

**LAPORAN PRAKTIKUM
ALGORITME DAN PEMROGRAMAN 2**

**MODUL 11
PENCARIAN NILAI ACAK PADA HIMPUNAN DATA**



Oleh:

RAIFANKA RAISA RAMADHAN

2311102205

IF - 11 - 07

**PROGRAM STUDI S1 TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2024**

I. Dasar Teori

Pencarian nilai acak pada himpunan data di Golang adalah suatu metode yang digunakan untuk mencari apakah sebuah nilai yang dipilih secara acak ada di dalam kumpulan data, seperti array atau slice. Proses ini dimulai dengan menghasilkan sebuah angka acak yang berfungsi sebagai indeks untuk memilih elemen yang akan diperiksa dalam himpunan data tersebut. Setelah mendapatkan angka acak sebagai indeks, program kemudian memeriksa apakah nilai yang terletak pada indeks tersebut ada dalam kumpulan data yang dimaksud.

Dalam proses pencarian ini, Golang memungkinkan pengguna untuk memanfaatkan teknik-teknik pencarian sederhana, seperti menggunakan loop atau perbandingan langsung, untuk memastikan apakah nilai yang dipilih acak tersebut memang ada dalam data atau tidak. Pencarian semacam ini dapat dilakukan pada berbagai jenis data, mulai dari angka, string, hingga tipe data lainnya yang ada.

II. Guided Guided 1

```
package main

import (
    "fmt"
)

// Definisi struct mahasiswa
type mahasiswa struct {
    nama      string
    nim       string
    kelas     string
    jurusan   string
    ipk       float64
}

// Definisi array bertipe struct mahasiswa
type arrMhs [2023]mahasiswa

// Sequential Search berdasarkan nama
func SeqSearch_3(T arrMhs, n int, X string) int {
    /* Mengembalikan indeks mahasiswa dengan nama X,
       atau -1 apabila tidak ditemukan pada array T
       yang berisi n data mahasiswa */
    var found int = -1
    var j int = 0
    for j < n && found == -1 {
        if T[j].nama == X {
            found = j
        }
        j = j + 1
    }
    return found
}

// Binary Search berdasarkan nim
func BinarySearch_3(T arrMhs, n int, X string) int {
    /* Mengembalikan indeks mahasiswa dengan nim X,
       atau -1 apabila tidak ditemukan pada array T
       yang berisi n data mahasiswa dan terurut membesar
       berdasarkan nim */
    var found int = -1
    var med int
    var kr int = 0
    var kn int = n - 1
```

```

        for kr <= kn && found == -1 {
            med = (kr + kn) / 2
            if X < T[med].nim {
                kn = med - 1
            } else if X > T[med].nim {
                kr = med + 1
            } else {
                found = med
            }
        }
        return found
    }

// Fungsi utama untuk demonstrasi
func main() {
    var data arrMhs
    var n int

    // Input jumlah mahasiswa
    fmt.Print("Masukkan jumlah mahasiswa: ")
    fmt.Scanln(&n)

    // Input data mahasiswa
    fmt.Println("Masukkan data mahasiswa:")
    for i := 0; i < n; i++ {
        fmt.Printf("Mahasiswa %d\n", i+1)
        fmt.Print("Nama: ")
        fmt.Scanln(&data[i].nama)
        fmt.Print("NIM: ")
        fmt.Scanln(&data[i].nim)
        fmt.Print("Kelas: ")
        fmt.Scanln(&data[i].kelas)
        fmt.Print("Jurusan: ")
        fmt.Scanln(&data[i].jurusan)
        fmt.Print("IPK: ")
        fmt.Scanln(&data[i].ipk)
    }

    // Pencarian berdasarkan nama
    var cariNama string
    fmt.Print("Masukkan nama mahasiswa yang ingin dicari: ")
    fmt.Scanln(&cariNama)
    posNama := SeqSearch_3(data, n, cariNama)
    if posNama == -1 {

