**BAB V**

**STRUKTUR DATA KOMPOSIT**

**5.1 Pendahuluan**

Dalam bab sebelumnya telah diperkenalkan tipe data yang termasuk dalam kelompok tipe data sederhana (*simple data type*) seperti integer, real dan karakter. Tipe data sederhana adalah data yang nilai datanya hanya terdiri satu data atomik, yaitu data yang hanya terdiri dari sebuah data tunggal yang tidak dapat dipecah-pecah lagi. Dalam banyak situasi, perlu diproses sekelompok data yang saling berhubungan seperti sederetan nilai ujian mahasiswa atau sekumpulan data hasil percobaan atau sebuah matriks. Dalam situasi ini, pemrosesan akan sulit dilakukan jika data tadi merupakan tipe data sederhana yang masing-masing berdiri sendiri. Untuk mengatasi hal ini, disediakan tipe data terstruktur (*structured data type*) seperti array, struct/record dan lainnya.

Dalam bab ini akan diperkenalkan konsep mengenai struktur data. Struktur data didefinisikan sebagai kumpulan dari item data dan bagaimana data tersebut saling berhubungan secara logik. Secara umum struktur data dikelompokkan menjadi struktur data sederhana/primitif dan struktur data komposit/non primitif.

Struktur data primitif adalah struktur data dalam bentuk paling dasar yang sudah dikodekan dalam memori komputer menjadi deretan angka biner. Contohnya integer, real, karakter, logika, pointer, dan *enumeration*. Struktur data non promitif/komposit adalah struktur data yang mengandung lebih dari satu struktur data primitif, atau struktur data yang dapat diuraikan lebih lanjut menjadi lebih dari satu atau struktur data primitif. Contohnya array, struct, file dan sebagainya. Dalam bab ini hanya akan dibahas struktur data komposit yang paling umum digunakan yaitu struktur data array atau dalam bahasa Indonesia disebut sebagai larik.

**5.2 Array**

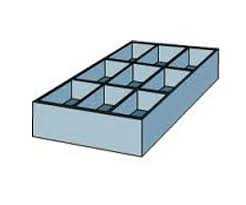
Array atau larik didefinisikan sebagai sejumlah tertentu deretan data yang homogen dan terdiri dari dua macam data yaitu:

1. *Base*/komponen: data yang disimpan dalam array, dapat bertipe integer, real, karakter dan sebagainya.
2. *Subscript*/indeks: untuk menunjukkan posisi data dalam array, bertipe integer, selalu dimulai dari angka 0.

Dari definisi di atas, dapat disimpulkan bahwa data yang disimpan dalam array:

1. tertentu jumlahnya (biasanya lebih dari satu),
2. mempunyai urutan, ada data yang pertama, kedua dan seterusnya
3. tipe datanya sama (homogen).

Array dapat dibayangkan sebagai suatu wadah besar yang mempunyai sekat-sekat tempat menyimpan data. Tiap sekat hanya dapat menyimpan sebuah data. Masing-masing sekat mempunyai nomor (indeks) yang berurutan sehingga suatu data diambil atau disimpan berdasarkan nomor tersebut (lihat gambar 5.1).



8

7

9

4

5

6

3

2

1

Gambar 5.1 Kotak bersekat dengan nomor sekat

Contoh data yang biasanya dibuat sebagai variabel array adalah sejumlah tertentu data NPM mahasiswa (tipe datanya string), kumpulan data hasil percobaan (tipe datanya real) dan sebagainya. Sebagai contoh, variabel NILAI\_MHS adalah variabel array yang menyimpan nilai dari 10 mahasiswa yang tipe data dari nilainya adalah real. Maka ilustrasi dari array ini dapat dilihat pada gambar 5.2 di halaman berikut ini.

NILAI\_MHS:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indeks: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Komponen: | 78.5 | 59.2 | 93.8 | 45.9 | 65.1 | 72.5 | 57.3 | 42.4 | 84.7 | 61.6 |

Gambar 5.2 Ilustrasi Array NILAI\_MHS yang berisi 10 komponen.

Array NILAI\_MHS pada gambar 5.2 berisi 10 data nilai (disebut komponen) yang semuanya bertipe real. Data nilai pertama disimpan dalam indeks 0, lalu berturut-turut data nilai yang kedua dalam indeks 1, dan seterusnya sampai data nilai yang ke 10 dalam indeks 9. Perhatikan, dalam array di atas, indeks dimulai dari 0 sampai 9, karena komponen array ada 10 buah.

