

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL VI
REKURSIF**



Disusun Oleh :

Muhammad Hamzah Haifan Ma'ruf

2311102091

IF-11-05

Dosen Pengampu :

Arif Amrulloh, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Rekursif adalah metode pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah secara langsung atau tidak langsung. Selama proses ini, masalah yang besar dipecah menjadi masalah yang lebih kecil dengan struktur yang sama hingga mencapai kondisi akhir yang sederhana yang disebut sebagai kasus dasar. Rekursi sangat berguna untuk memecahkan masalah yang memiliki pola dan struktur hierarkis berulang, seperti penghitungan faktorial, bilangan Fibonacci, atau pencarian dalam struktur data seperti pohon.

Base Case

Kondisi akhir di mana fungsi berhenti memanggil dirinya sendiri. Ini harus ditentukan dengan jelas agar fungsi tidak melakukannya terus-menerus, yang dapat menyebabkan error seperti stack overflow.

Rekor Hubungan

Situasi di mana fungsi memanggil dirinya sendiri dengan masalah yang lebih kecil untuk mendekati kasus dasar. Dalam rekursi, hubungan kembali adalah dasar yang memungkinkan pemecahan masalah secara bertahap.

Dalam Golang, dapat menggunakan rekursi dengan mendefinisikan fungsi yang memanggil dirinya sendiri. Fungsi ini akan berjalan terus hingga kondisi base case terpenuhi, dan rekursi akan berlanjut dengan melakukan pemanggilan ulang pada sub-masalah yang semakin dekat dengan base cas.

Keuntungan

- Menjadi lebih mudah untuk menulis kode, terutama untuk masalah dengan pola berulang atau struktur hierarkis.
- Mengurangi kompleksitas kode yang diperlukan untuk masalah seperti traversal pohon atau grafik.

Kekurangan

- Setiap pemanggilan fungsi disimpan dalam stack, yang membutuhkan lebih banyak memori.
- Jika base case tidak tercapai atau tidak ditentukan dengan benar, rentan terhadap error stack overflow.

Berbagai algoritma pemrosesan data yang kompleks sering menggunakan regresi, seperti

Traversal pohon

- Untuk melakukan pencarian atau traversal dalam struktur data pohon seperti pohon binar, rekursi digunakan.

Algoritma backtracking

- Digunakan dalam masalah pencarian solusi seperti Sudoku, n-queens, atau pemrosesan graf.

Pemrosesan data bersarang

- Rekursi dapat membantu mengakses komponen yang terstruktur dalam hirarki yang dalam, seperti data JSON yang bersarang (nested).

