

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 7
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA**



Oleh: Dimas Fanny Hebrasianto Permadi

NAMA: Dimas Ramadhani

NIM: 103112400065

KELAS: IF-12-01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2025

I. DASAR TEORI

1. Ide Pencarian Nilai Max/Min

Pencarian merupakan proses umum dalam kehidupan sehari-hari, seperti mencari file di komputer atau mencari buku di rak. Dalam konteks algoritma dan pemrograman, pencarian nilai ekstrem berarti menemukan nilai maksimum atau minimum dari sekumpulan data.

Cara kerjanya:

- Ambil nilai pertama dari data sebagai nilai awal.
- Bandingkan satu persatu dengan data berikutnya.
- Kalau ditemukan nilai yang lebih besar atau kecil, nilai awal tadi diganti dengan nilai baru tersebut.
- Setelah semua data dibandingkan, kita akan mendapatkan nilai ekstrem yang sebenarnya.

2. Pencarian Nilai Ekstrem di Array Bertipe Data Dasar

Pada array bertipe dasar seperti integer atau float, pencarian nilai ekstrem dilakukan dengan cara iterasi satu persatu terhadap seluruh elemen array. Dalam implementasinya, nilai minimum atau maksimum diperbarui jika ditemukan elemen yang lebih kecil atau lebih besar dari nilai saat ini.

3. Pencarian Nilai Ekstrem di Array Bertipe Data Terstruktur

Kalau data yang kita cari lebih kompleks, contohnya data mahasiswa yang terdiri dari nama, NIM, jurusan, dan IPK. Kita tetap bisa melakukan pencarian nilai ekstrem. Misalnya, kita ingin mencari mahasiswa dengan IPK tertinggi.

I. GUIDED

1. Nomor 1

- Source Code:

```
package main

import (
    "fmt"
)

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

    var tinggiTanaman [500]float64
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan tinggi tanaman ke-%d (cm): ", i+1)
        fmt.Scan(&tinggiTanaman[i])
    }

    min, max := tinggiTanaman[0], tinggiTanaman[0]
    for i := 1; i < n; i++ {
        if tinggiTanaman[i] < min {
            min = tinggiTanaman[i]
        }
        if tinggiTanaman[i] > max {
            max = tinggiTanaman[i]
        }
    }

    fmt.Printf("\nTinggi tanaman tertinggi: %.2f cm\n", max)
    fmt.Printf("Tinggi tanaman terpendek: %.2f cm\n", min)
    fmt.Printf("Array %.2f", tinggiTanaman[:n])
}
```

- **Screenshot Hasil Program:**

```
PS C:\Users\Dimas\OneDrive\Collage\Semester 2\Algoritma dan Pemrograman 2\Praktikum 1\
Masukkan jumlah tanaman: 4
Masukkan tinggi tanaman ke-1 (cm): 4
Masukkan tinggi tanaman ke-2 (cm): 2
Masukkan tinggi tanaman ke-3 (cm): 5
Masukkan tinggi tanaman ke-4 (cm): 7

Tinggi tanaman tertinggi: 7.00 cm
Tinggi tanaman terpendek: 2.00 cm
Array [4.00 2.00 5.00 7.00]
```

- **Penjelasan:**

Program ini bertujuan untuk membaca sejumlah tinggi tanaman yang diinputkan oleh pengguna, kemudian mencari dan menampilkan tinggi tanaman tertinggi dan terpendek. Selain itu, program juga menampilkan semua data tinggi tanaman yang telah diinputkan dalam bentuk array.

Deklarasi program, `n` bertipe `int` digunakan untuk menyimpan jumlah tanaman yang akan diinput oleh pengguna. `tinggiTanaman` bertipe array dengan 500 elemen `float64` untuk menyimpan data tinggi masing-masing tanaman. Pertama program meminta pengguna untuk memasukkan jumlah tanaman yang akan diinput, lalu membaca nilainya ke variabel `n`. Melalui perulangan `for` dari `i = 0` hingga `i < n`, program meminta input tinggi tanaman satu per satu dan menyimpannya ke dalam array `tinggiTanaman`. Untuk mencari nilai maksimum dan minimum, pertama variabel `min` dan `max` diinisialisasi dengan nilai dari `tinggiTanaman[0]` (tinggi tanaman pertama). Program melakukan perulangan dari `i = 1` hingga `i < n`, untuk membandingkan setiap tinggi tanaman dengan `min` dan `max`. Jika ditemukan nilai yang lebih kecil dari `min`, maka `min` diperbarui. Jika ditemukan nilai yang lebih besar dari `max`, maka `max` diperbarui.

