MODUL 12 REKURSI

1. Deskripsi Singkat

Rekursi adalah sub program yang memanggil dirinya sendiri baik secara langsung maupun tidak langsung. Pada modul 12 akan dibahas tentang rekursi, cara pembuatannya dan juga penggunaannya dalam program.

2. Tujuan Praktikum

Setelah praktikum pada modul 12 ini diharapkan:

1. mahasiswa memahami dan menguasai konsep rekursi
2. mahasiswa teknik memecahkan masalah pemrograman dengan menggunakan rekursi

3. Material Praktikum

Kegiatan pada modul 12 ini memerlukan material berupa program Pascal

4. Kegiatan Praktikum

Pada modul ini kita akan membahas tentang rekursi dimana pada bab sebelumnya kita telah membahas tentang sub program. Ingat bahwa suatu fungsi atau prosedur dapat dipanggil dari fungsi atau prosedur lainnya.

Rekursi terjadi ketika **sub program memanggil dirinya sendiri**. Rekursi banyak disandingkan dan dibandingkan dengan iterasi/perulangan untuk memudahkan pembelajaran dan membandingkan performanya.

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.

Sub Program Rekursif akan memanggil dirinya sendiri selama kondisi pemanggilan dipenuhi. Dengan melihat sifat sub program rekursif tersebut 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 penyesuaian parameter

else *lakukan instruksi* yang akan dieksekusi bila kondisi khusus dipenuhi

atau

if *kondisi khusus* terpenuhi

then *lakukan instruksi* yang akan dieksekusi bila kondisi khusus dipenuhi

else *panggil diri-sendiri* dengan penyesuaian parameter

A. Fungsi Rekursif

Berikut adalah contoh sebuah fungsi dengan parameter `x` untuk mengembalikan nilai jumlah dari `1` sampai `x`. Berikut adalah versi iterasinya:

```
Function Sum(x:integer):integer;
Var i, result: integer;
Begin
   Result:=0;
   For i:=1 to x do result:=result+i;
   Sum:=result;
End;
```

Ketika misalnya dipanggil `sum(5)` maka jalannya fungsi adalah sebagai berikut:

1. `x` akan berisi `5`;
2. `Result` bernilai awal `0`;
3. `For i:=1 to 5 do result:=result+1` akan menjumlahkan `1+2+3+4+5 = 15`.
4. Nilai `15` akan dikembalikan ke pemanggil fungsi tersebut.

Dan berikut ini adalah versi rekursinya. Simpan ulang dengan nama **praktikum12A.pas**

```
Function Sum(x: integer):integer;
Begin
   If x = 1 then Sum := 1
   Else Sum := x + Sum(x-1);
End;
```

Jalankan program **praktikum12A.pas**. Ketika misalnya dipanggil `sum(5)` maka jalannya fungsi adalah sebagai berikut:

1. `Sum(5)` menjadi `5 + sum(4)`
2. `Sum(4)` menjadi `4 + sum(3)`

3. `Sum(3)` menjadi `3 + sum(2)`
4. `Sum(2)` menjadi `2 + sum(1)`
5. `Sum(1)` hasilnya adalah `1`
6. `Sum(2)` hasilnya menjadi `2 + 1 = 3`
7. `Sum(3)` hasilnya menjadi `3 + 3 = 6`
8. `Sum(4)` hasilnya menjadi `4 + 6 = 10`
9. `Sum(5)` hasilnya menjadi `5 + 10 = 15`

Fungsi `Sum(5)` akan menghasilkan `15`.

Contoh lainnya adalah fungsi yang memanfaatkan iterasi untuk mengembalikan nilai faktorial dari sebuah bilangan.

```
Program faktorial_bilangan;
Uses crt;
Function Faktorial(x:integer):integer;
Var i, result:integer;
Begin
   Result:=1;
   For i:=x downto 1 do result:=result * i;
   Faktorial:=result;
End;
Var i: integer;
Begin
   Clrscr;
   i := Faktorial(5);
Writeln(i);
   Readln;
End.
```

Jalannya program dimulai dari baris pertama di program utama. Berikut adalah proses jalannya program:

1. `Clrscr` akan membersihkan layar dari kotoran
2. `i:=Faktorial(5))`
   Baris ini akan meng-*assign* nilai dari `Faktorial(5)` ke variabel `i`. Berapakah nilai `Faktorial(5)`? Di sini fungsi `Faktorial` akan dipanggil dari program

utama dengan argumen `5` yang akan ditransfer ke parameter `x` dalam fungsi `Faktorial`.
Maka `for i:=x downto 1` pada fungsi tersebut akan menjadi `for i:=5 downto 1`. Perulangan `result := result * i` akan diulang `5` kali sehingga hasilnya menjadi `5*4*3*2*1 = 120`.
Kembali ke program utama dengan mengembalikan nilai `120`, `i` akan berisi `120`.

