06. Rekursi

ARNA FARIZA

YULIANA SETIOWATI

Capaian Pembelajaran

1. Mahasiswa dapat mengimplementasikan operasi sisip dan hapus pada multiple list dalam bahasa pemrograman.

Materi

- Mengapa menggunakan rekursi
- Konsep dasar rekursi
- Penyelesaian Faktorial dengan Rekursi
- Rekursi Tail
- Fungsi Rekursi :
  - Faktorial
  - Fibonanci
  - Prima

Mengapa Menggunakan Rekursi

- Merumuskan solusi sederhana dalam sebuah permasalahan yang sulit untuk diselesaikan secara iteratif dengan menggunakan loop for, while, atau do while.
- Mendefinisikan permasalahan dengan konsisten dan sederhana.
- Membantu untuk mengekspresikan algoritma menjadi sebuah rumusan yang membuat tampilan algoritma tersebut lebih mudah untuk dianalisa.

Konsep Dasar

- Rekursi mempunyai arti suatu proses yang bisa memanggil dirinya sendiri.
- Dalam sebuah rekursi sebenarnya tekandung pengertian sebuah prosedur atau fungsi.
- Perbedaannya adalah bahwa rekursi bisa memanggil dirinya sendiri, kalau prosedur atau fungsi harus dipanggil melalui pemanggil prosedur atau fungsi.

Penyelesaian Faktorial dengan Rekursi

- ✓ Penyelesaian faktorial harus dilakukan secara iteratif, dimana

  n! = (n)(n-1)(n-2) …. (1)

  1! = 1

  0! = 1

- ✓ Dengan Rekursi dapat menggunakan rumus berikut

$$F(n) = \begin{cases} 1 & \text{jika } n=0,\ n=1 \\ nF(n-1) & \text{jika } n>1 \end{cases}$$

Fase pada Rekursi

- Fase awal
  - Masing-masing proses memanggil dirinya sendiri.
  - Fase berhenti ketika pemanggilan telah mencapai kondisi terminal.
  - Kondisi teminal adalah kondisi dimana sebuah fungsi rekursi kembali dari pemanggilan, artinya fungsi tersebut sudah tidak memanggil dirinya sendiri dan kembali pada sebuah nilai.
  - Contoh : dalam penghitungan faktorial dari n, kondisi terminal adalah *n = 1, n = 0.* Untuk setiap fungsi rekursi, minimal harus ada satu kondisi terminal.
- Fase balik
  - Fungsi sebelumnya akan dikunjungi lagi dalam fase balik ini.
  - Fase ini berlanjut sampai pemanggilan awal, hingga secara lengkap proses telah berjalan

Fase pada fungsi rekursi

$F(4) = 4 \times F(3)$	fase awal
$F(3) = 3 \times F(2)$	.
$F(2) = 2 \times F(1)$	.
$F(1) = 1$	kondisi terminal
$F(2) = (2) \times (1)$	fase balik
$F(3) = (3) \times (2)$	.
$F(4) = (4) \times (6)$	.
24	rekursi lengkap

Fungsi Rekursi : Faktorial

```
int fact(int n)
{
   if (n < 0)
      return 0;
   else if (n == 0)
      return 1;
   else if (n == 1)
      return 1;
   else
      return n * fact(n-1);
}
```

Faktorial tanpa Rekursi

```
int fact_it (int n)
{
    int temp;
    temp = 1;
    if (n < 0)
        return 0;
    else if (n == 0)
        return 1;
    else if (n == 1)
        return 1;
    else
        for (i=2; i<=n; ++i)
            temp = temp * i;
    return (temp);
}
```

Rekursi Tail

- Sebuah proses rekursi dikatakan rekursi tail jika pernyataan terakhir yang akan dieksekusi berada dalam tubuh fungsi dan hasil yang kembali pada fungsi tersebut bukanlah bagian dari fungsi tersebut.
- Ciri fungsi rekursi tail $\rightarrow$ tidak memiliki aktivitas selama fase balik.
- Pemanggilan rekursi adalah pernyataan terakhir yang dieksekusi dalam aktivitas yang sedang berlangsung, sehingga tidak ada aktivitas yang harus dikerjakan pada saat pemanggilan kembali.

Faktorial dg rekursi tail

$$F(n,a) = \begin{cases} a & \text{jika } n=0,\ n=1 \\ F(n-1,na) & \text{jika } n>1 \end{cases}$$

Faktorial dengan rekursi tail

$F(4,1) = F(3,4)$		fase awal $F(3,4)=F(2,12)$
$F(2,12) = F(1,24)$		.
$F(1,24) = 24$		kondisi terminal
	24	fase balik
		rekursi lengkap

Fungsi faktorial dg rekursi tail

```
int facttail(int n, int a)
{
    if (n < 0)
        return 0;
    else if (n == 0)
        return 1;
    else if (n == 1)
        return a;
    else
        return facttail(n-1,n*a);
}
```

Contoh Fungsi Rekursi Tanpa Batas Akhir

```
void Tidak_Berhenti()
{
    printf("Ctrl-Break untuk berhenti.\n");
    Tidak_Berhenti();
}

main()
{
    Tidak_Berhenti();
}
```

Contoh Fungsi Rekursi Dengan Batas Akhir

```
void Berhenti_N_Kali(int n)
{
    static int i=0;
    if (n<=0) return;
    printf("%d kali\n",++i);
    Berhenti_N_Kali(n-1);
}

main()
{
    int N=3;
    Berhenti_N_Kali(N);
}
```

Contoh Rekursi : Deret Fibonanci

- Deret Fibonanci

  1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

- Fungsi Fibonanci dg rekursi

$$F(n) = \begin{cases} 1 & \text{jika } n <= 2 \\ F(n-1) + F(n-2) & \text{jika } n>2 \end{cases}$$

Contoh Rekursi : Menentukan Bilangan Prima atau Buka Prima

- Bilangan Prima

  1, 2, 3, 5, 7, 11, 13, 17, 19, 23, ….

$$F(a,i) = \begin{cases} 1 & \text{jika } I = 1 \\ 0 & \text{jika } a \,\%\, I = 0 \\ F(a,i-1) & \text{lainnya} \end{cases}$$

Rangkuman

- Rekursi adalah proses yang memanggil dirinya sendiri
- Fungsi rekursi menggantikan proses iteratif sehingga program menjadi lebih sederhana dan mudah dianalisa

Latihan

1. Buatlah sebuah fungsi yang menulis angka dari n ke 0 dengan menggunakan proses rekursi.
2. Tuliskan sebuah fungsi untuk menulis angka dari 0 ke n dengan menggunakan proses rekursi.
3. Tuliskan sebuah fungsi rekursi yang melakukan pengecekan apakah sebuah elemen X merupakan anggota dari sebuah array a[n].
4. Tulis sebuah fungsi yang melakukan pengecekan apakah sebuah angka merupakan bilangan prima atau bukan (n bukan bilangan prima jika dapat dibagi dengan angka kurang dari n)