**LAPORAN PRAKTIKUM   
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL V**

**REKURSIF**

**Sebuah gambar berisi logo, simbol, Grafis, Font

Deskripsi dibuat secara otomatis**

**Disusun Oleh :**

**Fariz Ilham / 2311102275**

**IF-11-06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

1. **DASAR TEORI**

* Pengertian Rekursif

Rekursif adalah teknik pemrograman di mana sebuah fungsi atau prosedur memanggil dirinya sendiri untuk menyelesaikan suatu masalah. Dalam rekursif, masalah dipecah menjadi sub-masalah yang lebih kecil hingga mencapai suatu kondisi dasar, yaitu titik di mana fungsi tersebut berhenti memanggil dirinya sendiri.

* Elemen Rekursif

1. Basis Rekursi (Base Case)

Basis rekursi adalah kondisi di mana rekursi berhenti. Biasanya, ini adalah kasus paling sederhana dari masalah yang sedang diselesaikan dan memastikan bahwa fungsi tidak akan terus menerus memanggil dirinya sendiri.

2. Pemanggilan Rekursif (Recursive Call)

Ini adalah bagian di mana fungsi memanggil dirinya sendiri untuk menyelesaikan versi yang lebih kecil dari masalah. Pemanggilan rekursif ini mendekati kondisi dasar, sehingga rekursi akan berhenti setelah beberapa kali pemanggilan.

* Kelebihan dan Kekurangan Rekursif
* Kelebihan:

1. Sederhana dan Elegan: Rekursi sering kali membuat solusi lebih mudah dipahami karena langsung memecah masalah ke bentuk yang lebih sederhana dan berulang.
2. Efektif untuk Struktur Hierarkis: Rekursi sangat berguna untuk menyelesaikan masalah yang melibatkan struktur seperti pohon atau masalah yang dapat dipecah ke dalam bagian-bagian yang lebih kecil.

* Kekurangan:

1. Konsumsi Memori dan Kinerja: Setiap pemanggilan rekursif membutuhkan alokasi memori tambahan pada stack, yang bisa menyebabkan error \*stack overflow\* jika rekursinya terlalu dalam.
2. Alternatif Iteratif: Beberapa masalah rekursif dapat diselesaikan lebih efisien dengan pendekatan iteratif yang lebih hemat memori.

* Contoh Penerapan Rekursif

Rekursif banyak digunakan dalam berbagai jenis masalah, antara lain:

1. Faktorial: Menghitung hasil kali bilangan dari 1 hingga n.
2. Deret Fibonacci: Menghitung bilangan ke-n dalam deret Fibonacci.
3. Traversal Pohon: Mengunjungi setiap simpul pada struktur pohon, seperti pohon biner.
4. Divide and Conquer: Memecah masalah menjadi sub-masalah yang lebih kecil, seperti dalam algoritma quicksort dan mergesort.
5. **GUIDED**

**Soal Studi Case**

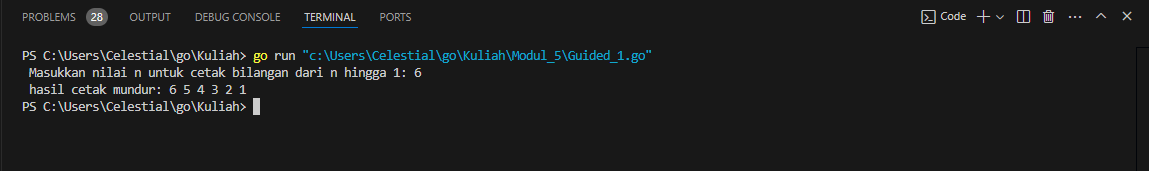
1. Membuat baris bilangan dari n hingga 1

Base-case: bilangan == 1

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  // Fungi untuk mencetak bilangan dari n hingga 1  func cetakMundur(n int) {  if n == 1 {  fmt.Println(n)  return  }  fmt.Print(n, " ")  cetakMundur(n - 1)  }  func main() {  var n int  fmt.Print(" Masukkan nilai n untuk cetak bilangan dari n hingga 1: ")  fmt.Scanln(&n)  fmt.Print(" hasil cetak mundur: ")  cetakMundur(n)  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini mencetak bilangan secara mundur dari nilai ‘n’ yang diberikan pengguna hingga 1. Proses pencetakan dilakukan menggunakan fungsi rekursif.

