Contoh ADT Sederhana 1

IF1210 – Algoritma dan Pemrograman 1
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Contoh spesifikasi ADT dengan notasi algoritmik

Struktur data

```
{ Modul ADT Time }
type Time: <  hours: integer[0..23],
             minutes: integer[0..59],
             seconds: integer[0..59]  >
```

Possible values

```
{ 0 ≤  hours  ≤ 23 }
{ 0 ≤ minutes ≤ 59 }
{ 0 ≤ seconds ≤ 59 }
```

Deklarasi operasi

```
{ Konstruktor: membentuk Time dari komponen-komponennya: h sebagai hours, m sebagai
               minutes, dan s sebagai seconds. }
procedure CreateTime(output t: Time, input h: integer[0..23],
                     input m: integer[0..59], input s: integer[0..59])
{ Mendapatkan komponen hours dari T }
function getHours(T: Time) → integer[0..23]
{ Mendapatkan komponen minutes dari T }
function getMinutes(T: Time) → integer[0..59]
{ Mendapatkan komponen seconds dari T }
function getSeconds(T: Time) → integer[0..59]
{ Selisih antara dua Time, dalam satuan detik }
function difference(start: Time, end: Time) → integer
```

Contoh realisasi operasi dalam notasi algoritmik

Signature operasi

Prasyarat operasi

```
d: Time) → integer
ime start dan end, dengan syarat start ≤ end. }
```

Ekspektasi hasil operasi pada setiap rentang kemungkinan masukan *(test case)*

```
{ EXPECT
      CreateTime(start, 1,2,3); CreateTime(end, 2,3,4)
          ⇒ difference(start,end) = 3661
      CreateTime(start, 2,3,4); CreateTime(end, 2,3,4)
          ⇒ difference(start,end) = 0
      CreateTime(start, 2,3,4); CreateTime(end, 1,2,3)
          ⇒ difference(start,end) = tak terdefinisi }
```

Body operasi

```
KAMUS LOKAL
    startSec, endSec: integer
ALGORITMA
    startSec ← getHours(start)*60*60 + getMinutes(start)*60 + getSeconds(start)
    endSec   ← getHours(end)*60*60   + getMinutes(end)*60   + getSeconds(end)
    → endSec-startSec
```

Operasi primitif

Beberapa operasi sangat bergantung pada struktur data yang digunakan.

Operasi-operasi tersebut adalah operasi primitif.

Pada contoh ADT Time sebelumnya, `CreateTime`, `getHours`, `getMinutes`, dan `getSeconds` akan memiliki implementasi yang berbeda jika Time menggunakan representasi detik saja.

Sementara itu, operasi selisih tidak harus bergantung pada struktur data karena implementasinya dapat memanfaatkan primitif yang sudah ada.

(Tidak mempertimbangkan efisiensi algoritma.)

Contoh: ADT Time alt-1 dan CreateTime

```
{ Modul ADT Time alt-1 }
type Time: <  hours: integer[0..23],    { 0 ≤  hours  ≤ 23 }
             minutes: integer[0..59],    { 0 ≤ minutes ≤ 59 }
             seconds: integer[0..59]  >  { 0 ≤ seconds ≤ 59 }

{ Konstruktor: membentuk Time t dari komponen-komponennya: h sebagai hours, m sebagai
               minutes, dan s sebagai seconds. }
procedure CreateTime(output t: Time, input h: integer[0..23],
                     input m: integer[0..59], input s: integer[0..59])
KAMUS
    -
ALGORITMA
    t.hours ← h
    t.minutes ← m
    t.seconds ← s
```

Contoh: ADT Time alt-2 dan CreateTime

```
{ Modul ADT Time alt-2 }
type Time: < seconds: integer[0..86399]  >  { 0 ≤ seconds ≤ 86400 }



{ Konstruktor: membentuk Time t dari komponen-komponennya: h sebagai hours, m sebagai
               minutes, dan s sebagai seconds. }
procedure CreateTime(output t: Time, input h: integer[0..23],
                     input m: integer[0..59], input s: integer[0..59])
KAMUS
    -
ALGORITMA
    t.seconds ← h*60*60 + m*60 + s
```

Contoh: implementasi getHours(t)

```
{ alt-1 }
{ Mendapatkan bagian hours dari t }
function getHours(t: Time) → integer[0..23]
KAMUS
    -
ALGORITMA
    → t.hours

-----------------------------------------------------------------------

{ alt-2 }
{ Mendapatkan bagian hours dari t }
function getHours(t: Time) → integer[0..23]
KAMUS
    -
ALGORITMA
    → t.seconds div (60*60)
```

Contoh: selisih dua waktu

```
function difference(start: Time, end: Time) → integer
{ Menghasilkan selisih antara dua Time start dan end, dengan syarat start ≤ end. }

{ EXPECT
      CreateTime(start, 1,2,3); CreateTime(end, 2,3,4)
          ⇒ difference(start,end) = 3661
      CreateTime(start, 2,3,4); CreateTime(end, 2,3,4)
          ⇒ difference(start,end) = 0
      CreateTime(start, 2,3,4); CreateTime(end, 1,2,3)
          ⇒ difference(start,end) = tak terdefinisi }
KAMUS LOKAL
    startSec, endSec: integer
ALGORITMA
    startSec ← getHours(start)*60*60 + getMinutes(start)*60 + getSeconds(start)
    endSec   ← getHours(end)*60*60   + getMinutes(end)*60   + getSeconds(end)
    → endSec-startSec
```