LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2

MODUL V

REKURSIF

Oleh:

FADHEL YUSSIE RAMADHAN

2311102322

S1IF-11-02

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pengantar Rekursif

Hal ini tidak menutup kemungkinan bahwa subprogram yang dipanggil adalah dirinya sendiri. Dalam pemrograman teknik ini dikenal dengan istilah rekursif. Rekursif secara sederhana dapat diartikan sebagai cara menyelesaikan suatu masalah dengan cara menyelesaikan sub-masalah yang identik dari masalah utama. Sebagai contoh perhatikan prosedur cetak berikut ini!

.

	Notasi Algoritma	Not Bah
1 2 3 4 5	Procedure cetak (In x: integer) Algoritma output (x) cetak (x+1) endprocedure	Fun int){ fmt ceta

Apabila diperhatikan subprogram cetak() di atas, terlihat pada baris ke-4 terdapat pemanggilan subprogram cetak() kembali. Misalnya apabila kita eksekusi perintah cetak(5) maka akan menampilkan angka 5 6 7 8 9...dst tanpa henti. Artinya setiap pemanggilan subprogram cetak() nilai x akan selalu P bertambah 1 (Increment by one) secara terus menerus tanpa henti.

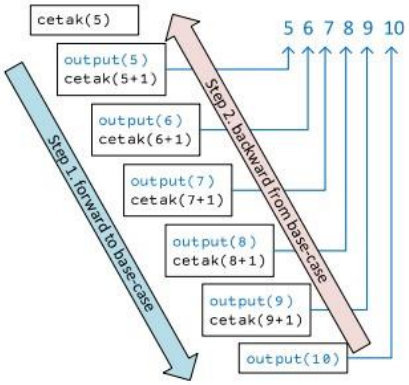
```
package
main
import "fmt"
func main(){
        cetak
(5)
}
  func cetak
(x int){
fmt.Println(x)
} cetak (x+1)
```

Oleh karena itu bisanya ditambahkan struktur kontrol percabangan (If-then) untuk menghentikan proses rekursif ini. Kondisi ini disebut juga dengan base-case, artinya apabila kondisi base-case bernilai true maka proses rekursif akan berhenti. Sebagai contoh misalnya base case adalah ketika x bernilal 10 atau x == 10, maka tidak perlu dilakukan rekursif.

```
procedure cetak (in x:integer) algoritma       if x ==10 then
output(x) else
     output(x)       cetak(x+1)   endif endprocedure
```

Apabila diperhatikan pada baris ke-3 di Program di atas, kita telah menambahkan base-case seperti penjelasan sebelumnya. Selanjutnya pada bagian aksi dari else di baris ke-6 dan ke-7 kita namakan recursive-case atau kasus pemanggilan dirinya sendiri tersebut terjadi. Kondisi dari recursive-case ini adalah negasi dari kondisi base-case atau ketika nilal x != 10.

```
package main  import "fmt"
func main(){      cetak (5)
} func cetak (x int){       if x== 10 {       fmt.Println(x)
}else{      fmt.Println(x) cetak (x+1)
}
}
```



Gambar ini memperlihatkan saat subprogram dipanggil secara rekursif, maka subprogram akan terus melakukan pemanggilan (forward) hingga berhenti pada saat kondisi base-case terpenuhi atau true. Setelah itu akan terjadi proses backward atau kembali ke subprogram yang sebelumnya. Artinya setelah semua instruksi cetak(10) selesai dieksekusi, maka program akan kembali ke cetak(9) yang memanggil cetak (10) tersebut. Begitu seterusnya hingga kembali ke cetak(5).