```

```

        fmt.Println("Mahasiswa dengan nama tersebut tidak
ditemukan.")
    } else {
        fmt.Printf("Mahasiswa ditemukan pada indeks: %d\n",
posNama)
    }

    // Pencarian berdasarkan nim
    var cariNIM string
    fmt.Print("Masukkan NIM mahasiswa yang ingin dicari: ")
    fmt.Scanln(&cariNIM)
    posNIM := BinarySearch_3(data, n, cariNIM)
    if posNIM == -1 {
        fmt.Println("Mahasiswa dengan NIM tersebut tidak
ditemukan.")
    } else {
        fmt.Printf("Mahasiswa ditemukan pada indeks: %d\n",
posNIM)
    }
}

```

Screenshots Output

The screenshot shows a Go IDE with the following components:

- EXPLORER:** Lists files including `guided1.go`, `guided2.go`, and `guided3.go`.
- EDITOR:** Displays the source code of `guided1.go`, which includes:
 - Package and import statements.
 - Struct definition for `mahasiswa` with fields: `nama` (string), `nim` (string), `kelas` (string), `jurusan` (string), and `ipk` (float64).
 - Array definition `arrMhs` of type `mahasiswa`.
 - Function `SeqSearch_3` for sequential search based on name.
 - Function `BinarySearch_3` for binary search based on NIM.
- TERMINAL:** Shows the execution output:
 - Initial prompt: `Masukkan jumlah mahasiswa: 2`
 - Data entry for 2 students:
 - Student 1: Name: raifanka, NIM: 1111, Kelas: 01, Jurusan: IF, IPK: 4.0
 - Student 2: Name: raisa, NIM: 2222, Kelas: 02, Jurusan: IF, IPK: 4.0
 - Search process:
 - Prompt: `Masukkan nama mahasiswa yang ingin dicari: ramadhan`
 - Output: `Mahasiswa dengan nama tersebut tidak ditemukan.`
 - Prompt: `Masukkan NIM mahasiswa yang ingin dicari: 2222`
 - Output: `Mahasiswa ditemukan pada indeks: 1`

Deskripsi:

Program ini mengelola data mahasiswa menggunakan struktur data array yang berisi elemen-elemen bertipe mahasiswa. Struktur mahasiswa mencakup atribut nama, NIM, kelas, jurusan, dan IPK. Program ini mendemonstrasikan dua metode pencarian pada array mahasiswa: Sequential Search berdasarkan nama dan Binary Search berdasarkan NIM.

Dalam pencarian Sequential Search, program akan mencari nama mahasiswa yang dimasukkan dan mengembalikan indeksinya, atau -1 jika nama tidak ditemukan. Sementara itu, Binary Search digunakan untuk mencari mahasiswa berdasarkan NIM, dengan asumsi data mahasiswa sudah terurut berdasarkan NIM. Setelah pengguna memasukkan jumlah mahasiswa dan data mereka, program akan meminta input nama dan NIM mahasiswa untuk dicari, dan menampilkan hasil pencarian, apakah mahasiswa ditemukan atau tidak.

