PRAKTIKUM 7

REKURSIF 1

A. TUJUAN PEMBELAJARAN

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

B. DASAR TEORI

Rekursif adalah suatu proses atau prosedur dari fungsi yang memanggil dirinya sendiri secara berulang-ulang. Karena proses dalam Rekursif ini terjadi secara berulang-ulang maka harus ada kondisi yang membatasi pengulangan persebut, jika tidak maka proses tidak akan pernah berhenti sampai memori yang digunakan untuk menampung proses tersebut tidak dapat menampung lagi/penuh.

Kelebihan Fungsi Rekursif adalah program menjadi lebih singkat. Pada beberapa kasus, lebih mudah menggunakan fungsi rekursif, contohnya: pangkat, factorial, dan fibonacci, dan beberapa proses deret lainnya. Fungsi rekursif lebih efisien dan cepat dibandingkan proses secara iteratif. Kekurangan Fungsi Rekursif adalah memakan memori lebih besar, karena setiap bagian dari dirinya dipanggil, akan membutuhkan sejumlah ruang memori untuk penyimpanan. Rekursif sering kali tidak bisa berhenti sehingga memori akan terpakai habis dan program bisa hang.

Contoh penerapan Rekursif :

1. Faktorial dari bilangan bulat positif n didefinisikan sebagai berikut.

n! = n x (n-1)! Untuk n > 1

0! = 1 Untuk n = 0 atau n = 1

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 masing-masing. Cukup sulit untuk menentukan mana yang paling sederhana, paling jelas, paling efisien dan paling mudah dibanding 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.

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

D. PERCOBAAN

Percobaan 1 : Fungsi rekursif untuk menghitung nilai faktorial

```
public class faktorialDemo {
    public static int faktorial(int x){
        if (x==1)
            return x ;
        else
            return x * faktorial(x-1);
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("N = ");
        int n = input.nextInt();
        System.out.println("Hasil = " + faktorial(n));
    }
}
```

Percobaan 2 : Fungsi rekursi untuk menampilkan deret fibonanci

```
public class Fibonacci {

    public static int fibbon(int x){
        if (x<=0 || x<=1)
            return x ;
        else
            return fibbon(x-2) + fibbon(x-1);
    }
    public static void main(String[] args) {
        int n = 10 ;
        for(int i=0;i<n;i++)
            System.out.println("f" + i + "=" + fibbon(i));
    }
}
```

Percobaan 3 : Fungsi rekursi untuk menentukan bilangan prima atau bukan prima

```
public class prima {
    private static int ambilNilaiRekursif(int number, int index){
        if (index == 1) {
            return 1;
         }
         else if (number % index == 0) {
            return 1 + ambilNilaiRekursif(number, --index);
         } else {
            return 0 + ambilNilaiRekursif(number, --index);
         }
    }

    public static boolean cekBilanganPrima(int num){
        if (num > 1) {
            return (ambilNilaiRekursif(num, num) == 2);
```

```
        }
         else
            return false;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan bilangan nya : ");
        int num = input.nextInt();

        if (cekBilanganPrima(num)) {
            System.out.println("Bilangan Prima");
        } else {
            System.out.println("Bukan Bilangan Prima");
        }
    }
}
```

Percobaan 4 : Fungsi rekursi untuk menampilkan kombinasi 2 karakter

```
public class karakter {
    public static void CharCombination(String a, int n){
         if (n==0)
            System.out.print(a+" ");
         else
             for(int i=97; i<99 ; i++)
                 CharCombination(a+(char)i,n-1);
    }
    public static void main(String[] args) {
       CharCombination("",2);
    }
}
```

Percobaan 5 : Fungsi rekursi untuk menghitung pangkat

```
public class pangkat {
    public static int pangkatrekursif(int x, int y){
        if (y==0)
            return 1 ;
        else return x * pangkatrekursif(x,y-1);
    }
    public static void main(String[] args) {
        System.out.println("Bilangan x pangkat y : ");
        Scanner input = new Scanner(System.in);
        System.out.println("Bilangan x : ");
        int x = input.nextInt() ;

        System.out.println("Bilangan y : ");
        int y = input.nextInt() ;

        System.out.println(x + " dipangkatkan " + y + " = " +
pangkatrekursif(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
```

