



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	<71230972>
Nama Lengkap	< Oktavian Christ Putranto >
Minggu ke / Materi	06 / Percabangan dan Perulangan Kompleks

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

BAGIAN 1: MATERI MINGGU INI (40%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

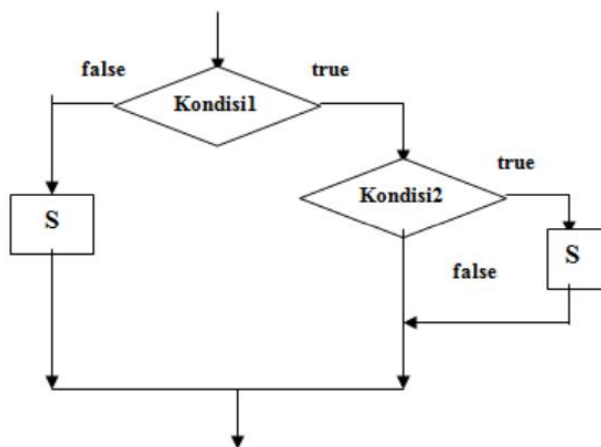
Struktur percabangan kompleks

Dalam membuat sebuah program yang terbilang rumit seringkali kita akan memakai percabangan yang kompleks/bertumpuk tumpuk agar permasalahan dapat diselesaikan oleh karena itu pada pembelajaran di minggu ke enam kita akan mulai belajar mengenai struktur perulangan dan percabangan yang kompleks.

Sederhananya sebuah bentuk kompleks dari percabangan ialah sebuah struktur percabangan Dimana kondisi dan operasi yang dilakukan lebih dari satu dan seringkali akan berjalan berdampingan dengan beragam perintah yang akan dijalankan.

Contoh contoh struktur percabangan kompleks :

Contoh 1

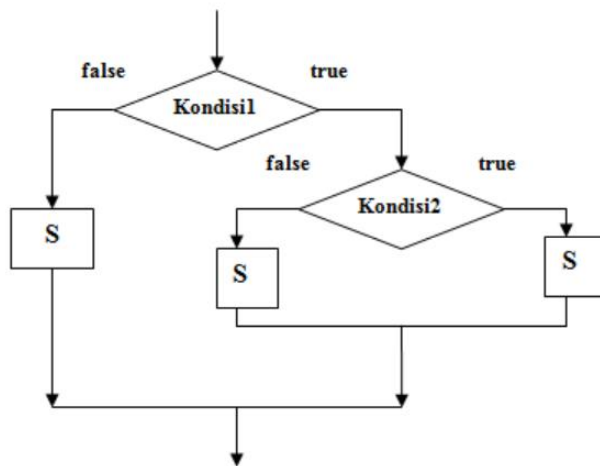


Gambar 6.1: Flowchart Percabangan Kompleks Bentuk 1

```
1  if kondisi1:
2      if kondisi2:
3          S
4          S
5  else:
6      S
7      S
```

Pada contoh pertama ini dapat dilihat sebuah bentuk percabangan Dimana setelah kita menentukan sebuah conditional maka dari percabangan tersebut kita akan diarahkan kepada conditional kedua yang memproses lebih dalam data tersebut kemudian mengeluarkan output yang sesuai dengan conditionalnya jika tidak memenuhi kondisi kondisi tersebut maka akan masuk ke kategori else.

Contoh 2:



Gambar 6.2: Flowchart Percabangan Kompleks Bentuk 2

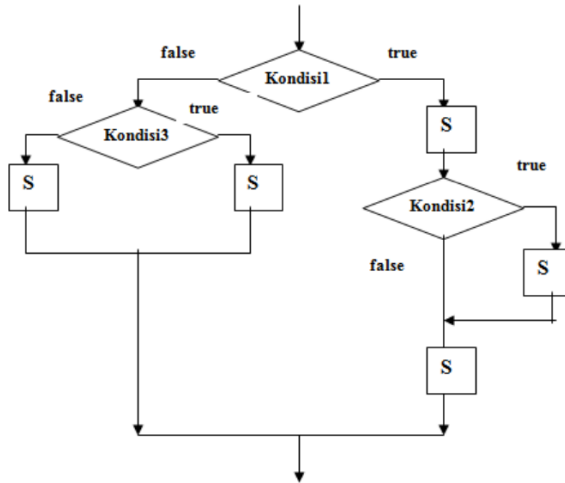
```
1  if kondisi1:
2      if kondisi2:
3          S
4          S
5      else:
6          S
7          S
8  else:
9      S
10     S
```

```
kondisi_2.py > ...
1  a=int(input("masukan nilai a: "))
2  b=int(input("masukan nilai b: "))
3
4  if a>b:
5      if a>=100:
6          print(f"{a} lebih besar dari {b} dan merupakan bilangan dengan 3 digit/lebih")
7      else:
8          print(f"{a} lebih besar dari {b} namun tidak lebih dari 2 digit")
9  else:
10     print(f"a tidak lebih besar dari b")
```

```
masukan nilai a: 120
masukan nilai b: 10
120 lebih besar dari 10 dan merupakan bilangan dengan 3 digit/lebih
```