Guided 2

```
package main

import "fmt"

// Define arrInt as an array of integers with a fixed size
type arrInt [4321]int

// Binary search function for descending sorted array
func BinarySearch_2(T arrInt, n int, X int) int {
    /* Mengembalikan indeks dari X apabila X ditemukan di dalam
    array T
        yang berisi n buah bilangan bulat terurut secara
    descending/menciu,
        atau -1 apabila X tidak ditemukan */
    var found int = -1
    var med int
    var kr int = 0
    var kn int = n - 1

    // Iterasi mencari elemen dengan binary search
    for kr <= kn && found == -1 {
        med = (kr + kn) / 2
        if X > T[med] { // Karena descending, jika X lebih
        besar, bergerak ke kiri
            kn = med - 1
        } else if X < T[med] { // Jika X lebih kecil, bergerak
        ke kanan
            kr = med + 1
        } else { // Jika ditemukan
            found = med
        }
    }
    return found
}

func main() {
    // Contoh penggunaan
    var data arrInt
    var n int
    // Input jumlah elemen
    fmt.Print("Masukkan jumlah elemen dalam array: ")
    fmt.Scanln(&n)
    // Input elemen array (harus terurut secara descending)
    fmt.Println("Masukkan elemen array (harus terurut
    menurun):")
}
```

```

    for i := 0; i < n; i++ {
        fmt.Printf("Elemen %d: ", i+1)
        fmt.Scanln(&data[i])
    }

    // Input elemen yang dicari
    var search int
    fmt.Print("Masukkan elemen yang ingin dicari: ")
    fmt.Scanln(&search)

    // Panggil fungsi binary search
    result := BinarySearch_2(data, n, search)

    // Cetak hasil
    if result == -1 {
        fmt.Println("Elemen tidak ditemukan dalam array.")
    } else {
        fmt.Printf("Elemen ditemukan pada indeks: %d\n",
result)
    }
}

```

Screenshots Output

The screenshot displays a Go IDE with the following components:

- EXPLORER:** Shows the project structure with files like `guided1.go`, `guided2.go`, and `guided3.go`.
- EDITOR:** Contains the source code for `BinarySearch_2`. The code defines an array of 4321 integers and implements a binary search function for a descending sorted array. Comments in Indonesian explain the logic, such as "Mencari elemen dengan binary search" and "Iterasi mencari elemen dengan binary search".
- TERMINAL:** Shows the execution output. It starts with a command to run the program, followed by the prompt "Masukkan jumlah elemen dalam array: 5". The user enters 5, and the program prints the first five elements: "Elemen 1: 5", "Elemen 2: 4", "Elemen 3: 3", "Elemen 4: 2", "Elemen 5: 1". Then, it prompts "Masukkan elemen yang ingin dicari: 3". The user enters 3, and the program outputs "Elemen ditemukan pada indeks: 2".

Deskripsi:

Program ini menggunakan metode Binary Search untuk mencari elemen dalam array yang telah terurut secara descending. Tipe data `arrInt` didefinisikan sebagai array bertipe integer dengan ukuran tetap 4321 elemen. Fungsi `BinarySearch_2` digunakan untuk mencari elemen dalam

array yang terurut menurun. Proses pencarian dimulai dengan membandingkan elemen tengah (med) dengan elemen yang dicari. Jika elemen yang dicari lebih besar dari nilai tengah, pencarian dilanjutkan ke sisi kiri array; jika lebih kecil, pencarian berlanjut ke sisi kanan. Fungsi ini mengembalikan indeks elemen yang ditemukan atau -1 jika elemen tersebut tidak ada. Program ini memungkinkan pengguna untuk memasukkan jumlah elemen dalam array, kemudian memasukkan nilai-nilai array yang sudah terurut secara menurun. Setelah itu, pengguna dapat mencari elemen tertentu dalam array tersebut. Hasil pencarian akan ditampilkan berupa indeks elemen jika ditemukan atau pesan bahwa elemen tidak ditemukan jika pencarian gagal.

