LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL XII & XIII

PENGURUTAN DATA

Disusun Oleh :

Marsep Trianto Pakondo / 2311102251

IF-11-06

Dosen Pengampu :

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

- **Definisi Pengurutan Data**

  Pengurutan data adalah proses penyusunan elemen-elemen dalam suatu himpunan data sesuai dengan urutan tertentu, baik secara menaik (ascending) maupun menurun (descending). Pengurutan memungkinkan pengelolaan data menjadi lebih sistematis, sehingga mempermudah proses pencarian, analisis, dan manipulasi data.

  Pengurutan data bertujuan untuk:

  1. Mempermudah pencarian elemen data, misalnya dalam penerapan algoritma binary search.
  2. Memberikan struktur data yang lebih teratur untuk analisis dan pelaporan.
  3. Meningkatkan efisiensi dalam operasi pada data besar, seperti penggabungan dan pengelompokan.

  Insertion Sort: Algoritma yang sederhana dan efisien untuk data yang kecil atau hampir terurut. Memasukkan setiap elemen ke dalam posisi yang benar dalam subarray yang sudah terurut.

II. GUIDED

Soal Studi Case

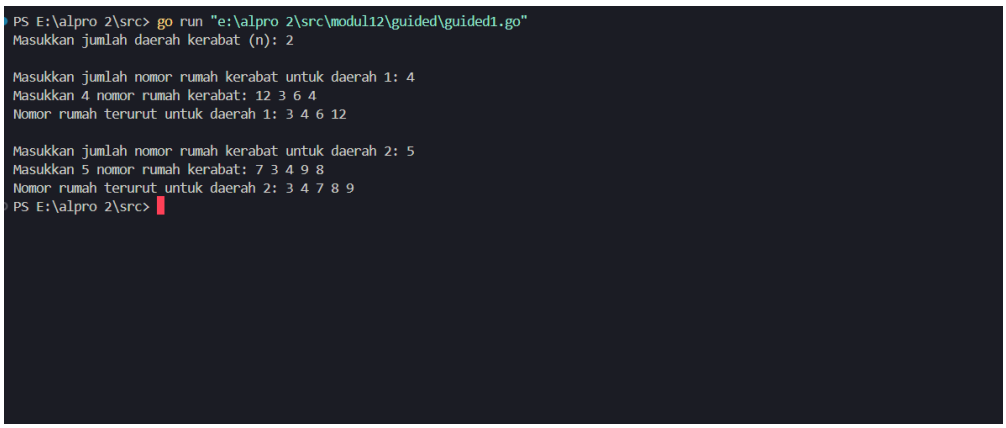
Xxxxxxxxxxxxxxx

Sourcecode

```
package main

import (
	"fmt"
)

// Fungsi untuk mengurutkan array menggunakan Selection Sort
func selectionSort(arr []int, n int) {
	for i := 0; i < n-1; i++ {
		idxMin := i
		for j := i + 1; j < n; j++ {
			// Cari elemen terkecil
			if arr[j] < arr[idxMin] {
				idxMin = j
			}
		}
		// Tukar elemen terkecil dengan elemen di posisi i
		arr[i], arr[idxMin] = arr[idxMin], arr[i]
	}
}

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

	// Proses tiap daerah
	for daerah := 1; daerah <= n; daerah++ {
		var m int
		fmt.Printf("\nMasukkan jumlah nomor rumah
kerabat untuk daerah %d: ", daerah)
		fmt.Scan(&m)

		// Membaca nomor rumah untuk daerah ini
		arr := make([]int, m)
		fmt.Printf("Masukkan %d nomor rumah kerabat:
", m)
```

```
                for i := 0; i < m; i++ {
                        fmt.Scan(&arr[i])
                }
                // Urutkan array dari terkecil ke terbesar
                selectionSort(arr, m)

                // Tampilkan hasil
                fmt.Printf("Nomor rumah terurut untuk daerah
        %d: ", daerah)
                for _, num := range arr {
                        fmt.Printf("%d ", num)
                }
                fmt.Println()
        }
}
```

