Muhammad Ghifari Habiburrahman
1105220036

Pencarian Versi 2

Studi Kasus

Studi kasus dari kode di atas adalah pencarian elemen tertentu dalam sebuah array yang sudah terurut menggunakan metode pencarian biner (binary search). Prosesnya dapat dijelaskan sebagai berikut:

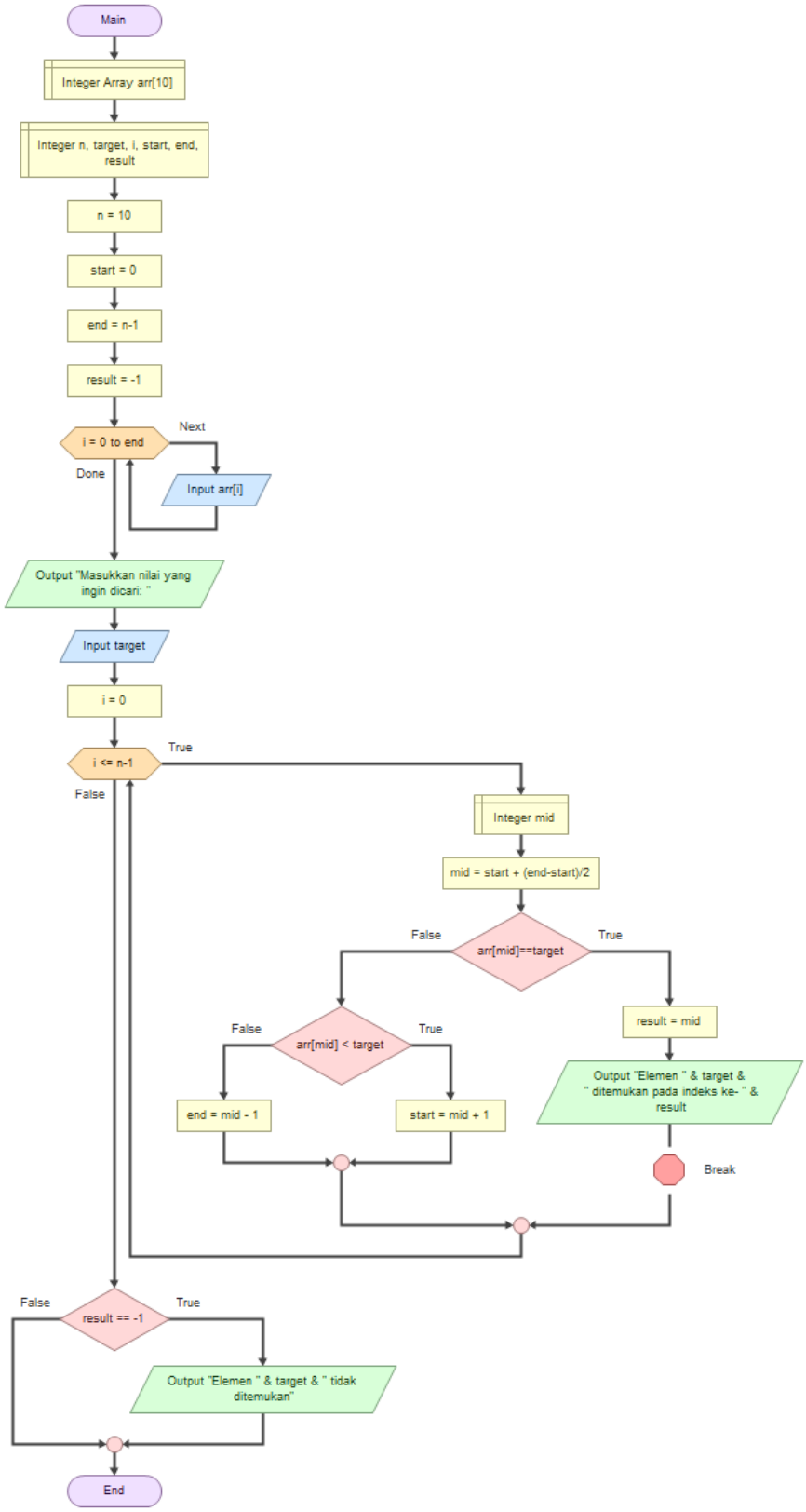
1. Inisialisasi array `arr` dengan elemen-elemen yang sudah terurut secara menaik.
2. Tentukan jumlah elemen dalam array (`n`), dalam kasus ini adalah 10.
3. Tentukan elemen yang ingin dicari, misalnya `target = 23`.
4. Tentukan indeks awal (`start = 0`) dan indeks akhir (`end = n - 1`) dari array.
5. Selama indeks awal kurang dari atau sama dengan indeks akhir, lakukan langkah-langkah berikut:
   a. Hitung indeks tengah (`mid`) sebagai rata-rata dari indeks awal dan indeks akhir.
   b. Jika elemen yang ditemukan di indeks tengah (`arr[mid]`) sama dengan elemen yang dicari (`target`), maka pencarian berhasil dan kembalikan indeks tengah sebagai hasil pencarian.
   c. Jika elemen yang ditemukan di indeks tengah lebih kecil dari elemen yang dicari, atur indeks awal menjadi `mid + 1` untuk mencari di setengah kanan array.
   d. Jika elemen yang ditemukan di indeks tengah lebih besar dari elemen yang dicari, atur indeks akhir menjadi `mid - 1` untuk mencari di setengah kiri array.
6. Jika elemen tidak ditemukan setelah iterasi selesai, kembalikan nilai -1 sebagai hasil pencarian.
7. Tampilkan pesan sesuai dengan hasil pencarian, apakah elemen ditemukan beserta indeksnya atau tidak ditemukan.

