Fakultas Informatika
School of Computing
Telkom University

CCH1A4 / Dasar Algoritma & Pemrogramanan

Yuliant Sibaroni M.T, Abdurahman Baizal M.Kom

KK Modeling and Computational Experiment

FUNGSI

- Overview Fungsi
- Konsep Fungsi
- Fungsi Sederhana
- Fungsi dengan Analisa Kasus
  - If ...Then ...Else
  - Depend .. On
- Fungsi dengan Perulangan
  - For ..to.. Do
  - Repeat Until /While ..Do
- Fungsi dengan Analisa kasus dan Perulangan

OVERVIEW

- Dalam pembuatan program yang memuat perhitungan yang kompleks, tidak disarankan untuk membuatnya dalam sebuah algoritma perhitungan yang utuh.
- Selain menyulitkan dalam proses validasinya, alur logika perhitungannya juga menjadi samar.
- Ketika pada bagian lain program memerlukan proses perhitungan yang sama, maka harus dilakukan penulisan ulang sehingga menjadi tidak efisien.
- Untuk mengatasi hal ini, bagian program yang terkait dengan komputasi, perlu dipecah-pecah dalam modul-modul yang lebih kecil, yaitu dalam bentuk **fungsi.**

KONSEP FUNGSI

Definisi

Dalam Kalkulus, sebuah fungsi didefinisikan sebagai pemetaan satu-satu ke suatu bilangan Real.

Sebuah fungsi setidaknya memerlukan satu inputan (variabel input) dan menghasilkan sebuah pengembalian (return)

Contoh fungsi dalam kalkulus:

$y = \sin(x)$ $\quad f(x, y) = z = x^2 + y^2$

Memiliki satu inputan : variabel x, bertipe Real
Menghasilkan sebuah keluaran : variabel y bertipe Real
Memiliki dua inputan : variabel x dan y, keduanya bertipe Real
Memiliki keluaran : *f*(*x*,*y*), bertipe Real

KONSEP FUNGSI

Notasi

function Nama**(parameter_1: Tipe, ..., parameter_n: Tipe) → Tipe_Return**

Kamus Lokal

....

Algoritma

.....

Bagian pertama

- **Bagian judul / spesifikasi fungsi :**

  Diawali dengan **function** ..., dilakukan pendefinisian **nama fungsi**, **parameter-parameter input** dan **tipenya** serta **tipe_return** dari fungsi

- **parameter_1...parameter_n** adalah **parameter input,** list parameter input boleh tidak ada (kosong).
- **Tipe_return** adalah tipe dari return/ pengembalian/ hasil fungsi tersebut. Untuk fungsi sederhana, tipe keluarannnya adalah tipe-tipe dasar seperti Real, Integer, Char, String dan Boolean.

KONSEP FUNGSI

Notasi

function Nama**(Variabel_1: Tipe, ..., Variabel_n: Tipe) → Tipe_Return**

Kamus Lokal

....

Algoritma

.....

Bagian pertama

Contoh :

function Hitung**(x: Real, y: Real) → Real**

Nama fungsi: Hitung

Tipe_Return : Real

Variabel input: x,y tipe: Real

KONSEP FUNGSI

Notasi

function Nama**(Variabel_1: Tipe, ..., Variabel_n: Tipe) → Tipe_Return**

Kamus Lokal

....

Algoritma

.....

Bagian kedua

- **Bagian kamus:**

  Diawali dengan **Kamus Lokal,** berisi pendefinisian variabel-variabel yang hanya digunakan secara lokal dalam fungsi ini

Kamus lokal

```
i : integer
```

KONSEP FUNGSI

Notasi

function Nama**(Variabel_1: Tipe, ..., Variabel_n: Tipe) → Tipe_Return**

Kamus Lokal

....

Algoritma

.....

Bagian ketiga

- **Bagian algoritma**

  Berisi langkah-langkah perhitungan, terakhir ditandai dengan notasi '→' diikuti dengan penulisan variabel yang akan dikeluarkan nilainya

Algoritma

```
Luas ← panjang * lebar
→ Luas
```

KONSEP FUNGSI

Pembuatan Program Utuh Yang Memuat Fungsi

- Sebuah fungsi tidak berdiri sendiri dalam sebuah program.
- Untuk dapat diakses/digunakan dalam sebuah program, spesifikasi fungsi harus didefinisikan terlebih dahulu.
- Karena sebuah fungsi menghasilkan sebuah nilai, untuk memanfaatkan fungsi yang telah dibuat, caranya seperti kita memanfaatkan sebuah nilai biasa. Jadi nilai suatu fungsi bisa dimanfaatkan pada proses *assignment*, pembuatan ekspresi dll.