Screenshoot Output



Deskripsi Program

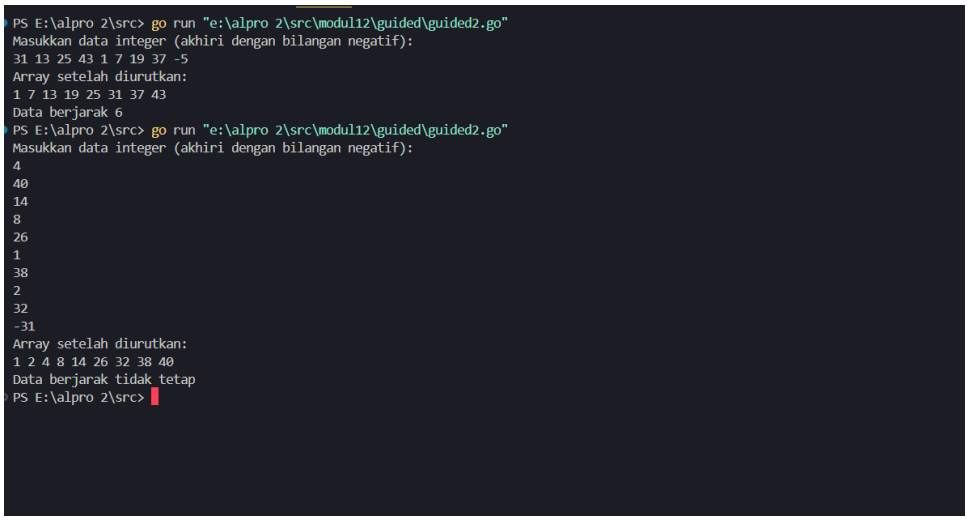
Program diatas digunakan untuk untuk mengorganisir data nomor rumah kerabat di berbagai daerah. Program akan meminta pengguna untuk memasukkan jumlah daerah dan jumlah nomor rumah di setiap daerah. Setelah itu, pengguna akan diminta untuk memasukkan nomor-nomor rumah tersebut secara manual. Selanjutnya, program akan mengurutkan nomor-nomor rumah di setiap daerah dari yang terkecil hingga terbesar menggunakan algoritma Selection Sort. Hasil pengurutan ini kemudian akan ditampilkan secara berurutan untuk setiap daerah, sehingga memudahkan dalam melihat urutan nomor rumah dari yang terkecil hingga terbesar.

Sourcecode

```
package main

import (
	"fmt"
)

// Fungsi untuk mengurutkan array menggunakan Insertion Sort
func insertionSort(arr []int, n int) {
	for i := 1; i < n; i++ {
		key := arr[i]
		j := i - 1

		// Geser elemen yang lebih besar dari key ke
kanan
		for j >= 0 && arr[j] > key {
			arr[j+1] = arr[j]
			j--
		}
		arr[j+1] = key
	}
}

// Fungsi untuk memeriksa apakah selisih elemen array tetap
func isConstantDifference(arr []int, n int) (bool, int) {
	if n < 2 {
		return true, 0
	}

	difference := arr[1] - arr[0]
	for i := 1; i < n-1; i++ {
		if arr[i+1]-arr[i] != difference {
			return false, 0
		}
	}
	return true, difference
}

func main() {
	var arr []int
	var num int
```

```
	// Input data hingga bilangan negatif ditemukan
	fmt.Println("Masukkan data integer (akhiri dengan
bilangan negatif):")
	for {
		fmt.Scan(&num)
		if num < 0 {
			break
		}
		arr = append(arr, num)
	}

	n := len(arr)

	// Urutkan array menggunakan Insertion Sort
	insertionSort(arr, n)

	// Periksa apakah selisih elemen tetap
	isConstant, difference := isConstantDifference(arr, n)

	// Tampilkan hasil pengurutan
	fmt.Println("Array setelah diurutkan:")
	for _, val := range arr {
		fmt.Printf("%d ", val)
	}
	fmt.Println()

	// Tampilkan status jarak
	if isConstant {
		fmt.Printf("Data berjarak %d\n", difference)
	} else {
		fmt.Println("Data berjarak tidak tetap")
	}
}
```

