**Bab 16 No. 1**

Pengurutan menaik untuk sekumpulan data integer berikut:

54, 23, 12, 56, 78, 50, 12, 89, 12

**procedure BubbleSort(input/output Arr = LarikInt, input N: integer)**

{Pengurutan menaik elemen larik Arr[1..N] dengan metode pengurutan apung}

{K. Awal: elemen larik Arr sudah terdefinisi nilainya}

{K. Akhir: elemen larik Arr sudah terurut menaik}

**DEKLARASI**

i, j : integer {indeks}

temp : integer

**ALGORITMA**

for i <- 1 to N do

for j <- N downto i+1 do

if Arr[j] < Arr[j-1] then

temp <- Arr[j]

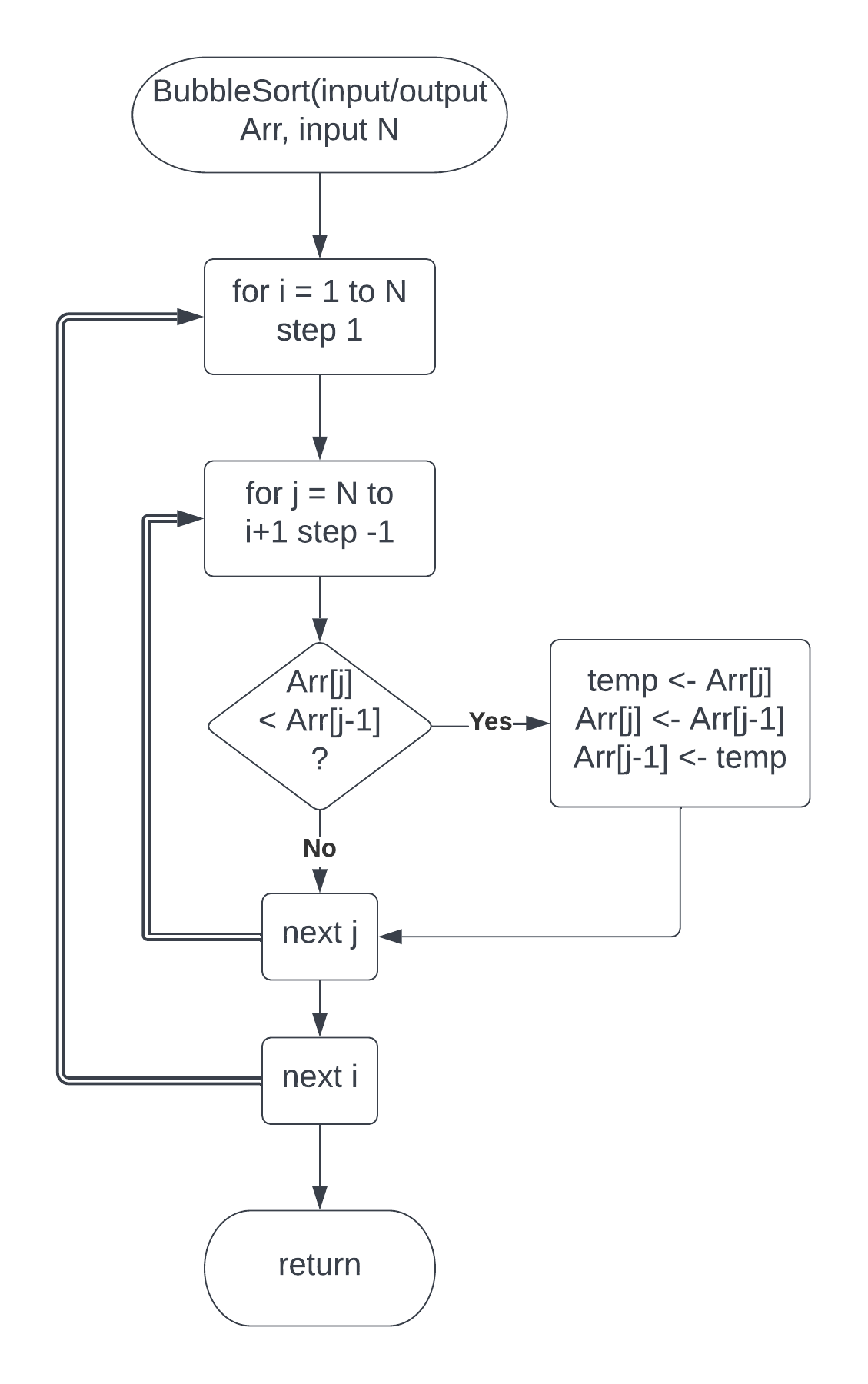
Arr[j] <- Arr[j-1]