II. GUIDED

1. GUIDED 1

Studi Case : Membuat baris bilangan dari n hingga 1

Base – case : bilangan == 1

Sourcecode

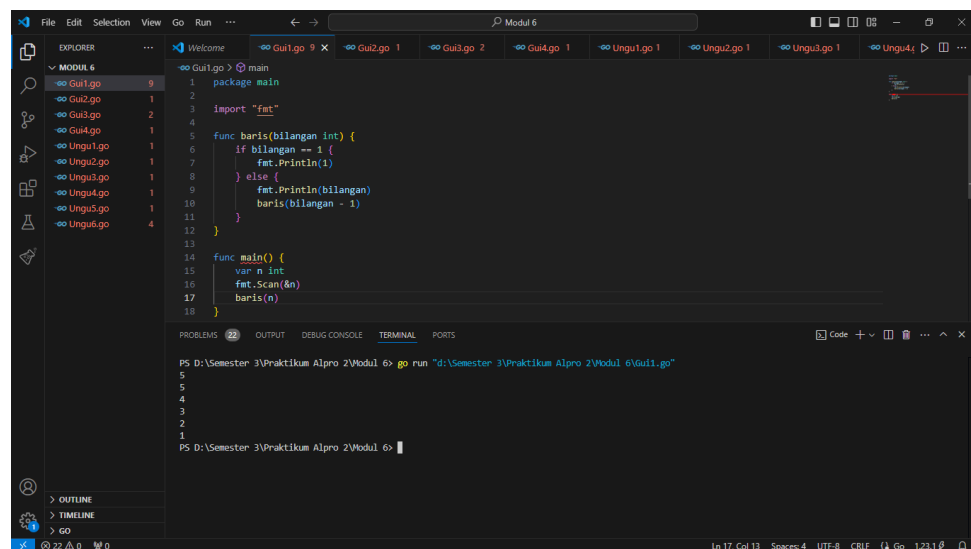
```
package main

import "fmt"

func baris(bilangan int) {
    if bilangan == 1 {
        fmt.Println(1)
    } else {
        fmt.Println(bilangan)
        baris(bilangan - 1)
    }
}

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

Screenshoot Output



Deskripsi Program

Program diatas menggunakan fungsi rekursif yang disebut "baris", program ini mencetak deret angka dari suatu bilangan bulat positif yang dimasukkan pengguna hingga mencapai angka 1. Pertama, pengguna memasukkan angka yang telah disimpan dalam variabel "n". Selanjutnya, fungsi "baris" dihubungkan dengan parameter "bilangan", yang akan berkurang setiap kali fungsi memanggil dirinya sendiri. Dalam kasus basis, jika "bilangan" bernilai 1, program mencetak angka 1 dan berhenti memanggil fungsi secara rekursif. Jika "bilangan" lebih besar dari 1, fungsi mencetak angka tersebut, lalu memanggil dirinya lagi dengan "bilangan-1", sehingga angka dari "n" hingga 1 ditampilkan secara berurutan. Sebagai contoh, jika pengguna memasukkan angka "5", aplikasi akan menampilkan deret angka yang turun dari 5 hingga 1.

2. GUIDED 2

Studi Case : Menghitung hasil penjumlahan 1 hingga n

Base – case : $n == 1$

Sourcecode

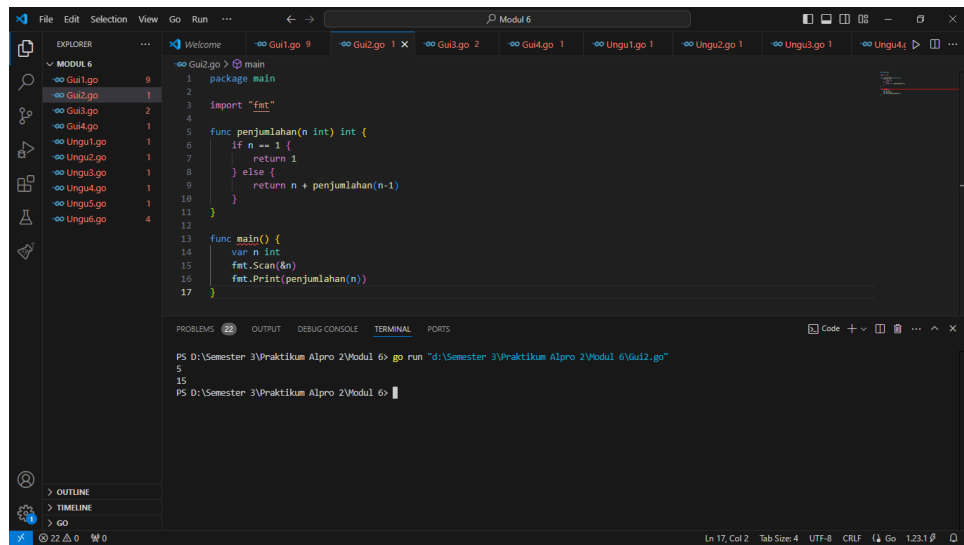
```
package main

import "fmt"

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

func main() {
    var n int
    fmt.Scan(&n)
    fmt.Print(penjumlahan(n))
}
```

Screenshoot Output



Deskripsi Program

Program diatas menghitung jumlah total dari bilangan 1 hingga bilangan bulat positif yang diinputkan oleh pengguna menggunakan fungsi rekursif 'penjumlahan'. Pertama, pengguna diminta memasukkan nilai 'n', kemudian fungsi 'penjumlahan' dipanggil dengan parameter 'n' untuk menghitung jumlah tersebut. Fungsi ini bekerja dengan logika rekursif di mana jika 'n' bernilai 1 (base case), fungsi mengembalikan nilai 1. Namun, jika 'n' lebih besar dari 1, fungsi akan mengembalikan nilai 'n' ditambah hasil dari 'penjumlahan(n-1)', sehingga terjadi proses penjumlahan secara bertahap dari 'n' hingga 1. Hasil akhirnya adalah jumlah dari semua bilangan dari 1 sampai 'n', yang kemudian dicetak. Sebagai contoh, jika pengguna memasukkan angka '5', program akan menghitung $5 + 4 + 3 + 2 + 1$ dan menghasilkan output '15'.

3. GUIDED 3

Studi Case : Mencari dua pangkat n atau $2n$

Base – case: $n == 0$

Sourcecode