Screenshoot Output



Deskripsi Program

Program diatas digunakan untuk mengurutkan dan menganalisis deret bilangan bulat yang diinputkan pengguna. Program akan meminta pengguna untuk memasukkan angka-angka secara berurutan hingga pengguna memasukkan angka negatif sebagai tanda berhenti. Setelah semua angka terkumpul, program akan mengurutkan angka-angka tersebut dari yang terkecil hingga terbesar menggunakan metode pengurutan Insertion Sort. Selanjutnya, program akan memeriksa apakah selisih antara setiap angka yang berurutan selalu sama. Jika selisihnya konsisten atau tetap, program akan menampilkan nilai selisih tersebut. Sebaliknya, jika selisihnya berbeda-beda, program akan menginformasikan bahwa deret angka tersebut tidak memiliki selisih yang tetap.

III. UNGUIDED

Soal Studi Case

Xxxxxxxxxxxxxxxx

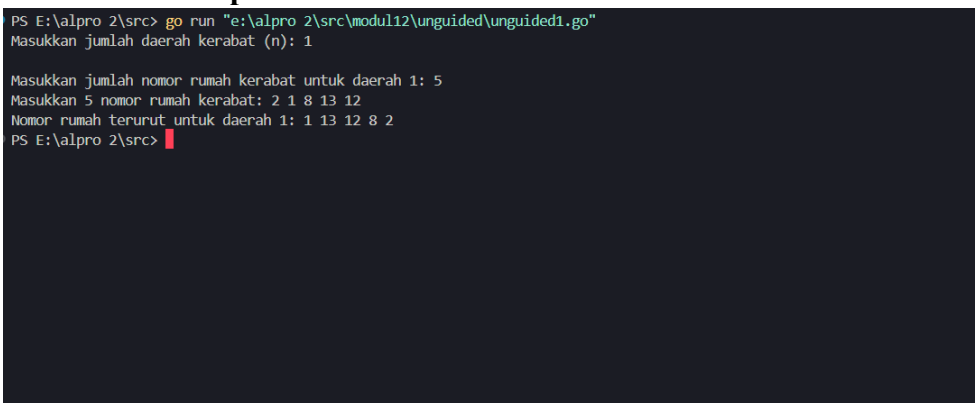
Sourcecode

```
package main

import "fmt"

func printGanjilGenap(arr []int, n int)  {
	arrGanjil := make([]int, n)
	arrGenap := make([]int, n)
	idxGenap := 0
	idxGanjil := 0

	for i := 0; i < n; i++ {
		if arr[i]%2 == 0 {
			arrGenap[idxGenap] = arr[i]
			idxGenap++
		} else {
			arrGanjil[idxGanjil] = arr[i]
			idxGanjil++
		}

	}

	selectionSort(arrGenap, arrGanjil, idxGenap, idxGanjil)

	for i := 0; i < idxGanjil; i++ {
		fmt.Print(arrGanjil[i], " ")
	}
	for i := 0; i < idxGenap; i++ {
		fmt.Print(arrGenap[i], " ")
	}

}

func selectionSort(arrGenap, arrGanjil []int, idxGenap, idxGanjil
int) {
	var arrass int
	var arrdess int
```

```
        for i := 0; i < idxGanjil; i++ {
            for j := i+1; j < idxGanjil; j++ {
                if arrGanjil[i] > arrGanjil[j] {
                    arrass = arrGanjil[j]
                    arrGanjil[j] = arrGanjil[i]
                    arrGanjil[i] = arrass
                }
            }
        }
        for i := 0; i < idxGenap; i++ {
            for j := i+1; j < idxGenap; j++ {
                if arrGenap[i] < arrGenap[j] {
                    arrdess = arrGenap[j]
                    arrGenap[j] = arrGenap[i]
                    arrGenap[i] = arrdess
                }
            }
        }
}

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

        // Proses tiap daerah
        for daerah := 1; daerah <= n; daerah++ {
            var m int
            fmt.Printf("\nMasukkan jumlah nomor rumah
kerabat untuk daerah %d: ", daerah)
            fmt.Scan(&m)

            // Membaca nomor rumah untuk daerah ini
            arr := make([]int, m)
            fmt.Printf("Masukkan %d nomor rumah kerabat:
", m)
            for i := 0; i < m; i++ {
                fmt.Scan(&arr[i])
            }

            fmt.Printf("Nomor rumah terurut untuk daerah
%d: ", daerah)
```

```
                        // Urutkan array dari terkecil ke terbesar
                        // Tampilkan hasil
                        printGanjilGenap(arr, m)

                        fmt.Println()
                }
}
```