* Algoritma:

1. Input ( n ) dari pengguna.

2. Panggil fungsi `cetakMundur` dengan parameter ( n ).

3. Di dalam fungsi `cetakMundur`:

- Jika ( n = 1 ), cetak 1 dan hentikan fungsi.

- Jika ( n > 1 ), cetak ( n ), lalu panggil kembali fungsi `cetakMundur` dengan parameter ( n - 1 ).

* Cara Kerja:

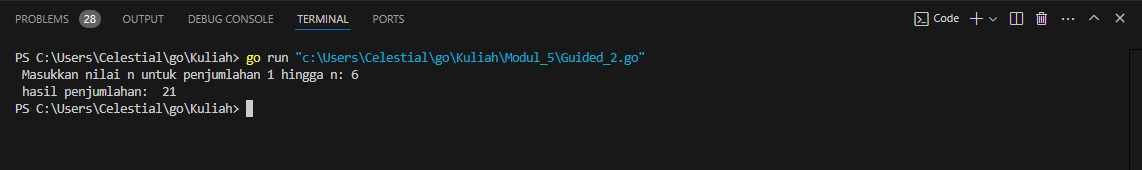
1. Program dimulai dengan meminta pengguna memasukkan nilai ( n ) untuk menentukan batas, atas cetakan mundur.
2. Fungsi `cetakMundur` dipanggil, memulai proses pencetakan dari ( n ) hingga 1.
3. Di setiap langkah rekursif, nilai ( n ) berkurang 1, dan dicetak terlebih dahulu jika ( n > 1 ).
4. Saat ( n ) mencapai 1, fungsi mencetak 1 dan berhenti, mengakhiri proses rekursi.
5. Menghitung hasil penjumlahan 1 hingga n

Base-case: n == 1

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  // fungsi untuk menghitung penjumlahan 1 hingga n  func jumlahRekursif(n int) int {  if n == 1 {  return 1  }  return n + jumlahRekursif(n-1)  }  func main() {  var n int  fmt.Print(" Masukkan nilai n untuk penjumlahan 1 hingga n: ")  fmt.Scanln(&n)  fmt.Println(" hasil penjumlahan: ", jumlahRekursif(n))  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini menggunakan rekursi untuk menghitung dan mencetak hasil penjumlahan bilangan dari 1 hingga ( n ). Pengguna memasukkan nilai ( n ), dan program akan menampilkan hasil penjumlahannya.

* Algoritma:

1. Meminta input nilai ( n ) dari pengguna.
2. Panggil fungsi `jumlahRekursif` dengan parameter ( n ).
3. Di dalam fungsi `jumlahRekursif`:

- Jika ( n = 1 ), kembalikan nilai 1.

- Jika ( n > 1 ), kembalikan hasil penjumlahan ( n ) dengan pemanggilan `jumlahRekursif` berikutnya dengan parameter ( n - 1 ).

4. Tampilkan hasil dari `jumlahRekursif(n)`.

* Cara Kerja:

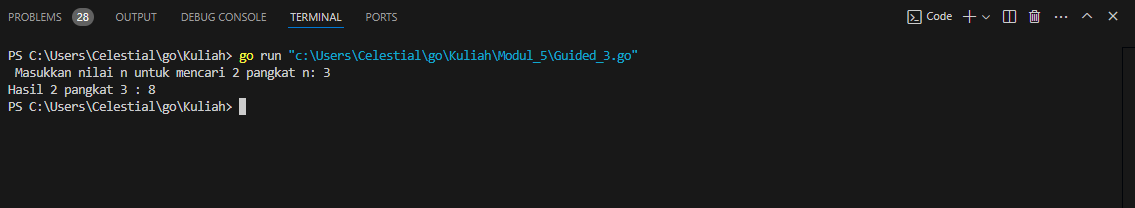
1. Program meminta input ( n ) dari pengguna untuk menentukan batas atas penjumlahan.
2. Fungsi `jumlahRekursif` dipanggil, mulai menghitung penjumlahan dari ( n ) hingga 1.
3. Di setiap langkah rekursif, nilai ( n ) dikurangi 1, dan ditambahkan ke hasil pemanggilan fungsi sebelumnya.
4. Proses berlanjut hingga mencapai kondisi ( n = 1 ), di mana nilai 1 dikembalikan.
5. Hasil akhir dari penjumlahan ditampilkan setelah seluruh proses rekursi selesai.
6. Mencari dua pangkat n atau 2^n

Base-case: n == 0

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  // Fungsi untuk mencari 2 pangkat n  func pangkatDua(n int) int {  if n == 0 {  return 1  }  return 2 \* pangkatDua(n-1)  }  func main() {  var n int  fmt.Print(" Masukkan nilai n untuk mencari 2 pangkat n: ")  fmt.Scanln(&n)  fmt.Println("Hasil 2 pangkat", n, ":", pangkatDua(n))  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini menghitung nilai ( 2^n ) menggunakan rekursi. Pengguna memasukkan nilai ( n ), dan program akan menampilkan hasil perhitungan ( 2^n ).

* Algoritma:

1. Ambil input nilai ( n ) dari pengguna.
2. Panggil fungsi `pangkatDua` dengan parameter ( n ).
3. Di dalam fungsi `pangkatDua`:

- Jika ( n = 0 ), kembalikan nilai 1 sebagai kondisi ( 2^0 = 1).

- Jika ( n > 0 ), kembalikan hasil perkalian 2 dengan hasil dari pemanggilan `pangkatDua(n - 1)`.

4. Tampilkan hasil perhitungan dari `pangkatDua(n)`.

* Cara Kerja:

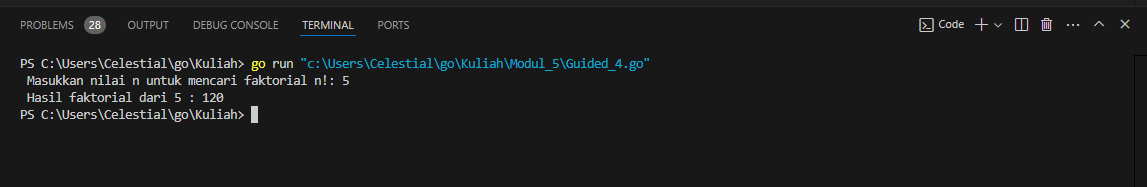
1. Program meminta input dari pengguna untuk menentukan pangkat yang ingin dihitung.
2. Fungsi `pangkatDua` dipanggil dan bekerja secara rekursif untuk menghitung ( 2^n ).
3. Setiap pemanggilan fungsi mengurangi nilai ( n ) sebanyak 1 hingga mencapai nilai (n = 0).
4. Saat ( n = 0 ), fungsi mengembalikan 1 sebagai dasar perhitungan.
5. Nilai 2 dikalikan secara bertahap pada setiap pemanggilan balik (return) hingga mencapai hasil akhir, yang kemudian ditampilkan di layar.
6. Mencari nilai faktorial atau n!

Base-case: n == 0 atau n == 1

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  // fungsi untuk menghitung faktorial n!  func faktorial(n int) int {  if n == 0 || n == 1 {  return 1  }  return n \* faktorial(n-1)  }  func main() {  var n int  fmt.Print(" Masukkan nilai n untuk mencari faktorial n!: ")  fmt.Scanln(&n)  fmt.Println(" Hasil faktorial dari", n, ":", faktorial(n))  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini menghitung faktorial dari bilangan bulat positif ( n ) menggunakan pendekatan rekursif. Pengguna diminta untuk memasukkan nilai ( n ), dan hasil faktorial akan ditampilkan .

