**LAPORAN PRAKTIKUM**

**ALGORITMA DAN PEMROGRAMAN 2**

**MODUL V**

**REKURSIF**

****

**DISUSUN OLEH :**

**WILDAN MAULANA ZIDAN**

**2311102162**

**IF 11 02**

**S1 TEKNIK INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

1. **DASAR TEORI**

Rekursif secara sederhana dapat diartikan sebagai cara menyelesaikan suatu masalah dengan cara menyelesaikan sub-masalah yang identic dari masalah utama. Sebagai contoh perhatikan tabel berikut!

|  |  |
| --- | --- |
| **Notasi algoritma** | **Notasi dalam Go** |
| procedure cetak (in x:integer)  algoritma  output(x)  cetak (x+1)  endprocedure | func cetak (x int ) {  fmt. Println(x)  cetak (x+1)  } |

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 cetalk() nilal x alkan selalu Iniversitu bertambah I (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 hetika x bernilai 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 be-7 bita namakan recursive-case atau kasus pemanggilan dirinya sendiri tersebut terjadi. Kondisi dari recursive-case in adalah negasi dari kondisi base-case atau ketika nilai 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)  }  } |

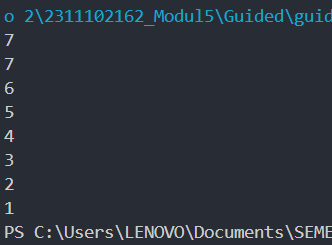
Apabila program di atas ini dijalankan maka akan tampil angka 5 6 7 8 9 10. Terlihat bahwa proses rekursif berhasil dihentikan ketika x == 10.

1. **GUIDED**

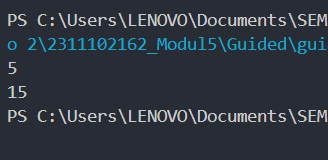
Guided 1

|  |
| --- |
| package main  import "fmt"  func main() {  var n int  fmt.Scan(&n)  baris(n)  }  func baris(bilangan int) {  if bilangan == 1 {  fmt.Println(1)  } else {  fmt.Println(bilangan)  baris(bilangan - 1)  }  } |

Kode di atas adalah membaca input bilangan bulat "n" dari pengguna, kemudian mencetak angka "n" menjadi 1 menggunakan fungsi pengembalian yang disebut "baris ()". Jika angka yang diterima oleh `line()` adalah 1, program akan mencetak 1 dan keluar. Jika tidak, angka tersebut akan dicetak, lalu 'list()' akan dipanggil lagi dengan nilai angka dikurangi 1, sehingga menghasilkan urutan angka dari 'n' ke 1.

Guided 2

|  |
| --- |
| 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)  }  } |

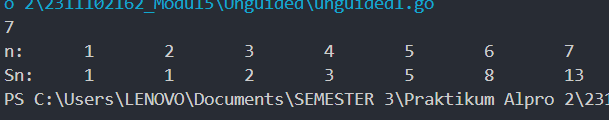
   
  
  
  
Kode di atas membaca input bilangan bulat "n" dari pengguna, menghitung dan mencetak jumlah angka dari 1 hingga "n" menggunakan fungsi pengembalian yang disebut "pejumlahan()". Jika n sama dengan 1, fungsi mengembalikan 1 sebagai nilai kembalian. Jika tidak, fungsi tersebut akan mengembalikan nilai "n" dan hasil pemanggilan fungsi "penjumlahan()" dengan argumen "n-1", sehingga fungsi ini diulangi hingga mencapai 1, lalu jumlahkan 1 menjadi "n".

1. **UNGUIDED**

Unguided 1

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.