Screenshoot Output



Deskripsi Program

Program diatas digunakan untuk mengorganisir dan menyajikan data nomor rumah kerabat secara lebih terstruktur, updatetan dari guided1. Program ini akan meminta pengguna untuk memasukkan sejumlah nomor rumah, kemudian mengelompokkan nomor-nomor tersebut berdasarkan bilangan genap dan ganjil. Setelah itu, setiap kelompok (genap dan ganjil) akan diurutkan dari yang terkecil hingga terbesar untuk ganjil dan besar ke kecil untuk genap. Hasil akhir yang ditampilkan adalah daftar nomor rumah yang telah diurutkan, dengan nomor-nomor ganjil muncul terlebih dahulu diikuti oleh nomor-nomor genap.

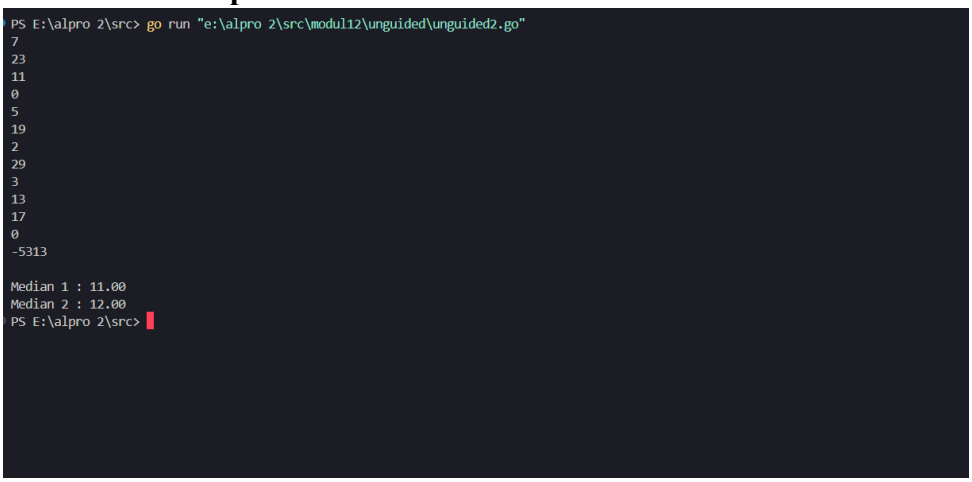
Sourcecode

```
        package main

        import (
                "fmt"
        )

        func findMedian(arrBilBul []int) float64 {
                for i := 1; i < len(arrBilBul); i++ {
                        key := arrBilBul[i]
```

```
                j := i - 1
                for j >= 0 && arrBilBul[j] > key {
                        arrBilBul[j+1] = arrBilBul[j]
                        j--
                }
                arrBilBul[j+1] = key
        }

        n := len(arrBilBul)
        if n%2 == 0 {
                return float64((arrBilBul[n/2-1] +
arrBilBul[n/2])) / 2.0
        }
        return float64(arrBilBul[n/2])
}

func main() {
        var arrBilBul []int
   var median []float64
   jumlahM := 0
        for {
                var input int
                fmt.Scan(&input)

                if input < 0 {
                        break
                }

                if input == 0 {
                        if len(arrBilBul) == 0 {
                                fmt.Println(0)
                        } else {
                                median = append(median,
findMedian(arrBilBul))
            jumlahM++
                        }
                        continue
                }

                arrBilBul = append(arrBilBul, input)
        }

   fmt.Println()
```

```
        for i := 0; i < jumlahM; i++ {
            fmt.Printf("Median %v : %.2f\n", i+1, median[i])
        }
}
```

