

LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 5
REKURSIF



DISUSUN OLEH:
ANASTASIA ADINDA NARENDRA INDRIANTO
103112400085
S1 IF-12-01
DOSEN:
Dimas Fanny Hebrasianto Permadi, S.ST., M.KOM

S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025

I. DASAR TEORI

1. Pengertian Bahasa Pemrograman Golang

Golang adalah bahasa pemrograman yang memiliki sejumlah kelebihan yang tidak dimiliki bahasa pemrograman yang lainnya. Hadirnya bahasa pemrograman Go Language (Golang) semakin dirasakan oleh para pengembang. Tidak heran jika banyak orang yang mulai belajar bahasa pemrograman yang satu ini.

Golang merupakan bahasa pemrograman yang dibuat Google dan tujuannya untuk menyempurnakan bahasa pemrograman yang ada, seperti C, Python dan yang lainnya. Golang bisa jadi pilihan yang tepat saat membuat aplikasi baru.

2. Pengertian Input dan Output

- i. *Input* atau masukan adalah data yang diberikan ke dalam program. Masukan ini bisa berasal dari berbagai sumber, seperti pengguna melalui keyboard, mouse, atau suara, dan juga bisa berasal dari sensor atau perangkat lain yang terhubung ke komputer. Dalam pemrograman, input diperlakukan sebagai bahan baku yang akan diproses oleh program.
- ii. *Output* atau keluaran adalah hasil yang diproduksi oleh program setelah mengolah input. Output bisa berbentuk tampilan pada layar, cetakan pada printer, suara, ataupun penyimpanan data ke file. Inti dari program yang kita tulis sebenarnya adalah menghasilkan output yang bermakna dari input yang diberikan.

3. Pengertian Tipe Data

Tipe data adalah kategori data yang akan diproses oleh sebuah program komputer. Dengan tipe data, program dapat mengetahui cara menangani data dengan benar, seperti melakukan operasi matematika pada angka atau memanipulasi string untuk tujuan tertentu. Secara sederhana, pengertian tipe data adalah kategori data yang akan diproses oleh sebuah program komputer. Dengan tipe data, program dapat mengetahui cara menangani data dengan benar, seperti melakukan operasi matematika pada angka atau memanipulasi string untuk tujuan tertentu. Contoh paling sederhana dari tipe data adalah tipe data **integer** yang digunakan untuk menyimpan angka bulat atau tipe data **string** yang digunakan untuk menyimpan rangkaian karakter. Penggunaan tipe data yang benar dalam suatu program memastikan bahwa data diolah secara tepat dan mengurangi risiko kesalahan atau bug dalam program tersebut.

4. Fungsi Tipe Data

- i. **Menentukan Jenis Nilai:** Tipe data memberi tahu program jenis nilai yang akan disimpan dalam variabel. Misalnya, jika kita ingin menyimpan angka, kita menggunakan tipe data integer atau float, sedangkan untuk menyimpan teks, kita menggunakan tipe data string. Ini membantu program memahami bagaimana cara menangani data tersebut.
- ii. **Efisiensi Penggunaan Memori:** Setiap tipe data memerlukan jumlah memori yang berbeda. Jika kita memilih tipe data yang tepat, program bisa menggunakan memori lebih efisien. Misalnya, menggunakan tipe data yang lebih kecil untuk angka yang tidak terlalu besar akan menghemat ruang di memori.
- iii. **Menjamin Konsistensi Data:** Dengan menentukan tipe data, kita memastikan bahwa variabel hanya bisa menyimpan jenis nilai yang sesuai. Misalnya, jika kita mendeklarasikan variabel sebagai tipe integer, program tidak akan bisa memasukkan teks atau jenis data lainnya ke dalamnya. Ini membantu menjaga agar data tetap konsisten dan sesuai dengan yang diharapkan.
- iv. **Memudahkan Operasi pada Data:** Tipe data juga menentukan operasi apa saja yang bisa dilakukan pada data. Misalnya, kita bisa melakukan perhitungan

matematika pada angka, tetapi kita tidak bisa melakukan hal yang sama pada teks. Jadi, dengan menentukan tipe data yang tepat, kita bisa melakukan operasi yang sesuai dengan jenis data yang kita miliki.