Setelah proses selesai, program mencetak tinggi tanaman (`max`) dengan format dua angka di belakang koma. Lalu tinggi tanaman terpendek (`min`) dengan format dua angka di belakang koma. Lalu

semua data tinggi tanaman yang telah diinputkan dalam bentuk array, hanya menampilkan elemen dari indeks ke-0 hingga ke-n-1.

2. Nomor 2

- Source Code:

```
package main

import (
    "fmt"
)

func main() {
    var x, y int
    fmt.Print("Masukkan jumlah buku dan jumlah buku per rak: ")
    fmt.Scan(&x, &y)

    var hargaBuku [500]float64
    fmt.Println("\nMasukkan harga setiap buku (dalam ribuan Rp):")
    for i := 0; i < x; i++ {
        fmt.Scan(&hargaBuku[i])
    }

    var hargaRatarata []float64
    for i := 0; i < x; i++ {
        total := 0.0
        for j := i; j < i+y && j < x; j++ {
            total += hargaBuku[j]
        }
        hargaRatarata = append(hargaRatarata, total/float64(y))
    }

    min, max := hargaBuku[0], hargaBuku[0]
    for _, harga := range hargaBuku[:x] {
        if harga < min {
            min = harga
        }
        if harga > max {
            max = harga
        }
    }

    fmt.Printf("\nRata-rata harga per rak: ")
    for _, avg := range hargaRatarata {
        fmt.Printf("%.2f ", avg)
    }
}
```

```

    }
    fmt.Printf("\nHarga termahal: Rp %.2f\n", max)
    fmt.Printf("Harga termurah: Rp %.2f\n", min)
}

```

- **Screenshot Hasil Program:**

```

Masukkan jumlah buku dan jumlah buku per rak: 5 2

Masukkan harga setiap buku (dalam ribuan Rp):
10000
25000
77000
50000
99000

Rata-rata harga per rak: 17500.00 51000.00 63500.00 74500.00 49500.00
Harga termahal: Rp 99000.00
Harga termurah: Rp 10000.00

```

- **Penjelasan:**

Program ini bertujuan untuk membaca harga dari sejumlah buku, menghitung rata-rata harga buku untuk setiap rak (dengan jumlah buku per rak ditentukan pengguna), serta menampilkan harga buku termahal dan termurah.

Deklarasi variabel `x` bertipe `int` untuk menyimpan jumlah buku yang akan diinput, lalu `y` bertipe `int` untuk menyimpan jumlah buku per rak, `hargaBuku` adalah array berukuran 500 bertipe `float64`, digunakan untuk menyimpan harga masing-masing buku. Lalu `hargaRataRata` adalah slice bertipe `float64`, digunakan untuk menyimpan hasil rata-rata harga buku per rak.

Program meminta pengguna untuk memasukkan dua angka: jumlah buku (`x`) dan jumlah buku per rak (`y`), lalu membaca input tersebut. Lalu program meminta pengguna memasukkan harga setiap buku satu per satu, lalu menyimpan harga tersebut ke dalam array `hargaBuku`.

Program menggunakan perulangan `for`, program menghitung rata-rata harga untuk setiap kelompok `y` buku. Untuk setiap iterasi `i`, dihitung total dari harga buku mulai dari indeks `i` hingga `i+y-1`, asalkan `j` masih dalam batas jumlah buku `j < x`. Hasil rata-rata (total dibagi jumlah buku per rak) disimpan ke dalam slice `hargaRatarata`.

Untuk mencari harga termahal dan termurah, program melakukan inisialisasi min dan max dengan harga buku pertama. Lalu melalui perulangan for dengan range array hargaBuku[:x], dibandingkan setiap harga untuk memperbarui min dan max sesuai dengan harga termurah dan termahal.

Program mencetak semua rata-rata harga per rak dengan format dua angka di belakang koma, dipisahkan oleh spasi. Program juga menampilkan harga buku termahal dan termurah dengan format dua angka di belakang koma.