Screenshoot Output



Deskripsi Program

Program diatas digunakan untuk menghitung nilai tengah (median) dari sekumpulan angka yang diinputkan pengguna secara berulang. Program ini akan terus meminta pengguna memasukkan angka hingga pengguna memasukkan angka kurang dari 0 sebagai tanda berhenti. Setiap kali pengguna memasukkan angka 0, program akan mengurutkan semua angka yang telah dimasukkan sebelumnya, kemudian mencari nilai tengahnya. Jika jumlah angka genap, nilai tengah adalah rata-rata dari dua angka di tengah. Hasil perhitungan median untuk setiap kelompok angka akan ditampilkan secara berurutan.

Sourcecode

```
package main

import "fmt"

const nMax = 7919

type Buku struct {
        id, judul, penulis, penerbit  string
        eksemplar, tahun, rating    int
}
```

```
type DaftarBuku struct {
	Pustaka  [nMax]Buku
	nPustaka int
}

func DaftarkanBuku(pustaka *DaftarBuku, n int) {
	for i := 0; i < n; i++ {
		fmt.Println("Masukkan data buku (id, judul,
penulis, penerbit, eksemplar, tahun, rating):")
		fmt.Scan(&pustaka.Pustaka[i].id,
&pustaka.Pustaka[i].judul, &pustaka.Pustaka[i].penulis,
&pustaka.Pustaka[i].penerbit, &pustaka.Pustaka[i].eksemplar,
&pustaka.Pustaka[i].tahun, &pustaka.Pustaka[i].rating)
	}
	pustaka.nPustaka = n
}

func CetakTerfavorit(pustaka DaftarBuku) {
	if pustaka.nPustaka == 0 {
		fmt.Println("Tidak ada buku dalam pustaka.")
		return
	}

	terfavorit := pustaka.Pustaka[0]
	for i := 1; i < pustaka.nPustaka; i++ {
		if pustaka.Pustaka[i].rating > terfavorit.rating {
			terfavorit = pustaka.Pustaka[i]
		}
	}

	fmt.Println("Buku terfavorit:")
	fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun:
%d, Rating: %d\n",
		terfavorit.judul, terfavorit.penulis,
terfavorit.penerbit, terfavorit.tahun, terfavorit.rating)
}

func UrutBuku(pustaka *DaftarBuku) {
	for i := 1; i < pustaka.nPustaka; i++ {
		key := pustaka.Pustaka[i]
		j := i - 1
```

```
                for j >= 0 && pustaka.Pustaka[j].rating <
key.rating {
                        pustaka.Pustaka[j+1] = pustaka.Pustaka[j]
                        j--
                }
                pustaka.Pustaka[j+1] = key
        }
}

func Cetak5Terbaru(pustaka DaftarBuku) {
        fmt.Println("5 Buku dengan rating tertinggi:")
        for i := 0; i < 5 && i < pustaka.nPustaka; i++ {
                buku := pustaka.Pustaka[i]
                fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s,
Tahun: %d, Rating: %d\n",
                        buku.judul, buku.penulis, buku.penerbit,
buku.tahun, buku.rating)
        }
}

func CariBuku(pustaka DaftarBuku, rating int) {
        low, high := 0, pustaka.nPustaka-1
        for low <= high {
                mid := (low + high) / 2
                if pustaka.Pustaka[mid].rating == rating {
                        buku := pustaka.Pustaka[mid]
                        fmt.Printf("Buku ditemukan: Judul: %s,
Penulis: %s, Penerbit: %s, Tahun: %d, Rating: %d\n",
                                buku.judul, buku.penulis,
buku.penerbit, buku.tahun, buku.rating)
                        return
                } else if pustaka.Pustaka[mid].rating < rating {
                        high = mid - 1
                } else {
                        low = mid + 1
                }
        }
        fmt.Println("Tidak ada buku dengan rating seperti itu.")
}

func main() {
        var pustaka DaftarBuku
        var n, rating int
```

```
        fmt.Print("Masukkan jumlah buku: ")
        fmt.Scan(&n)

        DaftarkanBuku(&pustaka, n)

        CetakTerfavorit(pustaka)

        UrutBuku(&pustaka)

        Cetak5Terbaru(pustaka)

        fmt.Print("Masukkan rating buku yang ingin dicari: ")
        fmt.Scan(&rating)
        CariBuku(pustaka, rating)
}
```