* Algoritma:

1. Meminta pengguna untuk memasukkan nilai ( n ).
2. Panggil fungsi `faktorial` dengan parameter ( n ).
3. Dalam fungsi `faktorial`:

- Jika ( n ) adalah 0 atau 1, kembalikan 1 karena ( 0! = 1||1! = 1 ).

- Jika ( n ) lebih dari 1, kembalikan hasil perkalian ( n ) dengan pemanggilan fungsi `faktorial` untuk ( n - 1 ).

4. Tampilkan hasil dari `faktorial(n)`.

* Cara Kerja:

1. Program dimulai dengan meminta pengguna untuk memasukkan nilai ( n ).
2. Fungsi `faktorial` digunakan untuk menghitung faktorial ( n ) secara rekursif.
3. Setiap pemanggilan fungsi mengurangi nilai ( n ) hingga mencapai 0 atau 1, yang merupakan kondisi dasar.
4. Ketika kondisi dasar tercapai, fungsi mengembalikan nilai 1.
5. Proses perkalian dilanjutkan dalam setiap pemanggilan, hingga semua nilai ( n ) dikalikan, menghasilkan faktorial yang diinginkan.
6. Hasil akhir dari faktorial ( n ) ditampilkan kepada pengguna.
7. **UNGUIDED**

**Soal Studi Case**

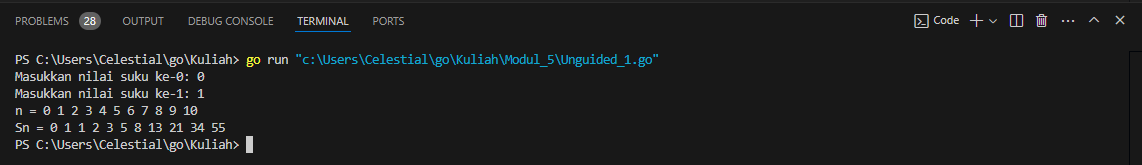
1. Deret fibonacci adalah sebuah deret dengan nilai suku ke-O dan ke-1 adalah 0 dan 1, dan nilai suku ke-n selanjutnya adalah hasil penjumlahan dua suku sebelumnya. Secara umum dapat diformulasikan Sn=Sn-1+Sn-2. Berikut ini adalah contoh nilai deret fibonacci hingga suku ke-10.

Buatlah program yang mengimplementasikan fungsi rekursif pada deret fibonacci tersebut.

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func fibonacci(suku0, suku1, n int) int {  if n == 0 {  return suku0  } else if n == 1 {  return suku1  } else {  return fibonacci(suku1, suku0+suku1, n-1)  }  }  func main() {  var suku0, suku1 int  fmt.Print("Masukkan nilai suku ke-0: ")  fmt.Scanln(&suku0)  fmt.Print("Masukkan nilai suku ke-1: ")  fmt.Scanln(&suku1)  fmt.Print("n = ")  for n := 0; n <= 10; n++ {  fmt.Print(n, " ")  }  fmt.Println()  fmt.Print("Sn = ")  for n := 0; n <= 10; n++ {  fmt.Print(fibonacci(suku0, suku1, n), " ")  }  fmt.Println()  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini menghitung deret Fibonacci dengan menggunakan rekursi, di mana pengguna dapat menentukan nilai untuk dua suku pertama dari deret. Program ini menampilkan sepuluh suku pertama dari deret Fibonacci berdasarkan nilai yang dimasukkan.

* Algoritma:

1. Ambil nilai untuk suku ke-0 (`suku0`) dan suku ke-1 (`suku1`) dari pengguna.

2. Definisikan fungsi `fibonacci` yang:

- Mengembalikan `suku0` jika ( n = 0 ).

- Mengembalikan `suku1` jika ( n = 1 ).

- Menghitung suku Fibonacci berikutnya untuk ( n > 1 ) dengan memanggil `fibonacci` secara rekursif.