Guided 3

```
package main

import "fmt"

// Define arrStr as an array of strings with a fixed size
type arrStr [1234]string

// Sequential search function
func SeqSearch_1(T arrStr, n int, X string) int {
    /* Mengembalikan indeks dari X apabila X ditemukan di dalam
    array T
        yang berisi n buah teks, atau -1 apabila X tidak
    ditemukan */
    var found int = -1
    var j int = 0

    // Iterasi mencari elemen yang cocok
    for j < n && found == -1 {
        if T[j] == X {
            found = j
        }
        j = j + 1
    }
    return found
}

func main() {
    // Contoh penggunaan
    var data arrStr
    var n int

    // Input jumlah elemen
    fmt.Print("Masukkan jumlah elemen dalam array: ")
    fmt.Scanln(&n)

    // Input elemen array
    fmt.Println("Masukkan elemen array:")
    for i := 0; i < n; i++ {
        fmt.Printf("Elemen %d: ", i+1)
        fmt.Scanln(&data[i])
    }
    // Input elemen yang dicari
    var search string
    fmt.Print("Masukkan elemen yang ingin dicari: ")
    fmt.Scanln(&search)
    // Panggil fungsi sequential search
```

```

    result := SeqSearch_1(data, n, search)
    // Cetak hasil
    if result == -1 {
        fmt.Println("Elemen tidak ditemukan dalam array.")
    } else {
        fmt.Printf("Elemen ditemukan pada indeks: %d\n",
result)
    }
}

```

Screenshots Output

The screenshot shows a Go IDE with the following components:

- EXPLORER:** Lists files including `guided1.go`, `guided2.go`, `guided3.go`, `tempCodeRunnerFile.go`, and `unguided1.go` through `unguided3.go`.
- EDITOR:** Displays the source code for `guided3.go`. The code defines a `SeqSearch_1` function that performs a sequential search on an array of strings. It includes comments in Indonesian explaining the function's purpose and logic.
- TERMINAL:** Shows the execution output. It prompts the user to enter the number of elements (6) and the elements themselves (2, 4, 6, 0, 0, 1). It then prompts for the element to search (4) and outputs the result: "Elemen ditemukan pada indeks: 1".

Deskripsi:

Program ini menggunakan metode **Sequential Search** untuk mencari elemen dalam array yang berisi string. Array `arrStr` didefinisikan dengan ukuran tetap 1234 elemen, dan program memungkinkan pengguna untuk memasukkan sejumlah elemen string yang ingin dimasukkan ke dalam array. Fungsi `SeqSearch_1` digunakan untuk mencari elemen tertentu dalam array dengan cara memeriksa satu per satu elemen dari awal hingga akhir. Jika elemen yang dicari ditemukan, fungsi akan mengembalikan indeks elemen tersebut. Jika tidak, fungsi akan mengembalikan nilai -1, menandakan bahwa elemen tidak ditemukan. Program meminta pengguna untuk memasukkan jumlah elemen array dan kemudian memasukkan string-string tersebut satu per satu. Setelah itu, pengguna diminta untuk memasukkan elemen yang ingin dicari. Fungsi `SeqSearch_1` kemudian akan mencari elemen tersebut dalam array. Hasil pencarian akan dicetak: jika elemen ditemukan, indeksnya akan ditampilkan; jika tidak, pesan bahwa elemen tidak ditemukan akan muncul.