Pada contoh kedua kita dapat melihat sedikit perbedaan dari contoh pertama Dimana pada contoh kedua ketika sampai pada kondisional kedua ketika kondisional ini tak terpenuhi maka akan menjalankan oprasi/perintah lain dalam kasus code yang saya buat diatas maka pertama tama code akan memeriksa apakah a lebih besar dari b atau tidak jika yam aka akan masuk ke conditional kedua jika tidak maka akan keluar output yang menyatakan "a tidak lebih besar dari b". untuk conditional kedua berguna untuk mengecek apakah a merupakan bilangan ratusan ke atas atau tidak jika conditional kedua terpenuhi maka line 6 yang akan ter eksekusi sedangkan jika tidak maka line 8 yang di eksekusi.

Contoh 3:



Gambar 6.3: Flowchart Percabangan Kompleks Bentuk 3

```
1  if kondisi1:
2      S
3      if kondisi2:
4          S
5          S
6  else:
7      if kondisi3:
8          S
9          S
10     else:
11         S
12         S
```

```
kondisi.py > ...
1  a=int(input("masukan nilai a: "))
2  b=int(input("masukan nilai b: "))
3
4  if a>b:
5      if a>=100:
6          print(f"{a} lebih besar dari {b} dan merupakan bilangan dengan 3 digit/lebih")
7      else:
8          print(f"{a} lebih besar dari {b} namun tidak lebih dari 2 digit")
9
10 else:
11     if a!=b:
12         print(f"{a} lebih kecil dari {b}")
13     else:
14         print("kedua bilangan bernilai sama besar")
```

```
masukan nilai a: 25
masukan nilai b: 25
kedua bilangan bernilai sama besar
```

Pada contoh ketiga diatas kita dapat melihat ada tambahan conditional ketiga di bagian else yang membuat code menjadi lebih kompleks dalam contoh code yang saya buat saya menambah conditional baru di line 11 untuk menentukan apakah bilangan a sama dengan b atau tidak jika tidak sama maka code yang dieksekusi adalah code line ke 12 yang mengeluarkan output "a lebih kecil dari b" sedangkan ketika tak terpenuhi maka code yang di eksekusi ialah code di line 14 yang memprint "kedua bilangan bernilai sama besar".

Selain dari ketiga contoh diatas masih ada banyak bentuk kompleks dari percabangan yang dapat digunakan sesuai dengan kondisi yang ingin dicapai, selain dari bentuk percabangan kompleks kita juga belajar mengenai perulangan kompleks/perulangan bertingkat.

Perulangan Kompleks

Perulangan yang biasanya kita gunakan ialah for dan while sederhana namun pada pertemuan ini kita belajar penggunaan for didalam for, dan while didalam while atau kombinasi dari keduanya, bentuk perulangan kompleks ini biasanya dipakai untuk membuat pola bilangan/bentuk bangun datar, dan masih banyak lagi, namun sebelum mempelajari perulangan kompleks/perulangan bertingkat yang rumit sebaiknya kita mengerti penggunaan break dan continue terlebih dahulu contoh penggunaan continue dan break.

Code perulangan biasa

```
perulangan.py > ...  
1  for i in range (10):  
2      print (i)
```

Output:

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9
```

penggunaan Break

```
perulangan.py > ...  
1  for i in range (10):  
2      if i==5:  
3          break  
4      print (i)
```

Output:

```
0  
1  
2  
3  
4
```

Penggunaan continue

```
perulangan.py > ...  
1  for i in range (10):  
2      if i==5:  
3          continue  
4      print (i)
```

Output :