Arr[j-1] <- temp

endif

endfor

endfor



**procedure SelectionSort(input/output Arr = LarikInt, input N: integer)**

{Pengurutan menaik dengan metode pengurutan seleski-minimum}

{K. Awal: elemen larik Arr sudah terdefinisi nilainya}

{K. Akhir: elemen larik Arr sudah terurut menaik}

**DEKLARASI**

i, j, min : integer {indeks}

temp = integer

**ALGORITMA**

for i <- 1 to N do

min <- i

for j <- i+1 to N do

if Arr[j] < Arr[min] then

min <- j

endif

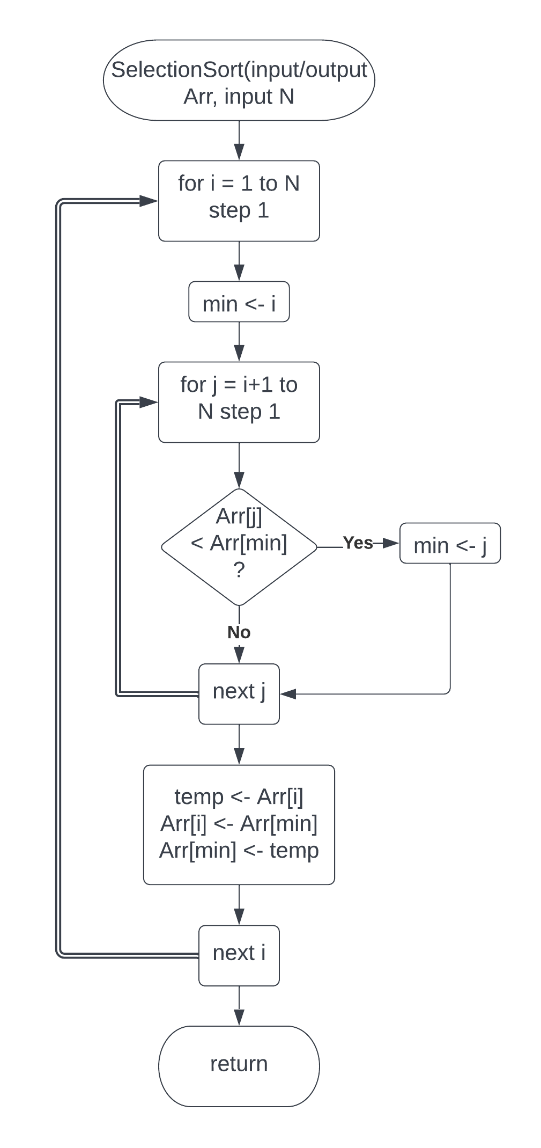
endfor

temp <- Arr[i]

Arr[i] <- Arr[min]

