Rekursi

Tim Olimpiade Komputer Indonesia

Pendahuluan

Melalui dokumen ini, kalian akan:

- Memahami konsep rekursi.
- Mempelajarai rekursi tak bercabang.

Pendahuluan (lanj.)

- Rekursi adalah keadaan dimana kita membuat sebuah fungsi untuk menyelesaikan sebuah permasalahan dengan cara memanggil diri sendiri secara berulang-ulang
- Ketika Anda memanggil fungsi, fungsi itu akan menyelesaikan masalah. Jika terlalu besar, fungsi itu akan memanggil diri sendiri tapi dengan permasalahan yang lebih kecil
- Fungsi itu akan memanggil diri sendiri sampai kita berhasil mengecilkan permasalahan itu ke base-case atau permasalah paling simpel

<TODO: Konten Wajib>

- Mengapa perlu ada rekursi

Cara mengerjakan soal dengan rekursi

Ketika kita mengerjakan soal dengan rekursi, kita harus memikirkan 3 perihal:

- *BaseCase* Apakah base-case (atau soal paling simpel) dari pertanyaan ini?
- Apa yang harus kita lakukan jika soal yang kita terima bukan *basecase*?
- Bagaimana cara implementasinya dalam fungsi tersebut?

Soal: Faktorial

Deskripsi:

- Operasi faktorial (dikenal dengan notasi $N!$) adalah operasi mengali bilangan dari 1 sampai N.
- Contoh: Jika $N = 4$, maka 4!=1 x 2 x 3 x 4 = 24
- Bantu Pak Dengklek cari hasil $N!$

Soal: Faktorial (lanj.)

Format masukan:

- Sebuah baris berisi sebuah bilangan N

Format keluaran:

- Sebuah baris berisi hasil $N!$

Batasan:

- $1 \leq N \leq 10$

Solusi

- Permasalahan: Bagaimana cara mengalihkan bilangan dari 1 sampai N?
- Ide 1: Loop
  - Kita set variabel *jawaban* $:= 1$
  - Buat sebuah loop dari i sampai N
  - Untuk setiap i, *jawaban* dikali dengan i
- Ide 2: Rekursi (Akan kita jelaskan)

Contoh Solusi: faktorial_loop.pas

Implementasinya cukup sederhana:

```
(* Faktorial dengan loop : *)
var i,N,jawaban:longint;
begin
  readln(N);
    jawaban:=1; (*Initialisasi*)
  for i:=1 to N do
      jawaban:= jawaban * i;
  writeln(jawaban);
end.
```

Penjelasan solusi Rekursi

Seperti yang kami jelaskan tadi, mengerjakan soal rekursi ada 3 tahap

- Tahap 1 : Base Case
  - Jika kita lihat di batasan soal, soal kita akan menjawab soal dari 1! sampai 10!
  - Pertanyaan : Dari 1! sampai 10!, mana yang paling gampang dihitung?
  - Yang paling gampang dihitung adalah 1!, jadi 1! adalah base casenya.
  - $1! = 1$

Penjelasan solusi Rekursi (lanj.)

- Tahap 2 : Bukan Base Case
  - Bagaimana kalau fungsi kita menerima soal yang bukan 1! ?
  - Sesuai penjelasan di pendahuluan, kita harus mengecilkan masalah!
  - Setelah dipikir-pikir untuk semua N lebih dari 1. N! = N X (N-1)!
  - Contoh: 5! = 5 X 4!
  - Dengan ini, kita telah mengecilkan permasalahan dari 5! jadi 4!
  - 4! lalu akan kita kecilkan terus menerus sampai menjadi *basecase* yaitu 1!

Contoh Solusi: faktorial_rekursi.pas

Perhatikan contoh berikut:

```
(* Faktorial dengan rekursi : *)
function faktorial(Z:longint):longint;
begin
  if(Z==1)then (* Base Case *)
    f:=1;
  else
    f:=Z*f(Z-1); (* Proses Mengecilkan Masalah*)
 end;
```

Penjelasan solusi rekursi

Perhatikan maksud kami:

- Anggap program kita ditanya nilai dari 5! (N=5)
- Kita akan panggil faktorial(5)
- faktorial(5) akan mengecek, apakah N=5 adalah *basecase*
- Ternyata bukan! Karena baru *basecase* kalau N=1
- Tadi kita ingat kalau 5! = 5 X 4!
- Untuk menjawab soal dari nilai 5!, faktorial(5) butuh nilai dari 4!
- Karena dia butuh nilai dari 4! , dia akan panggil faktorial(4)

Penjelasan solusi rekursi (lanj.)

- faktorial(4) akan mengecek apa sudah base case! Belum! Kita ingat 4!= 4 X 3! Dia butuh nilai dari 3! maka dia akan panggil faktorial(3)
- faktorial(3) akan mengecek apa sudah base case! Belum! Kita ingat 3!= 3 X 2! Dia butuh nilai dari 2! maka dia akan panggil faktorial(2)
- faktorial(2) akan mengecek apa sudah base case! Belum! Kita ingat 2!= 2 X 1! Dia butuh nilai dari 1! maka dia akan panggil faktorial (1)

Penjelasan solusi rekursi (lanj.)

- faktorial(1) akan mengecek apa sudah base case! Sudah! Karena 1! = 1, dia kembalikan angka 1
- faktorial(2) menangkap jawaban 1! dari fakotrial(1). 2! = 2 X 1! = 2 X 1 =2. faktorial(2) mengembalikan angka 2
- faktorial(3) menangkap jawaban 2! dari faktorial(2). 3!= 3 X 2! = 3 X 2 =6. faktorial(3) mengembalikan angka 6
- Dst sampai faktorial(5) menangkap jawaban 4! dari faktorial(4). 5!= 5 X 4! = 5 X 24 =120.

Kompleksitas solusi

- Kompleksitas solusi adalah O(N). Karena jika kita panggil **faktorial(N)**, pasti terpanggil semua dari **faktorial(1)** sampai **f(N)**.
- Jumlah pemanggilan adalah N kali. Makanya O(N).

Materi selanjutnya

- Rekursi bercabang. Di contoh faktorial tadi, program menangkap cuma 1 variabel. Bagaimana kalau program menangkap 2 variabel? (atau bahkan lebih)