```

package main

import "fmt"

func pangkat(n int) int {
    if n == 0 {
        return 1
    } else {
        return 2 * pangkat(n-1)
    }
}

```

```

    }
}

func main() {
    var n int
    fmt.Print("masukkan nilai n : ")
    fmt.Scan(&n)
    fmt.Println("hasil dari 2 pangkat", n, "adalah",
pangkat(n))
}

```

Screenshoot Output

```

1 package main
2
3 import "fmt"
4
5 func pangkat(n int) int {
6     if n == 0 {
7         return 1
8     } else {
9         return 2 * pangkat(n-1)
10    }
11 }
12
13 func main() {
14     var n int
15     fmt.Print("masukkan nilai n : ")
16     fmt.Scan(&n)
17     fmt.Println("hasil dari 2 pangkat", n, "adalah", pangkat(n))
18 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS D:\Semester 3\Praktikum Alpro 2\Modul 6> go run "d:\Semester 3\Praktikum Alpro 2\Modul 6\Gui3.go"
masukkan nilai n : 5
hasil dari 2 pangkat 5 adalah 32
PS D:\Semester 3\Praktikum Alpro 2\Modul 6>

```

Deskripsi Program

Program diatas menghitung nilai dari dua pangkat n, di mana n adalah bilangan bulat non-negatif yang dimasukkan pengguna, dihitung oleh program. Pertama, program meminta pengguna untuk memasukkan nilai n dengan perintah "fmt.Print". Setelah pengguna memasukkan nilai, fungsi rekursif yang disebut "pangkat" dihubungkan dengan parameter n. Logika fungsi ini adalah bahwa jika nilai n adalah 0 (base case), fungsi akan mengembalikan nilai 1, sesuai dengan aturan bahwa 2 pangkat 0 sama dengan 1. Jika nilai n lebih besar dari 0, fungsi akan mengembalikan hasil 2 dikali.

4. GUIDED 4

Studi Case : Mencari nilai factorial atau n!

Base – case: $n == 0$ atau $n == 1$

Sourcecode

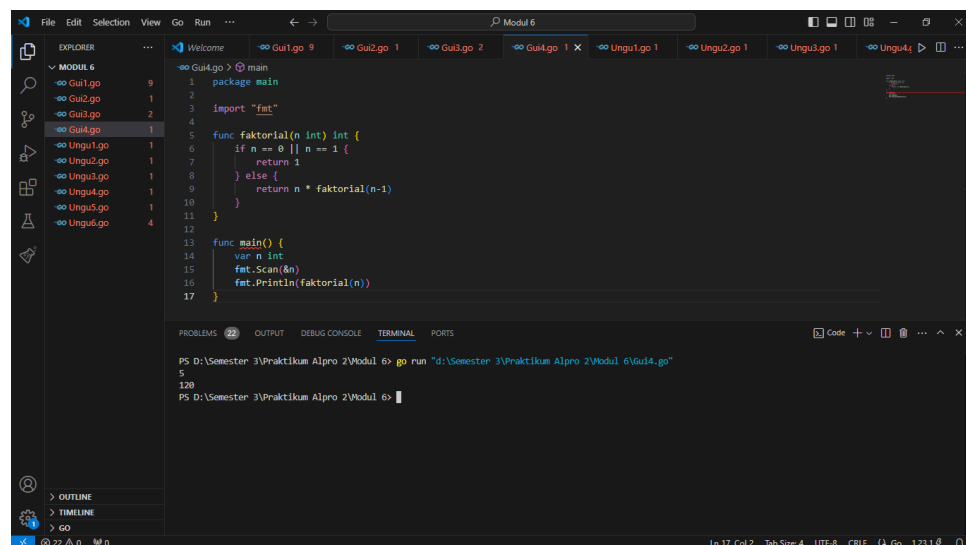
```
package main

import "fmt"

func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    } else {
        return n * faktorial(n-1)
    }
}

