LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 1
MODUL 10
“ELSE-IF”

DISUSUN OLEH:

DAFFA TSAQIFNA FAUZTSANY

103112400032

S1 IF-12-01

DOSEN:

Yohani Setiya Rafika Nur, M. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

DASAR TEORI:

1. Paradigma Percabangan

Percabangan dalam pemrograman memungkinkan eksekusi baris kode tertentu berdasarkan kondisi atau syarat tertentu. Salah satu struktur kontrol percabangan yang umum digunakan adalah else-if, yang memungkinkan program memilih antara beberapa aksi berdasarkan nilai kebenaran kondisi yang diberikan.

2. Karakteristik Else-If

- **Else-if serupa dengan struktur kontrol if-then, tetapi memiliki aksi alternatif jika suatu kondisi bernilai false.**
- **Struktur penulisan terdiri dari:**
    1. **Kondisi: Syarat atau ketentuan dalam bentuk nilai boolean.**
    2. **Aksi: Instruksi yang dijalankan jika kondisi bernilai true.**
    3. **Aksi Lain: Instruksi yang dijalankan jika kondisi bernilai false.**

3. Struktur Penulisan

Dalam pseudocode:

```
if kondisi_1 then
   // aksi 1
else if kondisi_2 then
   // aksi 2
else if kondisi_3 then
   // aksi 3
else
   // aksi lain
endif
```

Contoh dalam pseudocode:

```
if kondisi_1 {
   // aksi 1
} else if kondisi_2 {
   // aksi 2
} else if kondisi_3 {
   // aksi 3
} else {
   // aksi lain
}
```

4. Implementasi dengan Bahasa Go

Contoh implementasi else-if dalam Go melibatkan kasus-kasus seperti:

- **Menampilkan hasil pembagian hanya jika pembagi bukan nol.**
- **Menentukan indeks nilai berdasarkan parameter nilai, kehadiran, dan tugas.**

5. Aplikasi dan Contoh Kasus

Beberapa contoh soal yang menggunakan struktur else-if meliputi:

- **Menentukan kelayakan seseorang untuk membuat KTP berdasarkan usia dan kepemilikan kartu keluarga.**
- **Mengidentifikasi apakah suatu huruf adalah vokal, konsonan, atau bukan huruf.**
- **Menentukan apakah digit suatu bilangan empat digit terurut membesar, mengecil, atau tidak terurut.**

6. Signifikansi dan Penerapan

Else-if memungkinkan pengambilan keputusan yang lebih kompleks dalam algoritma, mendukung pengembangan program yang fleksibel dan terstruktur dengan logika bercabang.

CONTOH SOAL

Conso 1:

```
package main

import "fmt"

func main() {
    var usia int
    var kk bool
    fmt.Scan(&usia, &kk)
    if usia >= 17 && kk {
        fmt.Println("bisa membuat KTP")
    } else {
        fmt.Println("belum bisa membuat KTP")
    }
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-1.go'
17
true
bisa membuat KTP
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-1.go'
20
false
belum bisa membuat KTP
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-1.go'
15
true
belum bisa membuat KTP
```

Deskripsi Program:

Program di atas digunakan untuk menentukan apakah seseorang memenuhi syarat untuk membuat **KTP (Kartu Tanda Penduduk)** di Indonesia. Syaratnya adalah:

1. Usia minimal 17 tahun.
2. Memiliki **Kartu Keluarga (KK)**, yang direpresentasikan dengan nilai boolean (true jika memiliki KK, false jika tidak).

1. **Deklarasi variabel:**

```
var usia int
var kk bool
```

- **usia**: Variabel yang menyimpan usia pengguna dalam satuan tahun (integer).

- **kk**: Variabel boolean yang menunjukkan apakah pengguna memiliki Kartu Keluarga (KK). Nilai:
    - `true`: Memiliki KK.
    - `false`: Tidak memiliki KK.

2. Membaca input dari pengguna:

```
fmt.Scan(&usia, &kk)
```

- Program meminta pengguna memasukkan **usia** (bilangan bulat) dan **status KK** (boolean, `true` atau `false`).

