ADT TIME dalam Notasi Algoritmik

```
{ DEFINSI ADT TIME }
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 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])
{ I.S. h, m, s terdefinisi }
{ F.S. Terbentuk jam T dengan h sebagai hours, m sebagai minutes, dan
       s sebagai seconds.  }

{ Selectors }
function GetHours(T: Time) → integer[0..23]
{ Mendapatkan komponen hours dari T }
function GetMinutes(T: Time) → integer[0..59]
{ Mendapatkan komponen seconds dari T }
function GetSeconds(T: Time) → integer[0..59]
{ Mendapatkan komponen seconds dari T }

{ Komponen pengubah Time }
procedure SetHours(input/output T: Time, input newH: integer[0..23])
{ Mengubah komponen hours dari T menjadi newH }
procedure SetMinutes(input/output T: Time, input newM: integer[0..59])
{ Mengubah komponen minutes dari T menjadi newM }
procedure SetHours(input/output T: Time, input newS: integer[0..59])
{ Mengubah komponen seconds dari T menjadi newS }

{ Kelompok Validasi terhadap Type }
function IsValidTime (h,m,s : integer) → boolean
{ True jika h, m, s dapat membentuk Time yang valid sesuai definisi }
{ Dipakai untuk mengetes SEBELUM membentuk Time }

{ Kelompok Operasi Baca/Tulis }
procedure ReadTime (output T: Time)
{ I.S. T belum terdefinisi }
{ F.S. T terdefinisi dan merupakan Time yang valid }
{ Proses: mengulangi membaca komponen hours, minutes, dan seconds
  sehingga membentuk Time yang valid. }
procedure PrintTime (input T: Time)
{ I.S. T terdefinisi }
{ F.S. Nilai T tertulis di layar dengan format hours:minutes:seconds }
```

```
{ Kelompok Operasi Konversi terhadap Type }
function TimeToSeconds (T: Time) → integer
{ Menghasilkan konversi T ke jumlah detik }
{ Rumus: jumlah detik = 3600*hours + 60*minutes + seconds }
{ Nilai maksimum = 3600*23 + 60*59 + 59 = 86399 }
function SecondsToTime (n: integer) → Time
{ Menghasilkan konversi jumlah detik n ke Time }
{ Prekondisi: 0 ≤ n ≤ 86399 }

{ Kelompok Operasi Relasional }
function IsEQTime (T1: Time, T2: Time) → boolean
{ Menghasilkan true jika T1 = T2, false jika tidak }
function IsGTTime (T1: Time, T2: Time) → boolean
{ Menghasilkan true jika T1 > T2, false jika tidak }
function IsLTTime (T1: Time, T2: Time) → boolean
{ Menghasilkan true jika T1 < T2, false jika tidak }

{ Kelompok Operasi Aritmatika }
function NextSecond (T: Time) → Time
{ Menghasilkan Time 1 detik setelah T }
function PrevSecond (T: Time) → Time
{ Menghasilkan Time 1 detik sebelum T }
function Difference(start: Time, end: Time) → integer
{ Menghasilkan selisih antara dua Time, dalam satuan detik }
{ Hasilnya negatif jika start > end }
```

```
{ REALISASI ADT TIME }
{ Konstruktor }
procedure CreateTime (output T: Time, input h: integer[0..23],
                      input m: integer[0..59], input s:integer[0..59])
{ I.S. h, m, s terdefinisi }
{ F.S. Terbentuk jam T dengan h sebagai hours, m sebagai minutes, dan
       s sebagai seconds.  }
KAMUS LOKAL
ALGORITMA
    T.hours ← h
    T.minutes ← m
    T.seconds ← s
    { Alternatif: T ← <h,m,s> }
```

```
{ Selectors }
function GetHours(T: Time) → integer[0..23]