Komponen array diidentifikasikan oleh indeksnya atau *subscript* yang ditulis di antara tanda kurung siku setelah nama array. Sebagai contoh untuk mengidentifikasikan komponen yang ketiga dari array NILAI\_MHS yang berada pada indeks 2 maka penulisannya adalah:

NILAI\_MHS[2]

Dengan adanya *subscript* ini array juga dikenal sebagai ***subscripted variable*** (variabel berindeks). Berdasarkan banyaknya indeks dikenal macam-macam array yaitu:

1. Array satu dimensi: array yang mempunyai sebuah indeks, disebut juga vektor atau array linier
2. Array multidimensi: array yang mempunyai lebih dari satu indeks, macamnya adalah matriks (array dua dimensi), array volume (array tiga dimensi), dan seterusnya.

**5.3 Array 1 Dimensi**

Array 1 dimensi adalah array yang hanya mempunyai sebuah indeks. Sama seperti variabel lainnya, array harus dideklarasikan pada bagian spesifikasi program. Bentuk umum untuk mendeklarasikan array adalah:

**tipe\_data nama\_array[limit]**

dengan:

- tipe data : tipe data dari komponen array

- nama\_array: nama dari array. Kriteria membuat nama array sama dengan kriteria untuk membuat identifier

- limit: banyaknya komponen dalam array, merupakan bilangan integer.

Untuk contoh array pada gambar 5.2 maka deklarasinya adalah:

REAL NILAI\_MHS[10]

Tipe data Nama array Banyaknya komponen

Contoh deklarasi array yang lain adalah:

a. CHAR JWB[40] 🡪 JWB adalah variabel array yang mempunyai 40 komponen yang bertipe karakter.

b. REAL NILAI[60] 🡪 NILAI adalah variabel array yang mempunyai 60 komponen yang bertipe real

c. INTEGER KODE[110] 🡪 KODE adalah variabel array yang mempunyai 110 komponen yang bertipe integer

Setelah instruksi yang mendeklarasikan array dieksekusi, maka dalam memori komputer langsung disediakan tempat sebanyak komponen array. Sebagai contoh setelah instruksi **REAL NILAI[60]** dieksekusi maka di dalam memori komputer akan disediakan tempat untuk 60 komponen array NILAI yang masing-masing komponennya bertipe real, seperti gambar 5.3 berikut.

memori komputer:

..

..

..

..

..

..

..

..

..

NILAI[0] ⎫

NILAI[1] ⎪

NILAI[2] ⎪

... ⎪

... ⎬ NILAI

... ⎪

... ⎪

NILAI[59] ⎭

Gambar 5.3 Ilustrasi memori komputer untuk array REAL NILAI[60]

Tempat pertama (NILAI[0]) disediakan untuk komponen array NILAI yang pertama, tempat yang kedua (NILAI[1]) untuk komponen array NILAI yang kedua, demikian seterusnya berturut-turut sampai komponen array NILAI yang ke 60 pada NILAI[59]. Pada gambar 5.3 di atas, dapat dilihat bahwa tiap komponen array mempunyai tempat sendiri, dengan demikian tiap komponen array dapat diperlakukan sebagai variabel tunggal. Sebagai contoh instruksi:

NILAI[5] = 9.1

Akan menyimpan angka 9.1 ke indeks 5 dari array NILAI, dan instruksi:

WRITE (NILAI[20])

Akan menampilkan data yang disimpan di indeks 20 dari array NILAI, lihat ilustrasi pada gambar 5.4 berikut ini.

NILAI:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indeks: | 0 | .... | 4 | 5 | 6 | … | 20 | 21 | … | 59 |
| Komponen: | 7.5 | …. | 4.4 | 9.1 | 6.9 | … | 7.3 | 8.4 | … | 6.6 |

Gambar 5.4 Ilustrasi komponen array REAL NILAI[60]

# 5.3.1 Indeks array

Keistimewaan array adalah indeksnya dapat berupa variabel integer atau berupa ekspresi aritmatika (integer) yang sederhana seperti berikut :

n, k, k ± n, k \* n, k \* n ± m

dengan n, m = konstanta integer

k = variabel integer

Contoh: A adalah array linier yang bertipe integer, yang juga merupakan Deret Fibonacci dengan ketentuan :

Integer A[10]

A[0] = 0

A[1] = 1

A[J] = A[J – 2] + A[J – 1] untuk J > 1

Ditanya: Buatlah deret Fibonacci untuk 10 komponen yang pertama. Lihat ilustrasi gambar 5.5 di halaman berikut.

A:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Indeks: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Komponen: | 0 | 1 | …. | …. | …. | …. | …. | …. | …. | …. |

Gambar 5.5 Ilustrasi Array A

Ada beberapa hal yang perlu diperhatikan dalam menentukan indeks array yaitu:

1. Banyaknya komponen array harus terbatas.
2. Karena indeks selalu dimulai dari 0 maka maksimum indeks array adalah (banyaknya komponen – 1).
3. Jika array X adalah deretan komponen: X[L], X[L+1], …, X[H] dengan L dan H adalah integer dan L ≤ H maka L disebut batas bawah array X dan H disebut batas atas array X.