Arr[min] <- temp

endfor



**procedure InsertionSort(input/output Arr = LarikInt, input N: integer)**

{Pengurutan menaik dengan metode pengurutan sisip}

{K. Awal: elemen larik Arr sudah terdefinisi nilainya}

{K. Akhir: elemen larik Arr sudah terurut menaik}

**DEKLARASI**

i, j, key : integer

**ALGORITMA**

for i <- 2 to N do

key <- Arr[i]

j <- i - 1

while (j >= 1) and (Arr[j] > key) do

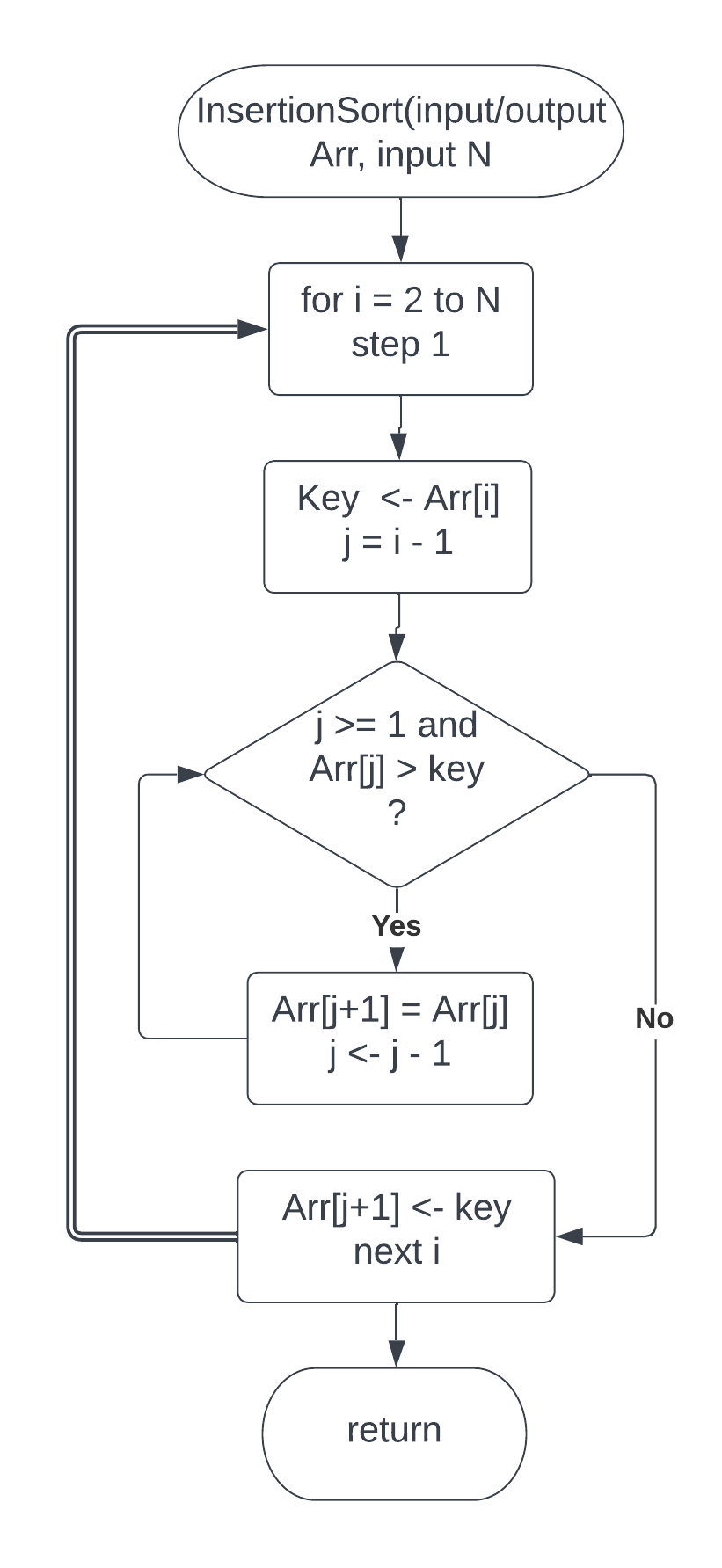
Arr[j+1] <- Arr[j]