5. Pengertian If-Else dan Fungsinya

If-else adalah struktur percabangan dalam pemrograman yang digunakan untuk mengeksekusi kode berdasarkan suatu kondisi. Jika kondisi dalam if bernilai true, maka blok kode di dalamnya akan dijalankan. Jika tidak, program akan memeriksa kondisi dalam else if sebagai alternatif. Jika semua kondisi false, maka else akan dijalankan sebagai pilihan terakhir. Dengan menggunakan if-else, program dapat mengambil keputusan dan menjalankan instruksi yang sesuai berdasarkan kondisi yang diberikan.

If-else berfungsi untuk mengontrol alur program dengan mengeksekusi kode berdasarkan suatu kondisi. If digunakan untuk memeriksa apakah suatu kondisi bernilai true, jika ya, maka blok kode di dalamnya akan dijalankan. Jika tidak, program dapat menggunakan else-if untuk mengevaluasi beberapa kondisi tambahan secara berurutan. Jika semua kondisi false, maka else akan dieksekusi sebagai pilihan terakhir. Selain itu, terdapat if-else bersarang, yang memungkinkan pengecekan kondisi di dalam kondisi lain untuk menangani keputusan yang lebih kompleks.

6. Pengertian While Loop dan Fungsinya

While loop adalah metode perulangan yang mengeksekusi blok kode selama kondisi yang diberikan bernilai true dan akan berhenti ketika kondisi berubah menjadi false. Perulangan ini sangat berguna dalam kasus di mana jumlah iterasi belum diketahui secara pasti, seperti membaca input hingga valid atau menjalankan suatu proses hingga syarat tertentu terpenuhi. Fungsinya adalah;

- i. **Menjalankan Perulangan Berdasarkan Kondisi** – While loop memastikan suatu proses terus berjalan selama kondisi bernilai **true**.
- ii. **Mengatasi Iterasi yang Tidak Diketahui Jumlahnya** – Digunakan ketika jumlah perulangan tidak bisa ditentukan sejak awal, seperti menunggu input yang valid.
- iii. **Mengoptimalkan Kontrol Program** – Membantu mengelola eksekusi kode agar hanya berjalan saat kondisi tertentu terpenuhi, meningkatkan efisiensi program.

7. Pengertian Fungsi dan Manfaatnya

Fungsi merupakan rangkaian instruksi yang menghasilkan suatu nilai dengan memetakan input ke output tertentu. Dalam pemrograman, suatu subprogram dikategorikan sebagai fungsi apabila memiliki deklarasi tipe nilai yang dikembalikan dan menggunakan kata kunci return dalam tubuhnya. Fungsi digunakan dalam berbagai situasi, seperti menetapkan nilai ke suatu variabel melalui assignment, menjadi bagian dari suatu ekspresi, atau digunakan sebagai argumen dalam subprogram lain. Karena fungsi selalu menghasilkan nilai, penamaannya sebaiknya bersifat deskriptif dan mencerminkan hasil yang diberikan, seperti *median*, *rerata*, *nilaiTerbesar*, atau *selesai*.

Function memungkinkan programmer membagi kode menjadi segmen-segmen kecil yang lebih terkelola, masing-masing melakukan bagian tertentu dari tugas yang lebih besar. Tidak hanya membantu dalam mengorganisasi kode secara lebih efisien, hal ini juga memudahkan pemeliharaan dan pengujian kode.

8. Pengertian Prosedur dan Manfaatnya

Prosedur adalah kumpulan instruksi yang dikemas menjadi satu kesatuan untuk menyederhanakan kode dalam program besar. Prosedur tidak mengembalikan nilai karena tidak memiliki deklarasi tipe nilai kembalian dan tidak menggunakan kata kunci *return*. Ketika dipanggil, prosedur langsung memberikan efek pada program utama, seperti halnya instruksi dasar atau fungsi bawaan (*built-in*). Nama prosedur sebaiknya menggunakan kata kerja atau kata yang menggambarkan proses, seperti *cetak*, *hitungRerata*, atau *cariNilai*. Hal ini bertujuan agar lebih mudah dipahami dan sesuai dengan fungsinya dalam program. Dengan menggunakan prosedur, kode program menjadi lebih terstruktur, mudah dibaca, dan dikelola.

