Algoritma dan Pemrograman Lanjut

Pertemuan Ke-5
Rekursif

Disusun Oleh :
Wilis Kaswidjanti, S.Si.,M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Pembangunan Nasional "Veteran"
Yogyakarta

Algoritma dan Pemrograman Lanjut

Judul Materi : Rekursif

Deskripsi Materi : Materi ini membahas bentuk algoritma rekursif dan penggunaan rekursif untuk menyelesaikan perosalan-persoalan iterasi

Tujuan Instruksional Khusus :

1. Memahami algoritma rekursif
2. Menjelaskan bentuk algoritma rekursif
3. Mendeskripsikan parameter-parameter pada algoritma rekursif
4. Membandingkan algoritma rekursif dengan algoritma non rekursif

Referensi :

- Buku Teks

Munir, Rinaldi (2005), *Algoritma dan Pemrograman dalam Bahasa Pascal dan C*, Buku 2, Edisi Ketiga, Penerbit Informatika Bandung, Bab 5, hal 169-226.
Charibaldi, N. (2004), *Modul Kuliah Algoritma Pemrograman II*, Edisi Kedua, Yogyakarta

- Buku Acuan/Referensi

Brassard, Gilles (1999), *Fundamentals of algorithma*, PrinteceHall.
Jarne, Stroustrup B. (1997), *C++ Programming language,* AT &T.
Kristanto, Andri (2003), *Algoritma pemrograman C++*, Graha Ilmu.
Schildt,Herbert (2000), *The Complete Reference C++*, McGraw-Hill.
Sedgewick, R. (2000), *Algoritma Third edition In C part 5*, Addison Wesley.

REKURSIF

PENDAHULUAN

Rekursif merupakan alat untuk memecahkan masalah dalam suatu fungsi atau procedure yang memanggil dirinya sendiri. Definisi menurut Niclaus Wirth : “ An object is said be recursive if it partially consist or is defines in terms of itself”

Mengapa kita memerlukan rekursif? Karena ada beberapa kasus yang akan jauh lebih mudah diselesaikan jika menggunakan rekursif.

Secara teori semua masalah yang dapat dipecahkan dengan rekursif dapat dipecahkan dengan tanpa rekursif. Meskipun dalam kenyataa ini, rekursif sangat bermanfaat sebab beberapa masalah mempunyai solusi yang sulit tanpa menggunakan rekursif.

Penggunaan rekursif kadang-kadang harus mengorbankan efisiensi dan kecepatan, disamping itu ada masalah yang sering muncul dalam rekursif, yaitu eksekusi yang tidak pernah berhenti, akibatnya memori tumpukan akan habis dan computer hank.

Pada dasarnya rekursif sering digunakan dalam perhitungan matematika, sebagai contoh pertimbangan fungsi factorial dan juga bilangan Fibonacci

ISI

Logika Rekursif adalah suatu fungsi berparameter yang memanggil dirinya sendiri dengan harga parameter yang berbeda. Logika ini dipakai sebagai pengganti proses iterasi. Kelebihan logika bentuk ini mudah dipahami alurnya, namun kelemahannya pada penggunaan register stack yang sangat membebani kecepatan jalannya program.

Bentuk dan sifat rekursif adalah sebagai berikut :

- Ada bagian base case dan ada bagian general case
- Paling sedikit mempunyai general base
- Selalu dalam bentuk fungsi-fungsi

- Selalu menggunakan statement percabangan

Berikut contoh fungsi rekursi, yaitu faktorial dan bilangan Fibonacci.

A. Faktorial

Salah satu contoh yang sering digunakan untuk menjelaskan rekursif adalah fungsi fakorial. Fungsi factorial dari bilangan bulat positif n didefinisikan sebagai berikut:

n!= n.(n-1)!, jika n>1

n!= 1, jika n=0, 1

contoh :

3!= 3. 2!

3!= 3. 2. 1!