KONSEP FUNGSI

Pendefinisian Fungsi dalam Program

Pada program, spesifikasi fungsi dituliskan dalam kamus. Setelah itu penulisan secara lengkap sebuah fungsi dilakukan (bisa dilakukan sebelum Algoritma program atau setelahnya).

Kamus

```
function  Hitung( x: Real, y: Real) → Real
```

Penggunaan atau Pemanggilan Fungsi

Setelah fungsi didefinisikan dibagian kamus dan ditulis secara lengkap, maka pada bagian algoritma program, sudah bisa dilakukan pemanggilan fungsi. Karena fungsi menghasilkan suatu nilai, maka pemanggilannya seperti kita memperlakukan suatu nilai.

```
z ← Hitung(a,b)
e ← Hitung(a,b) + Hitung(c,d)
output(Hitung(a,b))
```

Digunakan dalam *assignment*

Digunakan dalam pembuatan ekspresi

Ditampilkan dilayar secara langsung

Contoh 6.0 (Detail Program memuat fungsi)

```
Program  Hitung
{mencoba pembuatan fungsi dan pemanggilannya}
Kamus
    Function Jumlah (N : Integer) → Integer
    {mengembalikan jumlah dari 1 sampai dengan N}
    A,B,C  : integer
Algoritma
  input(A,B)
  Output(Jumlah(A), Jumlah(B))
  C ← Jumlah(A)+Jumlah(B)
  Output(C)

Function Jumlah (N : Integer) → Integer
{mengembalikan jumlah dari 1 sampai dengan N}
Kamus Lokal
  Sum, i :  Integer
Algoritma
  Sum ← 0
  For i ← 1 to N Do
     Sum  ← Sum + i
  → Sum
```

PEMBUATAN PROGRAM UTUH

Contoh 6.0 (Detail Program memuat fungsi)

```
Program Hitung
Kamus
    Function Jumlah (N : Integer) → Integer
    A,B,C  : integer
Algoritma
   input(A,B)
   Output(Jumlah(A), Jumlah(B))
   C ← Jumlah(A)+Jumlah(B)
   Output(C)

Function Jumlah (N : Integer) → Integer
Kamus Lokal
   Sum, i :  Integer
Algoritma
   Sum ← 0
   For i ← 1 to N Do
      Sum  ← Sum + i
   → Sum
```

Memanggil atau memanfaatkan fungsi

PEMBUATAN PROGRAM UTUH

Contoh 6.0 (Detail Program memuat fungsi)

```
Program  Hitung
Kamus
    Function Jumlah (N : Integer) → Integer
    A,B,C  : integer
Algoritma
  input(A,B)
  Output(Jumlah(A), Jumlah(B))
  C ← Jumlah(A)+Jumlah(B)
  Output(C)

Function Jumlah (N : Integer) → Integer
Kamus Lokal
  Sum, i :  Integer
Algoritma
  Sum ← 0
  For i ← 1 to N Do
     Sum  ← Sum + i
  → Sum
```

Parameter formal

Parameter Aktual

FUNGSI SEDERHANA

Definisi

Fungsi sederhana yang dimaksud disini adalah fungsi yang dalam algoritmanya hanya menggunakan perintah-perintah dasar, umumnya yang digunakan adalah **assignment** dan **ekspresi**. Dari pengertian ini, kompleksitas komputasi yang dilakukan oleh fungsi sederhana tergolong sederhana. Perintah **input** dan **output** tidak digunakan.

Contoh 6.1

Membuat fungsi untuk menghitung luas segitiga

Solusi

Untuk menghitung luas segitiga, diperlukan informasi nilai **alas** dan **tinggi**, dimana rumusnya : **Luas = 0,5 x alas x tinggi.** Ini berarti variabel input dalam fungsi ini ada 2, yaitu: **alas** dan **tinggi.** Misalkan nama fungsinya LuasSegitiga, maka spesifikasinya :

```
function LuasSegitiga( alas, tinggi : Real) → Real
```

FUNGSI SEDERHANA

Contoh 6.1 (Lanjutan)

Solusi

Pada spesifikasi fungsi tersebut, karena kedua tipe input adalah Real, maka hasil perkaliannya juga bertipe Real.

Pada bagian algoritma, hasil perhitungan alas x tinggi langsung dikirimkan sebagai keluaran fungsi:

→ 0.5*alas * tinggi

Fungsi secara lengkap:

```
function LuasSegitiga( Alas, Tinggi : Real) → Real

Kamus lokal

Algoritma
  → 0.5*alas * tinggi
```

FUNGSI DENGAN ANALISA KASUS

Menggunakan If ..Then ..

Fungsi yang dibuat saat ini memuat masalah yang yang memerlukan analisa kasus , dimana perintah analisa kasus yang digunakan adalah if ..then...