3. Untuk ( n ) dari 0 hingga 10, cetak nilai `n` dan hasil dari `fibonacci(suku0, suku1, n)`.

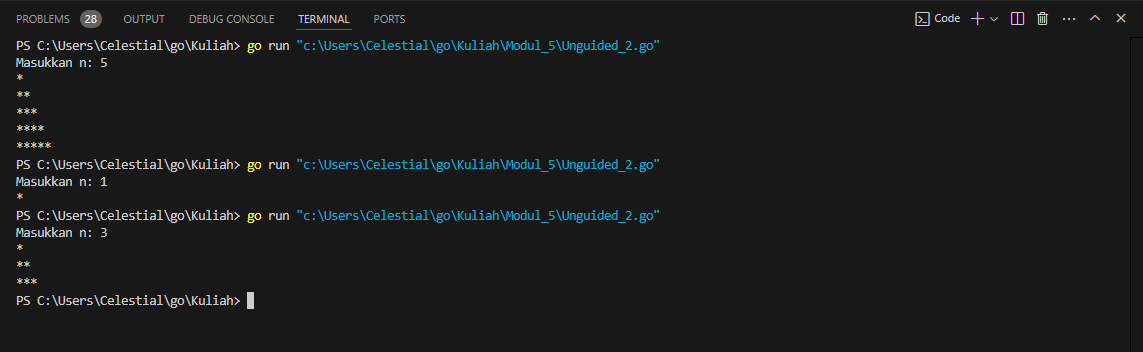
* Cara Kerja:

1. Program meminta pengguna untuk memasukkan nilai suku ke-0 dan suku ke-1 dari deret Fibonacci.
2. Fungsi `fibonacci` dipanggil dengan parameter yang sesuai untuk menghitung nilai suku Fibonacci untuk setiap ( n ) dari 0 hingga 10.
3. Setiap pemanggilan fungsi `fibonacci` memeriksa kondisi ( n ) untuk menentukan suku yang harus dikembalikan.
4. Hasil setiap perhitungan suku Fibonacci ditampilkan di layar dalam urutan dari 0 hingga 10.
5. Program memberikan keluaran yang menunjukkan suku deret Fibonacci sesuai dengan nilai awal yang diberikan pengguna.
6. Buatlah sebuah program yang digunakan untuk menampilkan pola bintang berikut ini dengan menggunakan fungsi rekursif. N adalah masukan dari user.

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func pola(i, n int, s string) {  if i <= n {  s = s + "\*"  fmt.Println(s)  pola(i+1, n, s)  }  }  func main() {  var n int  fmt.Print("Masukkan n: ")  fmt.Scanln(&n)  pola(1, n, "")  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini mencetak pola bintang (\*) dengan jumlah baris yang ditentukan oleh pengguna. Pengguna memasukkan nilai ( n ), dan program akan menampilkan pola bintang yang terdiri dari ( n ) baris, di mana setiap baris menambahkan satu bintang dari baris sebelumnya.

* Algoritma:

1. Minta pengguna untuk memasukkan nilai ( n ).

2. Definisikan fungsi `pola` yang:

* Menerima tiga parameter: `i` (indeks ), `n` (jumlah baris), dan `s` (string yang berisi bintang).
* Jika ( i ) kurang dari atau sama dengan ( n ):

- Tambahkan bintang ke string `s`.

- Cetak string `s`.

- Panggil `pola` secara rekursif dengan parameter ( i+1 ), ( n ), dan `s`.