3. Logika Utama:

```
if usia >= 17 && kk {
   fmt.Println("bisa membuat KTP")
} else {
   fmt.Println("belum bisa membuat KTP")
}
```

- **Kondisi usia ≥ 17**:
    - Mengecek apakah usia pengguna 17 tahun atau lebih.
- **Kondisi kk**:
    - Mengecek apakah pengguna memiliki KK (true).
- Operator Logika &&:
    - Kedua kondisi di atas harus terpenuhi secara bersamaan agar pengguna dapat membuat KTP.
- Kondisi Else:
    - Jika salah satu atau kedua kondisi tidak terpenuhi, program mencetak **"belum bisa membuat KTP"**.

Contoh: Jika pengguna memasukkan

```
18 true (input)

bisa membuat KTP(output)
```

2.cosol2

```
package main

import "fmt"

func main() {

   var x rune

   var huruf, vKecil, vBesar bool

   fmt.Scanf("%c", &x)

   huruf = (x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')

   vKecil = x == 'a' || x == 'i' || x == 'u' || x == 'e' || x == 'o'

   vBesar = x == 'A' || x == 'I' || x == 'U' || x == 'E' || x == 'O'
```

```
    if huruf && (vKecil || vBesar) {
        fmt.Println("vokal")
    } else if huruf && !(vKecil || vBesar) {
        fmt.Println("konsonan")
    } else {
        fmt.Println("bukan huruf")
    }
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-2.go'
A
vokal
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-2.go'
f
konsonan
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-2.go'
1
bukan huruf
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-2.go'
$
bukan huruf
```

Deskripsi Program:
Program di atas digunakan untuk menentukan jenis sebuah karakter (huruf) yang dimasukkan oleh pengguna. Program akan mengklasifikasikan karakter sebagai **huruf vokal**, **huruf konsonan**, atau **bukan huruf**.:

1. **Deklarasi variabel:**

```
var x rune
var huruf, vKecil, vBesar bool
```

- **x**: Variabel bertipe rune (karakter), digunakan untuk membaca input berupa satu karakter.
- **huruf**: Boolean, bernilai true jika karakter adalah huruf (baik huruf kecil maupun huruf besar).
- **vKecil**: Boolean, bernilai true jika karakter adalah vokal kecil ('a', 'i', 'u', 'e', 'o').
- **vBesar**: Boolean, bernilai true jika karakter adalah vokal besar ('A', 'I', 'U', 'E', 'O').

2. Membaca input dari pengguna:

```
fmt.Scanf("%c", &x)
```

- Membaca input berupa satu karakter dan menyimpannya di variabel

3. Logika Utama:

- Memeriksa Karakter Huruf:

```
huruf = (x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')
```

- huruf: Bernilai true jika karakter x berada dalam rentang huruf kecil ('a' sampai 'z') atau huruf besar ('A' sampai 'Z').

- Memeriksa Huruf Vokal Kecil:

```
vKecil = x == 'a' || x == 'i' || x == 'u' || x == 'e' || x == 'o'
```

- vKecil: Bernilai true jika karakter x adalah salah satu dari huruf vokal kecil.

- Memeriksa Huruf Vokal Besar:

```
vBesar = x == 'A' || x == 'I' || x == 'U' || x == 'E' || x == 'O'
```

- vBesar: Bernilai true jika karakter x adalah salah satu dari huruf vokal besar.

4. Output Berdasarkan Kondisi:

- Jika karakter adalah **huruf** dan termasuk **vokal**:

```
if huruf && (vKecil || vBesar) {
   fmt.Println("vokal")
}
```

- Cetak "vokal".

- Evaluasi Urutan Digit:

```
else if huruf && !(vKecil || vBesar) {
   fmt.Println("konsonan")
}
```

- Cetak "konsonan".

- Evaluasi Urutan Digit:

```
else {
   fmt.Println("bukan huruf")
}
```

- Cetak "bukan huruf".