j <- j - 1

endwhile

Arr[j+1] <- key

endfor



**procedure ShellSort(input/output Arr = LarikInt, input N: integer)**

{Pengurutan menaik dengan metode pengurutan shell dengan step 5,3,1}

{K. Awal: elemen larik Arr sudah terdefinisi nilainya}

{K. Akhir: elemen larik Arr sudah terurut menaik}

**DEKLARASI**

i, j, step, start : integer

**ALGORITMA**

step <- 5

while step >= 1 do

for start <- 1 to step do

i <- start+step

while i <= N do

key <- Arr[i]

j <- i - step

while (j >= 1) and (Arr[j] > key) do

Arr[j+step] <- Arr[j]

j <- j - step

endwhile

Arr[j+step] <- key

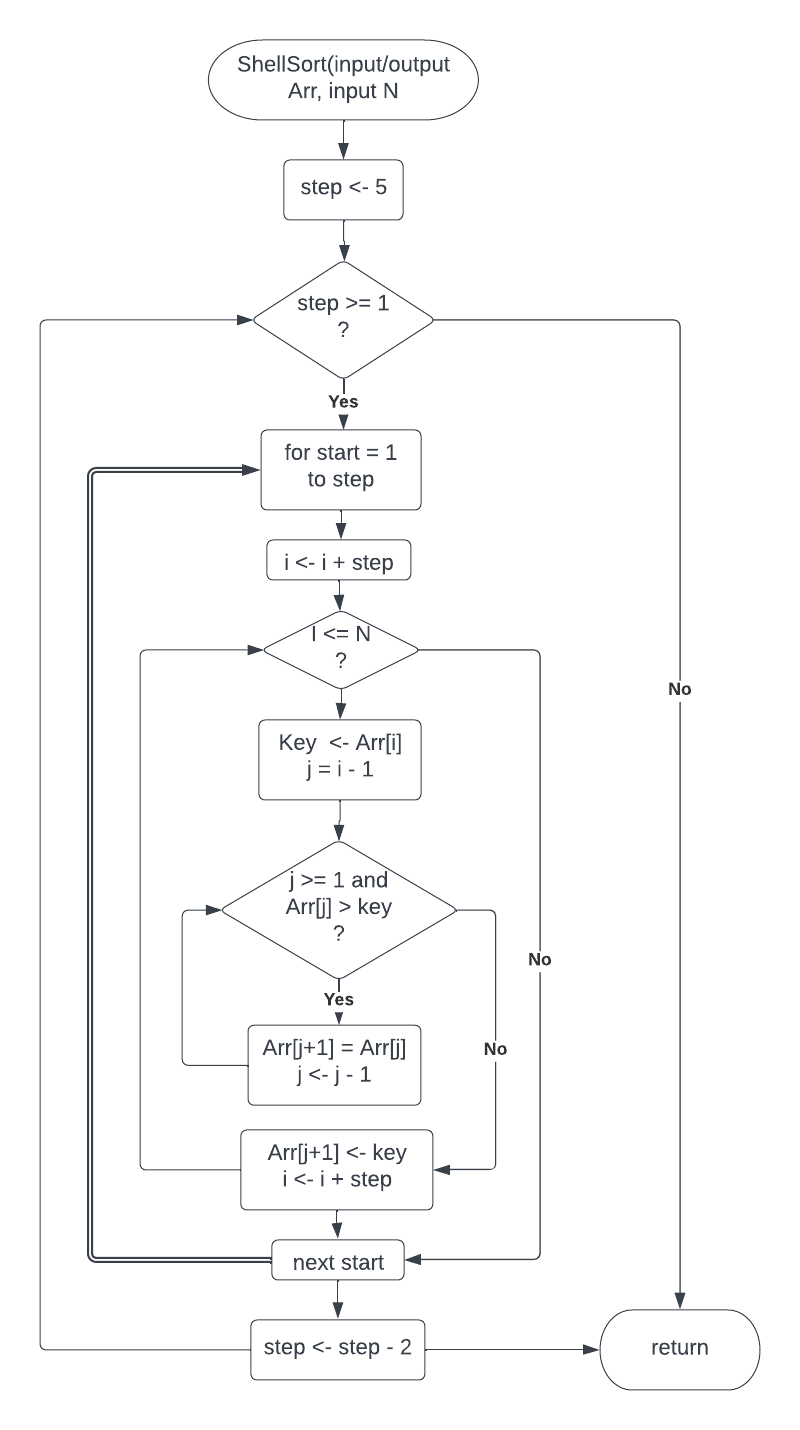
i <- i + step

endwhile

endfor

step <- step - 2

endwhile



**Bab 16 No. 2**

Suatu algoritma dikatakan tidak stabil jika misalnya terdapat elemen 1, 12(1), 0, 4, -1, 12(2), -5, dan di hasil pengurutan secara menaik 12(2) muncul lebih dulu daripada 12(1). Hal ini dapat terjadi karena kita menukar-nukar elemen yang berjauhan. Oleh karena itu, algoritma pengurutan yang stabil adalah algoritma yang tidak menggunakan metode tukar elemen atau menggunakan metode tukar elemen namun hanya menukar elemen dengan elemen tepat di sampingnya. Algoritma tersebut adalah pengurutan apung, pengurutan shell, dan pengurutan sisip. Algoritma pengurutan yang tidak stabil adalah penguruan seleksi.

misalkan 54, 23, 12', 56, 78, 50, 12'', 89, 12'''. Kita akan meninjau hasil pengurutannya.