3. `Writeln(i)` akan menuliskan `120` di layar, kemudian kursor turun 1 baris.
4. `Readln` akan membuat kursor berkedip sebelum program diakhiri.

Berikut ini adalah versi rekursi dari fungsi tersebut:

```
Function Faktorial(x:integer):integer;
Begin
   If x := 1 then faktorial := 1
Else Faktorial := x * Faktorial(x – 1);
End;
```

Berikut adalah jalannya fungsi ketika dipanggil `Faktorial(5)`:

1. `Faktorial(5)` menjadi `5 * Faktorial(4)`
2. `Faktorial(4)` menjadi `4 * Faktorial(3)`
3. `Faktorial(3)` menjadi `3 * Faktorial(2)`
4. `Faktorial(2)` menjadi `2 * Faktorial(1)`
5. `Faktorial(1)` hasilnya adalah `1`
6. `Faktorial(2)` hasilnya menjadi `2 * 1 = 2`
7. `Faktorial(3)` hasilnya menjadi `3 * 2 = 6`
8. `Faktorial(4)` hasilnya menjadi `4 * 6 = 24`
9. `Faktorial(5)` hasilnya menjadi `5 * 24 = 120`

Maka `Faktorial(5)` akan bernilai `120`. Fungsi `Faktorial` dipanggil sebanyak `5` kali.

Simpan ulang program kalkulator KATABAKU pada penugasan modul 9 dengan nama **praktikum12B.pas**. Tambahkan fungsi faktorial yang dihitung dengan menggunakan rekursi., sehingga keluaran program menjadi seperti berikut ini.

```
Selamat datang di Kalkulator Sederhana
Silahkan pilih menu berikut:
  1. Penjumlahan
  2. Pengurangan
  3. Perkalian
  4. Pembagian
```

```
  5. Faktorial
  6. Keluar
Pilihan Anda:
```

B. Prosedur Rekursif

Selain Function, Prosedur juga bisa bersifat rekursif, perhatikan contoh berikut ini. Simpan program dengan nama **praktikum12C.pas.**

```
PROCEDURE TULIS_1(banyak : integer; kata : string);
begin
if banyak > 1 then TULIS_1(banyak-1,kata);
writeln(kata, banyak);
end;
```

misal jika dipanggil `TULIS_1(5, 'Cetakan ke ')` maka outputnya adalah:

```
Cetakan ke 1
Cetakan ke 2
Cetakan ke 3
Cetakan ke 4
Cetakan ke 5
```

Mengapa bisa demikian? Begini kira-kira jalannya prosedur:

1. `Tulis_1(5, 'Cetakan ke ')` pada bagian `if` akan bernilai `TRUE` sehingga memanggil `Tulis_1(4, 'Cetakan ke ')`
2. `Tulis_1(4, 'Cetakan ke ')` pada bagian `if` akan bernilai `TRUE` sehingga memanggil `Tulis_1(3, 'Cetakan ke ')`
3. `Tulis_1(3, 'Cetakan ke ')` pada bagian `if` akan bernilai `TRUE` sehingga memanggil `Tulis_1(2, 'Cetakan ke ')`
4. `Tulis_1(2, 'Cetakan ke ')` pada bagian `if` akan bernilai `TRUE` sehingga memanggil `Tulis_1(1, 'Cetakan ke ')`
5. `Tulis_1(1, 'Cetakan ke ')` pada bagian `if` akan bernilai `FALSE` sehingga lanjut ke statemen selanjutnya yaitu `writeln(kata, banyak)` yang akan menuliskan `Cetakan ke 1` di layar. Prosedur `Tulis_1(1, 'Cetakan ke ')` selesai dijalankan dan kembali ke pemanggilnya yaitu `Tulis_1(2, 'Cetakan ke ')`
6. `Tulis_1(2, 'Cetakan ke ')` sudah selesai menjalankan bagian `if`, kemudian masuk ke `writeln(kata, banyak)` yang akan menuliskan `Cetakan ke 2` di layar. Prosedur `Tulis_1(2, 'Cetakan ke ')` selesai

dijalankan dan kembali ke pemanggilnya yaitu `Tulis_1(3, 'Cetakan ke ')`

7. `Tulis_1(3, 'Cetakan ke ')` sudah selesai menjalankan bagian `if`, kemudian masuk ke `writeln(kata, banyak)` yang akan menuliskan `Cetakan ke 3` di layar. Prosedur `Tulis_1(3, 'Cetakan ke ')` selesai dijalankan dan kembali ke pemanggilnya yaitu `Tulis_1(4, 'Cetakan ke ')`
8. `Tulis_1(4, 'Cetakan ke ')` sudah selesai menjalankan bagian `if`, kemudian masuk ke `writeln(kata, banyak)` yang akan menuliskan `Cetakan ke 4` di layar. Prosedur `Tulis_1(4, 'Cetakan ke ')` selesai dijalankan dan kembali ke pemanggilnya yaitu `Tulis_1(5, 'Cetakan ke ')`
9. `Tulis_1(5, 'Cetakan ke ')` sudah selesai menjalankan bagian `if`, kemudian masuk ke `writeln(kata, banyak)` yang akan menuliskan `Cetakan ke 5` di layar. Prosedur `Tulis_1(5, 'Cetakan ke ')` selesai dijalankan.
10. Prosedur sudah selesai dijalankan dan menghasilkan output seperti di atas.