3. Nomor 3

- Source Code:

```
package main

import (
    "fmt"
)

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

    var nilaiSiswa [200]float64
    var totalNilai float64 = 0

    fmt.Println("\nMasukkan nilai ujian masing-masing siswa:")
    for i := 0; i < n; i++ {
        fmt.Scan(&nilaiSiswa[i])
        totalNilai += nilaiSiswa[i]
    }

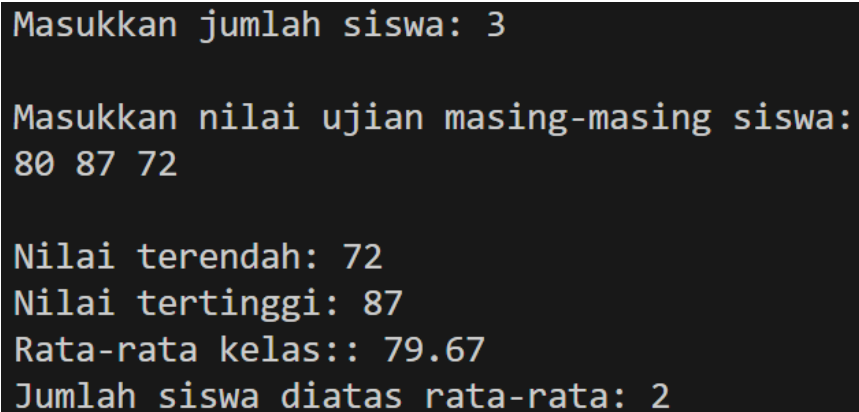
    rataRata := totalNilai / float64(n)

    min, max := nilaiSiswa[0], nilaiSiswa[0]
    var diAtasRataRata int = 0
    for _, nilai := range nilaiSiswa[:n] {
        if nilai < min {
            min = nilai
        }
        if nilai > max {
            max = nilai
        }
        if nilai > rataRata {
            diAtasRataRata++
        }
    }

    fmt.Printf("\nNilai terendah: %.0f\n", min)
    fmt.Printf("Nilai tertinggi: %.0f\n", max)
```

```
fmt.Printf("Rata-rata kelas:: %.2f\n", rataRata)
fmt.Printf("Jumlah siswa diatas rata-rata: %d\n", diAtasRataRata)
}
```

- **Screenshot Hasil Program:**



```
Masukkan jumlah siswa: 3

Masukkan nilai ujian masing-masing siswa:
80 87 72

Nilai terendah: 72
Nilai tertinggi: 87
Rata-rata kelas:: 79.67
Jumlah siswa diatas rata-rata: 2
```

- **Penjelasan:**

Program ini bertujuan untuk membaca nilai ujian dari sejumlah siswa, menghitung rata-rata nilai kelas, mencari nilai tertinggi dan terendah, serta menghitung jumlah siswa yang memiliki nilai di atas rata-rata.

Program melakukan deklarasi variabel `n` bertipe `int` untuk menyimpan jumlah siswa. Variabel `nilaiSiswa` adalah array berukuran 200 bertipe `float64`, digunakan untuk menyimpan nilai ujian masing-masing siswa. Variabel `totalNilai` bertipe `float64`, diinisialisasi dengan 0, untuk menjumlahkan semua nilai siswa. Variabel `rata-rata` bertipe `float64`, untuk menyimpan hasil perhitungan rata-rata nilai siswa. Variabel `min` dan `max` bertipe `float64`, untuk menyimpan nilai terendah dan tertinggi dari semua nilai siswa. Variabel `diAtasRataRata` bertipe `int`, digunakan untuk menghitung jumlah siswa yang nilai ujiannya lebih tinggi dari rata-rata.

Program akan meminta pengguna memasukkan jumlah siswa (`n`). Melalui perulangan `for` dari `i = 0` hingga `i < n`, program meminta input nilai ujian masing-masing siswa. Setiap nilai yang dimasukkan ditambahkan ke `totalNilai`.

Setelah semua nilai dimasukkan, rata-rata dihitung dengan membagi `totalNilai` dengan `n`. Inisialisasi `min` dan `max` dengan nilai siswa

pertama (`nilaiSiswa[0]`). Melalui perulangan `for` menggunakan `range`, program membandingkan setiap nilai, jika nilai lebih kecil dari `min` maka `min` diperbarui. Jika nilai lebih besar dari `max`, maka `max` diperbarui. Jika nilai lebih besar dari rata-rata, maka `diAtasRataRata` bertambah 1.