pada algoritma penguruan sisip, apung, dan shell:

12' 12'' 12''' 23 50 54 56 78 89 (setiap 12 diurutkan sesuai urutan yang tepat)

sedangkan pada algoritma pengurutan seleksi:

12''' 12'' 12' 23 50 54 56 78 89 (urutan angka 12 tidak tepat)

**Bab 16 No. 8**

**program sorting**

{Mengurutkan larik menggunakan metode counting sort (pengurutan pencacahan)}

**DEKLARASI**

NilaiMin, NilaiMaks, N : integer

type LarikInt : array[1..N] of integer

L : LarikInt

i : integer

procedure CountingSort(input/output arr : LarikInt, input N, min, max : integer)

{Mengurutkan larik [1..N] dengan metode pencacahan elemen larik, dimana data telah terdefinisi baik N, min, max, dan elemen larik L}

**ALGORITMA**

CountingSort(L, NilaiMin, NilaiMaks, N)

for i <- to N do

write(L[i])

endfor

**procedure CountingSort(input/output arr : LarikInt, input N, min, max : integer)**

{Mengurutkan larik [1..N] dengan metode pencacahan elemen larik}

{K. Awal: Larik L terdefinisi elemen-elemennya, N, min, dan max terdefinisi}

{K. Akhir: Larik L terurut menaik}

**DEKLARASI**

Result : LarikInt

count : array [min..max] of integer

i, range : integer

**ALGORITMA**

range <- max - min

for i <- 1 to N do {hitung jumlah tiap elemen dalam Larik L}

count[arr[i]-min] <- count[arr[i]-min] + 1

endfor

for i <- 2 to range do {hitung jumlah kumulatif}

count[i] <- count[i-1]

