

**LAPORAN PRAKTIKUM  
ALGORITME DAN PEMEROGRAMAN**

**MODUL 5  
REKURSIF**



Oleh:

ERVAN HAPIZ

2311102206

IF – 11- 02

**S1 TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## I. DASAR TEORI

### 6.1 Pengantar Rekursif

Pada modul-modul sebelumnya sudah dijelaskan bahwa suatu subprogram baik fungsi atau prosedur bisa memanggil subprogram lainnya. 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 berikut ini!

	Notasi Algoritma	Notasi dalam bahasa GO
1	procedure cetak(in x:integer)	func cetak(x int){
2	algoritma	fmt.Println(x)
3	output(x)	cetak(x+1 )
4	cetak(x+1 )	}
5	endprocedure	

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 bertambah 1 (increment by one) secara terus menerus tanpa henti.

```
1 package main
2 import "fmt"
3 func main(){
4     cetak(5)
5 }
6 func cetak(x int){
7     fmt.Println(x)
8     cetak(x+1 )
9 }
```

```
D:\DEV\DEMO>go build
contoh.go
D:\DEV\DEMO>contoh.exe
5
6
7
8
9
10
11
12
13
```

Oleh karena itu bisanya ditambahkan struktur Control 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 bernilai 10 atau  $x == 10$ , maka tidak perlu dilakukan rekursif.

```
1 procedure cetak(in x:integer)
2   algoritma
3     if x ==10 then
4       output (x)
5     else
6       output (x)
7       cetak(x+1 )
8     endif
9   endprocedure
```

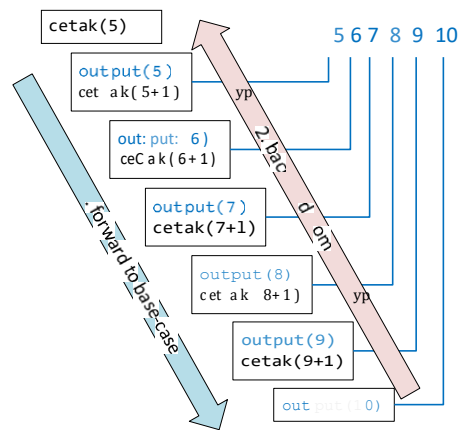
Apabila diperhatikan pada baris ke-3 di Program di atas, kita telah menambahkan base-case seperti penjelasan sebelumnya. Selanjutnya pada bagian aksi dari else di baris ke-6 dan ke-7 kita namakan recursive-case atau kasus pemanggilan dirinya sendiri tersebut terjadi. Kondisi dari **recursive-case** ini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

```
1 package main
2 import "fmt"
3 func main(){
4   cetak(5)
5 }
6 func cetak(x int){
7   if x == 10 {
8     fmt.Println(x)
9   }else(
10     fmt . Println(x)
11     cetak(x+1 )
12   }
13 }
```

```
D:\DEV\DEMO>go build
contoh.go
D:\DEV\DEMO>contoh.exe
5
6
7
8
9
10
```

Apabila program di atas ini dijalankan maka akan tampil angka 5 6 7 8 9 10. Terlihat bahwa proses rekursif berhasil dihentikan ketika  $x == 10$ .

---



Gambar 1. //ustrasi proses forward dan backward pada saat rekursi/.

Pada Gambar 2 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).

Perhatikan modifikasi program di atas dengan menukar posisi baris 10 dan 11, mengakibatkan ketika program dijalankan maka akan menampilkan 10 9 8 7 6 5. Kenapa bisa demikian? Pahami proses backward pada Gambar 2

```

1 package main
2 import "fmt"
3 func main(){
4     cetak(5)
5 }
6 func cetak(x int){
7     if x == 10 {
8         fmt.Println(x)
9     }else{
10        cetak(x+1)
11        fmt.Println(x)
12    }
13 }

```

```

D:\DEV\DEMO>go build
contoh.go
D:\DEV\DEMO>contoh.exe
10
9
8
7
6

```

Catatan:

- Teknik rekursif ini merupakan salah satu alternatif untuk mengganti struktur kontrol perulangan dengan memanfaatkan subprogram (bisa fungsi ataupun prosedur).
- Untuk menghentikan proses rekursif digunakan percabangan (if-then).
- Base-case adalah kondisi proses rekursif berhenti. Base-case merupakan hal terpenting dan pedoman yang harus diketahui oleh programmer untuk membuat program rekursif tanpa mengetahui 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 iteratif.