Contoh diketahui:

- X[0], X[1], ..., X[8] ⇒ L = 0 ; H = 8

- Y[0], Y[1], Y[2], Y[3] ⇒ L = 0; H = 3

1. Range dapat didefinisikan sebagai banyaknya komponen array

Range = H – L + 1

Contoh:

1. Range dari array X = 8 – 0 + 1 = 9
2. Range dari array Y = 3 – 0 + 1 = 4

# 5.3.2 Operasi input/output dengan array

Untuk memasukkan atau menampilkan sebuah komponen array cukup dengan menuliskan nama array dan indeksnya saja bersama perintah input/ouput (I/O), seperti contoh berikut ini untuk array X di atas:

- Read(X[3]) ⇒ membaca sebuah data lalu disimpan ke indeks 3 dari array X

- Write(X[1]) ⇒ menampilkan komponen yang ada pada indeks 1 dari array X

Untuk memasukkan atau menampilkan seluruh komponen dalam array linier, ada 2 cara yaitu:

a. Menggunakan nama array beserta indeksnya dalam perintah I/O

b. Menggunakan struktur pengulangan yang mengandung perintah I/O

Sebagai contoh diketahui deklarasi array VELO yang berisi data real seperti berikut ini:

REAL VELO[10]

Untuk membaca 10 buah data real dan menyimpannya ke masing-masing indeks pada array VELO maka instruksinya adalah:

a. Dengan menggunakan nama array beserta indeksnya:

READ (VELO[0])

READ (VELO[1])

READ (VELO[2])

READ (VELO[3])

READ (VELO[4])

READ (VELO[5])

READ (VELO[6])

READ (VELO[7])

READ (VELO[8])

READ (VELO[9])

Atau

READ (VELO[0], VELO[1], VELO[2], VELO[3], VELO[4], VELO[5],

VELO[6], VELO[7], VELO[8], VELO[9])

b. Dengan menggunakan instruksi pengulangan:

FOR (I = 0 ; I < 10 ; I ++)

{

READ (VELO[I])

}

Dari kedua cara di atas dapat dilihat bahwa cara yang pertama sama sekali tidak praktis dan dapat menjadi sumber kesalahan terutama jika komponen array sangat banyak. Dengan demikian maka untuk membaca atau menampilkan seluruh komponen array, disarankan untuk menggunakan instruksi pengulangan (cara yang kedua) karena lebih singkat dan lebih mudah dalam menuliskan instruksinya.

# 5.3.3 Perbedaan variabel primitif dan variabel array

Untuk melihat perbedaan antara penggunaan variabel primitif dan variabel array serta kelebihan dari variabel array, akan dibuat algoritma CLASS\_STAT untuk menyelesaikan masalah berikut ini:

Menghitung nilai rata-rata dari sejumlah data mahasiswa (yang terdiri dari nama mahasiswa dan nilainya), lalu mencetak daftar nama dan nilai mahasiswa yang mempunyai nilai akhir di atas nilai rata-rata.

Sebagai contoh, diketahui ada 5 mahasiswa. Dengan menggunakan variabel primitif maka diperlukan 10 variabel untuk menyimpan data mahasiswa yaitu NAME1, NAME2, NAME3, NAME4, dan NAME5 untuk menyimpan data nama serta GRADE1, GRADE2, GRADE3, GRADE4, dan GRADE5 untuk menyimpan data nilai. Dengan demikian algoritma untuk masalah ini adalah sebagai berikut:

Algoritma CLASS\_STAT

Algoritma untuk menghitung nilai rata-rata dari 5 mahasiswa. Variabel NAME1, NAME2, NAME3, NAME4 dan NAME5 bertipe string. Variabel GRADE1, GRADE2, GRADE3, GRADE4, GRADE5 dan AVERAGE bertipe real. Sebagai output adalah nama dan nilai mahasiswa yang nilainya ≥ nilai rata-rata.

1. [membaca data nama dan nilai dari 5 mahasiswa]

Read(NAME1, GRADE1, NAME2, GRADE2, NAME3, GRADE3, NAME4, GRADE4, NAME5, GRADE5)

1. [menghitung rata-rata]

AVERAGE = (GRADE1 + GRADE2 + GRADE3 + GRADE4 + GRADE5)/5

1. [Apakah nilai mahasiswa pertama ≥ nilai rata-rata ?]

If (GRADE1 ≥ AVERAGE)

{ [ya, cetak nama dan nilai mahasiswa]

Write(NAME1, GRADE1) }

1. [Apakah nilai mahasiswa ke dua ≥ nilai rata-rata ?]

If (GRADE2 ≥ AVERAGE)

{ [ya, cetak nama dan nilai mahasiswa]

Write(NAME2, GRADE2) }

1. [Apakah nilai mahasiswa ke tiga ≥ nilai rata-rata ?]