Contoh 6.2

Membuat fungsi IsOdd, yaitu fungsi untuk menentukan apakah suatu bilangan merupakan bilangan ganjil ataukah tidak

Solusi

Fungsi ini memerlukan input berupa sebuah bilangan bulat (karena bilangan ganjil adalah termasuk bilangan bulat) dan hasil dari fungsi ini adalah true/false (true bila bilangan yang diperiksa adalah ganjil). Ini berarti tipe keluaran dari fungsi ini adalah **boolean.** Berikut adalah penulisan spesifikasi fungsinya:

```
function IsOdd (A : integer) → boolean
```

FUNGSI DENGAN ANALISA KASUS

Contoh 6.2 (Lanjutan)

Solusi

Untuk menentukan apakah A bilangan ganjil / bukan, digunakan operator **mod**, lebih lengkapnya adalah : A mod 2. Jika hasilnya adalah 1, maka berarti A adalah bilangan ganjil (true), sebaliknya A adalah genap (false).

Jawaban lengkapnya adalah :

```
Function  IsOdd(A : Integer)→boolean
{mengembalikan True, jika ganjil}
Kamus lokal
Algoritma
  if (A mod 2 = 1) then → true
  else    → false
```

fungsi ini bisa dibuat lebih singkat :

```
Function  IsOdd(A : Integer)→ boolean
{mengembalikan True, jika ganjil}
Kamus lokal
Algoritma
   → (A mod 2 = 1)
```

FUNGSI DENGAN ANALISA KASUS

Menggunakan Depend On..

Fungsi yang dibuat saat ini memuat masalah yang yang memerlukan analisa kasus , dimana perintah analisa kasus yang digunakan adalah Depend ..On.

Contoh 6.3

Membuat fungsi Konversi , yaitu fungsi untuk mengkonversi nilai A,B,..., E menjadi dalam bentuk angka 4.0, 3.0, 2.0, 1.0 dan 0.0

Solusi

Fungsi ini memerlukan input berupa sebuah karakter, misalkan **Nilai** kemudian akan dikonversi menjadi sebuah nilai Real. Ini berarti tipe keluaran fungsi adalah **Real.** Sebagai contoh, nama fungsinya : **Konversi**, maka spesifikasi fungsinya :

```
Function Konversi (Nilai : Character) → Real
```

FUNGSI DENGAN ANALISA KASUS

Contoh 6.3 (Lanjutan)

Pada bagian Depend On, variabel yang terlibat hanya variabel **Nilai** saja, sedangkan Kondisi yang ada bisa diset dengan perbandingan antara Nilai dengan karakter A s.d. E tersebut, sebagai contoh :

```
Nilai = 'A'
```

Aksi yang dilakukan adalah mengeluarkan nilai yang bersesuaian dengan kondisi, misal untuk kondisi diatas, aksinya mengeluarkan nilai 4.0:

```
→ 4.0
```

Secara lengkap, isi dari Depend On adalah

```
Depend On (Nilai)
   Nilai='A'    : → 4.0
   Nilai='B'    : → 3.0
   Nilai='C'    : → 2.0
   Nilai='D'    : → 1.0
   Nilai='E'    : → 0.0
```

FUNGSI DENGAN ANALISA KASUS

Contoh 6.3 (Lanjutan)

Solusi

Fungsi secara lengkap adalah

```
Function Konversi (Nilai : Character) → Real
{mengkonversi nilai mutu menjadi nilai indeks real}
Kamus Lokal
Algoritma
Depend On (Nilai)
    Nilai='A' : → 4.0
    Nilai='B' : → 3.0
    Nilai='C' : → 2.0
    Nilai='D' : → 1.0
    Nilai='E' : → 0.0
```

FUNGSI DENGAN PERULANGAN

Menggunakan For..to..Do..

Fungsi yang dibuat saat ini memuat masalah yang yang memerlukan perulangan, dimana perintah analisa perulangan yang digunakan adalah For...to...Do. Pada perulangan ini, banyaknya perulangan adalah **pasti** dan **diketahui diawal**

Contoh 6.3

Membuat fungsi Jumlah , yaitu fungsi untuk menghitung 1+2+...+N, dimana nilai N bebas ditentukan oleh user

Solusi

Fungsi ini memerlukan input berupa sebuah bilangan bulat **N**. Hasil perhitungan 1+2+...+N juga sebuah bilangan bulat. Ini berarti tipe keluaran fungsi adalah **Integer.** Jadi spesifikasi fungsinya :

```
Function Jumlah (N : Integer) → Integer
```

FUNGSI DENGAN PERULANGAN

Menggunakan For..to..Do..