endfor

for i <- range downto 2 do {geser ke kanan sebanyak 1 untuk}

count[i] <- count[i-1] {menentukan posisi awal tiap elemen}

endfor

count[1] <- 1 {elemen diposisikan paling awal}

for i <- 1 to N do

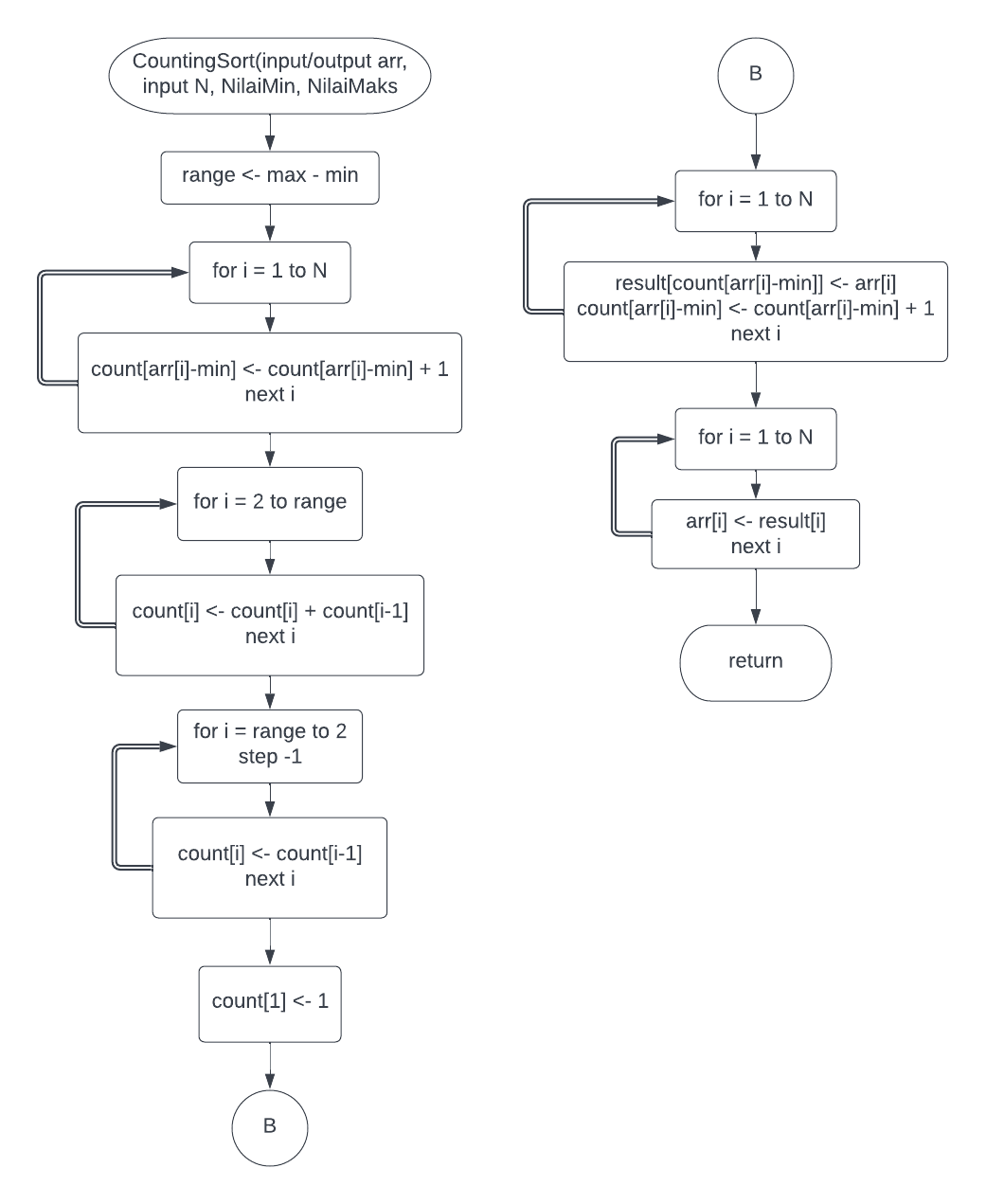
result[count[arr[i]-min]] <- arr[i]

count[arr[i]] <- count[arr[i]] + 1 {posisi awal bertambah 1}

endfor

for i <- 1 to N do

arr[i] <- result[i]

endfor