func main() {
    var n int
    fmt.Scan(&n)
    fmt.Println(faktorial(n))
}
```

Screenshoot Output



Deskripsi Program

Program diatas digunakan untuk menghitung nilai faktorial dari bilangan bulat non-negatif yang dimasukkan oleh pengguna. Setelah program dimulai, pengguna diminta untuk memasukkan bilangan bulat positif dengan perintah "fmt.Scan"; nilai yang dimasukkan akan disimpan dalam variabel "n", dan fungsi rekursif "faktorial" disebut parameter "n".

Fungsi faktorial bekerja dengan logika di mana jika nilai "n" adalah 0 atau 1, yaitu base case, fungsi akan mengembalikan nilai 1, sesuai dengan aturan

bahwa faktorial antara 0 dan 1 adalah 1. Jika nilai "n" lebih besar dari 1, fungsi akan mengembalikan hasil dari "n" dikalikan dengan pemanggilan rekursif "faktorial(n-1)", dan hasil perhitungan faktorial ini kemudian dicetak menggunakan " .

III. UNGUIDED

1. UNGUIDED 1

Studi Case : Deret fibonacci adalah sebuah deret dengan nilai suku ke-0 dan ke-1 adalah 0 dan 1, dan nilai suku ke-n selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Secara umum dapat diformulasikan $S_n = S_{n-1} + S_{n-2}$. Berikut ini adalah contoh nilai deret fibonacci hingga suku ke-10. Buatlah program yang mengimplementasikan fungsi rekursif pada deret fibonacci tersebut.

n	0	1	2	3	4	5	6	7	8	9	10
S_n	0	1	1	2	3	5	8	13	21	34	55

Sourcecode

```
package main

import (
    "fmt"
)

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

func main() {
    fmt.Printf("Deret fibonacci hingga suku ke-10\n")
    for i := 0; i <= 10; i++ {
        fmt.Printf("Fibonacci - %d = %d\n", i,
        fibonacci(i))
    }
}
```

Screenshoot Output

```

1 package main
2
3 import (
4     "fmt"
5 )
6
7 func fibonacci(n int) int {
8     if n == 0 {
9         return 0
10    } else if n == 1 {
11        return 1
12    }
13    return fibonacci(n-1) + fibonacci(n-2)
14 }
15
16 func main() {
17     fmt.Printf("Deret fibonacci hingga suku ke-10\n")
18     for i := 0; i <= 10; i++ {
19
20     }
21 }

```

```

PS D:\Semester 3\Praktikum Alpro 2\Modul 6> go run "d:\Semester 3\Praktikum Alpro 2\Modul 6\Ungu1.go"
Deret fibonacci hingga suku ke-10
Fibonacci - 0 = 0
Fibonacci - 1 = 1
Fibonacci - 2 = 1
Fibonacci - 3 = 2
Fibonacci - 4 = 3
Fibonacci - 5 = 5
Fibonacci - 6 = 8
Fibonacci - 7 = 13
Fibonacci - 8 = 21
Fibonacci - 9 = 34
Fibonacci - 10 = 55
PS D:\Semester 3\Praktikum Alpro 2\Modul 6>

```

Deskripsi Program

Program diatas menghitung dan mencetak deret Fibonacci hingga suku ke-10, program ini tidak memerlukan pengguna memasukkan input karena deret dihitung secara otomatis dari 0 hingga 10. Dimulai dengan mendefinisikan fungsi rekursif fibonacci, yang mengambil satu parameter n. Fungsi ini bekerja menggunakan logika rekursif, di mana jika n sama dengan 0, maka fungsi mengembalikan 0, dan jika n sama dengan 1, maka mengembalikan 1. Jika n lebih besar dari 1, fungsi akan mengembalikan hasil dari fibonacci(n-1) + fibonacci(n-2), yang merupakan jumlah dari dua suku Fibonacci sebelumnya.

2. UNGUIDED 2

Studi Case : Buatlah sebuah program yang digunakan untuk menampilkan pola bintang berikut ini dengan menggunakan fungsi rekursif. N adalah masukan dari user.

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	* ** *** **** *****
2	1	*
3	3	* ** ***

Sourcecode

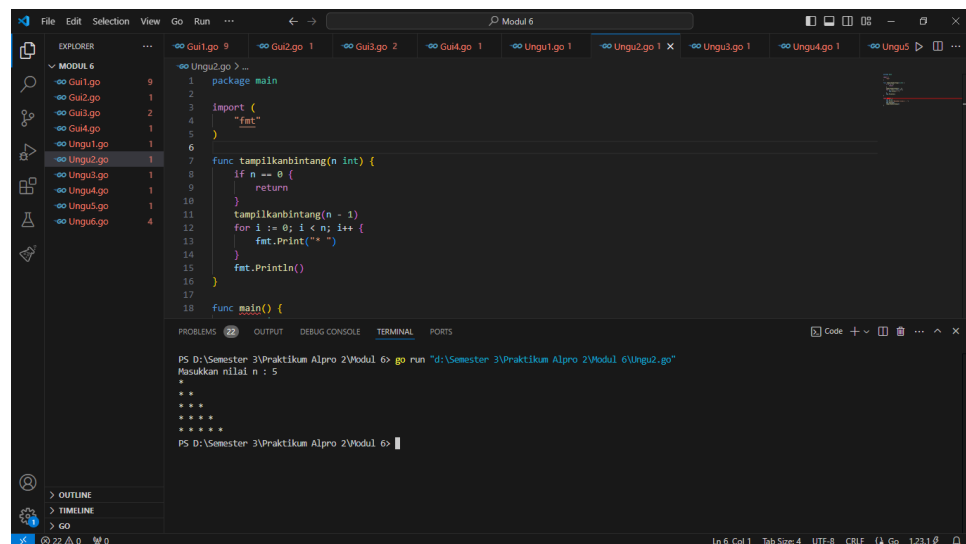
```
package main

