Pertama, inisialisasi jumlah\_disk dan array yang disesuaikan

Misal jika

Int jumlah\_disk = 3

Maka nilai masing masing tower berjumlah 3 , seperti berikut ini

Int A[3] = {1, 2, 3}

Int B[3] = {0, 0, 0}

Int C[3] = {0, 0, 0}

Int apop, bpop, dan cpop digunakan untuk menampung nilai terakhir dari array yang bukan = 0

Int indeks\_a,indeks\_b, dan indeks\_c digunakan agar langkah yang dibuat sesuai dengan kaidah tower-of-hanoi dimana disk dengan radius terkecil harus selalu di atas disk dengan radius terbesar. Dalam kasus ini radius disk atau ukuran disk direpresentasikan dengan angka, yaitu sebagai berikut.

A[jumlah\_disk] = {1, 2, 3} dengan 1 adalah disk dengan radius/ukuran terbesar diikuti dengan 2 yang lebih kecil dan 3 yang ukurannya terkecil.

Untuk memudahkan dalam memodifikasi kode maka dibuat sebuah outpun sebelum, dan output pada setiap langkah yang akan dijalankan.

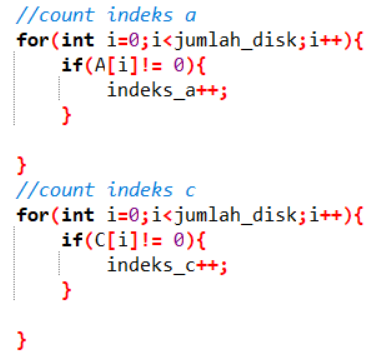
Selanjutnya maka program akan berjalan dengan cara berikut ini.

Jika jumlah\_disk yang dimasukan ganjil ( if (jumlah\_disk%2 != 0)) maka akan dijalankan kondisi perpindahan dari A ke C atau sebaliknya, lalu A ke B atau sebaliknya, dan C ke B atau sebaliknya. Adapun jika jumlah\_disk yang dimasukan genap (jumlah\_disk%2 == 0 atau bisa cukup diganti dengan else) maka maka akan dijalankan kondisi perpindahan dari A ke B atau sebaliknya, lalu A ke C atau sebaliknya, dan C ke B atau sebaliknya.

Kondisi perpindahan tersebut lebih detailnya seperti ini.

While(true) digunakan untuk memulai perulangan sampai terjadi kondisi false maka akan berhenti. Disini akan ada penambahan variabel loop untuk memeriksa jika kondisi sudah sesuai maka program akan menjadi false dan perulangan akan berhenti.

Hitung indeks tower yaitu jumlah bilangan yang bukan 0 yang akan dipindahkan. Misalnya jika akan memindahkan A ke C atau sebaliknya maka indeks yang dihitung adalah indeks tower A dan C.



Selanjutnya program akan mengambil nilai terakhir array yang akan dipindahkan yaitu apop dan cpop,

cpop = C[indeks\_c-1];

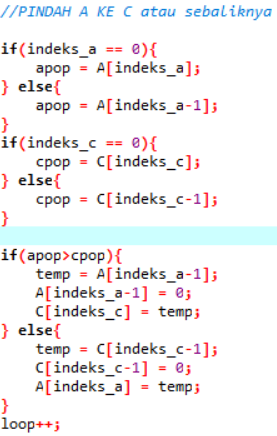
apop = A[indeks\_a-1];

Kemudian akan terjadi kondisi dimana jika apop > cpop maka pindah nilai terakhir array A ke slot setelah nilai terakhir dari array B.

Sebaliknya, Jika apop tidak lebih dari cpop maka maka pindah nilai terakhir array B ke slot setelah nilai terakhir dari array A.

Loop++ digunakan agar loop yang tadinya 0 akan selalu bertambah 1 ketika setiap langkah dijalankan.

Lebih jelasnya kondisinya seperti ini.



Setelah itu outputkan hasilnya agar dapat diketahui setiap langkahnya.

A screenshot of a cell phone

Description automatically generated

Selanjutnya jika jumlah loop sudah memenuhi n (jumlah langkah yang berdasarkan jumlah disk) maka kondisikan seperti berikut agar proses looping berhenti.

A screenshot of a cell phone

Description automatically generated

Kemudian apabila program masih berlanjut set indeks dari masing masing tower menjadi 0, indeks\_a = 0 dan indeks\_b = 0.



Program akan terus berjalan sampai memenuhi kondisi ketika loop == n dimana n adalah banyak langkahnya, maka pengulangan akan dipaksa berhenti dengan kata kunci break.

A screenshot of a cell phone

Description automatically generated