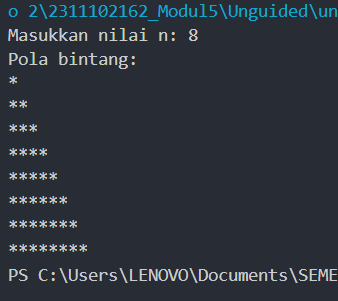
|  |
| --- |
| package main  import (  "fmt"  "strconv"  )  func fibonacci(n int) int {  if n <= 1 {  return n  } else {  return fibonacci(n-1) + fibonacci(n-2)  }  }  func main() {  var inp int  var bilke, fibo string  fmt.Scan(&inp)  for i := 1; i <= inp; i++ {  bilke += "\t" + strconv.Itoa(i)  fibo += "\t" + strconv.Itoa(fibonacci(i))  }  fmt.Printf("n: %s \nSn: %s \n", bilke, fibo)  } |



Kode di atas membaca input bilangan bulat "inp" dari pengguna dan mencetak deret angka dari 1 hingga "inp" termasuk nilai deret Fibonacci. Fungsi 'fibonacci()' menggunakan rekursi untuk menghitung nilai Fibonacci, yang mengembalikan nilai 'n' jika nilai 'n' kurang dari atau sama dengan 1. Jika tidak, ia akan mengembalikan jumlah dari dua Fibonacci sebelumnya. nilai. Dalam ``main()'', perulangan for menambahkan angka ke variabel ``bilke'' dan hasil kali Fibonaccinya ke variabel ``fibo'', dan hasilnya dicetak pada baris terpisah sebanyak dua: satu untuk ` nilai ``n'' dan lainnya. Untuk nilai Fibonacci (Sn).

Unguided 2

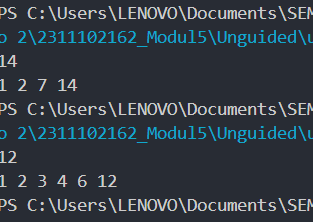
|  |
| --- |
| package main  import "fmt"  func cetakBintang(n int) {  if n == 0 {  return  }  fmt.Print("\*")  cetakBintang(n - 1)  }  func cetakPola(n, current int) {  if current > n {  return  }  cetakBintang(current)  fmt.Println()  cetakPola(n, current+1)  }  func main() {  var n int  fmt.Print("Masukkan nilai n: ")  fmt.Scan(&n)  fmt.Println("Pola bintang:")  cetakPola(n, 1)  } |



Kode di atas mencetak pola bintang bertingkat menggunakan dua fungsi rekursif: `cetakBintang()` dan `cetakPola()`. Fungsi `cetakBintang(n)` bertugas mencetak `n` buah bintang dalam satu baris, menghentikan pencetakan ketika `n` mencapai 0. Fungsi `cetakPola(n, current)` bertanggung jawab untuk mencetak pola bintang bertingkat dari 1 bintang hingga `n` bintang, di mana `current` menunjukkan jumlah bintang yang harus dicetak pada baris tersebut. Jika nilai `current` lebih besar dari `n`, fungsi berhenti. Pada `main()`, nilai `n` diinput oleh pengguna, kemudian pola bintang dicetak dengan memanggil fungsi `cetakPola(n, 1)`.

Unguided 3

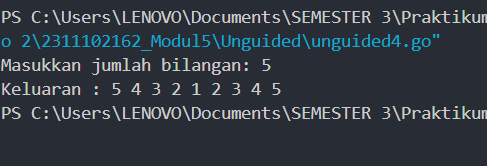
|  |
| --- |
| package main  import "fmt"  func rekursif(n int) {  for i := 1; i <= n; i++ {  if n%i == 0 {  fmt.Print(i, " ")  }  }  }  func main() {  var a int  fmt.Scan(&a)  rekursif(a)  } |



Kode di atas mencetak faktor-faktor dari bilangan bulat `a` yang diinput oleh pengguna. Fungsi `rekursif(n)` menggunakan loop `for` untuk memeriksa setiap bilangan `i` dari 1 hingga `n`. Jika `n` habis dibagi oleh `i` (yaitu `n % i == 0`), maka `i` adalah faktor dari `n` dan dicetak. Pada fungsi `main()`, pengguna diminta untuk menginput nilai `a`, lalu faktor-faktornya ditampilkan dengan memanggil fungsi `rekursif(a)`.