If (GRADE3 ≥ AVERAGE)

{ Write(NAME3, GRADE3) }

1. [Apakah nilai mahasiswa ke empat ≥ nilai rata-rata ?]

If (GRADE4 ≥ AVERAGE)

{ Write(NAME4, GRADE4) }

1. [Apakah nilai mahasiswa ke lima ≥ nilai rata-rata ?]

If (GRADE5 ≥ AVERAGE)

{ Write(NAME5, GRADE5)

}

1. [selesai]

Halt

Ilustrasi flowchart untuk algoritma CLASS\_STAT ini dapat dilihat pada gambar 5.6 di halaman berikut.

Jika banyaknya mahasiswa berubah menjadi 100 orang, dengan cara yang sama seperti di atas, maka diperlukan 200 variabel untuk menyimpan data nama dan nilai mahasiswa. Dengan demikian algoritma di atas perlu direvisi menjadi algoritma CLASS\_STAT\_revisi\_1 seperti berikut:

Algoritma CLASS\_STAT\_revisi\_1

Algoritma untuk menghitung nilai rata-rata dari 100 mahasiswa. Variabel NAME1, NAME2, ..., NAME99 dan NAME100 bertipe string. Variabel GRADE1, GRADE2, ..., GRADE99, GRADE100 dan AVERAGE bertipe real. Sebagai output adalah nama dan nilai mahasiswa yang nilainya ≥ nilai rata-rata.

1 [membaca data nama dan nilai dari 100 mahasiswa]

Read(NAME1, GRADE1, NAME2, GRADE2, NAME3, GRADE3, …, NAME100, GRADE100)

2. [menghitung rata-rata]

AVERAGE = (GRADE1+GRADE2+GRADE3+ …+GRADE100)/100

1. [Apakah nilai mahasiswa pertama ≥ nilai rata-rata ?]

If (GRADE1 ≥ AVERAGE)

{ [ya, cetak nama dan nilai mahasiswa]

Write(NAME1, GRADE1)

}

1. [Apakah nilai mahasiswa ke dua ≥ nilai rata-rata ?]

If (GRADE2 ≥ AVERAGE)

{ [ya, cetak nama dan nilai mahasiswa]

Write(NAME2, GRADE2)

}

….

….

102. [Apakah nilai mahasiswa ke 100 ≥ nilai rata-rata ?]

If (GRADE100 ≥ AVERAGE)

{ [ya, cetak nama dan nilai mahasiswa]

Write(NAME100, GRADE100)

}

103. [selesai]

Halt

Tanda titik-titik (......) pada algoritma CLASS\_STAT\_revisi\_1 di atas hanya dimaksudkan untuk menyingkat penulisan di dalam buku ini. Pada algoritma yang sebenarnya seluruh instruksi harus ditulis dengan lengkap.



Gambar 5. 6 Flowchart dari algoritma CLASS\_STAT

Algoritma CLASS\_STAT\_revisi1 di atas kelihatan tidak fleksibel dan tidak praktis terutama untuk jumlah mahasiswa yang bervariasi karena setiap kali ada perubahan jumlah mahasiswa maka algoritma tersebut harus ditulis ulang. Untuk mengatasi masalah ini, data nama dan nilai mahasiswa disimpan dalam variabel array NAME dan GRADE. Untuk menunjuk ke suatu data digunakan **indeks** yang ditulis di antara kurung siku. Keistimewaan variabel array ini adalah indeks array boleh berupa variabel. Dengan demikian algoritma CLASS\_STAT direvisi menjadi algoritma CLASS\_STAT\_revisi\_2 yang menggunakan variabel array untuk menyimpan data nama dan nilai mahasiswa.

Algoritma CLASS\_STAT\_revisi\_2

Algoritma untuk menghitung nilai rata-rata dari 5 mahasiswa. Data nama mahasiswa disimpan dalam variabel array NAME[5] yang bertipe karakter string. Data nilai mahasiswa disimpan dalam variabel array GRADE[5] yang bertipe real. Variabel NUMSTUDENT dan K bertipe integer. Variabel SUM dan AVERAGE bertipe real. Sebagai output adalah nama dan nilai mahasiswa yang nilainya ≥ nilai rata-rata.

[deklarasi array]

CHARACTER NAME[5]

REAL GRADE[5]

1. [Inisialisasi jumlah mahasiswa = 5]

NUM\_STUDENT = 5

1. [inisialisasi variabel SUM]

SUM = 0

1. [membuat loop untuk membaca data 5 mahasiswa]

For (K = 0 ; K < NUM\_STUDENT ; K ++)

3.a. { [baca data mahasiswa ke K]

Read(NAME[K],GRADE[K])

3.b. [menjumlahkan nilai mahasiswa]

SUM = SUM + GRADE[K]

}

1. [menghitung rata-rata]

AVERAGE = SUM / NUM\_STUDENT

1. [membuat loop untuk memeriksa nilai dan menampilkan data mahasiswa]

For (K = 0 ; K < NUM\_STUDENT ; K++)

5.a. { [Apakah nilai mahasiswa ke K ≥ nilai rata-rata ?]