3!= 3. 2. 1

3!= 6

Cara lain untuk mendefiniskan fungsi tersebut sebagai berikut:

- Dalam definisi ini, nilai 3! diekspresikan sebagai 3 x 2!, dengan kata lain untuk menghitung factorial 3, kita harus menghitung factorial lain yaitu factorial 2.
- Jika definisi nilai 2!=2 x 1!
- Factorial 1 didefinisikan dengan nilai 1.
- Jadi jika factorial 3!= 3 x 2 x 1, yang mana secara pasti mempunyai nilai sama yang diperoleh dari definisi tanpa rekursif.

Kita dapat menuliskan fungsi penghitung factorial seperti dibawah ini,

Faktorial(n)	→ hasilnya n*Faktorial(n-1),	jika n > 1	{general case}
	→ hasilnya 1,	jika n=1 atau 2	{base case}

Algoritma :

```
Function Faktorial (input n: integer) → integer
Deklarasi
 {tidak ada}
Deskripsi
  if (n = 0) or (n = 1) then
```

```
    return (1)
else
    return  (n * Faktorial(n-1))
endif
```

Bahasa C++ :

```
1. int Faktorial(int n)
2. {
3.   if ((n == 0) || (n == 1 ))
4.      return (1);
5.   else
6.      return (n * Faktorial(n-1));
7. }
```

- Pada *baris 3* dari fungsi diatas,

  nilai n dicek sama dengan 0 atau 1,

  jika ya, maka fungsi mengembalikan nilai 1 {*baris 4*},

  jika tidak, fungsi mengembalikan nilai n * Faktorial (n -1)

  {*baris 6*}

- disinilah letak proses rekursif itu, perhatikan fungsi factorial ini memanggil dirinya sendiri tetapi dengan parameter (n-1)

B. Bilangan Fibonacci

Fungsi lain yang dapat diubah kebentuk rekursif adalah perhitungan Fibonacci. Bilangan Fibonacci dapat didefinisikan sebagai berikut:

$f_n = f_{n-1} + f_{n-2}$ untuk n>1

$f_0 = 0$

$f_1 = 1$

berikut ini adalah barisan bilangan Fibonacci mulai dari n=1

1 1 2 3 5 8 13 21 34

Algoritma yang dipakai

```
Function Fibonacci(input n:integer) → integer
Deklarasi Lokal
  {tidak ada}
Deskripsi
  if (n ==1 || n==2) then
```

```
        return (l)
    else
        return (Fibonacci(n-1)+Fibonacci(n-2))
    endif
```

Contoh,

Untuk ukuran n= 4, proses perhitungan Fibonacci dapat dilakukan sebagai berikut:

$f_4 = f_3+f_2$

$f_4 = (f_2+f_1) + f_2$

$f_4 = (1+1) +1$

$f_4 = 3$

C. Penerapan faktorial

- **Kombinasi**

```
Function Kombinasi (input n, r : integer) → real
Deklarasi
 If (n < r) Then
        return (0)
 Else
        return (Faktorial(n)/(Faktorial(r)*Faktorial(n-r)))
 Endif
```

- **Permutasi**

```
Function Permutasi (input n, r : integer) → real
Deklarasi
    {tidak ada}
Deskripsi
    if (n< r) then
        return (0)
    else
        return (Faktorial(n) / Faktorial(n-r))
    endif
```

Contoh program menghitung permutasi dengan bahasa C++ (lengkap) :

```
#include <iostream.h>
int Faktorial(int n);
float Kombinasi(int n, int r);
main()
{
      cout << ”Kombinasi C(3,2) = ”<< Kombinasi(3,2);
```

```
}

int Faktorial(int n)
{
  if ((n == 0) || (n == 1 ))
        return(1);
  else
        return(n * Faktorial(n-1));
}

float Kombinasi(int n,int r)
{
  if (n < 1)
        return(0);
  else
        return(Faktorial(n)/(Faktorial(r)* Faktorial(n-r)));
}
```

PENUTUP

Persoalan yang diselesaikan dengan iterasi dapat juga diselesaikan dengan rekursif. Definisi rekursif harus tergantung kondisi, yang harus ada keadaan dimana kita dapat menghentikannya.

SOAL-SOAL

1. Buat program untuk menghitung deret S = 1+2+3+4+5 menggunakan function rekursi
2. Buat program untuk menghitung deret S = 2+4+6+8+10 menggunakan function rekursi