{ Mendapatkan komponen hours dari T }
KAMUS LOKAL
ALGORITMA
    → T.hours

function GetMinutes(T: Time) → integer[0..59]
{ Mendapatkan komponen seconds dari T }
KAMUS LOKAL
ALGORITMA
    → T.minutes

function GetSeconds(T: Time) → integer[0..59]
{ Mendapatkan komponen seconds dari T }
KAMUS LOKAL
ALGORITMA
    → T.seconds

{ Komponen pengubah Time }
procedure SetHours(input/output T: Time, input newH: integer[0..23])
{ Mengubah komponen hours dari T menjadi newH }
KAMUS LOKAL
ALGORITMA
    T.hours ← newH

procedure SetMinutes(input/output T: Time, input newM: integer[0..59])
{ Mengubah komponen minutes dari T menjadi newM }
KAMUS LOKAL
ALGORITMA
    T.minutes ← newM

procedure SetHours(input/output T: Time, input newS: integer[0..59])
{ Mengubah komponen seconds dari T menjadi newS }
KAMUS LOKAL
ALGORITMA
    T.seconds ← newS

{ Kelompok Validasi terhadap Type }
function IsValidTime (h,m,s : integer) → boolean
{ True jika h, m, s dapat membentuk Time yang valid sesuai definisi }
{ Dipakai untuk mengetes SEBELUM membentuk Time }
KAMUS LOKAL
ALGORITMA
    → (0 ≤ h ≤ 23) and (0 ≤ m ≤ 59) and (0 ≤ s ≤ 59)
```

```
{ Kelompok Operasi Baca/Tulis }
procedure ReadTime (output T: Time)
{ I.S. T belum terdefinisi }
{ F.S. T terdefinisi dan merupakan Time yang valid }
{ Proses: mengulangi membaca komponen hours, minutes, dan seconds
  sehingga membentuk Time yang valid. }
KAMUS LOKAL
    h, m, s: integer
ALGORITMA
    { Membaca komponen hours, minutes, seconds hingga didapatkan
      komponen yang valid}
    iterate
        input(h)
        input(m)
        input(s)
    stop: IsValidTime(h,m,s)
        output("Masukan Time tidak tepat. Ulangi.")

    { membentuk Time menggunakan konstruktor }
    CreateTime(T,h,m,s)

procedure PrintTime (input T: Time)
{ I.S. T terdefinisi }
{ F.S. Nilai T tertulis di layar dengan format hours:minutes:seconds }
KAMUS LOKAL
ALGORITMA
    output(GetHours(T), ":", GetMinutes(T), ":", GetSeconds(T))

{ Kelompok Operasi Konversi terhadap Type }
function TimeToSeconds (T: Time) → integer
{ Menghasilkan konversi T ke jumlah detik }
{ Rumus: jumlah detik = 3600*hours + 60*minutes + seconds }
{ Nilai maksimum = 3600*23 + 60*59 + 59 = 86399 }
KAMUS LOKAL
ALGORITMA
    → 3600*GetHours(T) + 60*GetMinutes(T) + GetSeconds(T)

function SecondsToTime (n: integer) → Time
{ Menghasilkan konversi jumlah detik n ke Time }
{ Prekondisi: 0 ≤ n ≤ 86399 }
KAMUS LOKAL
    T : Time
ALGORITMA
    CreateTime(T, n div 3600, (n mod 3600) div 60, n mod 60)
    → T
```

```
{ Kelompok Operasi Relasional }
function IsEQTime (T1: Time, T2: Time) → boolean
{ Menghasilkan true jika T1 = T2, false jika tidak }
KAMUS LOKAL
ALGORITMA
    → (GetHours(T1) = GetHours(T2)) and
       (GetMinutes(T1) = GetMinutes(T2)) and
       (GetSeconds(T1) = GetSeconds(T2))

function IsGTTime (T1: Time, T2: Time) → boolean
{ Menghasilkan true jika T1 > T2, false jika tidak }
KAMUS LOKAL
ALGORITMA
    depend on (GetHours(T1), GetHours(T2)):
        GetHours(T1) > GetHours(T2): → true
        GetHours(T1) < GetHours(T2): → false
    else { GetHours(T1) = GetHours(T2) }