If ( GRADE[K] ≥ AVERAGE )

{ [ya, cetak nama dan nilai mahasiswa]

Write(NAME[K],GRADE[K]) }

}

1. [selesai]

Halt

Perhatikan deklarasi untuk variabel NAME dan GRADE yang merupakan variabel array. Perhatikan pula serta instuksi untuk membaca data (langkah 3) dan proses untuk menentukan output (langkah 5), semuanya menggunakan instruksi pengulangan dan variabel sebagai indeks array. Ilustrasi flowchart untuk algoritma CLASS\_STAT\_revisi\_2 ini dapat dilihat pada gambar 5.7 di bawah ini.

Gambar 5.7 Flowchart dari Algoritma CLASS\_STAT revisi 2

Untuk lebih memahami algoritma di atas, dimasukkan contah data nama dan grade berikut ini untuk diproses dan dibuat trace table-nya.

Nomor Nama Grade

1 Ani 7.2

2 Ben 8.3

3 Cie 9.5

4 Dik 8.6

5 Eis 7.4

Tabel 5.1 Trace table Algoritma CLASS\_STAT\_revisi2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No | Na me [0] | Na me [1] | Na me [2] | Na me [3] | Na ma [4] | Grade [0] | Grade [1] | Grade [2] | Grade [3] | Grade [4] | Num\_student | Sum | K | For? | Average | IF? | Out put |
| 1 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | 5 | ? | ? |  | ? |  |  |
| 2 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | 5 | 0 | ? |  | ? |  |  |
| 3 | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? | 5 | 0 | 0 | T | ? |  |  |
| 3.a | Ani | ? | ? | ? | ? | 7.2 | ? | ? | ? | ? | 5 | 0 | 0 |  | ? |  |  |
| 3.b | Ani | ? | ? | ? | ? | 7.2 | ? | ? | ? | ? | 5 | 7.2 | 0 |  | ? |  |  |
| 3 | Ani | ? | ? | ? | ? | 7.2 | ? | ? | ? | ? | 5 | 7.2 | 1 | T | ? |  |  |
| 3.a | Ani | Ben | ? | ? | ? | 7.2 | 8.3 | ? | ? | ? | 5 | 7.2 | 1 |  | ? |  |  |
| 3.b | Ani | Ben | ? | ? | ? | 7.2 | 8.3 | ? | ? | ? | 5 | 15.5 | 1 |  | ? |  |  |
| 3 | Ani | Ben | ? | ? | ? | 7.2 | 8.3 | ? | ? | ? | 5 | 15.5 | 2 | T | ? |  |  |
| 3.a | Ani | Ben | Cie | ? | ? | 7.2 | 8.3 | 9.5 | ? | ? | 5 | 15.5 | 2 |  | ? |  |  |
| 3.b | Ani | Ben | Cie | ? | ? | 7.2 | 8.3 | 9.5 | ? | ? | 5 | 25.0 | 2 |  | ? |  |  |
| 3 | Ani | Ben | Cie | ? | ? | 7.2 | 8.3 | 9.5 | ? | ? | 5 | 25.0 | 3 | T | ? |  |  |
| 3.a | Ani | Ben | Cie | Dik | ? | 7.2 | 8.3 | 9.5 | 8.6 | ? | 5 | 25.0 | 3 |  | ? |  |  |
| 3.b | Ani | Ben | Cie | Dik | ? | 7.2 | 8.3 | 9.5 | 8.6 | ? | 5 | 33.6 | 3 |  | ? |  |  |
| 3 | Ani | Ben | Cie | Dik | ? | 7.2 | 8.3 | 9.5 | 8.6 | ? | 5 | 33.6 | 4 | T | ? |  |  |
| 3.a | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 33.6 | 4 |  | ? |  |  |
| 3.b | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 4 |  | ? |  |  |
| 3 | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 5 | F | ? |  |  |
| 4 | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 5 |  | 8.2 |  |  |
| 5 | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 0 | T | 8.2 |  |  |
| 5.a | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 0 |  | 8.2 | F |  |
| 5 | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 1 | T | 8.2 |  |  |
| 5.a | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 1 |  | 8.2 | T | Ben 8.3 |
| 5 | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 2 | T | 8.2 |  |  |
| 5.a | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 2 |  | 8.2 | T | Cie 9.5 |
| 5 | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 3 | T | 8.2 |  |  |
| 5.a | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 3 |  | 8.2 | T | Dik 8.6 |
| 5 | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 4 | T | 8.2 |  |  |
| 5.a | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 4 |  | 8.2 | F |  |
| 5 | Ani | Ben | Cie | Dik | Eis | 7.2 | 8.3 | 9.5 | 8.6 | 7.4 | 5 | 41.0 | 5 | F | 8.2 |  |  |
| 6 | Selesai | | | | | | | | | | | | | | | |  |