Modifikasi program pada **praktikum12C.**pas sehingga ouput yang dihasilkan menjadi seperti berikut ini. Misal, dipanggil `TULIS_1(5, 'Cetakan ke ')`, ouput yang dihasilkan:

```
Cetakan ke 5
Cetakan ke 4
Cetakan ke 3
Cetakan ke 2
Cetakan ke 1
```

C. Rekursi Tak Hingga

Sama dengan iterasi yang mungkin akan terjadi tak hingga bila kondisi pada `while` atau `until` tidak pernah terpenuhi, rekursi tak hingga juga dapat terjadi apabila jalannya pemanggilan sub program tidak pernah mencapai `true` pada kondisi `if`. Sebagai contoh, Apa yang terjadi bila fungsi pada **praktikum12A.pas** dipanggil dengan parameter bilangan negative.

```
Function Sum(x: integer):integer;
Begin
   If x=1 then Sum:=1
   Else Sum:=x+Sum(x-1);
End;
```

Misalnya, fungsi rekursi dipanggil dengan `sum(-1)` maka jalannya kira-kira seperti berikut:

a. `Sum(-1)` akan menjadi `-1 + sum (-2)`
b. `Sum(-2)` akan menjadi `-2 + sum(-3)`
c. `Sum(-3)` akan menjadi `-3 + sum(-4)`
d. `Sum(-4)` akan menjadi `-4 + sum(-5)`
e. `Sum(-5)` akan menjadi `-5 + sum(-6)`
f. `Sum(-6)` akan menjadi `-6 + sum(-7)`
g. `Sum(-7)` akan menjadi `-7 + sum(-8)`
h. ....
i. ....

Sampai kapan? Sampai selama-lamanya. Rekursi tak hingga akan terjadi jika kondisi pada if (disebut juga sebagai **base case** atau **special case**) tidak pernah terpenuhi. Modifikasi fungsi pada **praktikum12A.pas** agar fungsi tersebut juga dapat berjalan untuk parameter 0 dan juga parameter bilangan negatif.

D. Deret Fibonacci

Sebuah fungsi rekursif mungkin dapat kita ubah menjadi prosedur rekursif. Berikut ini adalah contoh fungsi rekursif untuk mengembalikan bilangan fibbonacci suku ke n. **Barisan** Bilangan **Fibonacci** adalah **barisan** yang nilai sukunya sama dengan jumlah dua suku di depannya. **Barisan**: 1, 1, 2, 3, 5, 8, 13, 21, 34, ... Simpan program untuk menghasilkan deret Fibonacci berikut dengan nama **praktikum12D.pas**. Jalankan program dan periksalah apakah output yang dihasilkan sudah sesuai.

```
program fibo_using_rekursif;
var
  x,i: integer;
function fib(n:integer):integer;
begin
 if(n=1) then fib:=1
 else if (n=2) then fib:=1
 else fib:=fib(n-1)+fib(n-2);
end;
begin
 writeln('deret fibonacci');
 write('input value : ');
 readln(x);
```

```
 writeln;
 for i := 1 to x do write(fib(i),'  ');
 readln;
end.
```

Modifikasi fungsi pada program sehingga menjadi prosedur seperti berikut ini. Jalankan ulang program. Periksalah apakah output yang dihasilkan sudah benar? Dimanakah letak perbedaan jalannya program dibandingkan dengan menggunakan fungsi yang sebelumnya?

```
procedure pib(n:integer; var hsl :integer);
var f1, f2: integer;
begin
      if(n=1) or (n=2) then hsl:=1
      else
        begin
                pib(n-1, f1);
                pib(n-2, f2);
                hsl:= f1 + f2;
        end;
end;
var x,i: integer; hsl:integer;
begin
      writeln('Barisan Bilangan Fibonacci');
        write('Jumlah bilangan Fibonnaci yang ingin
ditampilkan : ');
      readln(x); writeln;
        for i := 1 to x do
        begin
                pib(i,hsl);
                write(hsl,'  ');
        end;
        readln;
End.
```

5. Responsi

Kerjakan sesuai dengan yang dijelaskan pada bagian Kegiatan Praktikum. Simpan tangkapan layar hasil pekerjaan Anda untuk masing-masing kegiatan praktikum dalam file docx. Simpan ulang file tersebut dalam format pdf, dan beri nama dengan format *<<kelas>>_modul12_<<nim>>.pdf*, contoh: **1KS1_**modul12_192191234.pdf. Kumpulkan file tersebut sebagai responsi melalui Google Classroom