3. Panggil fungsi `pola` dengan nilai awal ( i = 1 ), ( n ) dari input pengguna, dan string kosong.

* Cara Kerja:

1. Program dimulai dengan meminta input ( n ) dari pengguna untuk menentukan berapa banyak baris bintang yang akan dicetak.
2. Fungsi `pola` dipanggil dengan nilai awal.
3. Setiap kali fungsi dipanggil, sebuah bintang ditambahkan ke string, dan string tersebut dicetak.
4. Proses ini berlanjut hingga ( i ) lebih besar dari ( n ), di mana fungsi akan berhenti memanggil dirinya sendiri.
5. Hasil akhirnya adalah pola bintang yang terlihat bertambah satu bintang di setiap baris, dimulai dari satu bintang hingga ( n ) bintang pada baris terakhir.
6. Buatlah program yang mengimplementasikan rekusif untuk menampilkan faktor bilangan dari suatu N stau bilangan yang apa saja yang habis membagi N

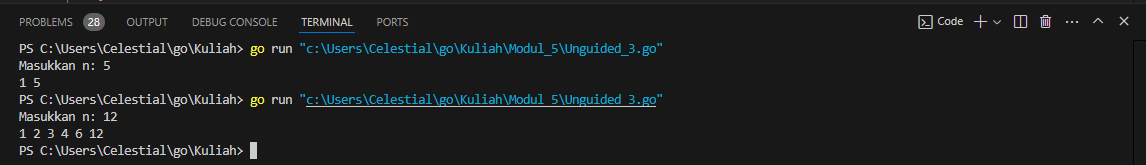
Masukan terdiri dari sebuah bilangan bulat positif N

Keluaran tardiri dari barisan bilangan yang menjadi faktor dart N (terurut dari 3 hingga Nyu),

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func faktor(n int, i int) {  if i > n {  return  }  if n%i == 0 {  fmt.Print(i, " ")  }  faktor(n, i+1)  }  func main() {  var n int  fmt.Print("Masukkan n: ")  fmt.Scanln(&n)  faktor(n, 1)  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini mencetak faktor dari bilangan bulat positif ( n ) menggunakan rekursif. Pengguna memasukkan nilai ( n ), dan program akan menampilkan semua faktor dari ( n ) mulai dari 1 hingga ( n ).

* Algoritma:

1. Minta pengguna untuk memasukkan nilai ( n ).

2. Definisikan fungsi `faktor` yang:

- Menerima dua parameter: ( n ) (bilangan positif) dan ( i ) (indeks ).

- Jika ( i ) lebih besar dari ( n ), fungsi berhenti.

- Jika ( n ) dapat dibagi habis oleh ( i ) artinya ( n % i == 0 ), cetak ( i ) sebagai faktor.

- Panggil fungsi `faktor` secara rekursif dengan ( n ) dan ( i + 1 ).

3. Panggil fungsi `faktor` dengan nilai awal ( i = 1 ).

* Cara Kerja:

1. Program dimulai dengan meminta input ( n ) dari pengguna.
2. Fungsi `faktor` dipanggil dengan ( n ) dan ( i ) diatur ke 1.
3. Untuk setiap nilai ( i ), fungsi memeriksa apakah ( i ) adalah faktor dari ( n ).
4. Jika ( i ) adalah faktor, maka ( i ) dicetak.
5. Fungsi kemudian dipanggil kembali dengan ( i ) yang ditambah satu, sehingga proses berlanjut hingga ( i ) melebihi ( n ).
6. Pada akhir proses, semua faktor dari ( n ) akan dicetak ke layar, dipisahkan oleh spasi.
7. Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan tertentu

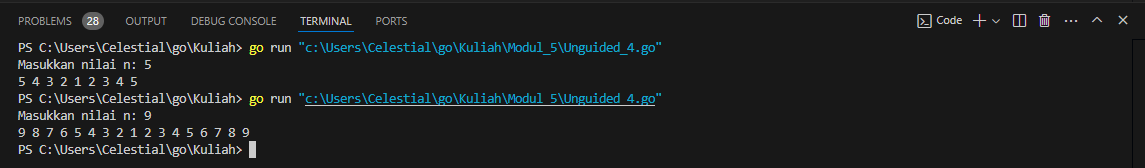
Masukan terdiri dari sebuah bilangan bulat positif N.

Kaluaran terdiri dari barisan bilangan dari N hingga i dan kembali ke N.

