang-ulang?
- Sudahkah kita menginisialisasi semua variabel dengan benar sebelum loop dimulai?
- Akankah ekspresi decision akan selalu bernilai “Yes” saat dieksekusi?
- Jika loop adalah counter loop, apakah loop akan dieksekusi sebanyak jumlah yang benar?
  - Kesalahan paling umum adalah error “off-by-one”, dimana ketika kita ingin loop dikerjakan sebanyak N kali, tetapi nyatanya dikerjakan sebanyak (N-1) atau (N+1) kali

Yang seharusnya sudah Anda pahami sekarang .....

- Urutan statement pemrograman merupakan elemen kunci dari pembuatan sebuah program
- Ada 3 tipe dasar alur program: Sequential, Selection, dan Loop (Iterasi)
- Ekspresi keputusan, yang menghasilkan nilai True/False, digunakan untuk menentukan jalur mana yang akan dikerjakan oleh program pada step selanjutnya
- Kapan harus menggunakan Selection dan/atau Loop untuk menyelesaikan sebuah permasalahan
- Perbedaan antara “Counter Loop”, “Input Loop”, “Loop Menghitung Total”
- Loop tak terbatas itu buruk, dan perhatian khusus perlu diberikan untuk menjamin bahwa loop yang kita buat selalu berakhir/selesai

Latihan Soal

- Which control structure would be most appropriate for the following problems: **Sequential, Selection, Cascading Selection, or a Loop**
-

_____ Printing an appropriate message for a cadet's class year.
_____ Checking for a correct input and continually re-checking if incorrect.

_____ Computing the average GPA of your CS110 section.
_____ Determining the volume of a sphere given a radius.
_____ Initializing all of the variables at the beginning of a program.
_____ Determining whether a vowel, constant or digit has been typed in.
_____ Writing out a message if an integer variable contains a negative value.

_____ Writing "Odd" or "Even" depending on an integer variable's value.
_____ Writing out the squares of the numbers 1 though 100.
_____ Reading in scores until a user enters a negative number.

- Which of the following Decision expressions will always evaluate to "Yes", always evaluate to "No", or could possibly be either "Yes" or "No"?

- _____ GR_Score > 100 **or** GR_Score < 90.
  _____ GR_Score > 100 **and** GR_Score < 90.

  _____ GR_Score < 100 **or** GR_Score > 90.
  _____ GR_Score < 100 **and** GR_Score > 90.

- Write a series of RAPTOR statements that determines if X has the value 1, 2, or 3, and prints out “ONE”, “TWO”, or “THREE” accordingly.
- Write a complete program that converts between degrees Fahrenheit and Celsius. The user must first enter the conversion that is desired (F to C or C to F) using any means you want and then enter the value to be converted. The formulas for conversion are:

  F = 9/5 C + 32 and C = 5/9 (F – 32)