Untuk mengantisipasi adanya perubahan dalam banyaknya mahasiswa, isi variabel NUM\_STUDENT diinput ketika algoritma dijalankan sehingga jika ada perubahan dalam banyaknya mahasiswa, algoritma tidak perlu ditulis ulang. Dalam hal ini, yang perlu diperhatikan adalah banyaknya komponen array harus dideklarasikan ketika algoritma dibuat. Programmer harus mempertimbangkan maksimum banyaknya mahasiswa yang akan diproses oleh algoritma ini. Misalnya algoritma akan digunakan untuk memroses data mahasiswa dalam satu kelas. Berdasarkan data yang sudah ada, banyaknya mahasiswa dalam satu kelas tidak lebih dari 100 orang. Dengan demikian angka 100 dapat digunakan sebagai banyaknya komponen array NAME dan GRADE ketika dideklarasikan. Hasil revisi dapat dilihat pada algoritma CLASS\_STAT\_revisi\_3 berikut ini dan ilustrasi flowchart-nya dapat dilihat pada gambar 5.8 di halaman berikut.

Algoritma CLASS\_STAT\_revisi\_3

Algoritma untuk menghitung nilai rata-rata dari maksimum 100 mahasiswa. Data nama mahasiswa disimpan dalam variabel array NAME[100] yang bertipe karakter string. Data nilai mahasiswa disimpan dalam variabel array GRADE[100] yang bertipe real. Variabel NUMSTUDENT dan K bertipe integer. Variabel SUM dan AVERAGE bertipe real. Sebagai output adalah nama dan nilai mahasiswa yang nilainya ≥ nilai rata-rata.

[deklarasi array]

CHARACTER NAME[100]

REAL GRADE[100]

1. [Memasukkan data banyaknya mahasiswa, maksimum 100]

Read(NUM\_STUDENT)

1. [inisialisasi variabel SUM]

SUM = 0

1. [membuat loop untuk membaca data mahasiswa]

For ( K = 0 ; K < NUM\_STUDENT ; K++ )

{

3.a. [baca data mahasiswa ke K]

Read(NAME[K],GRADE[K])

3.b. [menjumlahkan nilai mahasiswa]

SUM = SUM + GRADE[K]

}

1. [menghitung rata-rata]

AVERAGE = SUM / NUM\_STUDENT

1. [membuat loop untuk memeriksa nilai mahasiswa]

For ( K = 0 ; K < NUM\_STUDENT ; K++ )

{

5.a . [Apakah nilai mahasiswa ke K ≥ nilai rata-rata ?]

If ( GRADE[K] ≥ AVERAGE )

{ Write(NAME[K], GRADE[K])

}

}

1. [selesai]

Halt

Gambar 5.8 Flowchart untuk Algoritma CLASS\_STAT revisi 3

Dengan menggunakan struktur data array untuk menyimpan data nama dan nilai mahasiswa, dapat dilihat bahwa algoritma CLASS\_STAT\_revisi\_3 ini lebih singkat dan efisien dibandingkan dengan algoritma CLASS\_STAT dan algoritma CLASS\_STAT\_revisi\_1 yang menggunakan struktur data primitif untuk menyimpan data. Algoritma ini dapat menampung perubahan banyaknya mahasiswa tanpa perlu mengubah instruksi program, selama banyaknya mahasiswa yang akan diproses tidak lebih dari 100 orang karena array yang dideklarasikan hanya dapat menampung 100 komponen (lihat bagian deklarasi array). Selain itu, dengan diperbolehkannya penggunaan variabel sebagai indeks array, maka pembacaan dan pemrosesan data serta penulisan hasil (lihat langkah 3 dan langkah 5) menjadi lebih singkat.

Sebagai latihan, tambahkan dalam algoritma di atas perintah untuk mencari dan menampilkan nama dan nilai mahasiswa yang terbesar serta nama dan nilai mahasiswa yang terkecil. Bandingkan panjang algoritma jika data nama dan nilai mahasiswa tersebut tidak disimpan dengan menggunakan array.

# 5.3.4 Contoh penggunaan array 1 dimensi

Sebagai contoh penggunaan variabel array, akan dibuat algoritma dan flowchart untuk membaca data, menghitung rata-rata nilai data dan mencari nilai terbesar dari data yang dimasukkan. Diasumsikan banyaknya data yang dimasukkan tidak lebih dari 500 data sehingga array dideklarasikan untuk menampung maksimum 500 data.