import (
    "fmt"
)

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

func main() {
    var n int
    fmt.Print("Masukkan nilai n : ")
    fmt.Scan(&n)
    tampilkanbintang(n)
}
```

Screenshoot Output

The screenshot shows a Go IDE with a dark theme. The Explorer panel on the left shows a project named 'Modul 6' with several Go files. The main editor displays the source code for 'Ungu2.go', which is the same code as shown in the 'Sourcecode' section. Below the editor, the 'TERMINAL' panel shows the command 'go run "d:\Semester 3\Praktikum Alpro 2\Modul 6\Ungu2.go"' and its output. The output shows the prompt 'Masukkan nilai n : 5', followed by the execution of the program which prints a right-angled triangle of stars: five rows of stars, with the first row having 5 stars and each subsequent row having one fewer star. The status bar at the bottom indicates 'Ln 6, Col 1 Tab Size: 4 UTF-8 CRLF' and the Go version '1.23.1'.

Deskripsi Program

Program diatas menggunakan nilai integer positif yang dimasukkan oleh pengguna, program ini mencetak pola bintang segitiga terbalik. Setelah program dimulai, pengguna diminta untuk memasukkan nilai n, yang

merupakan tinggi segitiga bintang yang akan dicetak, dan nilai ini disimpan dalam variabel n.

Fungsi rekursif "Tampilkan bintang" berfungsi untuk mencetak pola bintang. Parameter n diberikan kepada fungsi ini. Jika n bernilai 0, fungsi akan menghentikan pemanggilan dan kembali tanpa melakukan tindakan apa pun. Namun, jika n lebih besar dari 0, fungsi pertama-tama memanggil dirinya sendiri dengan parameter n-1, yang menghasilkan efek rekursif yang mendalam. Fungsi mencetak n bintang di baris saat ini setelah pemanggilan rekursif, menggunakan loop for untuk mengulang pencetakan bintang sebanyak n kali. Kemudian, fungsi memanggil `fmt.Println`.

3. UNGUIDED 3

Studi Case : Buatlah program yang mengimplementasikan rekursif untuk menampilkan faktor bilangan dari suatu N, atau bilangan yang apa saja yang habis membagi N. Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan yang menjadi faktor dari N (terurut dari 1 hingga N ya).

Contoh masukan dan keluaran :

No	Masukan	Keluaran
1	5	1 5
2	12	1 2 3 4 6 12

Sourcecode

```
package main

import (
    "fmt"
)