III. Unguided Unguided 1

```
package main

import (
    "fmt"
)

func main() {
    var suara int
    var totalSuaraMasuk, totalSuaraSah int

    suaraCalon := [21]int{}

    for {
        fmt.Scan(&suara)

        if suara == 0 {
            break
        }

        totalSuaraMasuk++

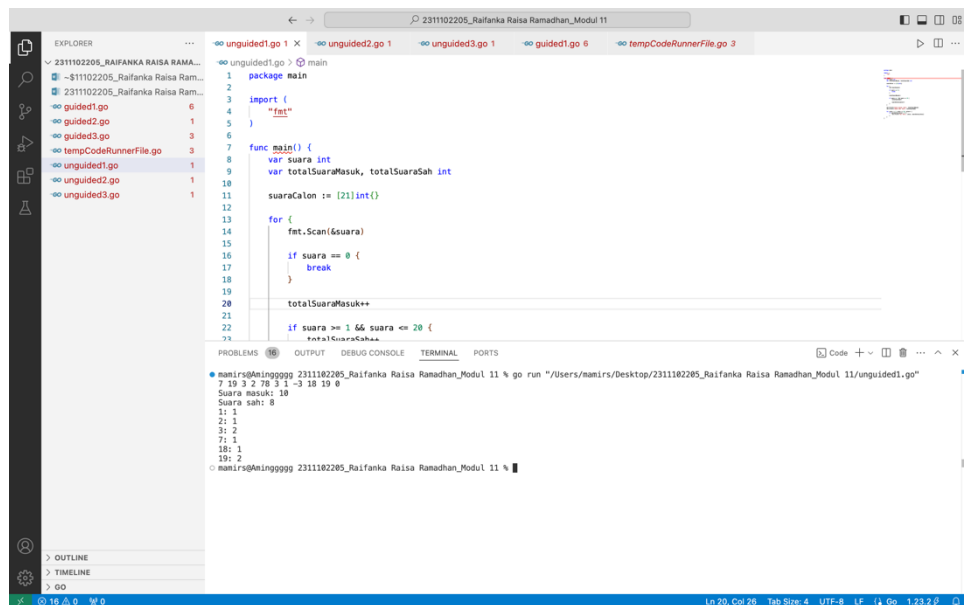
        if suara >= 1 && suara <= 20 {
            totalSuaraSah++

            suaraCalon[suara]++
        }
    }

    fmt.Printf("Suara masuk: %d\n", totalSuaraMasuk)
    fmt.Printf("Suara sah: %d\n", totalSuaraSah)

    for calon := 1; calon <= 20; calon++ {
        if suaraCalon[calon] > 0 {
            fmt.Printf("%d: %d\n", calon, suaraCalon[calon])
        }
    }
}
```

Screenshots Output



```
package main

import (
    "fmt"
)

func main() {
    var suara int
    var totalSuaraMasuk, totalSuaraSah int
    suaraCalon := [21]int{}

    for {
        fmt.Scan(&suara)
        if suara == 0 {
            break
        }
        totalSuaraMasuk++
        if suara >= 1 && suara <= 20 {
            totalSuaraSah++
            suaraCalon[suara-1]++
        }
    }

    fmt.Println("Total Suara Masuk:", totalSuaraMasuk)
    fmt.Println("Total Suara Sah:", totalSuaraSah)
    for i, v := range suaraCalon {
        if v > 0 {
            fmt.Printf("Calon %d: %d suara\n", i+1, v)
        }
    }
}
```

Terminal Output:

```
mamirs@Amingggg 2311182205_Raifanka Raisa Ramadhan_Modul 11 % go run *.go
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
1: 1
2: 1
3: 2
7: 1
18: 1
19: 2
Total Suara Masuk: 10
Total Suara Sah: 8
Calon 1: 1 suara
Calon 2: 1 suara
Calon 3: 2 suara
Calon 7: 1 suara
Calon 18: 1 suara
Calon 19: 2 suara
```

Deskripsi:

Program ini digunakan untuk menghitung hasil pemilu dengan cara menerima input suara untuk masing-masing calon. Suara dimasukkan secara berulang, dan program akan terus menerima input hingga pengguna memasukkan angka 0. Setiap kali suara dimasukkan, program menghitung total suara yang masuk. Program juga memeriksa apakah suara yang dimasukkan valid, yaitu berupa angka antara 1 hingga 20. Jika suara berada dalam rentang tersebut, suara dianggap sah dan dihitung. Suara yang tidak valid tidak akan dihitung. Semua suara untuk masing-masing calon disimpan dalam array yang mewakili nomor calon, dengan nilai pada indeks array menunjukkan jumlah suara yang diterima oleh calon tersebut. Setelah input selesai, program mencetak total suara yang masuk, total suara sah, dan jumlah suara yang diterima oleh setiap calon yang memiliki suara lebih dari 0.