Manfaat utama prosedur adalah meningkatkan keterbacaan dan keteraturan kode, sehingga lebih mudah dikelola dan diperbaiki. Dengan memecah program menjadi bagian-bagian kecil, prosedur juga membantu dalam mengurangi pengulangan kode (code redundancy), meningkatkan efisiensi, serta mempermudah debugging dan pengembangan program. Hal ini membuat program lebih modular dan fleksibel untuk diperbarui di masa mendatang.

9. Pengertian Rekursif dan Fungsinya

Rekursi adalah teknik dalam pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah yang lebih kecil dari masalah utama.

Pendekatan ini sering digunakan untuk menyelesaikan permasalahan yang dapat dipecah menjadi submasalah serupa, seperti pencarian dalam struktur data pohon atau perhitungan faktorial. Agar tidak berjalan tanpa henti, rekursi memerlukan kondisi dasar yang menentukan kapan fungsi harus berhenti memanggil dirinya sendiri.

Fungsi rekursif memiliki keunggulan dalam menyederhanakan kode untuk masalah yang memiliki pola berulang, seperti algoritma Divide and Conquer atau pemrosesan data yang bersifat hierarkis. Namun, jika tidak dirancang dengan baik, rekursi dapat menyebabkan penggunaan memori yang berlebihan atau bahkan error karena stack overflow. Oleh karena itu, memahami kapan dan bagaimana menggunakan rekursi dengan efisien sangat penting dalam pemrograman.

II. GUIDED

1. Guide 1

Source Code:

```
// Anastasia Adinda N.I
// 103112400085
// Iterasi
package main

import "fmt"

func pangkatIteratif(base, exp int) int {
    hasil := 1
    for i := 0; i < exp; i++ {
        hasil *= base
    }
    return hasil
}

func faktorialIteratif(n int) int {
    hasil := 1
    for i := 2; i <= n; i++ {
        hasil *= i
    }
    return hasil
}

func main() {
    var base, exp, n int

    fmt.Print("Masukkan bilangan: ")
    fmt.Scanln(&base)
    fmt.Print("Masukkan pangkat: ")
    fmt.Scanln(&exp)

    fmt.Printf("%d^%d = %d\n", base, exp, pangkatIteratif(base, exp))

    fmt.Print("Masukkan angka untuk faktorial: ")
    fmt.Scanln(&n)

    fmt.Printf("%d! = %d\n", n, faktorialIteratif(n))
}
```

Output:

```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL5> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODUL5\Guided\Guided1.go"
Masukkan bilangan: 15
Masukkan pangkat: 5
15^5 = 759375
Masukkan angka untuk faktorial: 5
5! = 120
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL5> |
```

Deskripsi Program:

Program Guided1.go dengan bahasa Go lang dibuat untuk menghitung perpangkatan dan faktorial secara iteratif. Perpangkatan dihitung dengan mengalikan bilangan dasar berulang kali sesuai pangkatnya, menggunakan rumus $a^b = a * a * \dots * a$. Faktorial dihitung dengan mengalikan bilangan dari 2 hingga n, sesuai rumus $n! = n * (n-1) * \dots * 1$. Metode iteratif ini memastikan efisiensi tanpa memerlukan rekursi.

2. Guide 2

Source Code:

```
// Anastasia Adinda N.I
// 103112400085
// Rekursif
package main

import "fmt"

func pangkatRekursif(base, exp int) int {
    if exp == 0 {
        return 1
    }
    return base * pangkatRekursif(base, exp-1)
}

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

func main() {
    var base, exp, n int

    fmt.Print("Masukkan bilangan: ")
    fmt.Scanln(&base)
    fmt.Print("Masukkan pangkat: ")
    fmt.Scanln(&exp)
```

