**LAPORAN PRAKTIKUM   
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL X1**

**PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATASebuah gambar berisi logo, simbol, Grafis, Font

Deskripsi dibuat secara otomatis**

**Disusun Oleh :**

**Didik Setiawan**

**IF 11 06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi,S.Kom., M.Kom**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

1. **DASAR TEORI**

Dasar Teori

encarian nilai ekstrem (nilai minimum dan maksimum) adalah operasi penting dalam algoritma pemrosesan data. Tujuannya adalah untuk menemukan nilai terkecil (minimum) dan terbesar (maksimum) dalam suatu kumpulan data. Dalam konteks pemrograman Golang, operasi ini melibatkan iterasi melalui elemen-elemen dalam slice atau array dan membandingkan setiap elemen untuk memperbarui nilai ekstrem

1. **GUIDED**
   * + 1. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan. Source Code diberi penjelasan maka akan menjadi nilai ++

**Soal Studi Case**

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  // Mendeklarasikan tipe data array arrInt dengan panjang 2023  type arrInt [2023]int  // Fungsi untuk mencari indeks elemen terkecil dalam array  func terkecil(tabInt arrInt, n int) int {      var idx int = 0 // idx menyimpan indeks elemen terkecil      var j int = 1      for j < n {          if tabInt[idx] > tabInt[j] {              idx = j // Simpan indeks j jika elemen di indeks j lebih kecil          }          j = j + 1      }      return idx  }  // Fungsi main untuk menguji fungsi terkecil  func main() {      var n int      var tab arrInt      // Meminta input jumlah elemen array      fmt.Print("Masukkan jumlah elemen (maks 2023): ")      fmt.Scan(&n)      // Validasi input jumlah elemen      if n < 1 || n > 2023 {          fmt.Println("Jumlah elemen harus antara 1 dan 2023.")          return      }      // Memasukkan elemen-elemen array      fmt.Println("Masukkan elemen-elemen array:")      for i := 0; i < n; i++ {          fmt.Print("Elemen ke-", i+1, ": ")          fmt.Scan(&tab[i])      }      // Memanggil fungsi terkecil untuk menemukan indeks elemen terkecil      idxMin := terkecil(tab, n)      // Menampilkan nilai dan indeks terkecil      fmt.Println("Nilai terkecil dalam array adalah:", tab[idxMin], "pada indeks:", idxMin)  } |

**Screenshoot Output**

**Deskripsi Program**

User memasukkan jumlah elemen (dengan batas 1 hingga 2023) dan nilai-nilai array tersebut. Fungsi **terkecil** akan menentukan indeks elemen dengan nilai terkecil, yang kemudian ditampilkan bersama nilainya.

* + - 1. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan. Source Code diberi penjelasan maka akan menjadi nilai ++

**Soal Studi Case**

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  // Definisi struct mahasiswa dengan atribut nama, nim, kelas, jurusan, dan ipk  type mahasiswa struct {      nama, nim, kelas, jurusan string      ipk                       float64  }  // Definisi tipe data array mahasiswa dengan kapasitas maksimal 2023  type arrMhs [2023]mahasiswa  // Fungsi untuk mencari IPK tertinggi dalam array mahasiswa  func ipk(T arrMhs, n int) float64 {      var tertinggi float64 = T[0].ipk      var j int = 1      for j < n {          if tertinggi < T[j].ipk {              tertinggi = T[j].ipk          }          j = j + 1      }      return tertinggi  }  // Fungsi main untuk mengisi data mahasiswa dan mencari IPK tertinggi  func main() {      var n int      var dataMhs arrMhs      // Meminta input jumlah mahasiswa      fmt.Print("Masukkan jumlah mahasiswa (maks 2023): ")      fmt.Scan(&n)      // Validasi jumlah mahasiswa yang dimasukkan      if n < 1 || n > 2023 {          fmt.Println("Jumlah mahasiswa harus antara 1 dan 2023.")          return      }      // Mengisi data mahasiswa      for i := 0; i < n; i++ {          fmt.Printf("\nMasukkan data mahasiswa ke-%d\n", i+1)          fmt.Print("Nama: ")          fmt.Scan(&dataMhs[i].nama)          fmt.Print("NIM: ")          fmt.Scan(&dataMhs[i].nim)          fmt.Print("Kelas: ")          fmt.Scan(&dataMhs[i].kelas)          fmt.Print("Jurusan: ")          fmt.Scan(&dataMhs[i].jurusan)          fmt.Print("IPK: ")          fmt.Scan(&dataMhs[i].ipk)      }      // Mencari dan menampilkan IPK tertinggi      tertinggi := ipk(dataMhs, n)      fmt.Printf("\nIPK tertinggi dari %d mahasiswa adalah: %.2f\n", n, tertinggi)  } |

**Screenshoot Output**

**A screenshot of a computer

Description automatically generated**

**Deskripsi Program**

mendefinisikan struktur data untuk menyimpan informasi mahasiswa, termasuk nama, NIM, kelas, jurusan, dan IPK, serta menyediakan fungsi untuk mencari IPK tertinggi di antara sekumpulan data mahasiswa. Pengguna diminta memasukkan jumlah mahasiswa dan mengisi data masing-masing mahasiswa. Program kemudian menghitung dan menampilkan IPK tertinggi dari data yang dimasukkan, membantu dalam menganalisis performa akademik mahasiswa secara sederhana.