Program akan mencetak nilai terendah (`min`) dan tertinggi (`max`) dengan format tanpa angka di belakang koma. Rata-rata kelas dicetak dua angka di belakang koma. Jumlah siswa yang memiliki nilai di atas rata-rata juga ditampilkan.

II. UNGUIDED

1. Nomor 1

- **Source Code:**

```
package main

import "fmt"

func main() {
    var N int
    var berat [1000]float64
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&N)
    for i := 0; i < N; i++ {
        fmt.Printf("Masukkan berat anak kelinci ke-%d (kg): ", i+1)
        fmt.Scan(&berat[i])
    }

    /*
    Dimas Ramadhani
    103112400065
    */
    min, max := berat[0], berat[0]
    for i := 1; i < N; i++ {
        if berat[i] < min {
            min = berat[i]
        }
        if berat[i] > max {
            max = berat[i]
        }
    }
    fmt.Println("Berat kelinci terkecil:", min)
    fmt.Println("Berat kelinci terbesar:", max)
}
```

- **Screenshot Hasil Pembahasan:**

```
Masukkan jumlah anak kelinci: 4
Masukkan berat anak kelinci ke-1 (kg): 3
Masukkan berat anak kelinci ke-2 (kg): 2.5
Masukkan berat anak kelinci ke-3 (kg): 5
Masukkan berat anak kelinci ke-4 (kg): 3.3
Berat kelinci terkecil: 2.5
Berat kelinci terbesar: 5
```

- **Penjelasan:**

Program ini bertujuan untuk membaca berat dari sejumlah anak kelinci, lalu menentukan dan menampilkan berat anak kelinci terkecil dan terbesar.

Program melakukan deklarasi variabel N bertipe `int` untuk menyimpan jumlah anak kelinci yang akan diinput. Berat adalah array berukuran 1000 bertipe `float64`, digunakan untuk menyimpan berat masing-masing anak kelinci.

Program meminta pengguna untuk memasukkan jumlah anak kelinci (N). Melalui perulangan `for` dari $i = 0$ hingga $i < N$, program meminta pengguna memasukkan berat setiap anak kelinci. Setiap berat yang dimasukkan disimpan dalam array `berat`.

Untuk mencari berat terkecil dan terbesar, variabel `min` dan `max` diinisialisasi dengan berat anak kelinci pertama (`berat[0]`). Melalui perulangan `for` dari $i = 1$ hingga $1 < N$, program membandingkan setiap berat, jika berat lebih kecil dari `min`, maka `min` diperbarui, jika berat lebih besar dari `max`, maka `max` diperbarui.

Output program mencetak berat kelinci terkecil (`min`) dan berat anak kelinci terbesar (`max`).

2. Nomor 2

- Source Code:

```
package main

import "fmt"

func main() {
    var x, y int
    var ikan [1000]float64
    fmt.Print("Masukkan jumlah ikan (x) dan jumlah ikan per wadah (y): ")
    fmt.Scan(&x, &y)

    fmt.Print("Masukkan berat setiap ikan: ")
    for i := 0; i < x; i++ {
        fmt.Scan(&ikan[i])
    }

    banyakWadah := (x + y - 1) / y
    totalBeratWadah := make([]float64, banyakWadah)
    jumlahIkanWadah := make([]float64, banyakWadah)
    for dimasRamadhani := 0; dimasRamadhani < x;
    dimasRamadhani++ {
        posisiWadah := dimasRamadhani / y
        totalBeratWadah[posisiWadah] += ikan[dimasRamadhani]
        jumlahIkanWadah[posisiWadah]++
    }

    for i, beratWadah := range totalBeratWadah {
        fmt.Printf("| Wadah ke-%d: %.2f |", i+1, beratWadah)
    }
    // 103112400065
    fmt.Println()

    for i := 0; i < banyakWadah; i++ {
        rataRata := totalBeratWadah[i] / jumlahIkanWadah[i]
        fmt.Printf("| Berat rata-rata wadah ke-%d: %.2f |", i+1,
rataRata)
    }
}
```

- **Screenshot Hasil Pembahasan:**

```
Masukkan jumlah ikan (x) dan jumlah ikan per wadah (y): 5 3
Masukkan berat setiap ikan: 3 2 5 4 4
| Wadah ke-1: 10.00 || Wadah ke-2: 8.00 |
| Berat rata-rata wadah ke-1: 3.33 || Berat rata-rata wadah ke-2: 4.00 |
```

- **Penjelasan:**

Program ini bertujuan untuk membaca berat beberapa ekor ikan, mengelompokkan ikan ke dalam wadah dengan jumlah ikan tertentu per wadah, menghitung total berat ikan dalam setiap wadah, serta menghitung dan menampilkan rata-rata berat ikan per wadah.