Contoh 6.3

Solusi

Langkah perhitungannya:

Karena menggunakan perulangan, maka **perintah** pada setiap pengulangan harus **sama.** Pada kasus ini, untuk setiap bertambahnya i maka kita akan tambahkan i pada jumlah sebelumnya. Misal untuk menyimpan jumlah bilangan, digunakan variabel **Sum,** maka perintah perulangannya adalah:

```
For i ← 1 to N Do

Sum  ← Sum + i
```

Terkait dengan perintah perulangan tersebut, maka sebelumnya harus didefinisikan variabel **Sum** dan **i**, dan **inisialisasi** Sum dengan 0 (Sum ← 0). Sedangkan setelah perulangan, aksi yang dilakukan (terminasi) adalah mengeluarkan hasil sum:

```
→ Sum
```

FUNGSI DENGAN PERULANGAN

Menggunakan For..to..Do..

Contoh 6.3

Solusi

Berikut jawaban fungsi secara lengkapnya

```
Function Jumlah (N : Integer) → Integer
{mengembalikan jumlah dari 1 sampai dengan N}
Kamus Lokal
   Sum, i :  Integer
Algoritma
   Sum ← 0
   For i ← 1 to N Do
      Sum  ← Sum + i
   → Sum
```

FUNGSI DENGAN PERULANGAN

Menggunakan Repeat ..Until / While ..Do

Fungsi yang dibuat saat ini memuat masalah yang yang memerlukan perulangan, dimana perintah analisa perulangan yang digunakan adalah **Repeat ...Until** atau **While ...Do**. Pada perulangan ini, banyaknya perulangan adalah **Pasti** / **tidak Pasti**, tergantung **kondisi.**

Contoh 6.4

Membuat fungsi Jumlah , yaitu fungsi untuk menghitung 1+2+...+N, dimana nilai N bebas ditentukan oleh user, mengunakan While ...Do.

Solusi

Sama seperti proses sebelumnya, spesifikasi fungsi tidak berbeda yaitu:

```
Function Jumlah (N : Integer) → Integer
```

FUNGSI DENGAN PERULANGAN

Menggunakan Repeat ..Until / While ..Do

Contoh 6.4 (Lanjutan)

Solusi

Proses yang berbeda yang dilakukan ketika menggunakan While ... do adalah menentukan **Kondisi Perulangan,** yaitu:

```
i <= N
```

(Jika menggunakan repeat until, maka harus ditentukan Kondisi Penghentian Perulangan, yaitu: i>N)

Dimana pada langkah awal, harus dilakukan Inisialisasi terhadap **Sum** dan **i** :

```
Sum ← 0
i ← 1
```

Tambahan aksi lainnya pada bagian pengulangan adalah :

```
i ← i+1
```

Ini dillakukan karena nilai i tidak bisa bertambah secara otomatis

FUNGSI DENGAN PERULANGAN

Menggunakan Repeat ..Until / While ..Do

Contoh 6.4 (Lanjutan)

Solusi

Berikut jawaban secara lengkap :

```
Function Jumlah (N : Integer) → Integer
{mengembalikan jumlah dari 1 sampai dengan N}
Kamus Lokal
   Sum, i :  Integer
Algoritma
   Sum ← 0; i ←0
   While  ( i<=N)  Do
      Sum  ← Sum + i
      i ← i+1
   → Sum
```

FUNGSI DENGAN ANALISA KASUS DAN PERULANGAN

Menggunakan If...Then dan For...to...Do

Fungsi yang dibuat saat ini memuat masalah yang yang memerlukan analisa kasus dan perulangan. Jenis analisa kasus dan perulangan yang bisa digunakan sebenarnya bebas, tetapi pada kasus ini kita akan menggunakan If...Then dan For..To ...Do.

Contoh 6.5

Membuat fungsi JumlahGanjil , yaitu fungsi untuk menghitung bilangan ganjil dari 1 sampai dengan N, dimana N ditentukan oleh user.

Solusi

Sama seperti proses sebelumnya, spesifikasi fungsi tidak berbeda yaitu:

```
Function JumlahGanjil (N : Integer) → Integer
```

FUNGSI DENGAN ANALISA KASUS DAN PERULANGAN

Menggunakan If...Then dan For...to...Do

Contoh 6.5 (Lanjutan)

Solusi

Proses yang dilakukan mirip dengan perulangan menggunakan for...to...do..., yang berbeda adalah kita akan menambahkan i ke Sum bila i adalah ganjil:

```
if ( i mod 2=1) then Sum ← Sum + i
```

Langkah lainnya sama.

```
Function JumlahGanjil(N : Integer) → Integer
{mengembalikan jumlah bilangan ganjil}
Kamus Lokal
  Sum, i :  Integer
Algoritma
  Sum ← 0
  For i ← 1 to N Do
     if ( i mod 2=1) then Sum ← Sum + i
  → Sum
```

REFERENSI

- Inggriani Liem, Diktat Kuliah IF223 Algoritma Dan Pemrograman, Jurusan Teknik Informatika Bandung, 1999

Fakultas Informatika
School of Computing
Telkom University

THANK YOU