## 6.2 Komponen Rekursif

Algoritma rekursif terdiri dari dua komponen utama:

- Base-case (Basis), yaitu bagian untuk menghentikan proses rekursif dan menjadi komponen terpenting di dalam sebuah rekursif.
- Recursive-case, yaitu bagian pemanggilan subprogramnya.

## II. GUIDED

### 1. Guided 1

#### Source Code

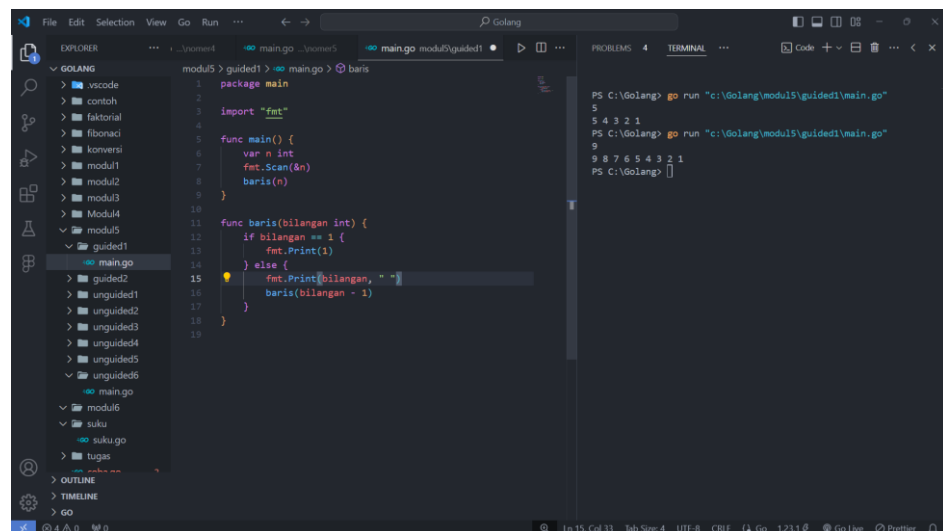
```
Package main

import "fmt"

func main() {
    var n int
    fmt.Scan(&n)
    baris(n)
}

func baris(bilangan int) {
    if bilangan == 1 {
        fmt.Print(1)
    } else {
        fmt.Print(bilangan, " ")
        baris(bilangan - 1)
    }
}
```

#### Screenshot



#### Deskripsi

Program ini adalah program untuk mengurutkan bilangan dengan rekursif sesuai dengan input user. Pada func baris() terdapat if bilangan ==1 (kondisi ini sebagai base case dari rekursif). Yang akan menampilkan 1. Jika kondisi if tidak terpenuhi maka program akan menampilkan nilai dari variable bilangan. Kemudian program akan

memanggil Kembali func baris() dengan parameter bilangan – 1. Kondisi ini berlangsung hingga kondisi base case terpenuhi dan program akan berhenti. Output berupa barisan bilangan dari n hingga ke 1