```
0  
1  
2  
3  
4  
6  
7  
8  
9
```

Dari ketiga contoh diatas dapat terlihat jelas perbedaan diantara ketiganya Dimana pada code biasa maka data yang ditampilkan ialah 0-9 sedangkan pada code perulangan yang memakai break maka perulangan hanya dari 0-4 dan pada perulangan yang memakai continue angka 5 dalam 0-9 tidak ikut ter print.

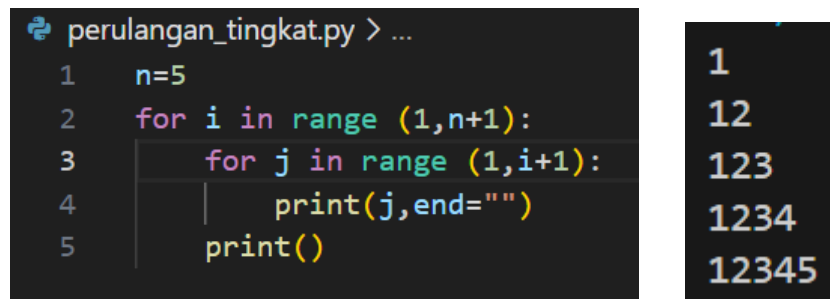
Dari contoh cotntoh tersebut kita dapat menarik kesimpulan bahwa Sederhananya penggunaan Break berfungsi untuk menghentikan perulangan ketika conditional terpenuhi, dan continue berfungsi untuk melompati/menskip perulangan yang memenuhi conditional yang kita tentukan.

Setelah mempelajari break dan continue selanjutnya ialah mempelajari bentuk perulangan kompleks Dimana terdapat perulangan bertingkat didalamnya bentuk perulangan kompleks ini biasanya digunakan untuk masalah masalah yang berhubungan dengan pola pola.

Contoh dari perulangan kompleks:

```
1  for i in range(m):
2      for j in range(n):
3          <lakukan perintah ini di inner>
4          <lakukan perintah itu di inner>
5      <lakukan perintah lain di outer>
6      <lakukan perintah lain lagi di outer>
```

Penggunaan bentuk tersebut dalam code:



```
perulangan_tingkat.py > ...
1  n=5
2  for i in range (1,n+1):
3      for j in range (1,i+1):
4          print(j,end="")
5      print()

1
12
123
1234
12345
```

Pada code ini terlihat bahwa perulangan pertama yang berada pada line 2 berfungsi untuk mengatur tinggi output/jumlah baris dari outputnya selanjutnya diberikan perulangan kedua yang berada di line 3, perulangan ini berada di dalam perulangan lain dan perulangan ini berfungsi untuk memprint barisan bilangan – bilangan pada tiap baris nya, dan perintah yang ada di line 4 ialah perintah yang berada dalam inner atau berada dalam perulangan kedua sedangkan perintah di line ke 5 merupakan perintah yang berada di outer atau perintah yang ada dalam perulangan pertama.

BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

SOAL 1

Input:

```
lat_1.py > ...
1  def cek_prima(n):
2      if n<2:
3          return False
4      elif n==2:
5          return True
6      for i in range(2, int(n**0.5) + 1):
7          if n % i == 0:
8              return False
9      else:
10         return True
11     # prima terdekat
12     def nearest_prime(n):
13         angka=n-1
14         for i in range(angka,0,-1):
15             if cek_prima(i)==True:
16                 return i
17
18     n=int(input("masukan n:"))
19     cek_prima(n)
20     nearest_prime(n)
21     print(f"Bilangan prima terdekat < {n} adalah {nearest_prime(n)}")
```

Output:

```
masukan n:15
Bilangan prima terdekat < 15 adalah 13
```

Penjelasan code:

Pada code ini pertama tama saya membuat fungsi cek_prima(n):

```

lat_1.py > ...
1  def cek_prima(n):
2      if n<2:
3          return False
4      elif n==2:
5          return True
6      for i in range(2, int(n**0.5) + 1):
7          if n % i == 0:
8              return False
9      else:
10         return True

```

Di dalam fungsi ini pertama tama dalam line 2 saya membuat conditional $n < 2$ untuk mengecek apakah bialngan tersebut lebih dari 2 atau tidak jika tidak maka akan return false jika iya maka akan pergi mengecek conditional berikutnya.

Conditional di line 4 ialah if $n == 2$ yang mengecek jika n ilah 2 atau tidak jika n ialah 2 maka akan langsung return True, jika tidak maka akan lanjut ke line 6

Pada line 6 saya menggunakan perulangan Dimana bilangan tersebut akan di modulus dengan angka 2 sampai akar dari $n + 1$ rumus ini berfungsi untuk mengecek apakah ada bilangan yang mampu membagi habis n atau tidak jika ada maka akan return False.

Pada line terakhir di fungsi cek_prima(n) ialah line 10 dimana ketika semua syarat diatas sudah di cek satu persatu dan tidak memenuhi maka akan mereturn True.

