Praktikum 9

Rekusif

A. TUJUAN PEMBELAJARAN

Setelah melakukan praktikum dalam bab ini, mahasiswa diharapkan mampu:

1. Memahami mengenai konsep rekursif
2. Mampu memecahkan permasalahan dengan konsep rekursif

B. DASAR TEORI

Rekursif berarti bahwa suatu proses bisa memanggil dirinya sendiri. Rekursif adalah kemampuan suatu rutin untuk memanggil dirinya sendiri. Dalam Rekursif sebenarnya terkandung pengertian prosedur dan fungsi. Perbedaannya adalah bahwa rekursif bisa memanggil ke dirinya sendiri, tetapi prosedur dan fungsi harus dipanggil lewat pemanggil prosedur dan fungsi. Rekursif merupakan teknik pemrograman yang penting dan beberapa bahasa pemrograman mendukung keberadaan proses rekursif ini. Dalam prosedur dan fungsi, pemanggilan ke dirinya sendiri bisa berarti proses berulang yang tidak bisa diketahui kapan akan berakhir.

Contoh paling sederhana dari proses rekursif ini adalah proses menghitung nilai faktorial dari suatu bilangan bulat positif dan mencari deret Fibbonacci dari suatu bilangan bulat.

1. Nilai faktorial secara rekursif dapat ditulis sebagai

   0 ! = 1

   N ! = N x (N-1) !

   yang secara pemrograman dapat ditulis sebagai

   Faktorial(0) = 1 (1)

   Faktorial(N) = N*Faktorial(N-1) (2)

Persamaan (2) di atas adalah contoh hubungan rekurens (*recurrence relation*), yang berarti bahwa nilai suatu fungsi dengan argumen tertentu bisa dihitung dari fungsi yang sama dengan argumen yang lebih kecil. Persamaan (1) tidak bersifat rekursif, disebut nilai awal atau basis. Setiap fungsi rekursif paling sedikit mempunyai satu nilai awal, jika tidak fungsi tersebut tidak bisa dihitung secara eksplisit.

2. Bilangan Fibbonacci didefinisikan sebagai berikut

   1 1 2 3 5 8 13 21 34 55 89 …

   dari barisan tersebut dapat dilihat bahwa bilangan ke-N (N>2) dalam barisan dapat dicari dari dua bilangan sebelumnya yang terdekat dengan bilangan N, yaitu bilangan ke-(N-1) dan bilangan ke-(N-2), sehingga dapat dirumuskan sebagai

   Fibbonacci(1) = 1 (1)

   Fibbonacci(2) = 1 (2)

   Fibbonacci(N) = Fibbonacci(N-1) + Fibbonacci(N-2) (3)

   Dengan persamaan (1) dan (2) adalah basis dan persamaan (3) adalah rekurensnya

Dalam beberapa situasi, pemecahan secara rekursif maupun secara iteratif mempunyai keuntungan dan kekurangan yang bisa saling diperbandingkan. Adalah cukup sulit untuk menentukan mana yang paling sederhana, paling jelas, paling efisien dan paling mudah disbanding yang lain. Boleh dikatakan pemilihan cara iterative maupun rekursif merupakan kesenangan seorang programmer dan tergantung konteks permasalahan yang akan dipecahkan sesuai dengan kesanggupan yang bersangkutan.

Prosedur Dan Fungsi Rekursif

Prosedur dan fungsi merupakan sub program yang sangat bermanfaat dalam pemrograman, terutama untuk program atau proyek yang besar. Manfaat penggunaan sub program antara lain adalah :

1. meningkatkan *readibility*, yaitu mempermudah pembacaan program
2. meningkatkan *modularity*, yaitu memecah sesuatu yang besar menjadi modul-modul atau bagian-bagian yang lebih kecil sesuai dengan fungsinya, sehingga mempermudah pengecekan, testing dan lokalisasi kesalahan.

3. meningkatkan *reusability*, yaitu suatu sub program dapat dipakai berulang kali dengan hanya memanggil sub program tersebut tanpa menuliskan perintah-perintah yang semestinya diulang-ulang.

Sub Program Rekursif adalah sub program yang memanggil dirinya sendiri selama kondisi pemanggilan dipenuhi. Dengan melihat sifat sub program rekursif di atas maka sub program rekursif harus memiliki :

1. kondisi yang menyebabkan pemanggilan dirinya berhenti (disebut **kondisi khusus** atau **special condition**)
2. pemanggilan diri sub program (yaitu bila kondisi khusus tidak dipenuhi)

Secara umum bentuk dari sub program rekursif memiliki statemen kondisional :

if *kondisi khusus tak dipenuhi*
then *panggil diri-sendiri dengan parameter yang sesuai*
else *lakukan instruksi yang akan dieksekusi bila kondisi khusus dipenuhi*

Sub program rekursif umumnya dipakai untuk permasalahan yang memiliki langkah penyelesaian yang terpola atau langkah-langkah yang teratur. Bila kita memiliki suatu permasalahan dan kita mengetahui algoritma penyelesaiannya, kadang-kadang sub program rekursif menjadi pilihan kita bila memang memungkinkan untuk dipergunakan. Secara algoritmis (dari segi algoritma, yaitu bila kita mempertimbangkan penggunaan memori, waktu eksekusi sub program) sub program rekursif sering bersifat tidak efisien.
Dengan demikian sub program rekursif umumnya memiliki efisiensi dalam penulisan perintah, tetapi kadang tidak efisien secara algoritmis. Meskipun demikian banyak pula permasalahan-permasalahan yang lebih sesuai diselesaikan dengan cara rekursif (misalnya dalam algoritma pengurutan dan pencarian).