## 2. Guided 2

### Source Code

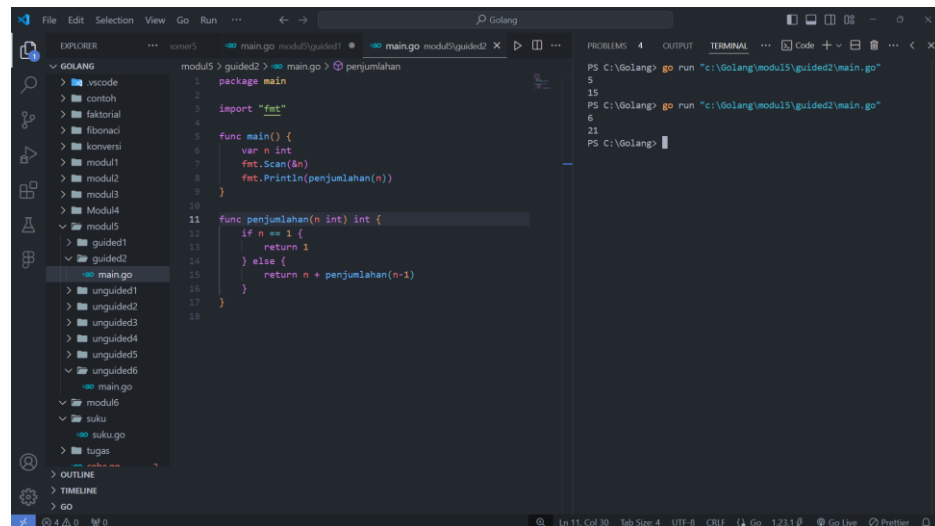
```
package main

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)
    }
}
```

### Screenshot



### Deskripsi

Program ini adalah program untuk menjumlahkan bilangan urut dari n hingga 1. Dalam func penjumlahan () dengan parameter n int. Dalam fungsi terdapa if sebagai base case untuk rekursif dengan kondisi n == 1 maka program akan membalikan nilai 1. Ketika tidak terpenuhi maka func akan membalikan nilai n + penjumlahan (n-1). Yang mana

penjumlahan(n-1) memanggil Kembali func penjumlahan dengan parameter n – 1. Kemudian akan berlangsung hingga kondisi mencapai base case dan fungsi akan terhenti. Dan program akan menampilkan hasil akhir dari penjumlahan berurut. Pada func main terdapat deklarasi variable n dengan tipe data int. kemudian program akan menerima input dari user. Pemanggilan func penjumlahan() kemudian program akan menampilkan nilai dari func penjumlahan.

### III. UNGUIDED

#### 1. Unguided 1

##### Source Code

```
package main

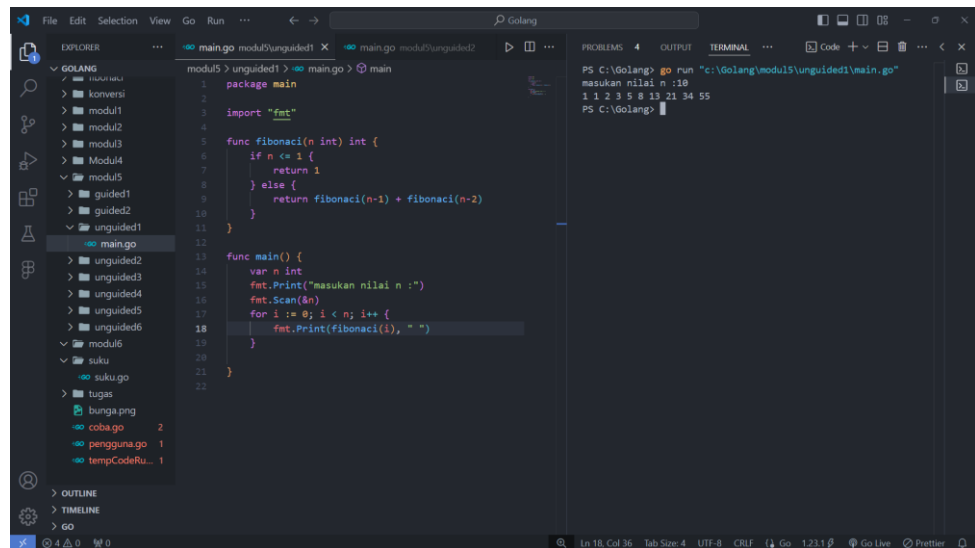
import "fmt"

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

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

##### Screenshot





## Deskripsi

Program ini adalah program untuk menampilkan deret fibonacci. Dalam program terdapat fungsi rekursif yaitu func fibonacci() dengan parameter n int. Dalam fungsi terdapat if sebagai kondisi base case dari rekursif yaitu  $n \leq 1$ , jika kondisi terpenuhi maka program akan membalikan nilai 1. Terdapat else jika kondisi if tidak terpenuhi, maka program akan mengembalikan nilai dengan memanggil fungsi fibonacci(n-1) + fibonacci(n-2) ini artinya fungsi akan menjumlahkan nilai dari func fibonacci n -1 dengan func fibonacci n-2. Missal  $n = 2$  maka program akan mengecek apakah n masuk ke kondisi base case atau tidak jika tidak maka program akan menjumlahkan nilai dari fibonacci 1 + fibonacci 0. Kemudian pada func main terdapat deklarasi variable n . kemudain input dari user untuk nilai n. dan untuk menampilkan deret fibonacci digunakan perulangan for dari i 0 sampai n , kemudian pemanggilan fungsi fibonacci.

## 2. Unguided 2

### Source Code