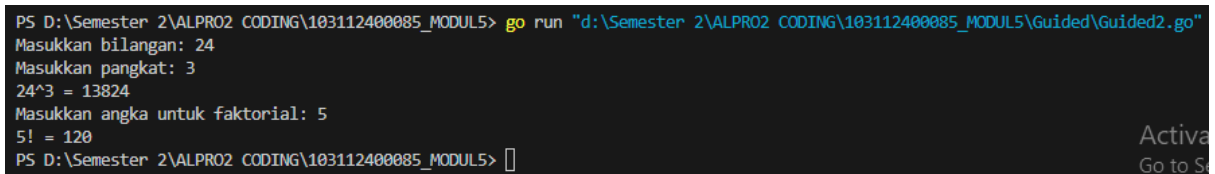
```
    fmt.Printf("%d^%d = %d\n", base, exp, pangkatRekursif(base, exp))

    fmt.Print("Masukkan angka untuk faktorial: ")
    fmt.Scanln(&n)

    fmt.Printf("%d! = %d\n", n, faktorialRekursif(n))

}
```

Output:



```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL5> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODUL5\Guided\Guided2.go"
Masukkan bilangan: 24
Masukkan pangkat: 3
24^3 = 13824
Masukkan angka untuk faktorial: 5
5! = 120
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL5> 
```

Deskripsi Program:

Program Guided2.go dengan Bahasa Go dibuat dengan tujuan Program ini menghitung perpangkatan dan faktorial menggunakan metode rekursif. Perpangkatan dihitung dengan rumus $a^b = a * b^{(b-1)}$ hingga mencapai pangkat nol sebagai kondisi dasar. Faktorial dihitung dengan rumus $n! = n * (n-1)!$ dengan kondisi dasar saat $n=0$ atau $n=1$. Dengan rekursi, fungsi terus memanggil dirinya sendiri hingga mencapai kondisi dasar, menghasilkan perhitungan yang lebih elegan dibandingkan metode iteratif.

III. UNGUIDED

1. Unguided 1

Source Code:

```
// Anastasia Adinda N.I
// 103112400085
package main

import "fmt"

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

func main() {
    for i := 0; i <= 10; i++ {
        fmt.Printf("%v ", fibonacci(i))
    }
    fmt.Println()
}
```

Output:

```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL5> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODUL5\Unguided\Unguided1.go"
0 1 1 2 3 5 8 13 21 34 55
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL5> █
```

Deskripsi Program:

Program Unguided1.go dengan bahasa Go lang digunakan menghitung deret Fibonacci menggunakan metode rekursif dengan rumus 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 $S_n = S_{n-1} + S_{n-2}$.

2. Unguided 2

Source Code:

```
// Anastasia Adinda N.I
// 103112400085
package main

import "fmt"

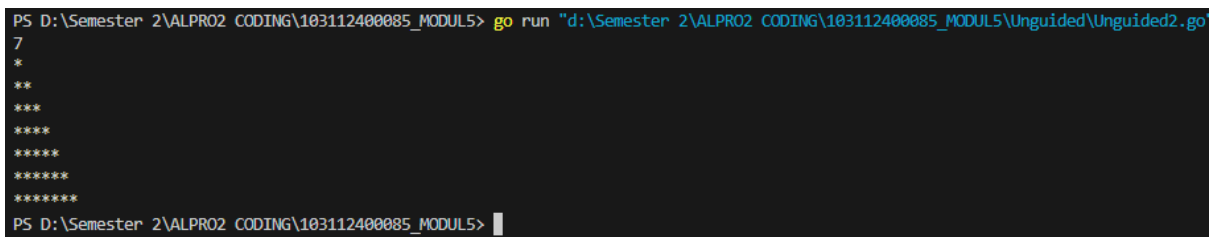
func cetakBaris(n int) {
    if n > 0 {
        fmt.Print("*")
        cetakBaris(n - 1)
    }
}

func cetakBintang(n, i int) {
    if i > n {
        return
    }
    cetakBaris(i)
    fmt.Println()
    cetakBintang(n, i+1)
}

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

    cetakBintang(n, 1)
}
```

Output:



```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODULE5> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODULE5\Unguided\Unguided2.go"
7
*
**
***
****
*****
*****
*****
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODULE5> |
```

Deskripsi Program:

Program Unguided2.go dengan bahasa Go lang dibuat mencetak pola segitiga Bintang menggunakan metode rekusif. Fungsi cetakBaris(n) mencetak n bintang dalam satu baris, sementara cetakBintang(n, i) mencetak baris bertahap dari i = 1 hingga n. Rekursi digunakan untuk mengulang proses tanpa perulangan eksplisit, sehingga setiap pemanggilan fungsi menghasilkan pola bertingkat hingga mencapai batas yang ditentukan.

