Algoritma, Struktur Data, dan *Abstract Data Type* (ADT)

IF2110/IF2111 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Syarat

Dasar pemrograman (imperatif ~ prosedural):

- Variabel, *assignment*
- Array
- *Control flow:*
  - Percabangan
  - Perulangan
- Fungsi & Prosedur

Algoritma

Definisi umum: suatu prosedur, **langkah demi langkah**, untuk *menyelesaikan suatu masalah* atau *mencapai sebuah tujuan*.

Definisi khusus: suatu rangkaian **terhingga** dari instruksi-instruksi yang **terdefinisi** dan dapat diimplementasikan pada komputer untuk *menyelesaikan himpunan permasalahan spesifik* yang ***computable***.

> **Terhingga**: langkah-langkah harus berhenti *at some point*.
>
> **Terdefinisi**: memenuhi semua klausa dari suatu definisi.
>
> ***Computable***: ada himpunan masalah yang belum diketahui apakah dapat diselesaikan dengan komputer atau tidak → bukan lingkup mata kuliah ini.

Contoh: algoritma Euclid untuk mencari FPB.

Struktur data

Adalah bagaimana data **diorganisasikan** dan **dikelola** agar dapat digunakan secara **efektif** dan **efisien**.

Contoh: merepresentasikan waktu (*time*) pukul 01:02:03

Cara I	Cara II
3 byte: h=01, m=02, s=03 `0x01 0x02 0x03` • Menghitung selisih waktu lebih rumit • Mencetak waktu dalam format “HH:MM:SS” lebih mudah	2 byte: detik sejak tengah malam = 3723 `0x0e 0x8b` • Menghitung selisih waktu lebih mudah • Mencetak waktu dalam format “HH:MM:SS” lebih rumit

Dari contoh tersebut juga terlihat bahwa **struktur data yang berbeda** membutuhkan **algoritma yang berbeda** meskipun merepresentasikan **permasalahan yang sama**.

“Algoritma + Struktur Data = Program”

(Niklaus Wirth)

Abstract data type (ADT)

ADT adalah pemodelan suatu tipe data yang didefinisikan perilakunya berdasarkan:

data yang terkandung di dalamnya,

himpunan nilai yang mungkin dimiliki oleh data tersebut, serta

operasi yang dapat diterapkan terhadap data tersebut.

Implementasi dari suatu ADT mencakup **struktur data** untuk data yang didefinisikan oleh ADT dan **algoritma** untuk setiap operasi terhadap data tersebut.

ADT, struktur data, dan algoritma adalah salah satu mekanisme modularitas (→ *reusability*) dalam paradigma prosedural.

Modularitas ADT, struktur data, dan algoritma

Menghitung selisih antara dua instan waktu, tanpa ADT. Bayangkan jika operasi ini perlu dilakukan berulang kali di program yang Anda buat:

```
{ waktu #1, 13:45:00 }                  { waktu #2, 14:30:00 }
h1 ← 13                          ;      h2 ← 14
m1 ← 45                          ;      m2 ← 30
s1 ← 0                           ;      s2 ← 0
{ selisih waktu #2 dengan waktu #1 }
ss1 ← h1*60*60 + m1*60 + s1      ;      ss2 ← h2*60*60 + m2*60 + s2
d_ss ← ss2-ss1
```

Dengan ADT:

```
{ waktu #1, 13:45:00 }                  { waktu #2, 14:30:00 }
CreateTime(t1, 13,45,0)           ; CreateTime(t2, 14,30,0)
d_t ← difference(t1,t2)
```

Modularitas tanpa ADT?

Arguably, beberapa hal dapat dimodularkan dengan fungsi/prosedur, tanpa ADT:

```
{ waktu #1, 13:45:00 }                    { waktu #2, 14:30:00 }
h1 ← 13                                ;  h2 ← 14
m1 ← 45                                ;  m2 ← 30
s1 ← 0                                 ;  s2 ← 0
{ selisih waktu #2 dengan waktu #1 }
ss1 ← hhmmssToSeconds(h1,m1,s1)        ;  ss2 ← hhmmssToSeconds(h2,m2,s2)
d_ss ← ss2-ss1
{ atau: }
d_ss ← difference(h2,m2,s2,h1,m1,s1)
```

Namun mekanisme ini tidak mencegah pemrogram merangkai waktu secara salah, misalnya: `output(h1 + ":" + m2 + ":" + s1)` yang mencetak “13:30:00”, yang secara lojik tidak pernah relevan pada program di atas.

Ringkasan: ADT

Membuat suatu ADT mencakup:

Spesifikasi:

Definisi tipe data

Himpunan nilai yang mungkin

Daftar **operasi-operasi** (primitif maupun penunjang)

Prekondisi yang harus terpenuhi sebelum operasi dimulai

State setelah operasi selesai

Implementasi:

Struktur data konkret

Implementasi algoritma setiap operasi

"**Test case**" untuk setiap operasi pada setiap rentang kemungkinan nilai