Algoritma HITUNG

Algoritma untuk menerima data yang bertipe real (maksimum 500 buah) dan menyimpannya dalam array DATA, lalu menghitung rata-rata, mencari data dengan nilai terbesar dan menampilkan hasilnya. Variabel n untuk menyimpan banyaknya data, bertipe integer. Variabel JUM, RATA, MAKS digunakan berturut-turut untuk menyimpan hasil penjumlahan, rata-rata data dan nilai maksimum data. Semua variabel tersebut bertipe real.

[deklarasi array]

Real DATA[500]

1. [membaca banyaknya data yang akan diproses, maksimum 500 buah]

Read (n)

2. [inisialisasi variabel jum dan maks]

JUM = 0

MAKS = 0

3. [membuat loop untuk menerima data dan menyimpannya ke array DATA, serta menjumlahkannya]

For (i = 0 ; i < n ; i++)

{ Read (DATA[i])

JUM = JUM + DATA[i]

}

4. [menghitung rata-rata]

RATA = JUM / n

5. [membuat loop untuk mencari nilai maksimum]

For (i = 0 ; i < n ; i++)

{ If (MAKS < DATA[i])

{ MAKS = DATA[i]

}

}

6. [menampilkan hasil]

Write(“Rata-rata dari data yang dimasukkan = “, RATA)

Write(“Nilai maksimum dari data yang dimasukkan = “, MAKS)

1. [selesai]

Halt

Ilustrasi flowchart untuk algoritma di atas dapat dilihat pada gamber 5.9 di halaman berikut.

Sebagai contoh berikutnya dari penggunaan variabel array ini, akan dibuat algoritma dan flowchart untuk menghitung jawaban dari sebuah hasil survey. Diasumsikan banyaknya responden ada 40 orang. Tiap responden menjawab “Y” jika ia setuju dengan pernyataan, atau “T” jika ia tidak setuju dengan pernyataan atau “A” jika ia tidak mempunyai pendapat atas pertanyaan yang diajukan (tidak menjawab). Jawaban responden disimpan dalam array JWB[40] yang bertipe character. Algoritma akan menyimpan jawaban seluruh responden lebih dulu, kemudian dianalisis hasil surveynya. Hasil analisis adalah banyaknya dan persentase responden yang menjawab Y, T atau A.

Algoritma Survey

Algoritma untuk menyimpan data hasil survey yang bertipe charakter (sebanyak 40 buah) dan menyimpannya dalam array JWB, lalu menghitung banyaknya responden yang menjawab Y, atau T atau A serta persentasi dari masing-masing jawaban. Variabel i untuk pengontrol loop, bertipe integer. Variabel JUM1, JUM2, dan JUM3 bertipe integer dan digunakan berturut-turut untuk menyimpan banyaknya responden yang menjawab Y, T, dan A. Variabel P1, P2, dan P3 bertipe real dan digunakan untuk menyimpan persentasi dari responden yang menjawab Y, T dan A.

[deklarasi array]

Character JWB[40]



Gambar 5.9 Ilustrasi flowchart untuk algoritma Hitung

1. [inisialisasi variabel]

JUM1 = 0

JUM2 = 0

JUM3 = 0

2. [membuat loop untuk menyimpan hasil survey ke array JWB]

For (i = 0 ; i < 40 ; i++)

{ Read (JWB[i])

}

3. [membuat loop untuk menganalisis hasil survey]

For (i = 0 ; i < 40 ; i++)

{

3.a. If (JWB[i] == “Y”)

3.a.i. { JUM1= JUM1+1

}

Else

3.a.ii { If (JWB[i] == “T”)

{ JUM2= JUM2+1

}

Else

{ JUM3 = JUM3 + 1

}

}

}

4. [Menghitung persentasi jawaban]

P1 = JUM1\*100/40

P2 = JUM2\*100/40

P3 = JUM3\*100/40

5. [menampilkan hasil]

Write (“Hasil survey terhadap 40 responden”)

Write (“Banyaknya responden yang menjawab YA = “, JUM1, “ atau “, P1, “%”)

Write (“Banyaknya responden yang menjawab TIDAK = “, JUM2, “ atau “, P2, “%”)

Write (“Banyaknya responden yang tidak menjawab = “, JUM3, “ atau “, P3, “%”)

6. [selesai]

Halt

Flowchart untuk algoritma SURVEY ini dapat dilihat pada gambar 5.10 di halaman berikut.

Sebagai contoh yang lain dari penggunaan struktur data ini akan dibuat algoritma untuk membaca data curah hujan dari suatu stasiun pengamat cuaca dan mengelompokkan stasiun pengamat cuaca berdasarkan curah hujan dan menghitung banyaknya stasiun cuaca dalam suatu kelompok sesuai contoh tabel 5.2 di bawah ini. Diasumsikan curah hujan yang diukur oleh seluruh stasiun berada di antara 0 sampai 49 cm.