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

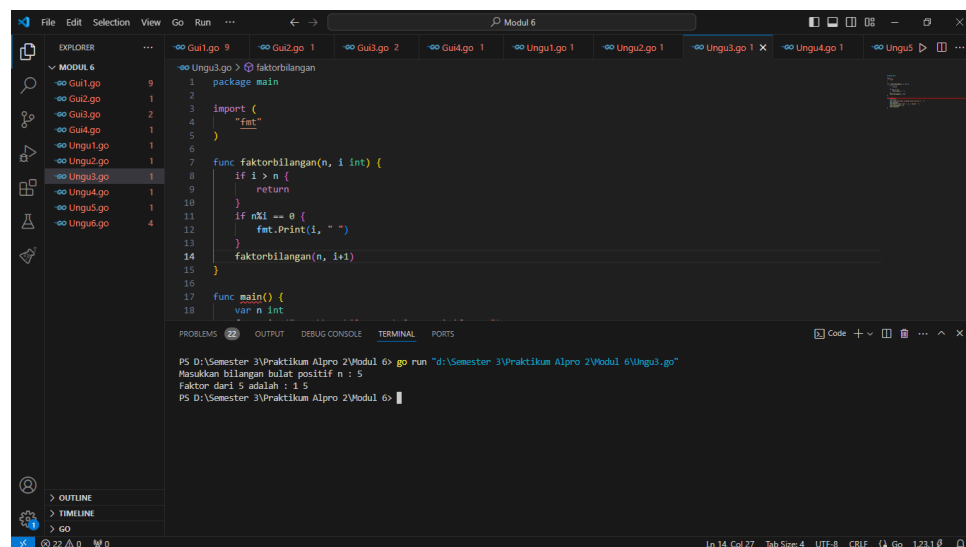
func main() {
    var n int
```

```

    fmt.Print("Masukkan bilangan bulat positif n : ")
    fmt.Scan(&n)
    fmt.Print("Faktor dari ", n, " adalah : ")
    faktorbilangan(n, 1)
    fmt.Println()
}

```

Screenshoot Output



Deskripsi Program

Program diatas untuk mencetak faktor-faktor dari bilangan bulat positif yang dimasukkan oleh pengguna. Setelah program dimulai, pengguna diminta untuk memasukkan bilangan bulat positif n, yang akan dianalisis untuk mengidentifikasi faktor-faktornya. Variabel n menyimpan nilai input.

Fungsi faktorbilangan adalah fungsi rekursif dengan dua parameter: n, bilangan faktor yang ingin dicetak, dan i, yang digunakan untuk mengecek setiap angka dari 1 hingga n. Fungsi ini mulai dengan memeriksa apakah nilai i lebih besar dari n, dan jika benar, fungsi akan menghentikan panggilan dan kembali tanpa melakukan tindakan tambahan. i dicetak sebagai faktor dari n jika n dibagi i memiliki sisa 0.

4. UNGUIDED 4

Studi Case : Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan tertentu. Masukan terdiri dari sebuah bilangan bulat positif N. Keluaran terdiri dari barisan bilangan dari N hingga 1 dan kembali ke N.

Contoh masukan dan keluaran :

No	Masukan	Keluaran
1	5	5 4 3 2 1 2 3 4 5
2	9	9 8 7 6 5 4 3 2 1 2 3 4 5 6 7 8 9

Sourcecode

```
package main

import (
    "fmt"
)

func turun(n int) {
    if n == 1 {
        fmt.Printf("%d ", n)
    } else {
        fmt.Printf("%d ", n)
        turun(n - 1)
    }
}

func naik(n, i int) {
    if i == n {
        fmt.Printf("%d ", i)
    } else {
        fmt.Printf("%d ", i)
        naik(n, i+1)
    }
}