C. TUGAS PENDAHULUAN

Jawablah pertanyaan berikut ini :

1. Apa yang dimaksud dengan rekursi?
2. Tuliskan fungsi untuk menghitung nilai faktorial
3. Tuliskan fungsi untuk menampilkan nilai fibonanci dari deret fibonanci
4. Tuliskan fungsi untuk menentukan sebuah bilangan apakah termasuk bilangan prima atau bukan prima

D. PERCOBAAN

Percobaan 1 : Fungsi rekursif untuk menghitung nilai faktorial

```
#include<stdio.h>

int faktorial(int x)
{
      if(x==1)
            return x;
      else
            return x * faktorial(x-1);
}

void main()
{
      int N;
      printf("Masukkan N = ");
      scanf("%d", &N);
      printf("Hasil %d! = %d\n", N, faktorial(N));
}
```

Percobaan 2 : Fungsi menghitung nilai faktorial dengan rekursi tail

```
#include<stdio.h>

int faktorial(int x, int a)
{
      if(x==1)
            return a;
      else
            return faktorial(x-1,x*a);
}

void main()
{
      int N;
      printf("Masukkan N = ");
      scanf("%d", &N);
      printf("Hasil %d! = %d\n", N, faktorial(N,1));
}
```

Percobaan 3 : Fungsi rekursi untuk menampilkan deret fibonanci

```
#include<stdio.h>

int fibo(int x)
{
      if (x<=0 || x<=1)
        return x ;
    else
        return fibo(x-2) + fibo(x-1);
}

void main()
{
      int n;
      printf("Masukkan jumlah deret = ");
```

```
      scanf("%d", &n);
      printf("Deret fibonanci dari %d = ", n);
      for(int i=0;i<n;i++)
        printf("%d ", fibo(i));

}
```

Percobaan 4 : Fungsi rekursi untuk menentukan bilangan prima atau bukan prima

```
#include<stdio.h>
#include<math.h>

int prime(int number, int index)
{
      if(index == 1)
            return 1;
      else if(number % index == 0)
            return 0;
      else
            return prime(number, --index);
}

void main()
{
      int num;
      printf("Masukkan bilangan sampai dengan : ");
      scanf("%d", &num);
      printf("Deret bilangan prima : ");
      for(int i=1;i<=num;i++)
            if(prime(i,(int)sqrt(i)))
                  printf("%d ", i);
}
```

Percobaan 5 : Fungsi rekursi untuk menghitung pangkat

```
#include<stdio.h>

int pangkat(int x, int y)
{
      if(y==0)
            return 1;
      else
            return x * pangkat(x,y-1);
}

void main()
{
      int x, y;
      printf("Bilangan x pangkat y : ");
      scanf ("%d %d", &x, &y);
      printf("%d pangkat %d = %d\n", x, y, pangkat(x,y));
}
```

E. LATIHAN

1. Buatlah program rekursif untuk menghitung segitiga Pascal !

```
F1                    1
F2                  1   1
F3                1   2   1
F4              1   3   3   1
F5            1   4   6   4   1
F6          1   5   10  10  5   1
```

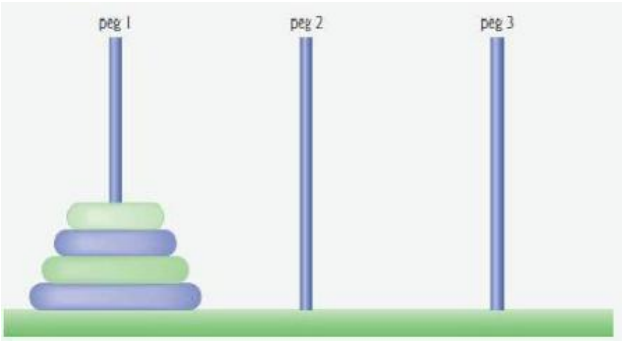
2. Buatlah program secara rekursif, masukkan jumlah N karakter dan cetak dalam semua kombinasi !

```
Jumlah karakter = 3
aaa aab aac aba abb abc aca acb acc baa bab bac bba bbb bbc
bca bcb bcc caa cab cac cba cbb cbc cca ccb ccc BUILD
SUCCESSFUL (total time: 1 second)
```

3. Buat program BinarySearch dengan Rekursif ! (data tentukan sendiri)

```
Data : 2,5,8,10,14,32, 35, 41, 67, 88, 90, 101, 109
Data yang dicari : 10
Data 10 berada pada indek ke – 3
```

4. Buatlah program rekursif untuk memecahkan permasalahan Menara Hanoi !



Program ini merupakan program untuk menampilkan pergerakkan menara hanoi, yang merujuk pada class menaraHanoi. Secara umum algoritma menara hanoi, adalah memindahkan sub menara hanoi dengan n – 1 pin dari n pin ke tiang perantara. Lalu memindahkan pin ke n ke tiang tujuan, lalu memindahkan sub menara hanoi dengan n – 1 pin yang ada di tiang perantara, ke tiang tujuan. StopCase nya jika n == 1.

Jumlah disk : 3
Langkah-langkah nya adalah dengan :
1. Pindahkan disc 1 dari pasak A ke pasak C
2. Pindahkan disc 2 dari pasak A ke pasak B

3. Pindahkan disc 1 dari pasak C ke pasak B
4. Pindahkan disc 3 dari pasak A ke pasak C
5. Pindahkan disc 1 dari pasak B ke pasak A
6. Pindahkan disc 2 dari pasak B ke pasak C
7. Pindahkan disc 1 dari pasak A ke pasak C

F. LAPORAN RESMI

1. Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.
2. Tuliskan kesimpulan dari percobaan dan latihan yang telah anda lakukan.