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func printDescending(n int) {  if n < 1 {  return  }  fmt.Print(n, " ")  printDescending(n - 1)  }  func printAscending(start, n int) {  if start > n {  return  }  fmt.Print(start, " ")  printAscending(start+1, n)  }  func printPattern(n int) {  printDescending(n)  printAscending(2, n)  fmt.Println()  }  func main() {  var n int  fmt.Print("Masukkan nilai n: ")  fmt.Scanln(&n)  printPattern(n)  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini mencetak pola angka yang dimulai dengan menampilkan angka secara menurun dari ( n ) hingga 1, kemudian melanjutkan dengan menampilkan angka secara meningkat dari 2 hingga ( n ). Pengguna memasukkan nilai ( n ), dan program akan menampilkan pola angka tersebut.

* Algoritma:

1. Minta pengguna untuk memasukkan nilai ( n ).

2. Definisikan fungsi `printDescending(n int)` yang:

- Mencetak ( n ) jika lebih besar atau sama dengan 1.

- Memanggil dirinya sendiri dengan parameter ( n-1 ) untuk mencetak angka berikutnya.

3. Definisikan fungsi `printAscending(start int, n int)` yang:

- Mencetak angka mulai dari `start` hingga ( n ).

- Memanggil dirinya sendiri dengan parameter `start+1` untuk mencetak angka berikutnya.

4. Definisikan fungsi `printPattern(n int)` yang:

- Memanggil `printDescending(n)` untuk mencetak angka menurun.

- Memanggil `printAscending(2, n)` untuk mencetak angka meningkat.

5. Panggil `printPattern(n)` dari fungsi `main()`.

* Cara Kerja:

1. Program dimulai dengan meminta input ( n ) dari pengguna.
2. Fungsi `printPattern` dipanggil dengan nilai ( n ).
3. Fungsi `printDescending` mencetak angka dari ( n ) hingga 1 dengan memanggil dirinya sendiri secara rekursif hingga mencapai angka kurang dari 1.
4. Setelah selesai mencetak angka menurun, fungsi `printAscending` dipanggil untuk mencetak angka dari 2 hingga ( n ), juga menggunakan rekursi.
5. Setelah kedua pola angka dicetak, hasilnya ditampilkan di layar dalam satu baris, memisahkan angka dengan spasi.
6. Buatlah program yang mengimplementasikan rekursif untuk menampilkan barisan bilangan ganjil.

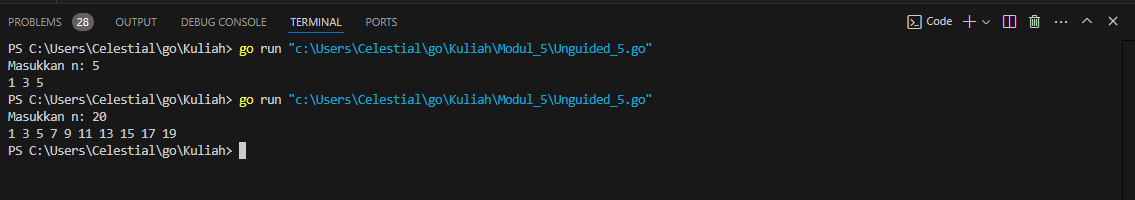
Masukan terdiri dari sebuah bilangan bulat positif N.

Keluaran terdiri dari barisan bilangan ganjil dari 1 hingga N.

**Sourcecode**

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

**Screenshoot Output**



**Deskripsi Program**

Program ini mencetak semua bilangan bulat ganjil dari 1 hingga ( n ) menggunakan pendekatan rekursif. Pengguna memasukkan nilai ( n ), dan program akan menampilkan semua bilangan ganjil yang tidak melebihi ( n ).

* Algoritma:

1. Minta pengguna untuk memasukkan nilai ( n ).

2. Definisikan fungsi `ganjil(n int)` yang:

- Menghentikan eksekusi jika ( n < 1 ).

- Jika ( n ) adalah bilangan ganjil ( n % 2 != 0 ), panggil `ganjil (n - 2)` untuk mendapatkan bilangan ganjil sebelumnya, kemudian cetak ( n ).