Unguided 2

```
package main

import (
    "fmt"
)

func main() {
    var suara int
    var totalSuaraMasuk, totalSuaraSah int

    suaraCalon := [21]int{}

    for {
        fmt.Scan(&suara)

        if suara == 0 {
            break
        }

        totalSuaraMasuk++

        if suara >= 1 && suara <= 20 {
            totalSuaraSah++

            suaraCalon[suara]++
        }
    }

    fmt.Printf("Suara masuk: %d\n", totalSuaraMasuk)
    fmt.Printf("Suara sah: %d\n", totalSuaraSah)

    var ketua, wakil int
    var suaraTerbanyak, suaraWakil int

    for calon := 1; calon <= 20; calon++ {
        if suaraCalon[calon] > suaraTerbanyak ||
            (suaraCalon[calon] == suaraTerbanyak && calon <
ketua) {
            wakil = ketua
            suaraWakil = suaraTerbanyak
            ketua = calon
            suaraTerbanyak = suaraCalon[calon]
        } else if suaraCalon[calon] > suaraWakil ||
            (suaraCalon[calon] == suaraWakil && calon < wakil)
    }
}
```

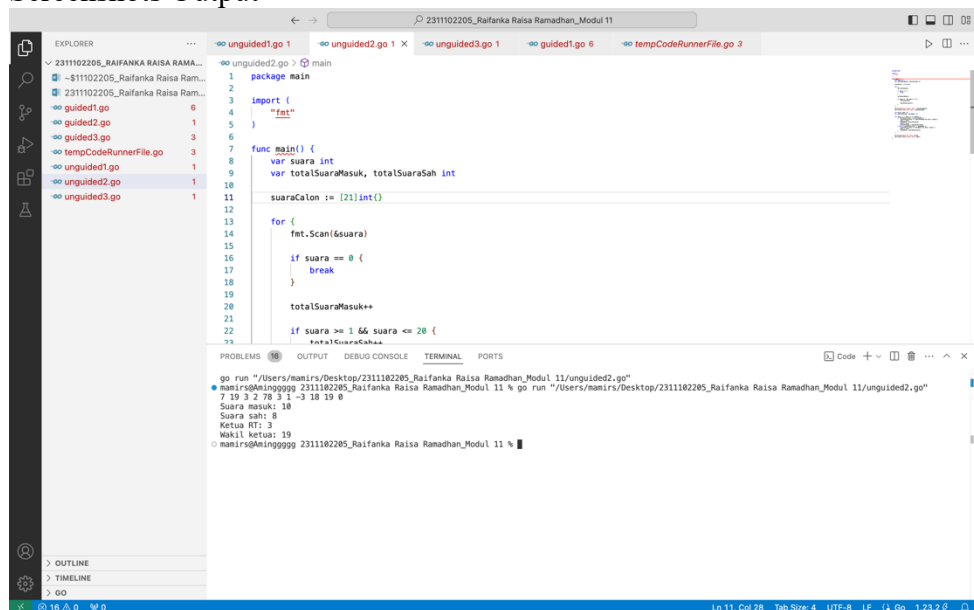
```

        wakil = calon
        suaraWakil = suaraCalon[calon]
    }
}

fmt.Printf("Ketua RT: %d\n", ketua)
fmt.Printf("Wakil ketua: %d\n", wakil)
}

```

Screenshots Output



Deskripsi:

Program ini digunakan untuk menghitung suara pemilu ketua RT dan wakil ketua berdasarkan input suara yang dimasukkan oleh pengguna. Suara dimasukkan secara berulang, dan program akan terus menerima input hingga pengguna memasukkan angka 0. Program menghitung total suara yang masuk dan total suara sah (suara yang berada dalam rentang 1 hingga 20). Suara untuk masing-masing calon disimpan dalam array, dengan indeks array yang sesuai mewakili calon. Setelah semua suara masuk, program kemudian mencari calon dengan suara terbanyak untuk menjabat sebagai ketua dan calon dengan suara terbanyak kedua sebagai wakil ketua. Proses pemilihan ketua dan wakil ketua mempertimbangkan suara terbanyak, dengan aturan jika ada calon dengan jumlah suara yang sama, maka calon dengan nomor lebih kecil akan dipilih terlebih dahulu. Di akhir, program mencetak jumlah suara yang masuk, jumlah suara sah, serta nomor calon yang terpilih sebagai ketua dan wakil ketua berdasarkan suara terbanyak.