Dalam studi kasus ini, kita menggunakan metode pencarian biner (binary search) karena array sudah terurut. Metode ini dapat mengurangi jumlah iterasi yang diperlukan untuk menemukan elemen dibandingkan dengan pencarian linear dalam array yang tidak terurut.

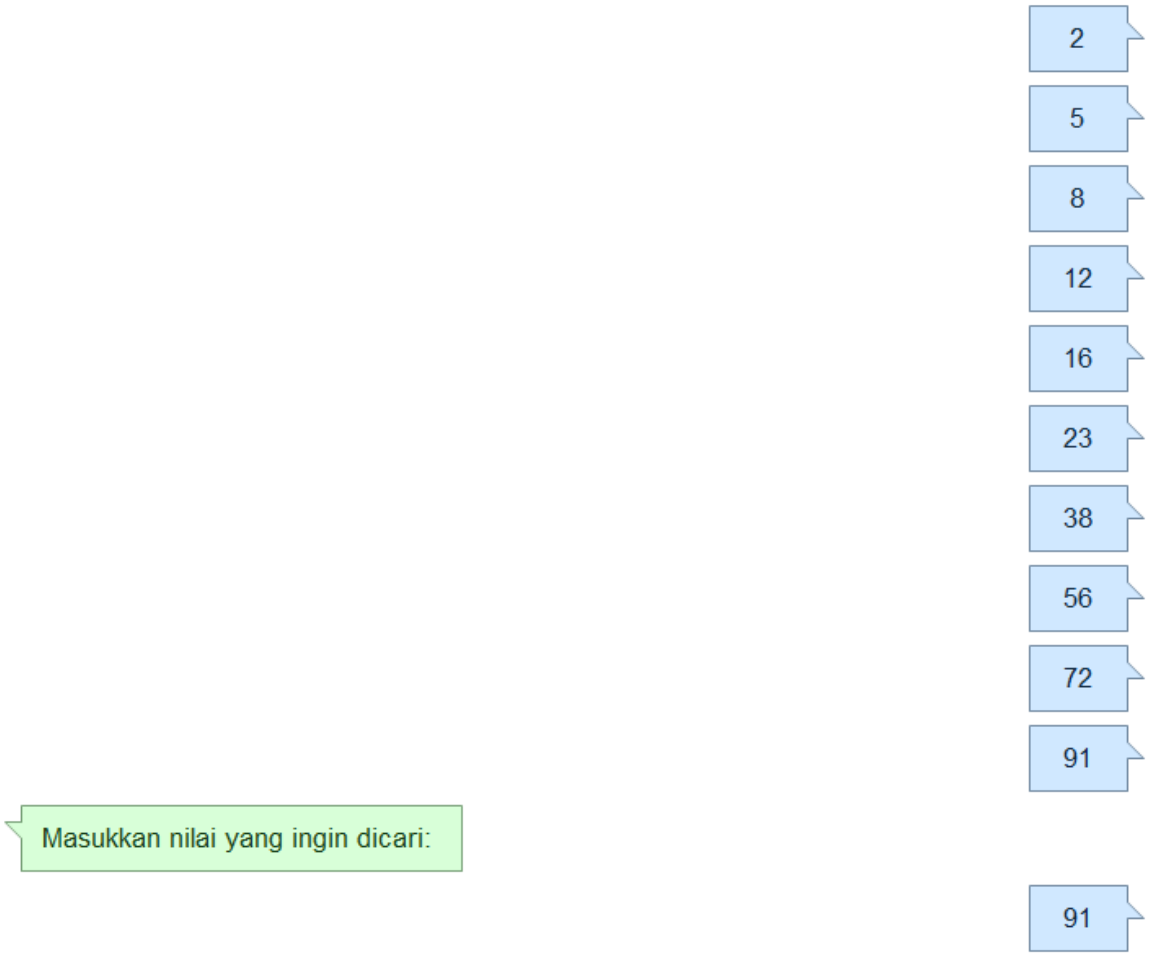
Notasi Algoritmik

```
ALGORITMA CariElemenTerurut()
    ARRAY arr[MAX_ELEMS] // Inisialisasi array yang sudah terurut
    n ← Jumlah elemen dalam array
    start ← 0 // Indeks awal
    end ← n - 1 // Indeks akhir
    found ← 0 // Penanda elemen ditemukan

    TULISKAN "Masukkan elemen yang ingin dicari:"
    BACA elemenCari

    SELAMA start ≤ end DAN found == 0
        mid ← (start + end) / 2 // Hitung indeks tengah
        JIKA arr[mid] == elemenCari MAKA
            TULISKAN "Elemen 'elemenCari' ditemukan pada indeks mid."
            found ← 1 // Set penanda elemen ditemukan
        JIKA arr[mid] < elemenCari MAKA
            start ← mid + 1 // Geser indeks awal
        SELAIN ITU
            end ← mid - 1 // Geser indeks akhir
    SELESAI

    JIKA found == 0 MAKA
        TULISKAN "Elemen 'elemenCari' tidak ditemukan dalam array."
    AKHIR ALGORITMA
```

Flowchart


Main
Integer Array arr[10]
Integer n, target, i, start, end, result