Catatan:

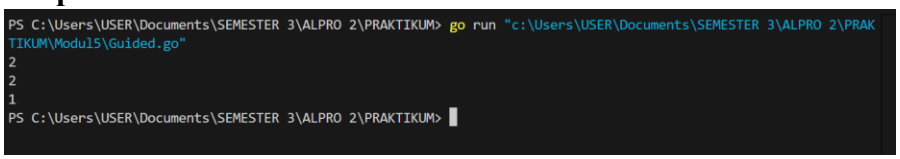
- Teknik rekursif ini merupakan salah satu alternatif untuk mengganti struktur kontrol perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedure).
- Untuk menghentikan proses rekursif digunakan percabangan (if-then).
- Base-case adalah kondisi proses rekursif berhenti. Base-case merupakan hal terpenting dan
- pertama yang harus diketahui ketika akan membuat program rekursif. Mustahil membuat program rekursif tanpa mengetahul base-case terlebih dahulu.
- Recursive-case adalah kondisi dimana proses pemanggilan dirinya sendiri dilakukan. Kondisi recursive-case adalah komplemen atau negasi dari base-case.
- Setiap algoritma rekursif selalu memiliki padanan dalam bentuk algoritma Interatif.

II. GUIDED

1. Source Code

```
package main

import "fmt"

func main() {
    var n int
    fmt.Scan(&n) //membaca input pengguna
    baris(n)     //memanggil fungsi rekursif 'baris'

}

func baris(bilangan int) {
    if bilangan == 1 { //base case: jika bilangan sama dengan
1
        fmt.Println(1) //cetak angka 1
    } else { //jika bilangan lebih besar daripada 1
        fmt.Println(bilangan) //cetak bilangan saat ini
        baris(bilangan - 1)   //panggil fungsi 'baris' dengan
bilangan -1
    }
}
```

Output



Keterangan menghitung dan

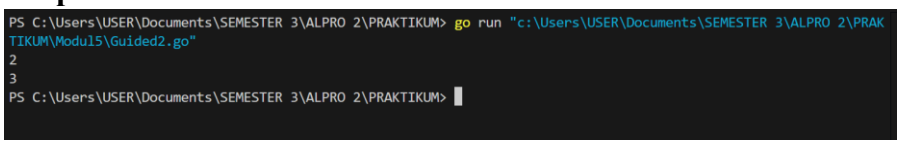
menampilkan permutasi.

Fungsi factorial menghitung faktorial suatu angka secara iteratif. Prosedur permutasi menghitung permutasi P(n,r) menggunakan rumus n! / (n-r)! dan menampilkan hasilnya. Fungsi main memanggil permutasi dengan nilai n=5 dan r=3 sebagai contoh. Program ini mendemonstrasikan penggunaan fungsi dan prosedur untuk menghitung permutasi dalam matematika kombinatorik.

2. Source Code

```go
package main

import "fmt"

func main() {
    var n int
    fmt.Scan(&n)
    fmt.Println(penjumlahan(n))

}

func penjumlahan(n int) int {
    if n == 1 {
        return 1
    } else {
        return n + penjumlahan(n - 1)
    }
}
```

Output



Keterangan

III. UNGUIDED

1. Source Code

```
package main

import "fmt"

func main() {
    var n int
    fmt.Print("masukkan nilai: ")
    fmt.Scan(&n)
    for i := 0; i <= n; i++ {
        fmt.Print(fibonacci(i), " ")
    }
}

func fibonacci(n int) int {
        if n == 0 {
            return 0
        } else if n == 1 {
            return 1
        } else {
    return fibonacci(n-1) + fibonacci(n-2)
        }
}
```

Output

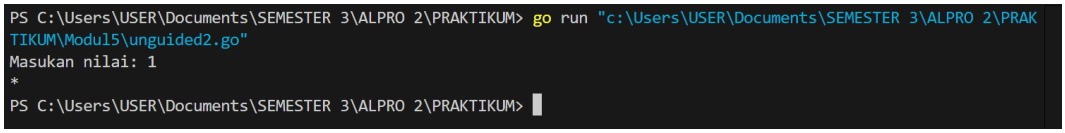
```
PS C:\Users\USER\Documents\SEMESTER 3\ALPRO 2\PRAKTIKUM> go run "c:\Users\USER\Documents\SEMESTER 3\ALPRO 2\PRAK
TIKUM\Modul5\Unguided1.go"
masukkan nilai: 1
0 1
PS C:\Users\USER\Documents\SEMESTER 3\ALPRO 2\PRAKTIKUM>
```