Contoh: Jika pengguna memasukkan

```
a(input)
vokal
```

3.cosol3

```
package main
import "fmt"
func main() {
   var bilangan, d1, d2, d3, d4 int
```

```
    var teks string
    fmt.Print("Bilangan: ")
    fmt.Scan(&bilangan)
    d4 = bilangan % 10
    d3 = (bilangan % 100) / 10
    d2 = (bilangan % 1000) / 100
    d1 = bilangan / 1000
    if d1 < d2 && d2 < d3 && d3 < d4 {
        teks = "terurut membesar"
    } else if d1 > d2 && d2 > d3 && d3 > d4 {
        teks = "terurut mengecil"
    } else {
        teks = "tidak terurut"
    }
    fmt.Println("Digit pada bilangan", bilangan, teks)
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-3.go'
Bilangan: 2489
Digit pada bilangan 2489 terurut membesar
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-3.go'
Bilangan: 3861
Digit pada bilangan 3861 tidak terurut
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\conso7\conso7-3.go'
Bilangan: 9651
Digit pada bilangan 9651 terurut mengecil
```

Deskripsi Program:

Program di atas digunakan untuk mengevaluasi **susunan digit pada bilangan empat digit** (ribuan) yang dimasukkan pengguna. Program akan mengklasifikasikan apakah digit-digit dalam bilangan tersebut **terurut membesar**, **terurut mengecil**, atau **tidak terurut**

1. **Deklarasi variabel:**

```
var bilangan, d1, d2, d3, d4 int
var teks string
```

- **bilangan**: Menyimpan bilangan empat digit yang dimasukkan pengguna.
- **d1, d2, d3, d4**: Digit-digit dari bilangan (dari ribuan hingga satuan):
  - d1: Digit ribuan.
  - d2: Digit ratusan.

- d3: Digit puluhan.
- d4: Digit satuan.

- **teks: String yang digunakan untuk menyimpan hasil evaluasi.**

2. Membaca input dari pengguna:

```
fmt.Scan(&bilangan)
```

- Pengguna diminta untuk memasukkan bilangan dengan 4 digit, dan nilainya disimpan di variabel bilangan.

3. Memisahkan Digit-digit:

```
d4 = bilangan % 10
d3 = (bilangan % 100) / 10
d2 = (bilangan % 1000) / 100
d1 = bilangan / 1000
```

- **d4**: Digit satuan, diperoleh dengan operasi modulus bilangan % 10.
- **d3**: Digit puluhan, diperoleh dengan bilangan % 100 (dua digit terakhir), lalu dibagi 10.
- **d2**: Digit ratusan, diperoleh dengan bilangan % 1000 (tiga digit terakhir), lalu dibagi 100.
- **d1**: Digit ribuan, diperoleh dengan pembagian langsung bilangan / 1000.

4. Evaluasi Urutan Digit:

- **Terurut Membesar**:

```
if d1 < d2 && d2 < d3 && d3 < d4 {
   teks = "terurut membesar"
}
```

  - Digit ribuan harus lebih kecil dari digit ratusan
  - Digit ratusan harus lebih kecil dari digit puluhan.
  - Digit puluhan harus lebih kecil dari digit satuan.

- Evaluasi Urutan Digit:

```
else if d1 > d2 && d2 > d3 && d3 > d4 {
   teks = "terurut mengecil"
}
```

  - Digit ribuan harus lebih besar dari digit ratusan.
  - Digit ratusan harus lebih besar dari digit puluhan.
  - Digit puluhan harus lebih besar dari digit satuan.

- Evaluasi Urutan Digit:

```
else {
   teks = "tidak terurut"
}
```

  - Jika kedua kondisi di atas tidak terpenuhi, maka digit bilangan dianggap tidak terurut.

5. Output:

```
fmt.Println("Digit pada bilangan", bilangan, teks)
```

- Program mencetak hasil evaluasi digit bilangan, apakah **terurut membesar**, **terurut mengecil**, atau **tidak terurut**.