n = 10
start = 0
end = n-1
result = -1
i = 0 to end
Next
Done
Input arr[i]
Output "Masukkan nilai yang ingin dicari: "
Input target
i = 0
i <= n-1
True
False
Integer mid
mid = start + (end-start)/2
arr[mid]==target
False
True
result = mid
Output "Elemen " & target & " ditemukan pada indeks ke- " & result
Break
arr[mid] < target
False
True
end = mid - 1
start = mid + 1
result == -1
False
True
Output "Elemen " & target & " tidak ditemukan"
End

Output Flowchart



Code C

```c
#include <stdio.h>

int main() {
    int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};
    int n = sizeof(arr) / sizeof(arr[0]);
    int target;

    int start = 0;
    int end = n - 1;
    int result = -1;

    printf("Masukkan nilai yang ingin dicari: ");
    scanf("%d", &target);

    while (start <= end) {
        int mid = start + (end - start) / 2;

        if (arr[mid] == target) {
            result = mid;
            break;
        } else if (arr[mid] < target) {
            start = mid + 1;
        } else {
            end = mid - 1;
        }
    }

    if (result != -1) {
        printf("Elemen %d ditemukan pada indeks %d.\n", target,
result);
    } else {
        printf("Elemen %d tidak ditemukan.\n", target);
    }

    return 0;
}
```

Output