Keterangan

2. Source Code

```
package main

import "fmt"

func main () {
    var a int
    fmt.Print ("Masukan nilai: ")
    fmt.Scan(&a)
    pola(a, 1)

}

func bintang (B int) {
    if B == 0 {
        return
    }
    fmt.Print ("* ")
    bintang (B - 1)
}

func pola (n, c int) {
    if c > n {
        return
    }
    bintang (c)
    fmt.Println()
    pola (n, c+1)
}
```

Output



Keterangan

3. Source Code

```
package main

import (
    "fmt"
)

func main() {
    var N int

    fmt.Print("masukkan angka: ")
    fmt.Scan(&N)


    fmt.Print("Faktor dari ", N, ": ")
    findFactors(N, 1)
    fmt.Println()
}

func findFactors(n, i int) {
    if i > n {
        return
    }
    if n%i == 0 {
        fmt.Print(i, " ")
    }
    findFactors(n, i+1)
}
```

Output

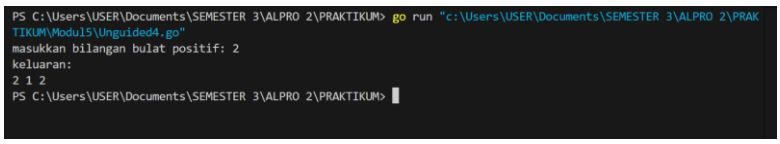
```
PS C:\Users\USER\Documents\SEMESTER 3\ALPRO 2\PRAKTIKUM> go run "c:\Users\USER\Documents\SEMESTER 3\ALPRO 2\PRAK
TIKUM\Modul5\Unguided3.go"
masukkan angka: 1
Faktor dari 1: 1
PS C:\Users\USER\Documents\SEMESTER 3\ALPRO 2\PRAKTIKUM>
```

Keterangan

4. Source Code

```
package main

import "fmt"

func main() {
    var N int
    fmt.Print("masukkan bilangan bulat positif: ")
    fmt.Scanln(&N)

    fmt.Println("keluaran:")
    printSequence(N)
    fmt.Println()
}

func printSequence(n int) {
    if n == 1 {
        fmt.Print(n, " ")
        return
    }
    fmt.Print(n, " ")

    printSequence(n - 1)
    fmt.Print(n, " ")
}
```

Output

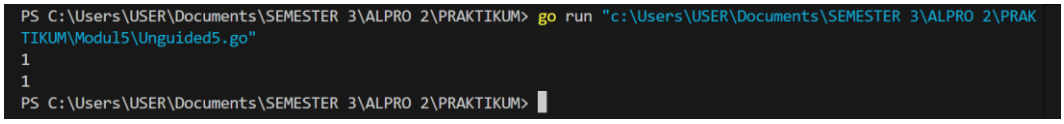


Keterangan

5. Source Code

```
package main

import "fmt"

func main() {
    var num int
    fmt.Scanln(&num)
    barisGanjil(num)
}

func barisGanjil(num int) {
    if num <= 1 {
        fmt.Print("1 ")
    } else if num%2 != 0 {
        barisGanjil(num - 1)
        fmt.Printf("%v ", num)
    } else {
        barisGanjil(num - 1)
    }
}
```

Output

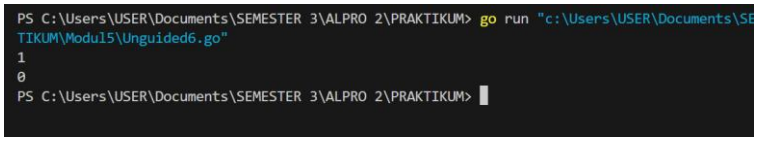


Keterangan

1. Source Code

```
package main

import "fmt"

func main() {
    var num, power_of int
    fmt.Scanln(&num, &power_of)
    fmt.Println(power(num, power_of))
}

func power(num int, power_of int) int {
    if power_of <= 1 {
        return num * power_of
    } else {
        return num * power(num, power_of-1)
    }
}
```

Output



Keterangan