        depend on (GetMinutes(T1), GetMinutes(T2)):
            GetMinutes(T1) > GetMinutes(T2): → true
            GetMinutes(T1) < GetMinutes(T2): → false
        else { GetMinutes(T1) = GetMinutes(T2) }
            → GetSeconds(T1) > GetSeconds(T2)
{ Alternatif: menggunakan TimeToSeconds }

function IsLTTime (T1: Time, T2: Time) → boolean
{ Menghasilkan true jika T1 < T2, false jika tidak }
KAMUS LOKAL
ALGORITMA
    depend on (GetHours(T1),GetHours(T2)):
        GetHours(T1) > GetHours(T2): → false
        GetHours(T1) < GetHours(T2): → true
    else { GetHours(T1) = GetHours(T2) }
        depend on (GetMinutes, GetMinutes(T2)):
            GetMinutes(T1) > GetMinutes(T2): → false
            GetMinutes(T1) < GetMinutes(T2): → true
        else { GetMinutes(T1) = GetMinutes(T2) }
            → GetSeconds(T1) < GetSeconds(T2)
{ Alternatif: menggunakan TimeToSeconds }
```

```
{ Kelompok Operasi Aritmatika }
function NextSecond (T: Time) → Time
{ Menghasilkan Time 1 detik setelah T }
KAMUS LOKAL
    T1 : Time
ALGORITMA
    if (GetSeconds(T) = 59) then
        if (GetMinutes(T) = 59) then
            if (GetHours(T) = 23) then
                CreateTime(T1,0,0,0) { hari berikutnya }
            else { GetHours(T) < 23 }
                CreateTime(T1, GetHours(T)+1, 0, 0)
        else { GetMinutes(T) < 59 }
            CreateTime(T1, GetHours(T), GetMinutes(T)+1, 0)
    else { GetSeconds(T) < 59 }
        CreateTime(T1, GetHours(T), GetMinutes(T), GetSeconds(T)+1)
    → T1
{ Alternatif: menggunakan TimeToSeconds dan SecondsToTime }

function PrevSecond (T: Time) → Time
{ Menghasilkan Time 1 detik sebelum T }
KAMUS LOKAL
    T1 : Time
ALGORITMA
    if (GetSeconds(T) = 0) then
        if (GetMinutes(T) = 0) then
            if (GetHours(T) = 0) then
                CreateTime(T1,23,59,59) { hari sebelumnya }
            else { GetHours(T) > 0 }
                CreateTime(T1, GetHours(T)-1, 59, 59)
        else { GetMinutes(T) > 0 }
            CreateTime(T1, GetHours(T), GetMinutes(T)-1, 59)
    else { GetSeconds(T) > 0 }
        CreateTime(T1, GetHours(T), GetMinutes(T), GetSeconds(T)-1)
    → T1
{ Alternatif: menggunakan TimeToSeconds dan SecondsToTime }
```

```
function Difference(start: Time, end: Time) → integer
{ Menghasilkan selisih antara dua Time, dalam satuan detik }
{ Hasilnya negatif jika start > end }
KAMUS LOKAL
    startSec, endSec: integer
ALGORITMA-1
    startSec ← GetHours(start)*3600 + GetMinutes(start)*60 +
               GetSeconds(start)
    endSec ← GetHours(end)*3600 + GetMinutes(end)*60 + GetSeconds(end)
    → endSec-startSec
ALGORITMA-2
    startSec ← TimeToSeconds(start)
    endSec ← TimeToSeconds(end)
    → endSec-startSec
```

```
{ Contoh Driver ADT TIME }

Program TestADTTime
{ Input: Membaca 2 ADT Time }
{ Output: Menuliskan selisih kedua Time dalam detik }
uses adtpoint

KAMUS
    T1, T2 : Time

ALGORITMA
    ReadTime(T1)
    ReadTime(T2)

    output("Time 1 = ")
    PrintTime(T1)
    output("Time 2 = ")
    PrintTime(T2)

    output("Selisih Time 1 dan Time 2 = ",Difference(T1,T2)," detik")
```