Contoh: jika pengguna memasukkan:

```
1234(input)

Digit pada bilangan 1234 terurut membesar(output)
```

SOAL LATIHAN

1. Latihan1

Source Code:

```
package main

import "fmt"

func main() {
    var x, y, z, b1, b2, bt int
    fmt.Print("Berat parcel (gram): ")
    fmt.Scan(&x)
    y = x / 1000
    z = x % 1000
    b1 = y * 10000
    fmt.Println("Detai berat:  ", y, "Kg + ", z, " gr")
    if z >= 500 {
        b2 = z * 5
    } else {
        b2 = z * 15
    }
    fmt.Println("Detail biaya: Rp.", b1, " + Rp. ", b2)
    if y >= 10 {
        bt = b1
        fmt.Println("Total biaya: Rp. ", bt)
    } else {
        bt = b1 + b2
        fmt.Println("Total biaya: Rp. ", bt)
    }
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\latso7\latso7-1.go'
Berat parcel (gram): 8500
Detai berat:   8 Kg +  500  gr
Detail biaya: Rp. 80000  + Rp.  2500
Total biaya: Rp.  82500
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\latso7\latso7-1.go'
Berat parcel (gram): 9250
Detai berat:   9 Kg +  250  gr
Detail biaya: Rp. 90000  + Rp.  3750
Total biaya: Rp.  93750
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\latso7\latso7-1.go'
Berat parcel (gram): 11750
Detai berat:   11 Kg +  750  gr
Detail biaya: Rp. 110000  + Rp.  3750
Total biaya: Rp.  110000
```

Deskripsi Program:
Program di atas menghitung biaya pengiriman **parcel** berdasarkan berat total yang diinputkan oleh pengguna. Berat parcel dikonversi menjadi satuan kilogram dan gram, kemudian dihitung biaya pengirimannya dengan tarif tertentu.:

1. **Deklarasi variabel:**

```
var x, y, z, b1, b2, bt int
```

6. **x**: Berat parcel dalam gram yang diinputkan pengguna.
7. **y**: Berat dalam kilogram (kg), dihitung dari x / 1000.
8. **z**: Sisa berat dalam gram setelah dikonversi ke kilogram, dihitung dari x % 1000.
9. **b1**: Biaya berdasarkan berat dalam kilogram.
10. **b2**: Biaya tambahan berdasarkan sisa berat dalam gram.
11. **bt**: Total biaya pengiriman.

2. Membaca Berat Parcel:

```
fmt.Print("Berat parcel (gram): ")
fmt.Scan(&x)
```

12. Pengguna diminta memasukkan berat parcel dalam gram, lalu disimpan di variabel `x`.

3. Logika utama:

```
y = x / 1000
z = x % 1000
```

13. **y**: Berat dalam kilogram diperoleh dengan membagi berat total (x) dengan 1000.
14. **z**: Berat dalam gram (sisa dari konversi) dihitung dengan operasi modulus (x % 1000).

4. Biaya Berdasarkan Berat dalam Kg:

```
b1 = y * 10000
```

15. Setiap 1 kg dikenakan biaya sebesar Rp. 10.000. Nilainya dihitung dengan mengalikan berat dalam kg (y) dengan 10.000.
5. Biaya Berdasarkan Berat dalam Gram:

```
if z >= 500 {
   b2 = z * 5
} else {
   b2 = z * 15
}
```

16. Jika sisa berat (gram) **lebih dari atau sama dengan 500**, tarifnya Rp. 5/gram.
17. Jika kurang dari 500, tarifnya Rp. 15/gram.

6. Total Biaya:

```
if y >= 10 {
   bt = b1
   fmt.Println("Total biaya: Rp. ", bt)
} else {
```

```
    bt = b1 + b2
    fmt.Println("Total biaya: Rp. ", bt)
}
```

18. Jika berat parcel lebih dari atau sama dengan **10 kg**, hanya biaya berdasarkan kilogram (b1) yang dihitung.
19. Jika berat kurang dari **10 kg**, total biaya dihitung sebagai penjumlahan biaya kilogram (b1) dan biaya gram (b2).