```

12  def nearest_prime(n):
13      angka=n-1
14      for i in range(angka,0,-1):
15          if cek_prima(i)==True:
16              return i
17

```

Fungsi kedua yang saya buat ialah nearest_prime(n): yang berguna untuk menemukan bialngan prima terdekat yang lebih kecil dari n

Pada line 13 saya membuat variable baru yaitu angka yang bernilai $n-1$ karena kita ingin mencari bilangan dibawah n

Pada line 14 saya membuat perulangan yang berfungsi untuk mengecek tiap angka yang lebih kecil dari n mulai dari angka hingga 0, kemudian tiap angka itu saya masukan ke line 15 untuk mengecek apakah dari tiap bilangan tersebut ada yg prima ketika ditemukan maka akan langsung me return l yang menghentikan perulangan dan menghasilkan nilai i yang merupakan bilan prima dibawah n

```

18  n=int(input("masukan n:"))
19  cek_prima(n)
20  nearest_prime(n)
21  print(f"Bilangan prima terdekat < {n} adalah {nearest_prime(n)}")
22

```


Pada bagian terakhir barulah saya membuat inputan nya Dimana n di input dan dimasukan ke fungsi cek_prima(n) dan nearest_prime(n) yang kemudian akan mengeluarkan output seperti di line 21

SOAL 2

Input :

```
lat_2.py > faktorial
1  def faktorial(n):
2      for i in range(n,0,-1):
3          tot=1
4          for j in range(i,0,-1):
5              tot=tot*j
6          print (tot,end=" ")
7          for k in range(i,0,-1):
8              print(k," ",end="")
9          print()
10
11  n=int(input("input n:"))
12  faktorial(n)
```

Output:

```
input n:6
720 6 5 4 3 2 1
120 5 4 3 2 1
24 4 3 2 1
6 3 2 1
2 2 1
1 1
```

Penjelasan :

```

lat_2.py > faktorial
1  def faktorial(n):
2      for i in range(n,0,-1):
3          tot=1
4          for j in range(i,0,-1):
5              tot=tot*j
6          print (tot,end=" ")
7          for k in range(i,0,-1):
8              print(k," ",end="")
9          print()

```

Pada bagian ini saya mendefine sebuah fungsi Bernama faktorial(n):

Dimana pada line 2 saya membuat perulangan dengan nilai minus atau perulangan menurun dimana bagian awal bernilai n sampai 0.

Pada line 3 saya membuat variable tot=1 saya buat ini di bagian outer karena saya ingin pada tiap barisnya nilai tot Kembali jadi bernilai 1 lagi

Pada line 4 saya buat inner loop yaitu j in range (i,0,-1): yang berisi tot=tot*j perulangan ini berfungsi untuk menghitung dan menampilkan tot pada awal baris untuk output tot saya buat pada outter loop agar hanya ter print sekali pada awal.

Pada line 6 saya membuat perulangan yang menampilkan bilangan-bilangan mulai dari n-1 kemudian di print berurutan dengan tot karena keduanya memakai end=" " yang berfungsi untuk membuat barisan tetap dalam satu baris

Kemudian pada line 9 saya membuat print() pada outer loop untuk membuat baris berakhir/membuat tab

Sisa dari code berfungsi untuk meminta input user dan menjalankan fungsi factorial(n)

SOAL 3

Input:

```

lat_3.py > ...
1  tinggi=int(input("masukan tinggi:"))
2  lebar=int(input("masukan lebar:"))
3  angka=0
4  for i in range(tinggi):
5      for j in range(1,lebar+1):
6          angka+=1
7          print(angka,end=" ")
8      print ()

```

Output :

```
masukan tinggi:5
masukan lebar:4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
```

Penjelasan:

```
lat_3.py > ...
1  tinggi=int(input("masukan tinggi:"))
2  lebar=int(input("masukan lebar:"))
```

pada bagian ini pertama code akan meminta inputan user yang menentukan value tinggi dan lebar

```
3  angka=0
4  for i in range(tinggi):
5      for j in range(1,lebar+1):
6          angka+=1
7          print(angka,end=" ")
8      print ()
```

Pada line ke 3 saya menentukan variable angka yang bernilai 0, variable angka saya buat diluar semua perulangan agar ketika perulangan berjalan value ini tidak ter reset dan akan terus bertambah value nya sejalan dengan looping pada line 5

Pada line 4 saya membuat looping untuk menentukan tinggi nya atau banyak barisnya dibawah kemudian saya membuat perulangan dalam perulangan ini yakni pada line 5

Di line 5 saya membuat perulangan dari 1 hingga lebarnya selama perulangan ini berjalan maka akan mengubah nilai angka ditambah 1 setiap perulangan berjalan kemudian memprint nilai angka diakhiri dengan end=" " agar tetap dalam satu baris hingga sepanjang lebarnya

Setelah itu print() untuk memisahkan barisnya

Pada baris berikutnya akan terprint angka dengan nilai value angka yang tetap berjalan sehingga akan menghasilkan output yang diinginkan.

Source: https://github.com/vianchr/71230972_tugas6_vian