```

package main

import "fmt"

func segitiga(i, n int) {
    if i > n {
        return
    }
    for j := 0; j < i; j++ {
        fmt.Print("*")
    }
    fmt.Println()
}

```

```

        segitiga(i+1, n)
    }

    func main() {
        var n int
        fmt.Print("masukan tinggi segitiga : ")
        fmt.Scan(&n)
        segitiga(1, n)
    }

```

## Screenshot

The screenshot shows a Go IDE with a file explorer on the left, a code editor in the center, and a terminal on the right. The code editor displays the following Go code:

```

package main

import "fmt"

func segitiga(i, n int) {
    if i > n {
        return
    }
    for j := 0; j < i; j++ {
        fmt.Print("*")
    }
    fmt.Println()
    segitiga(i+1, n)
}

func main() {
    var n int
    fmt.Print("masukan tinggi segitiga : ")
    fmt.Scan(&n)
    segitiga(1, n)
}

```

The terminal on the right shows the execution of the program with the following output:

```

PS C:\Golang> go run "c:\Golang\modul5\unguided2\main.go"
masukan tinggi segitiga : 6
*
**
***
****
*****
PS C:\Golang> go run "c:\Golang\modul5\unguided2\main.go"
masukan tinggi segitiga : 3
*
**
***
PS C:\Golang> go run "c:\Golang\modul5\unguided2\main.go"
masukan tinggi segitiga : 1
*
PS C:\Golang>

```

## Deskripsi

Program ini adalah program untuk membuat sebuah segitiga siku-siku. Pada program untuk membuat segitiga menggunakan rekursif, terdapat func segitiga dengan dua parameter yaitu i (untuk baris) dan n (untuk tinggi). Dalam func segitiga terdapat if dengan kondisi  $i > n$  sebagai base case untuk menghentikan rekursif. Kemudian ketika kondisi if tidak terpenuhi maka terdapat else yang didalamnya ada for j dari 0 hingga i maka akan mencetak Bintang sebanyak i. ketika perulangan selesai terdapat `fmt.Println()` yang berfungsi untuk membuat baris baru setelah mencetak Bintang. Kemudian terdapat rekursif dengan memanggil func `segitiga(i + 1, n)` dengan i increment untuk mencetak baris baru. Kemudian pada func `main()` terdapat deklarasi variable n, user akan menginput nilai n. kemudian pemanggilan fungsi segitiga dengan nilai n dan  $i = 1$ .

### 3. Unguided 3

#### Source Code

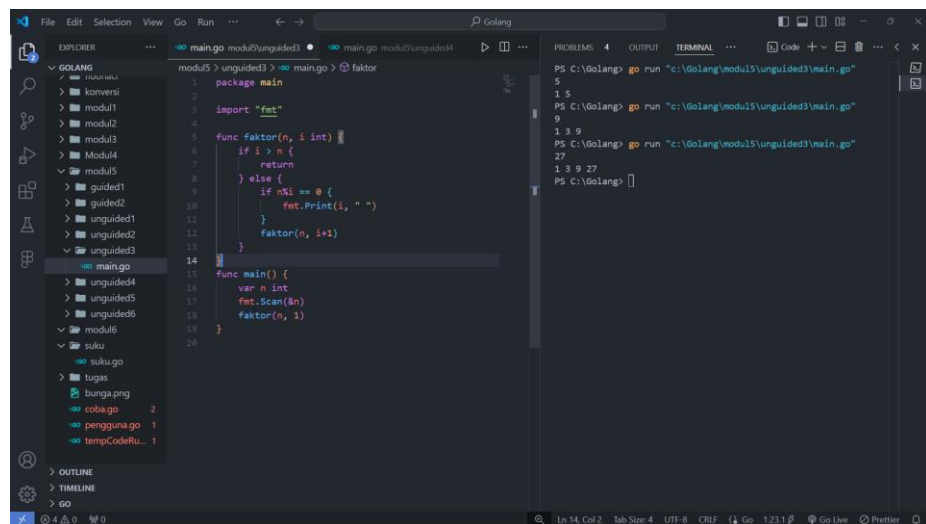
```
package main

