**ALGORITMA & STRUKTUR DATA**

**Nama : Andi Farhan Sappewali**

**Nim : D121211078**

**Bab 14**

1. Pada deklarasi yang pertama, dibuat record yang memiliki 2 buah field bertipe real. Lalu dibuat matriks dengan tipe data record tersebut. Jika dibuat dalam kode C, hasilnya:

const NbarisMaks = 5;

const MkolomMaks = 5;

struct TP{

float T;

float P;

}:

struct TP B[NBarisMaks][MkolomMaks];

Sehingga terdapat 25 buah indeks dalam array B. Indeks martriks dapat diakses dengan B[0][0].T;

atau

B[0][0].P;

Sedangkan pada deklarasi yang kedua, dibuat record yang memiliki 2 buah field dengan matriks bertipe data float. Lalu dibuat variabel dengan tipe data struct tersebut. Jika dibuat dalam kode C, hasilnya:

const NbarisMaks = 5;

const NkolomMaks = 5;

struct TP{

float T[NBarisMaks][NKolomMaks];

float P[NbarisMaks][NKolomMaks];

}:

struct TP B;

Indeks matriks dapat diakses dengan :

B.T[0][0];

Atau

B.P[0][0];

6. PROCEDURE JumlahNilai (input m, n : int, NR : array[1..m] of real, NilaiMhs[1..m, 1..n] of Nilai, output NR : array [1..m] of real)

{prosedur untuk memasukkan dan menjumlahkan nilai sks mahasiswa}

{K.Awal : nilai m dan n terdefinisi isinya}

{K.Akhir : larik NR dan NilaiMhs terdefinisi isinya}

DEKLARASI

sigma1 : real {menyimpan nilai total sigma di pembilang}

sigma 2 : real {menyimpan nilai total sigma di penyebut}

ALGORITMA

sigma1 🡨 0

sigma2 🡨 0

{looping untuk menginput nilai dan sks mahasiswa}

for i 🡨 0 to m do

write(“Data Mahasiswa i+1”)

for j 🡨 0 to n do

ulang:

{input indeks nilai dan SKS mata kuliah}

write(“Indeks Nilai Mata Kuliah j+1: ”)

read(NilaiMhs[i][j].indeksNilai)

case NilaiMhs[i][j].indeksNilai

‘A’ ‘a’ : NilaiMhs[i][j].indeksNilai 🡨 4

‘B’ ‘b’ : NilaiMhs[i][j].indeksNilai 🡨 3

‘C’ ‘c’ : NilaiMhs[i][j].indeksNilai 🡨 2

‘D’ ‘d’ : NilaiMhs[i][j].indeksNilai 🡨 1

‘E’ ‘e’ : NilaiMhs[i][j].indeksNilai 🡨 0

otherwise : goto ulang

write(“SKS Mata Kuliah j+1 : “)

read(NilaiMhs[i][j].SKS)

{menjumlahkan nilai sigma1 dan sigma 2}

sigma1 🡨 sigma1 + (NilaiMhs[i][j].indeksNilai \* NilaiMhs[i][j].SKS)

sigma2 🡨 sigma2 + NilaiMhs[i][j].SKS

endfor

{menjumlahkan nilai rata-rata}

NR[i] 🡨 sigma1 / sigma 2

endfor

END

PROGRAM HitungNilaiRata-rata

{program untuk menghitung nilai rata-rata mahasiswa}

DEKLARASI

m : int {jumlah mahasiswa}

n : int {jumlah mata kuliah}

Nilai : record < SKS : integer, indeksNilai : integer >

NilaiMhs : array [1..m, 1..n] of Nilai

NR : array [1..m] of real

procedure JumlahNilai (input m, n : integer, NR : array [1..m] of real, NilaiMhs : array[1..m, 1..n] of Nilai, output NR : array[1..m] of real)

{prosedur untuk memasukkan dan menjumlahkan nilai dan sks mahasiwa}

ALGORITMA

write(“Jumlah Mahasiswa : ”)

read(m)

write(“Jumlah Mata Kuliah: ”)

read(n)

JumlahNilai(NR, NilaiMhs, m,n)

{looping untuk mencetak nilai rata-rata tiap mahasiswa}

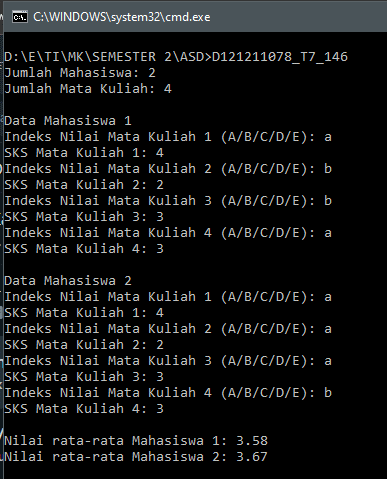
for i 🡨 0 to m do

write(“Nilai rata-rata Mahasiswa i+r : NR[i]\n”)

endfor

END

**Hasil Program**



**FLOWCHART**

Start Prosedure JumlahNilai

1

case ‘c’ ‘C’

case ‘b’ ‘B’

case ‘a’ ‘A’

read(NilaiMhs[i][j].indeksNilai)

write(“Data Mahasiswa i+1”)

j = 0 to n

write(“Data Mahasiswa i=1”)

i = 0 to m

sigma1 🡨 0

sigma2 🡨 0

2

True

NilaiMhs[i][j].indeksNilai 🡨 4

False

True

3

NilaiMhs[i][j].indeksNilai 🡨 3

False

True

NilaiMhs[i][j].indeksNilai 🡨 2

False

NR[i] 🡨 sigma1/sigma2

sigma1 🡨 sigma1 + (NilaiMhs[i][j].indeks \* NilaiMhs[i][j].SKS

sigma2 🡨 sigma2 + NilaiMhs[i][j].SKS

read(NilaiMhs[i][j].indeksNilai)