1. **UNGUIDED**
   * + 1. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan. Source Code diberi penjelasan maka akan menjadi nilai ++

**Soal Studi Case**

**Sourcecode**

|  |
| --- |
| package main  import (      "fmt"  )  func main() {      var N int      // Membaca jumlah anak kelinci      fmt.Print("Masukkan jumlah anak kelinci (N): ")      fmt.Scan(&N)      // Validasi jumlah anak kelinci      if N <= 0 || N > 1000 {          fmt.Println("Jumlah anak kelinci harus antara 1 dan 1000.")          return      }      // Inisialisasi berat terkecil dan terbesar      var berat, terkecil, terbesar float64      fmt.Println("Masukkan berat anak kelinci:")      for i := 0; i < N; i++ {          fmt.Printf("Berat ke-%d: ", i+1)          fmt.Scan(&berat)          // Validasi berat positif          if berat <= 0 {              fmt.Println("Berat harus angka positif.")              return          }          // Inisialisasi nilai pertama sebagai acuan          if i == 0 {              terkecil, terbesar = berat, berat          } else {              if berat < terkecil {                  terkecil = berat              }              if berat > terbesar {                  terbesar = berat              }          }      }      // Output hasil      fmt.Printf("Berat terkecil: %.2f\n", terkecil)      fmt.Printf("Berat terbesar: %.2f\n", terbesar)  } |

**Screenshoot Output**

**A screen shot of a computer

Description automatically generated**

**Deskripsi Program**

User memasukkan jumlah anak kelinci (dengan batas 1 hingga 1000) dan berat masing-masing anak kelinci. Program memvalidasi bahwa semua berat yang dimasukkan adalah angka positif. Hasil akhirnya adalah berat terkecil dan terbesar, yang ditampilkan dengan format dua angka desimal

2. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan. Source Code diberi penjelasan maka akan menjadi nilai ++

**Soal Studi Case**

**Sourcecode**

|  |
| --- |
| package main  import (      "fmt"      "math"  )  func main() {      // Masukkan jumlah ikan (x) dan jumlah ikan per wadah (y)      var x, y int      fmt.Print("Masukkan jumlah ikan (x) dan jumlah ikan per wadah (y): ")      fmt.Scan(&x, &y)      // Validasi input      if x <= 0 || y <= 0 || x > 1000 {          fmt.Println("Input tidak valid. x harus > 0, y harus > 0, dan x <= 1000.")          return      }      // Masukkan berat ikan      beratIkan := make([]float64, x)      fmt.Println("Masukkan berat ikan (pisahkan dengan spasi):")      for i := 0; i < x; i++ {          fmt.Scan(&beratIkan[i])          if beratIkan[i] <= 0 {              fmt.Println("Berat ikan harus bernilai positif.")              return          }      }      // Menghitung jumlah wadah      jumlahWadah := int(math.Ceil(float64(x) / float64(y)))      totalBeratPerWadah := make([]float64, jumlahWadah)      // Distribusi ikan ke wadah      for i := 0; i < x; i++ {          indexWadah := i / y          totalBeratPerWadah[indexWadah] += beratIkan[i]      }      // Menampilkan total berat di setiap wadah      fmt.Println("Total berat di setiap wadah:")      for i, total := range totalBeratPerWadah {          fmt.Printf("Wadah %d: %.2f\n", i+1, total)      }      // Menghitung dan menampilkan rata-rata berat per wadah      totalBeratKeseluruhan := 0.0      for \_, total := range totalBeratPerWadah {          totalBeratKeseluruhan += total      }      rataRataBerat := totalBeratKeseluruhan / float64(jumlahWadah)      fmt.Printf("Rata-rata berat per wadah: %.2f\n", rataRataBerat)  } |

**Screenshoot Output**

**A screen shot of a computer

Description automatically generated**

**Deskripsi Program**

user memasukkan jumlah ikan (x), jumlah maksimum ikan per wadah (y), dan berat masing-masing ikan. Program memvalidasi input agar sesuai dengan ketentuan (positif dan dalam batasan tertentu). Ikan didistribusikan secara berurutan ke wadah, dan total berat setiap wadah dihitung serta ditampilkan. Selain itu, program menghitung rata-rata berat per wadah untuk memberikan gambaran distribusi berat.

1. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan. Source Code diberi penjelasan maka akan menjadi nilai ++

**Soal Studi Case**

**Sourcecode**

|  |
| --- |
| package main  import (      "fmt"  )  // hitungStatistik menghitung berat minimum, maksimum, dan rata-rata  func hitungStatistik(arrBerat []float64) (float64, float64, float64) {      if len(arrBerat) == 0 {          return 0, 0, 0      }      bMin := arrBerat[0]      bMax := arrBerat[0]      total := 0.0      for \_, berat := range arrBerat {          if berat < bMin {              bMin = berat          }          if berat > bMax {              bMax = berat          }          total += berat      }      rata := total / float64(len(arrBerat))      return bMin, bMax, rata  }  func main() {      var n int      // Input jumlah data balita      fmt.Print("Masukkan banyak data berat balita: ")      fmt.Scan(&n)      // Validasi jumlah balita      if n <= 0 {          fmt.Println("Jumlah data balita harus lebih dari 0.")          return      }      // Input berat masing-masing balita      arrBerat := make([]float64, n)      for i := 0; i < n; i++ {          fmt.Printf("Masukkan berat balita ke-%d: ", i+1)          fmt.Scan(&arrBerat[i])      }      // Hitung statistik      bMin, bMax, rata := hitungStatistik(arrBerat)      // Output hasil      fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)      fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)      fmt.Printf("Rerata berat balita: %.2f kg\n", rata)  } |

**Screenshoot Output**

**A screen shot of a computer

Description automatically generated**

**Deskripsi Program**

User memasukkan jumlah data balita dan berat masing-masing balita. Program memvalidasi bahwa jumlah data lebih dari 0 dan menghitung statistik menggunakan fungsi khusus. Hasil berupa berat minimum, maksimum, dan rata-rata ditampilkan dalam format desimal dua angka.