import "fmt"

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

func main() {
    var n int
    fmt.Scan(&n)
    faktor(n, 1)
}
```

#### Screenshot



#### Deskripsi

Program ini adalah program untuk mencari factor dari bilangan yang diinput. Terdapat func faktor() serbagai fungsi rekursif dengan parameter n sebagai bilangan yang akan dicari faktornya, dan i sebagai bilangan atau factor dari n. dalam fungsi terdapat if dengan kondisi  $i > n$  sebagai base case dari rekursif. Kemudian  $n \% i == 0$  akan mencetak nilai dari

i jika kondisi benar, kondisi ini digunakan untuk mengecek apakah I factor dari n. kemudian pemanggilan fungsi factor (n, i+1) sebagai pemanggilan rekursif dengan I di increment. Pada func main() terdapat deklarasi variable n , program akan menerima input untuk nilai n. kemudian pemanggilan func factor (n, 1) i=1 untuk nilai awal.

#### 4. Unguided 4

##### Source Code

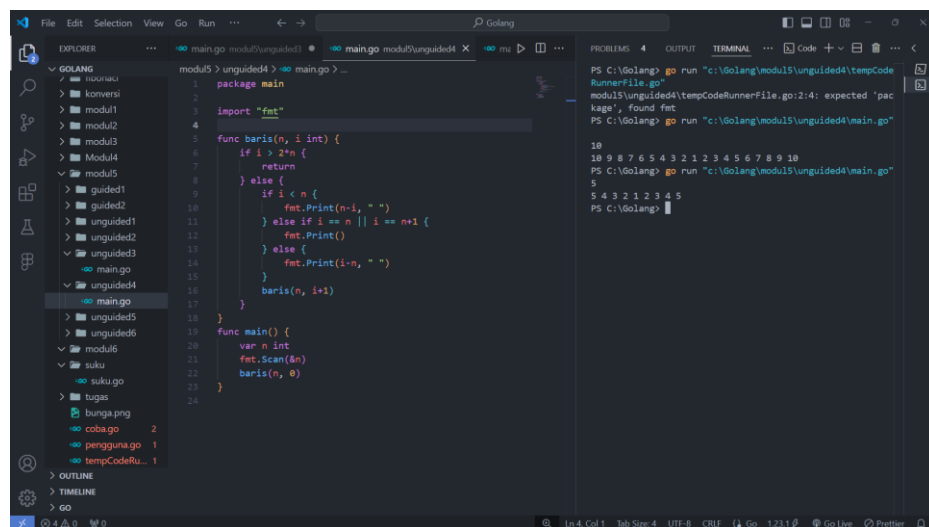
```
package main