func tampilkanbarisan(n int) {
    turun(n)
    naik(n, 2)
    fmt.Println()
}

func main() {
    var n int
    fmt.Print("Masukkan bilangan bulat positif n : ")
    fmt.Scanln(&n)
    fmt.Print("Hasil barisan bilangan : ")
    tampilkanbarisan(n)
}
```

Screenshoot Output

Deskripsi Program

Pogram diatas menghitung barisan angka dibuat oleh program ini, yang dimulai dengan bilangan bulat positif yang dimasukkan oleh pengguna. Angka ditampilkan dalam urutan menurun terlebih dahulu dan kemudian diikuti dengan urutan menaik. Setelah program dimulai, pengguna diminta untuk memasukkan bilangan bulat positif n , yang kemudian disimpan dalam variabel n . Fungsi turun adalah fungsi rekursif yang ditugaskan untuk mencetak angka dari n hingga 1. Jika nilai n sama dengan 1, fungsi akan mencetak angka 1, tetapi jika tidak, fungsi akan mencetak nilai n . Selanjutnya, fungsi memanggil dirinya sendiri dengan parameter $n - 1$, sehingga menghasilkan urutan menurun.

Fungsi naik adalah fungsi rekursif yang mencetak angka mulai dari 2 hingga n . Nilai i akan dicetak jika nilai i sama dengan n , dan nilai i saat ini akan dicetak jika tidak. Kemudian fungsi memanggil dirinya sendiri dengan parameter $i + 1$, sehingga menghasilkan urutan naik. Kedua fungsi tersebut dapat digunakan dengan menggunakan fungsi tampilanbarisan. Fungsi turun(n) dapat digunakan untuk mencetak angka menurun dari n hingga 1, dan fungsi naik($n, 2$) dapat digunakan untuk mencetak angka menaik dari n hingga 2.

5. UNGUIDED 5

Studi Case : Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan ganjil. Masukan terdiri dari sebuah bilangan bulat positif N . Keluaran terdiri dari barisan bilangan ganjil dari 1 hingga N .

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	5	1 3 5
2	20	1 3 5 7 9 11 13 15 17 19

Sourcecode

```
package main

import (
    "fmt"
)

func tampilkanganjil(n int, i int) {
    if i > n {
        return
    }
    if i%2 != 0 {
        fmt.Print(i, " ")
    }
    tampilkanganjil(n, i+1)
}

func main() {
    var n int
    fmt.Print("Masukkan bilangan bulat positif n : ")
    fmt.Scan(&n)
    fmt.Print("Barisan bilangan ganjil dari 1 - ", n, "
adalah : ")
    tampilkanganjil(n, 1)
    fmt.Println()
}
```

Screenshoot Output

```

1  func tampilKanganjil(n int, i int) {
2      if i%2 != 0 {
3          fmt.Println(i, " ")
4          tampilKanganjil(n, i+1)
5      }
6  }
7
8  func main() {
9      var n int
10     fmt.Println("Masukkan bilangan bulat positif n : ")
11     fmt.Scan(&n)
12     fmt.Println("Barisan bilangan ganjil dari 1 - ", n, " adalah : ")
13     tampilKanganjil(n, 1)
14     fmt.Println()
15 }

```

```

PS D:\Semester 3\Praktikum Alpro 2\Modul 6> go run "d:\Semester 3\Praktikum Alpro 2\Modul 6\Ungu5.go"
Masukkan bilangan bulat positif n : 5
Barisan bilangan ganjil dari 1 - 5 adalah : 1 3 5
PS D:\Semester 3\Praktikum Alpro 2\Modul 6>

```

Deskripsi Program

Program diatas dapat mencetak bilangan ganjil mulai dari 1 hingga bilangan bulat positif yang dimasukkan pengguna. Saat program dimulai, pengguna diminta untuk memasukkan bilangan bulat positif n, yang akan digunakan sebagai batas atas untuk mencetak bilangan ganjil.

Fungsi rekursif tampilKanganjil mengambil dua parameter: n, batas atas, dan i, angka yang saat ini diperiksa. Fungsi ini mulai dengan menentukan apakah i lebih besar dari n. Jika itu benar, fungsi akan menghentikan pemanggilan dan kembali tanpa melakukan tindakan lebih lanjut. i dicetak sebagai bilangan ganjil dalam kasus di mana i dibagi dua dan sisanya tidak sama dengan 0.

6. UNGUIDED 6

Studi Case : Buatlah program yang mengimplementasikan rekursif untuk mencari hasil pangkat dari dua buah bilangan. Masukan terdiri dari bilangan bulat x dan y. Keluaran terdiri dari hasil x dipangkatkan y. Catatan : diperbolehkan menggunakan asterisk "*", tapi dilarang menggunakan import "math".

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	2 2	4
2	5 3	125

Sourcecode

```

package main

import (
    "fmt"
)