Contoh: Jika pengguna memasukkan

```
Berat parcel (gram): 10500(output)
Detai berat:   10 Kg +  500  gr(output)
Detail biaya: Rp. 100000  + Rp.  0(output)
Total biaya: Rp.  100000(output)
```

2. Latihan2

Source Code SALAH:

```
package main

import "fmt"

func main() {
    var nam float64
    var nmk string
    fmt.Print("Nilai akhir mata kuliah: ")
    fmt.Scan(&nam)
    if nam > 80 {
        nam = "A"
    }
    if nam > 72.5 {
        nam = "AB"
    }
    if nam > 65 {
        nam = "B"
    }
    if nam > 57.5 {
        nam = "BC"
    }
    if nam > 50 {
        nam = "C"
```

```
    }
    if nam > 40 {
        nam = "D"
    } else if nam <= 40 {
        nam = "E"
    }
    fmt.Println("Nilai mata kuliah: ", nmk)

}
```

1. Jika nam diberikan adalah 80.1, apa keluaran dari program tersebut? Apakah eksekusi program tersebut sesuai spesifikasi soal?

```
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\latso7\latso7-2wrong.go'
# command-line-arguments
lab shit\7th meet\latso7\latso7-2wrong.go:11:9: cannot use "A" (untyped string constant) as float64 value in assignment
lab shit\7th meet\latso7\latso7-2wrong.go:14:9: cannot use "AB" (untyped string constant) as float64 value in assignment
lab shit\7th meet\latso7\latso7-2wrong.go:17:9: cannot use "B" (untyped string constant) as float64 value in assignment
lab shit\7th meet\latso7\latso7-2wrong.go:20:9: cannot use "BC" (untyped string constant) as float64 value in assignment
lab shit\7th meet\latso7\latso7-2wrong.go:23:9: cannot use "C" (untyped string constant) as float64 value in assignment
lab shit\7th meet\latso7\latso7-2wrong.go:26:9: cannot use "D" (untyped string constant) as float64 value in assignment
lab shit\7th meet\latso7\latso7-2wrong.go:28:9: cannot use "E" (untyped string constant) as float64 value in assignment
```

2. Apa saja kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur program seharusnya!
    a. Tipe Data Variabel nam Tidak Konsisten

    20. Variabel nam dideklarasikan sebagai float64 (angka desimal), tetapi di bagian berikutnya, mencoba menetapkan nilai string ("A", "B", dll.) ke variabel tersebut. Ini menyebabkan type mismatch karena tipe data float64 tidak dapat menyimpan nilai string.
    21. Solusi: Gunakan dua variabel terpisah: satu untuk menyimpan nilai angka (float64) dan satu untuk menyimpan nilai huruf (string).

    a. Variabel nmk Tidak Digunakan

    22. mendeklarasikan variabel nmk dengan tipe data string, tetapi tidak pernah memberikan nilai padanya. Akhirnya, saat mencetak nilai nmk, hasilnya selalu string kosong.
    23. Solusi: Gunakan variabel nmk untuk menyimpan nilai huruf hasil konversi dari angka.

    a. Logika if Tidak Konsisten

    24. Semua kondisi if dievaluasi, tanpa memperhatikan urutan prioritas. Ini mengakibatkan nilai pada variabel nam terus berubah hingga kondisi terakhir (if nam > 40), sehingga hasil akhirnya salah.
    25. Solusi: Gunakan struktur if-else if untuk memastikan hanya satu kondisi yang dieksekusi berdasarkan nilai nam.

    a. Output Tidak Sesuai

    26. Pada bagian terakhir, mencetak nmk, tetapi nilai sebenarnya tidak pernah diubah. Jadi, output selalu kosong atau default.
    27. Solusi: Perbaiki logika penentuan nilai nmk dan pastikan mencetak variabel yang sesuai.

1. Perbaiki program tersebut! Ujilah dengan masukan: 93.5; 70.6; dan 49.5. Seharusnya keluaran yang diperoleh adalah ‘A’, ‘B’, dan ‘D’.