Unguided 4

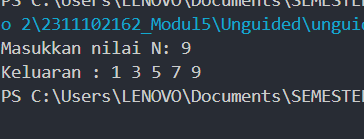
|  |
| --- |
| package main  import "fmt"  func cetakUrutan(jumlah int) {  if jumlah == 1 {  fmt.Print(jumlah, " ")  return  }  fmt.Print(jumlah, " ")  cetakUrutan(jumlah - 1)  fmt.Print(jumlah, " ")  }  func main() {  var jumlah int  fmt.Print("Masukkan jumlah bilangan: ")  fmt.Scanln(&jumlah)  fmt.Print("Keluaran : ")  cetakUrutan(jumlah)  fmt.Println()  } |



Kode di atas mencetak pola bilangan secara rekursif menggunakan fungsi printPattern(n, current). Fungsi ini mencetak angka-angka dari current samapai 1 secara menurun, lalu mencetak angka-angka tersebut kembali secara naik setelah pemanggilan rekursif. Ketika current == 1, fungsi berhenti (basis rekursi). Setelah setiap pemanggilan rekursif, program mencetak tambahan format [n+current-1] di antara angka-angka yang naik. Pada fungsi main(), pengguna menginput nilai nVerse, yang kemudian diproses untuk mencetak pola bilangan sesuai dengan logika rekursif tersebut.

Unguided 5

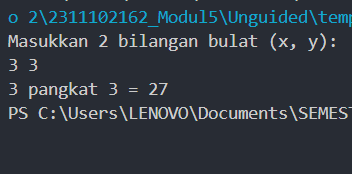
|  |
| --- |
| package main  import "fmt"  func cetakBilanganGanjil(n int) {  if n < 1 {  return  }  cetakBilanganGanjil(n - 2)  if n%2 != 0 {  fmt.Print(n, " ")  }  }  func main() {  var N int  fmt.Print("Masukkan nilai N: ")  fmt.Scanln(&N)  fmt.Print("Keluaran : ")  cetakBilanganGanjil(N)  fmt.Println()  } |



Kode di atas mencetak bilangan ganjil secara rekursif dari 1 hingga bilangan `N` yang diinput oleh user. Fungsi `cetakBilanganGanjil(n)` bekerja dengan cara menurunkan nilai `n` sebesar 2 setiap kali dipanggil, sehingga fungsi hanya akan memproses bilangan ganjil. Fungsi akan berhenti (basis rekursi) ketika `n` kurang dari 1. Setelah kembali dari rekursi, jika `n` adalah bilangan ganjil (dicek dengan `n%2 != 0`), bilangan tersebut dicetak. Pada `main()`, pengguna diminta untuk menginput nilai `N`, dan hasil keluaran berupa bilangan-bilangan ganjil dari 1 sampai `N` dicetak oleh fungsi `cetakBilanganGanjil(N)`.

Unguided 6

|  |
| --- |
| package main  import "fmt"  func power(x, y int) int {  result := 1  for i := 0; i < y; i++ {  result \*= x  }  return result  }  func main() {  fmt.Println("Masukkan 2 bilangan bulat (x, y):")  var x, y int  fmt.Scanln(&x, &y)  result := power(x, y)  fmt.Printf("%d pangkat %d = %d\n", x, y, result)  } |



Kode di atas menghitung hasil pemangkatan bilangan `x` dengan `y` menggunakan fungsi `power(x, y)`. Fungsi ini menggunakan loop `for` untuk mengalikan nilai `x` sebanyak `y` kali, dengan nilai awal `result` dimulai dari 1. Pada fungsi `main()`, pengguna diminta memasukkan dua bilangan bulat `x` dan `y`, kemudian hasil pemangkatan `x` pangkat `y` dihitung dengan memanggil fungsi `power(x, y)` dan hasilnya ditampilkan dalam format `x pangkat y = hasil`.