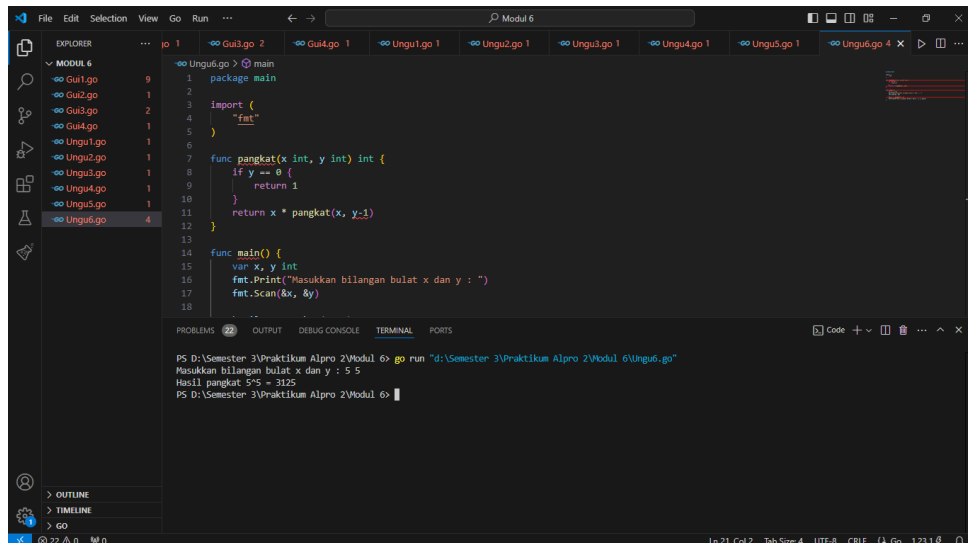
func pangkat(x int, y int) int {
    if y == 0 {
        return 1
    }
    return x * pangkat(x, y-1)
}

func main() {
    var x, y int
    fmt.Print("Masukkan bilangan bulat x dan y : ")
    fmt.Scan(&x, &y)

    hasil := pangkat(x, y)
    fmt.Printf("Hasil pangkat %d^%d = %d\n", x, y, hasil)
}

```

Screenshoot Output



Deskripsi Program

Program diatas berfungsi untuk menghitung nilai pangkat dari bilangan bulat x yang dipangkatkan dengan bilangan bulat y yang dimasukkan oleh pengguna. Setelah program dimulai, pengguna diminta untuk memasukkan dua bilangan bulat, x dan y, yang akan digunakan untuk perhitungan pangkat.

Fungsi pangkat adalah fungsi rekursif dengan dua parameter: x , yang merupakan basis (bilangan yang akan dipangkatkan), dan y , yang merupakan eksponen (kuasa yang akan diterapkan pada basis). Dalam logika rekursif, jika y sama dengan 0 dan y lebih besar dari 0, fungsi akan mengembalikan nilai 1, sesuai dengan aturan bahwa setiap bilangan pangkat 0 sama dengan 1.

IV. KESIMPULAN

Semua program menggunakan fungsi rekursif untuk melakukan tugas tertentu, seperti menghitung faktorial, menemukan faktor bilangan, dan mencetak deret angka. Untuk memudahkan perhitungan, rekursi memecah masalah menjadi bagian yang lebih kecil hingga mencapai kondisi dasar. Program interaktif ini memungkinkan pengguna memasukkan nilai yang akan diproses dengan menggunakan `fmt.Scan` dan menampilkan hasilnya dengan `fmt.Print` atau `fmt.Println`, yang memberikan umpan balik yang jelas. Setiap program menggunakan algoritma yang berbeda, termasuk menghitung nilai pangkat, menampilkan pola bintang, dan mencetak bilangan ganjil. Algoritma ini menunjukkan kemampuan rekursi untuk menyelesaikan berbagai jenis masalah.

Struktur program serupa, dengan fungsi `main` sebagai titik masuk dan definisi fungsi rekursif untuk melakukan perhitungan, yang membuat program lebih terorganisir dan mudah dibaca. Meskipun metode rekursif seringkali lebih mudah dibaca dan dipahami, perlu diperhatikan bahwa dalam kasus masalah yang signifikan atau pemanggilan fungsi yang kompleks, rekursi dapat menjadi kurang efisien dibandingkan dengan iterasi karena dapat membutuhkan lebih banyak memori.

V. REFERENSI

- [1] Modul VI Praktikum Algoritma dan Pemrograman 2.
- [2] Fazry, M. R. (2021b, December 9). Cara Rekursif di Golang - Muhammad Redyka Fazry - Medium. *Medium*. <https://medium.com/@fazry/cara-rekursif-di-golang-cc202f55e336>.
- [3] School, D. (2019, March 4). *Recursion dalam Bahasa Golang - Kursus Web Programming*. Kursus Web Programming. <https://kursuswebprogramming.com/recursion-dalam-bahasa-golang/>.