Program melakukan deklarasi variabel `x` bertipe `int` untuk menyimpan jumlah total ikan. Variabel `ya` bertipe `int` digunakan untuk menyimpan jumlah ikan per wadah. Variabel `ikan` adalah array berukuran 1000 bertipe `float64` untuk menyimpan berat masing-masing ikan.

Program meminta pengguna memasukkan jumlah total ikan (`x`) dan jumlah ikan yang akan di tempatkan dalam satu wadah (`y`). Lalu program meminta pengguna untuk memasukkan berat masing-masing ikan satu per satu, dan menyimpannya ke dalam array `ikan`.

Untuk menghitung jumlah wadah dan mengelompokkan ikan menggunakan variabel `banyakWadah` dihitung menggunakan rumus $(x+y-1)/y$, supaya jika jumlah ikan tidak habis dibagi, tetap dibuatkan satu wadah tambahan untuk sisa ikan. Lalu variabel `totalBeratWadah` adalah slice bertipe `float64` untuk menyimpan jumlah ikan dalam tiap wadah.

Dengan perulangan `for`, untuk setiap ikan (menggunakan variabel `dimasramadhani`), ditentukan `posisiWadah` berdasarkan indeks ikan dibagi jumlah ikan per wadah (`dimasramadhani/y`). Lalu berat ikan ditambahkan ke total berat wadah sesuai `posisiWadah`. Dan jumlah ikan dalam wadah tersebut ditambah satu.

Program akan mencetak total berat setiap wadah dengan format dua angka di belakang koma. Dan program menghitung rata-rata berat ikan dengan membagi total berat dengan jumlah ikan di dalam wadah, lalu mencetak hasilnya.

3. Nomor 3

- Source Code:

```
package main

import "fmt"

type arrBalita [100]float64

func hitungMinMax(arrBerat arrBalita, bMin, bMax *float64) {
    *bMin, *bMax = arrBerat[0], arrBerat[0]
    for i := 0; i < len(arrBerat); i++ {
        if arrBerat[i] != 0 {
            if arrBerat[i] < *bMin {
                *bMin = arrBerat[i]
            }
            if arrBerat[i] > *bMax {
                *bMax = arrBerat[i]
            }
        }
    }
    fmt.Printf("Berat balita minimum: %.2f kg\n", *bMin)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", *bMax)
}

func rerata(arrBerat arrBalita) float64 {
    var pembagi, jumlah float64
    for i := 0; i < len(arrBerat); i++ {
        if arrBerat[i] > 0 {
            jumlah += arrBerat[i]
            pembagi++
        }
    }
    return jumlah / pembagi
}

/*
Dimas Ramadhani
103112400065
*/

func main() {
    var dataBalita arrBalita
    var min, max float64
    var n int
```

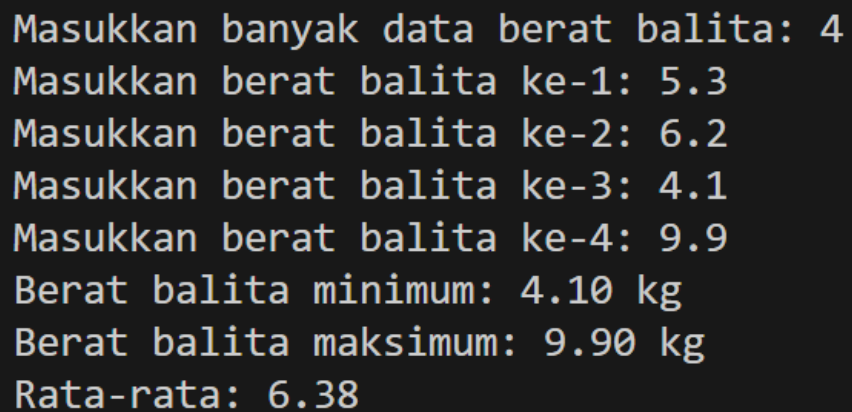


```

fmt.Print("Masukkan banyak data berat balita: ")
fmt.Scan(&n)
for i := 0; i < n; i++ {
    fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
    fmt.Scan(&dataBalita[i])
}
hitungMinMax(dataBalita, &min, &max)
fmt.Printf("Rata-rata: %.2f", rerata(dataBalita))
}