Screenshoot Output

```
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul12\unguided\unguided3.go"
Masukkan jumlah buku: 2
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 Harry Stone Jeka 50 1999 4
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 Naruto Masashi Roky 260 1998 5
Buku terfavorit:
Judul: Naruto, Penulis: Masashi, Penerbit: Roky, Tahun: 1998, Rating: 5
PS E:\alpro 2\src> go run "e:\alpro 2\src\modul12\unguided\unguided3.go"
Masukkan jumlah buku: 2
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1 Harry Stone Jeka 50 1999 4
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
2 Naruto Masashi Roky 260 1998 5
Buku terfavorit:
Judul: Naruto, Penulis: Masashi, Penerbit: Roky, Tahun: 1998, Rating: 5
5 Buku dengan rating tertinggi:
Judul: Naruto, Penulis: Masashi, Penerbit: Roky, Tahun: 1998, Rating: 5
Judul: Harry, Penulis: Stone, Penerbit: Jeka, Tahun: 1999, Rating: 4
Masukkan rating buku yang ingin dicari: 4
Buku ditemukan: Judul: Harry, Penulis: Stone, Penerbit: Jeka, Tahun: 1999, Rating: 4
PS E:\alpro 2\src>
```

Deskripsi Program

Program diatas digunakan untuk mengelola daftar buku. Program dimulai dengan mendefinisikan tipe data Buku yang berisi atribut seperti id, judul, penulis, penerbit, eksemplar, tahun, dan rating. Struktur DaftarBuku berisi array buku dengan kapasitas maksimum nMax (7919) dan jumlah buku saat ini (nPustaka). Fungsi DaftarkanBuku mengisi daftar buku berdasarkan input pengguna, sedangkan CetakTerfavorit mencari buku dengan rating tertinggi menggunakan iterasi sederhana. Selanjutnya, UrutBuku mengurutkan buku berdasarkan rating secara menurun menggunakan algoritma insertion sort. Fungsi Cetak5Terbaru mencetak lima buku dengan rating tertinggi dari daftar yang sudah diurutkan. Untuk pencarian buku berdasarkan rating tertentu, digunakan fungsi CariBuku

yang mengimplementasikan binary search, dengan asumsi bahwa array sudah terurut sebelumnya. Pada bagian utama (main), program meminta jumlah buku, mengisi daftar buku, mencetak buku terfavorit, mengurutkan buku, mencetak lima buku terbaik, dan melakukan pencarian berdasarkan rating.