import "fmt"

func baris(n, i int) {
    if i > 2*n {
        return
    } else {
        if i < n {
            fmt.Print(n-i, " ")
        } else if i == n || i == n+1 {
            fmt.Print()
        } else {
            fmt.Print(i-n, " ")
        }
        baris(n, i+1)
    }
}

func main() {
    var n int
    fmt.Scan(&n)
    baris(n, 0)
}
```

##### Screenshot



### Deskripsi

Program ini adalah program untuk menampilkan suatu baris bilang dari n hingga 1 dan balik ke n. Pada program terdapat func baris () dengan parameter n ( sebagai batas angka) dan i . pada fungsi terdapat if dengan kondisi  $i > 2 * n$  sebagai base case dari rekursif. Kemudian ketika kondisi if tidak terpenuhi maka terdapat else if  $i == n$  atau  $i == n+1$  yang akan menjalankan fmt.print(), ini digunakan untuk mengilangkan angka 1 lebih dari 1 dan angka 0. Kemudian else akan memanggil fungsi rekursif baris(n, i + 1). Pada func main() terdapat deklarasi variable n , program akan menerima input untuk nilai n. kemudian pemanggilan func baris (n, 0 ) i=0 untuk nilai awal.

## 5. Unguided 2

### Source Code

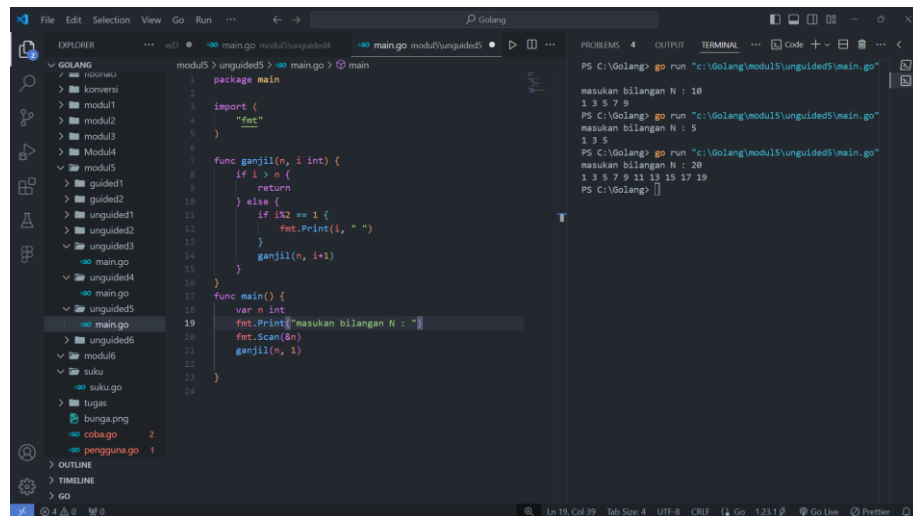
```
package main

import (
    "fmt"
)

func ganjil(n, i int) {
    if i > n {
        return
    } else {
        if i%2 == 1 {
            fmt.Print(i, " ")
        }
        ganjil(n, i+1)
    }
}

func main() {
    var n int
    fmt.Print("masukan bilangan N : ")
    fmt.Scan(&n)
    ganjil(n, 1)
}
```

### Screenshot



## Deskripsi

Program ini adalah program untuk menampilkan bilangan ganjil dari 1 sampai `n`. terdapat dunc `ganjil ()` sebagai fungsi rekursif dengan parameter `n` (sebagai batas angka ) dan `i` ( sebagai angka yang akan dicek). If `i > n` sebagai batas atau base case dari rekursif. Pada else terdapat if `i % 2` yang akan mengecek I apakah bilangan ganjil atau tidak. Jika memenuhi maka akan dicetak `i`. dan terjadi pemanggilan fungsi rekursif `ganjil(n, i + 1)`. Pada func `main()` terdapat deklarasi variable `n` , program akan menerima input untuk nilai `n`. kemudian pemanggilan func `ganjil (n, 1) i=1` untuk nilai awal.

## 6. Unguided 6

### Source Code