Source Code BENAR:

```
package main

import "fmt"
```

```
func main() {
    var nam float64
    var nmk string

    fmt.Print("Nilai akhir mata kuliah: ")
    fmt.Scan(&nam)

    if nam > 80 {
        nmk = "A"
    } else if nam > 72.5 {
        nmk = "AB"
    } else if nam > 65 {
        nmk = "B"
    } else if nam > 57.5 {
        nmk = "BC"
    } else if nam > 50 {
        nmk = "C"
    } else if nam > 40 {
        nmk = "D"
    } else {
        nmk = "E"
    }

    fmt.Println("Nilai mata kuliah: ", nmk)
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\latso7\latso7-2right.go'
Nilai akhir mata kuliah: 93.5
Nilai mata kuliah:  A
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\latso7\latso7-2right.go'
Nilai akhir mata kuliah: 70.6
Nilai mata kuliah:  B
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\latso7\latso7-2right.go'
Nilai akhir mata kuliah: 49.5
Nilai mata kuliah:  D
```

3. Latihan3

Source Code:

```
package main

import "fmt"

func main() {
    var x, y, i int
    var a bool
    fmt.Print("bilangan: ")
    fmt.Scan(&x)
```

```
    fmt.Print("faktor: ")
    for i = 1; i <= x; i++ {
        if x%i == 0 {
            y++
            fmt.Print(i, " ")
        }
    }
    fmt.Print("\nprima: ")
    if y <= 2 {
        a = true
        fmt.Print(a)
    } else {
        fmt.Print(a)
    }
}
```

Output:

```
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\latso7\latso7-3.go'
bilangan: 12
faktor: 1 2 3 4 6 12
prima: false
PS D:\test bs> go run  'd:\test bs\lab shit\7th meet\latso7\latso7-3.go'
bilangan: 7
faktor: 1 7
prima: true
```

Deskripsi Program:

Program ini memeriksa faktor dari suatu bilangan dan menentukan apakah bilangan tersebut merupakan bilangan prima. Program juga mencetak semua faktor bilangan tersebut.

1. **Deklarasi variabel:**

```
var x, y, i int
var a bool
```

28. **x**: Bilangan yang diinputkan oleh pengguna.
29. **y**: Counter untuk menghitung jumlah faktor bilangan **x**.
30. **i**: Variabel loop untuk mencari faktor bilangan.
31. **a**: Variabel boolean yang digunakan untuk menentukan apakah bilangan tersebut prima (true jika prima, false jika tidak).

2. Input Bilangan:

```
fmt.Print("bilangan: ")
fmt.Scan(&x)
```

32. Pengguna diminta untuk memasukkan sebuah bilangan, yang disimpan dalam variabel **x**.

3. Mencari Faktor Bilangan:

```
fmt.Print("faktor: ")
for i = 1; i <= x; i++ {
    if x%i == 0 {
```

```
        y++
        fmt.Print(i, " ")
    }
}
```

33. Program menggunakan loop dari **1** hingga **x** untuk mencari faktor bilangan.
34. Jika xmod i==0x \mod i == 0xmodi==0, maka **i** adalah faktor bilangan **x**.
35. Faktor bilangan akan dicetak ke layar, dan variabel **y** akan bertambah 1 untuk setiap faktor yang ditemukan.

4. Mengecek Apakah Bilangan Prima:

```
fmt.Print("\nprima: ")
if y <= 2 {
    a = true
    fmt.Print(a)
} else {
    fmt.Print(a)
}
```

- Sebuah bilangan prima adalah bilangan yang hanya memiliki **2 faktor**, yaitu **1** dan dirinya sendiri.
- Jika jumlah faktor (**y**) kurang dari atau sama dengan 2, maka **a** diatur menjadi true dan program mencetak true (bilangan tersebut adalah bilangan prima).
- Jika jumlah faktor lebih dari 2, **a** tetap false dan program mencetak false.

Contoh: Jika pengguna memasukkan:

```
bilangan: 7(input)

faktor: 1 7 (output)

prima: true(output)
```