```

- **Screenshoot Hasil Pembahasan:**



```

Masukkan banyak data berat balita: 4
Masukkan berat balita ke-1: 5.3
Masukkan berat balita ke-2: 6.2
Masukkan berat balita ke-3: 4.1
Masukkan berat balita ke-4: 9.9
Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rata-rata: 6.38

```

- **Penjelasan:**

Program ini bertujuan untuk membaca data berat badan beberapa balita, menghitung berat balita minimum dan maksimum, serta menghitung rata-rata berat badan balita.

Program menggunakan tipe data array `arrBalita` berukuran 100 tipe `float64` digunakan untuk menyimpan data berat badan balita.

Fungsi `hitungMinMax` menggunakan parameter `arrBalita`, `bMin` dan `bMax` bertipe pointer ke `float64`. Fungsi ini mencari nilai berat minimum dan maksimum dari array `arrBerat`. Awalnya `bMin` dan `bMax` diset ke elemen pertama `arrBerat[0]`. Melalui perulangan `for`, fungsi mencetak seluruh elemen array dengan kondisi jika elemen tidak sama dengan 0 dan lebih kecil dari `bMin`, maka `bMin` diperbarui. Jika lebih besar dari `bMax`, maka `bmax` diperbarui. Setelah selesai, fungsi akan menampilkan berat minimum dan maksimum.

Fungsi `rerata` menggunakan parameter `arrBerat` bertipe `arrBalita`. Fungsi ini menghitung rata-rata berat badan balita. Dengan

perulangan for, fungsi menjumlahkan semua berat balita yang lebih besar dari 0 dan menghitung berapa banyak data (pembagi). Rata-rata dihitung dengan membagi total berat dengan jumlah data yang valid.

Pada program utama, melakukan deklarasi variabel dataBalita bertipe arrBalita untuk menyimpan data berat balita. Variabel min dan max bertipe float64 untuk menyimpan berat minimum dan maksimum. Variabel n bertipe int untuk menyimpan jumlah data yang akan dimasukkan. Program meminta pengguna memasukkan jumlah data berat balita (n). Dengan perulangan for dari $i = 0$ hingga $i < n$ pengguna diminta memasukkan berat balita satu per satu ke dalam array arrBalita. Lalu program memanggil fungsi hitungMinMax untuk menentukan berat minimum dan maksimum, sekaligus mencetak hasilnya. Lalu program memanggil fungsi rerata untuk menghitung rata-rata berat balita dan menampilkannya.

III. KESIMPULAN

Pada praktikum ini, telah dipelajari dan diterapkan algoritma pencarian nilai ekstrem, yaitu pencarian nilai maksimum dan minimum dalam sekumpulan data. Proses pencarian nilai ekstrem dilakukan dengan prinsip dasar membandingkan elemen-elemen data secara berurutan, lalu memperbarui nilai ekstrem jika ditemukan nilai yang lebih sesuai.

Penerapan algoritma ini mencakup dua jenis data, yaitu:

- Array bertipe data dasar, seperti bilangan bulat (integer) atau bilangan riil (float), di mana pencarian dilakukan langsung terhadap nilai.
- Array bertipe dasar terstruktur, seperti array yang berisi record mahasiswa, di mana pencarian nilai ekstrem dilakukan berdasarkan salah satu atribut dan hasil pencarian dapat berupa nilai maupun indeks dari data tersebut.

Pemahaman Konsep pencarian nilai ekstrem ini sangat penting dalam pengolahan data karena tidak hanya memungkinkan identifikasi nilai terbesar atau terkecil, tetapi juga memungkinkan pengaksesan data terkait yang lebih lengkap atau melalui indeks.

IV. REFERENSI

Prayogo, N. A. (2024). *Dasar Pemrograman Golang* (Versi 4.0.20240830). Retrieved from <https://github.com/novalagung/dasarpemrogramangolang>

Selly Meliana, S.Kom., M.Kom.(2024) *Modul Praktikum ALGORITMA DAN PEMROGRAMAN 2*.