Unguided 3

```
package main

import "fmt"

const NMAX = 1000000

var data [NMAX]int

func isiArray(n int) {
    for i := 0; i < n; i++ {
        fmt.Scan(&data[i])
    }
}

func posisi(n, k int) int {
    kiri := 0
    kanan := n - 1

    for kiri <= kanan {
        tengah := (kiri + kanan) / 2

        if data[tengah] == k {
            return tengah
        } else if data[tengah] < k {
            kiri = tengah + 1
        } else {
            kanan = tengah - 1
        }
    }

    return -1
}

func main() {
    var n, k int

    fmt.Scan(&n, &k)
    isiArray(n)

    hasil := posisi(n, k)

    if hasil == -1 {
        fmt.Println("TIDAK ADA")
    } else {
        fmt.Println("Output: ", hasil)
    }
}
```




Screenshots Output

```

1 package main
2
3 import "fmt"
4
5 const NMAX = 1000000
6
7 var data [NMAX]int
8
9 func isiArray(n int) {
10     for i := 0; i < n; i++ {
11         fmt.Scan(&data[i])
12     }
13 }
14
15 func posisi(n, k int) int {
16     kiri := 0
17     kanan := n - 1
18
19     for kiri <= kanan {
20         tengah := (kiri + kanan) / 2
21         if data[tengah] == k {
22             return tengah
23         }
24     }
25     return -1
26 }
27
28 func main() {
29     n, k := 0, 0
30     isiArray(n)
31     posisi(n, k)
32 }

```

```

go run "/Users/mamirs/Desktop/2311102205_Raifanka Raisa Ramadhan_Modul 11/unguided3.go"
mamirs@Mingggggg 2311102205_Raifanka Raisa Ramadhan_Modul 11 % go run "/Users/mamirs/Desktop/2311102205_Raifanka Raisa Ramadhan_Modul 11/unguided3.go"
12 534
1 3 8 16 32 123 323 323 534 543 823 999
Output: 8
mamirs@Mingggggg 2311102205_Raifanka Raisa Ramadhan_Modul 11 %

```

Deskripsi:

Program ini berfungsi untuk mencari elemen dalam array yang sudah terurut menggunakan algoritma binary search. Pada awal program, terdapat sebuah konstanta NMAX yang mengatur batas maksimum ukuran array, yaitu 1 juta elemen. Variabel global data digunakan untuk menyimpan elemen-elemen yang dimasukkan oleh pengguna. Fungsi isiArray digunakan untuk mengisi array dengan nilai-nilai yang dimasukkan oleh pengguna, berdasarkan jumlah elemen n yang diberikan. Setelah array diisi, fungsi posisi digunakan untuk mencari posisi elemen k dalam array menggunakan metode binary search, yang membagi array menjadi dua bagian secara efisien untuk menemukan elemen yang dicari. Jika elemen ditemukan, fungsi ini akan mengembalikan indeks elemen tersebut. Jika tidak ditemukan, maka akan mengembalikan nilai -1. Fungsi main pertama-tama membaca dua input, yaitu jumlah elemen n dan elemen yang dicari k. Setelah itu, array diisi dengan elemen-elemen yang diberikan dan pencarian dilakukan dengan memanggil fungsi posisi. Hasilnya, jika elemen ditemukan, program akan mencetak posisi elemen tersebut, sementara jika tidak ditemukan, akan mencetak pesan "TIDAK ADA".