3

write(“Data Mahasiswa i+1”)

3

goto

2

NilaiMhs[i][j].indeksNilai 🡨 0

NilaiMhs[i][j].indeksNilai 🡨 1

False

False

default:

case ‘e’ ‘E’

1

case ‘d’ ‘D’

True

True

True

4

Prosedur JumlahNilai

JumlahNilai

i = 0 to m

4

write(“Nilai rata-rata Mahasiswa %d : %.2f \n”, i+1, NR[i])

write(“Jumlah Mahasiswa: ”)

read(n)

write(“Jumlah Mata Kuliah: ”)

read(m)

**Bab 15**

1. Procedure CariNilai(input A : array[1..n] of int, n : int, x : int, output id : int)

{prosedur untuk mencari nilai x di dalam larik A dan mengisi id dengan index ditemukannya nilai x. Jika tidak ditemukan, maka id diisi dengan -1}

{K.Awal : larik A, n, dan x terdifinisi nilainya}

{K.Awal : id terisi nilainya}

DEKLARASI

i 🡨 integer

ALGORITMA

id 🡨 1

for i 🡨 n-1 down to 0 do

if A[i] = x then

id 🡨 n

break

endif

endfor

END

PROGRAM PencarianBeruntunDariAkhir

{program pencarian beruntun yang memberikan hasil indeks elemen yang larik yang mengandung x, tetapi pencarian dimulai dari elemen terakhir}

DEKLARASI

n : integer

A : array [1..n] of integer

x : integer {nilai yang akan dicari indeksnya}

id : integer {nilai yang akan menjadi output}

ALGORITMA

read(n)

for i 🡨 0 to n do

read (A[i])

endfor

read(x)

CariNilai (A, n, x, id)

{mencetak hasil akhir}

if id != -1 then

write(“Nilai x berada di index ke-“, x, id)

else

write(“Nilai x tidak ditemukan!”)

endif

END

5. Procedure CariNilai(input A : array[1..n] of integer, x, n : integer, output ketemu : boolean, id : integer)

{prosedur untuk mencari nilai x di larik lalu mengubah nilai ketemu menjadi benar dan nilai id terisi dengan k jika didapat nilai x dalam larik}

{K.Awal : larik A, x, dan n terdefinisi nilainya}

{K.Akhir : ketemu menjadi benar atau salah dan id diisi dengan nilai k}

DEKLARASI

i : integer {peubah untuk menentukan indeks awal pencarian}

j : integer {peubah untuk menentukan indeks akhir pencarian}

k : integer {peubah untuk menentukan indeks tengah pencarian}

ALGORITMA

i 🡨 0

j 🡨 n – 1

while (not ketemu) and (i <= j) do

k 🡨 i + (j – i) \* (x – A[i] div A[j] – A[i])

if A[k] = x then

ketemu 🡨 benar

id 🡨 k

else

if A[k] > x then

i 🡨 k + 1

else

j 🡨 k – 1

endif

endif

endwhile

END

PROGRAM PencarianInterpolasi

{program untuk mencari nilai x di suatu larik dengan metode interpolasi. Nilai n, larik A, dan x telah terdefinisi}

DEKLARASI

n : integer {banyaknya elemen larik}

x : integer {nilai yang dicari}

A : array [1..n] of integer

ketemu : boolean

i : integer {peubah untuk menentukan index awal pencarian}

id : integer {index dimana nilai x ditemukan}

ALGORITMA

ketemu 🡨 salah

read(n)

for i 🡨 0 to n do

read(A[n])

endfor

read(x)

CariNilai(A, x, n, ketemu)

if ketemu then

write(“Nilai x terdapat di index ke-“, x, id)

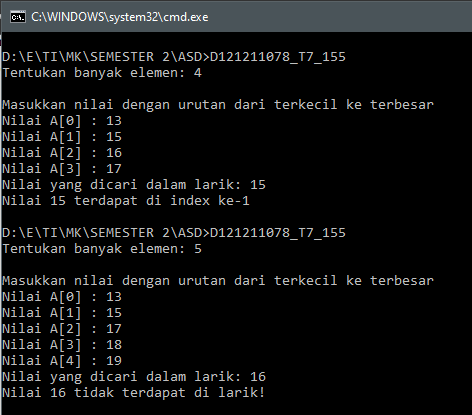
else

write(“Nilai x terdapat di larik!”, x)

endif

END

**Hasil Program**

****

**FLOWCHART**

False

True

ketemu 🡨 benar

id 🡨 k

A[k] = x

K 🡨 i + (j-i) \* (x-A[i] div (A[j] – A[i])

Tidak ketemu && i <= j

i 🡨 0

j 🡨 n-1

True

A[k] > x

else

j 🡨 k - 1

i 🡨 k + 1

False

True

Procedure CariNilai

False

True

True

write(“Nilai x terdapat di larik!“, x)

write(“Nilai x terdapat di index ke-“, x, id)

else

ketemu

CariNilai

read(m)

read(m)

i = 0 to n

read(m)

ketemu 🡨 salah