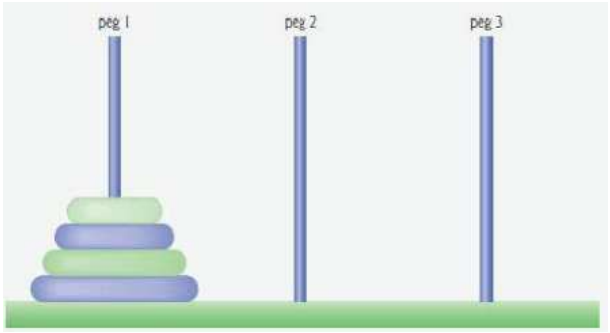
2. Buatlah program secara rekursif, masukkan jumlah N karakter dan cetak dalam semua kombinasi !

```
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 !



Gambar 1. 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

5. Jelaskan proses rekursif untuk program dibawah ini !

```
public static void decToBin(int num)
{
      if (num > 0)
      { decToBin(num / 2);
      System.out.print(num % 2);
      }
}
```

6. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil `test("01101", 4)`!

```
int test(String s, int last) {
      if (last < 0) {
            return 0;
      }
      if (s.charAt(last) == "0") {
            return 2 * test(s, last-1);
      }
      return 1 + 2 * test(s, last-1);
}
```

7. Jelaskan proses rekursif untuk program dibawah ini !

```
boolean search(int[] x, int size, int n) {
      if (size > 0) {
            if (x[size-1] == n) {
                  return true;
            } else {
                  return search(x, size-1, n);
            }
      }
return false;
}
```

8. Jelaskan proses rekursif untuk program dibawah ini !

```
boolean binarySearch(int[] x, int start, int end, int n) {
   if (end < start) return false;
   int mid = (start+end) / 2;
   if (x[mid] == n) {
         return true;
   } else {
         if (x[mid] < n) {
         return search(x, mid+1, end, n);
   } else {
         return search(x, start, mid-1, n);
   }
   }
}
```

9. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil `mystery(2, 25) and mystery(3, 11)`!

```
public static int mystery(int a, int b) {
   if (b == 0)     return 0;
   if (b % 2 == 0) return mystery(a+a, b/2);
   return mystery(a+a, b/2) + a;
}
```

10. Jelaskan proses rekursif untuk program dibawah ini !

```
public static boolean gcdlike(int p, int q) {
   if (q == 0) return (p == 1);
   return gcdlike(q, p % q);
}
```

11. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil `ex234(6)`!

```
public static String ex234(int n) {
    if (n <= 0) return "";
    return ex234(n-3) + n + ex234(n-2) + n;
}
```

12. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil collatz(7)!

```
public static void collatz(int n) {
   System.out.print(n + " ");
   if (n == 1) return;
   if (n % 2 == 0) collatz(n / 2);
   else            collatz(3*n + 1);
}
```

13. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil mystery(0, 8)!

```
public static int mystery(int a, int b) {
    if (a == b) StdOut.println(a);
    else {
       int m1 = (a + b) / 2;
       int m2 = (a + b + 1) / 2;
       mystery(a, m1);
       mystery(m2, b);
    }
}
```

14. Jelaskan proses rekursif untuk program dibawah ini !

```
public static int f(int n) {
   if (n == 0) return 0;
   if (n == 1) return 1;
   if (n == 2) return 1;
   return 2*f(n-2) + f(n-3);
```

15. Jelaskan proses rekursif untuk program dibawah ini dengan memanggil `square(5), cube(5), cube(123)`?

```
public static int square(int n) {
   if (n == 0) return 0;
   return square(n-1) + 2*n - 1;
}
public static int cube(int n) {
   if (n == 0) return 0;
```

```
    return cube(n-1) + 3*(square(n)) - 3*n + 1;
}
```

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.