- Jika ( n ) adalah bilangan genap, panggil `ganjil(n - 1)` untuk memeriksa bilangan ganjil sebelumnya

* Cara Kerja:

1. Meminta input ( n ) dari pengguna untuk menentukan batas atas pencetakan bilangan ganjil.
2. Fungsi `ganjil` dipanggil dengan parameter ( n ).
3. Jika ( n ) adalah bilangan ganjil, fungsi akan memanggil dirinya sendiri dengan ( n - 2 ) untuk mencari dan mencetak bilangan ganjil hingga mencapai 1.
4. Jika ( n ) adalah bilangan genap, fungsi akan memanggil dirinya sendiri dengan ( n - 1 ) untuk menemukan bilangan ganjil yang lebih kecil.
5. Setelah seluruh proses rekursif selesai, bilangan ganjil yang ditemukan dicetak , dengan urutan yang mengikuti dari yang terkecil hingga yang terbesar.
6. Buatlah program yang mengimplementasikan rekursif untuk mencari hasil I pangkat dari dua buah bilangan.

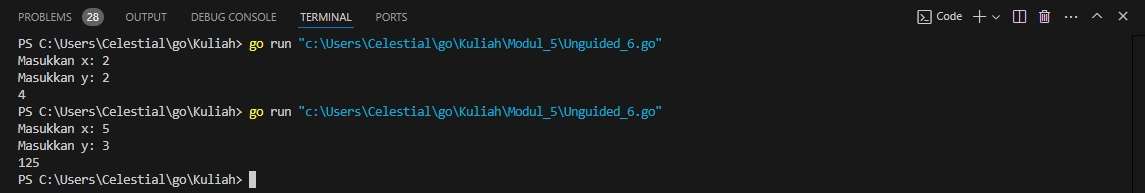
Masukan terdiri dari bilangan bulat x dan y.

Keluaran terdiri dari hasil x dipangkatkan y.

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func pangkat(x, y int) int {  if y == 0 {  return 1  } else if y == 1 {  return x  } else {  return x \* pangkat(x, y-1)  }  }  func main() {  var x, y int  fmt.Print("Masukkan x: ")  fmt.Scanln(&x)  fmt.Print("Masukkan y: ")  fmt.Scanln(&y)  fmt.Println(pangkat(x, y))  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini menghitung nilai pangkat ( x^y ) menggunakan rekursi. Pengguna memasukkan nilai basis ( x ) dan eksponen ( y ), kemudian program akan menampilkan hasil dari ( x^y ).

* Algoritma:

1. Minta pengguna untuk memasukkan nilai ( x ) dan ( y ).

2. Definisikan fungsi `pangkat(x int, y int)` yang:

- Jika ( y = 0 ), kembalikan 1 karena ( x^0 = 1 ) untuk semua ( x ).

- Jika ( y = 1 ), kembalikan ( x ) karena ( x^1 = x ).

- Jika ( y > 1 ), kembalikan hasil perkalian ( x ) dengan `pangkat (x, y-1)` untuk menghitung pangkat secara rekursif.

3. Cetak hasil dari `pangkat(x, y)`.

* Cara Kerja:

1. Program meminta input nilai ( x ) dan ( y ) dari pengguna.
2. Fungsi `pangkat` dipanggil dengan parameter ( x ) dan ( y ).
3. Jika ( y = 0 ), fungsi mengembalikan nilai 1 sebagai hasil akhir.
4. Jika ( y = 1 ), fungsi mengembalikan nilai ( x ) sebagai hasil akhir.
5. Jika ( y > 1 ), fungsi memanggil dirinya sendiri dengan ( y - 1 ) hingga mencapai kondisi dasar, yaitu ( y = 0 ) atau ( y = 1 ).
6. Pada setiap pemanggilan balik, hasil perkalian ( x ) dikalikan dengan hasil pemanggilan sebelumnya, sehingga menghasilkan ( x^y ).
7. Hasil akhir ditampilkan di layar sebagai output dari perhitungan pangkat.