3. Unguided 3

Source Code:

```
// Anastasia Adinda N.I
// 103112400085
package main

import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan bilangan: ")
    fmt.Scan(&n)

    fmt.Printf("Faktor dari %d: ", n)
    cetakFaktor(n, 1)
    fmt.Println()
}
```

Output:

```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL5> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODUL5\Unguided\Unguided3.go"
Masukkan bilangan: 12
Faktor dari 12: 1 2 3 4 6 12
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL5> |
```

Deskripsi Program:

Program Unguided3.go dengan bahasa Go dibuat untuk mengimplementasikan rekursif untuk menampilkan faktor bilangan dari suatu N, atau bilangan yang apa saja yang habis membagi N. Fungsi cetakFaktor(n, i) memeriksa apakah i merupakan factor dari n dengan kondisi $n \% i == 0$. Jika iya nilai i dicetak, lalu fungsi dipanggil Kembali dengan i+1 hingga mencapai n.

IV. KESIMPULAN

Rekursi adalah teknik pemrograman dimana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah yang lebih kecil dari masalah utama. Teknik ini sangat berguna untuk menyelesaikan permasalahan yang dapat dipecah menjadi submasalah serupa. Agar rekursi berjalan dengan benar, diperlukan kondisi dasar (base case) yang menentukan kapan fungsi harus berhenti memanggil dirinya sendiri.

Dalam praktikum ini, telah dilakukan implementasi beberapa algoritma menggunakan teknik rekursif, antara lain:

1. Perhitungan Pangkat dan Faktorial

- Implementasi dengan dua pendekatan: iteratif dan rekursif
- Pada pendekatan rekursif, perhitungan pangkat menggunakan rumus $a^b = a * a^{(b-1)}$ dengan kondisi dasar $b=0$
- Faktorial dihitung dengan rumus $n! = n * (n-1)!$ dengan kondisi dasar $n=0$ atau $n=1$

2. Deret Fibonacci

- Deret ini menggunakan konsep rekursif dimana nilai suku ke- n adalah penjumlahan dari dua suku sebelumnya
- Implementasi menggunakan rumus $S_n = S_{n-1} + S_{n-2}$ dengan kondisi dasar $n \leq 1$

3. Pola Segitiga Bintang

- Implementasi rekursif untuk mencetak pola visual tanpa menggunakan perulangan eksplisit
- Menggunakan dua fungsi rekursif: satu untuk mencetak baris dan satu untuk mengontrol jumlah bintang per baris

4. Pencarian Faktor Bilangan

- Implementasi rekursif untuk menemukan semua bilangan yang habis membagi n
- Menggunakan kondisi $n \% i == 0$ untuk menentukan apakah i adalah faktor dari n

Dari hasil praktikum, terlihat bahwa teknik rekursif memberikan pendekatan yang lebih elegan dalam penyelesaian masalah tertentu dibandingkan dengan metode iteratif. Namun, perlu diperhatikan bahwa penggunaan rekursi yang tidak tepat dapat menyebabkan penggunaan memori yang berlebihan atau bahkan stack overflow jika tidak memiliki kondisi dasar yang tepat.

V. REFRENSI

<https://sko.dev/wiki/input-dan-output>

<https://codingstudio.id/blog/golang-adalah/>

<https://dif.telkomuniversity.ac.id/tipe-data-pemrograman/>

<https://rpubs.com/maulidyarahmah/829044>

https://repository.unikom.ac.id/62967/1/Materi%20Pertemuan%204_Labview%201%2BWhile%20Loop%20%2B%20Shift%20Register.pdf

<https://www.revou.co/kosakata/function>

<https://sko.dev/wiki/fungsi>

<https://drive.google.com/file/d/1tMnxOLAYoMqLB9cKyeJHwyFjmyVtKqMV/view?usp=sharing>

<https://janabadra.ac.id/2023/algoritma-rekursi/#:~:text=Rekursi%20adalah%20sebuah%20metode%20pengulangan,seperti%20definisi%20fungsi%20pada%20umumnya.>