```

package main

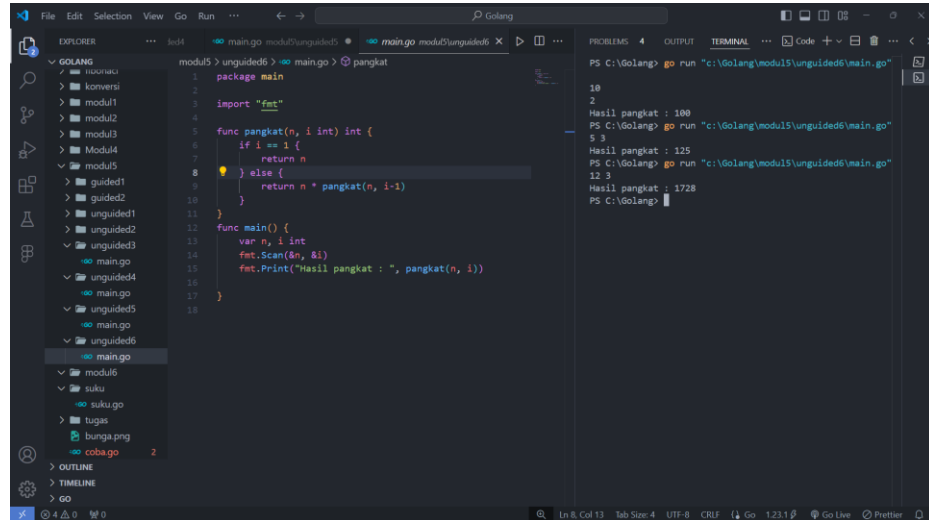
import "fmt"

func pangkat(n, i int) int {
    if i == 1 {
        return n
    } else {
        return n * pangkat(n, i-1)
    }
}

func main() {
    var n, i int
    fmt.Scan(&n, &i)
    fmt.Print("Hasil pangkat : ", pangkat(n, i))
}

```

## Screenshot



```
package main

import "fmt"

func pangkat(n, i int) int {
    if i == 1 {
        return n
    } else {
        return n * pangkat(n, i-1)
    }
}

func main() {
    var n, i int
    fmt.Scan(&n, &i)
    fmt.Print("Hasil pangkat : ", pangkat(n, i))
}
```

Terminal Output:

```
PS C:\Golang> go run "c:\Golang\modul5\unguided6\main.go"
10
2
Hasil pangkat : 100
PS C:\Golang> go run "c:\Golang\modul5\unguided6\main.go"
5 3
Hasil pangkat : 125
PS C:\Golang> go run "c:\Golang\modul5\unguided6\main.go"
12 3
Hasil pangkat : 1728
PS C:\Golang>
```

## Deskripsi

Program ini adalah program untuk menampilkan hasil pangkat dari  $n$  pangkat  $i$ . pada func pangkat() sebagai fungsi rekursif dengan parameter  $n$  sebagai bilangan yang akan dipangkatkan dan  $i$  sebagai bilangan pangkat. If  $i == 1$  maka fungsi rekursif akan berhenti. Pada else terdapat  $\text{return } n * \text{pangkat}(n, i-1)$  ini akan mengalikan  $n$  dengan fungsi rekursif yang dipanggil dengan  $i$  decrement. Pada func main() terdapat deklarasi variable  $n$ , program akan menerima input untuk nilai  $n$ . kemudian pemanggilan func ganjil  $(n, 1) i=1$  untuk nilai awal.