Tabel 5.2 Pengelompokan Stasiun Berdasarkan Curah Hujan

Kelompok Curah Hujan (cm)

* 1. 0 – 4
  2. 5 – 9
  3. 10 – 14
  4. 15 – 19
  5. 20 – 24
  6. 25 – 29
  7. 30 – 34
  8. 35 – 39
  9. 40 – 44
  10. 45 – 49



Gambar 5.10 Flowchart untuk algoritma SURVEY

Sebagai input adalah nama stasiun dan curah hujan yang diukurnya. Contoh data input dapat dilihat pada tabel 5.3 di bawah ini. Diasumsikan semua data bertipe integer. Sebagai output adalah nomor kelompok dan banyaknya stasiun yang ada dalam kelompok tersebut, format output dapat dilihat pada tabel 5.4 di bawah ini. Untuk mengetahui kelompok dari suatu stasiun digunakan statement IF bertingkat. Kondisi yang digunakan untuk mengelompokkan stasiun adalah range curah hujan pada tabel 5.2 di atas.

Tabel 5.3 Contoh input untuk pengelompokan stasiun pengamat cuaca

Nama Stasiun Curah Hujan, cm

Jakarta 23

Depok 31

Surabaya 18

Bandung 32

Bogor 48

… ....

… ....

Tabel 5.4 Contoh output hasil pengelompokan stasiun pengamat cuaca

Kelompok Banyak Stasiun

1 ....

2 ....

3 ....

... ....

.... ....

10 ....

Algoritma KELOMPOK\_STASIUN

Algoritma untuk membaca curah hujan dari setiap stasiun pengamat cuaca, lalu mengelompokkannya berdasarkan tabel 5.2 dan menampilkan banyaknya stasiun pengamat cuaca yang ada dalam tiap kelompok. Variabel NAMA\_STASIUN adalah variabel string untuk menyimpan nama stasiun pengamat cuaca. Variabel CURAH\_HUJAN adalah variabel integer untuk menyimpan data curah hujan. Variabel array KELOMPOK[9] bertipe integer untuk menyimpan banyaknya stasiun dalam suatu kelompok, indeksnya merupakan kelompok stasiun dan komponennya menyimpan banyaknya stasiun dalam kelompok tersebut. Data nama stasiun dan curah hujan disimpan dalam file sehingga digunakan sentinel eof untuk mengakhiri pembacaan data. K adalah variabel integer.

[deklarasi array]

INTEGER KELOMPOK[9]

1. [inisialisasi array KELOMPOK, semua komponennya angka 0]

For ( K = 0 ; K <= 9 ; K++ )

{ KELOMPOK[K] =0

}

1. [membaca data nama stasiun dan curah hujan]

Read (NAMA\_STASIUN, CURAH\_HUJAN)

1. [membuat pengulangan untuk menentukan kelompok dan membaca data berikutnya]

While ( not eof )

{

3.a [menentukan kelompok berdasarkan curah hujan]

If ( CURAH\_HUJAN <=4 )

{ KELOMPOK[0] = KELOMPOK[0] + 1 }

else

{ if ( CURAH\_HUJAN <= 9 )

{ KELOMPOK[1] = KELOMPOK[1] + 1 }

else

{ if ( CURAH\_HUJAN <= 14 )

{ KELOMPOK[2] = KELOMPOK[2] + 1 }

else

{ if ( CURAH\_HUJAN <= 19 )

{ KELOMPOK[3] = KELOMPOK[3] + 1 }

else

{ if ( CURAH\_HUJAN <= 24 )

{ KELOMPOK[4] = KELOMPOK[4] + 1 }

else

{ if ( CURAH\_HUJAN <= 29 )

{ KELOMPOK[5] = KELOMPOK[5] + 1 }

else

{ if ( CURAH\_HUJAN <= 34 )

{ KELOMPOK[6] = KELOMPOK[6] + 1 }

else

{ if ( CURAH\_HUJAN <= 39 )

{ KELOMPOK[7] = KELOMPOK[7] + 1 }

else

{ if ( CURAH\_HUJAN <= 44 )

{ KELOMPOK[8] = KELOMPOK[8] + 1 }

else

{ if ( CURAH\_HUJAN <= 49 )

{ KELOMPOK[9] = KELOMPOK[9] + 1}

else

{ Write(‘Salah memasukkan data’) }

}

}

}

}

}

}

}

}

}

3.b. [membaca data berikutnya]

Read (NAMA\_STASIUN, CURAH­\_HUJAN)

}

1. [menuliskan judul tabel]

Write(‘Kelompok Banyak Stasiun’)

1. [menuliskan hasil pengelompokan]

For ( K = 0 ; K <= 9 ; K++ )

{ Write (K+1, KELOMPOK[K])

}

1. [selesai]

Halt

Penentuan kelompok suatu stasiun pada langkah 3.a di atas yang menggunakan statement IF bertingkat, masih dapat disederhanakan dengan menggunakan operator div. Caranya dengan membagi (div) angka curah hujan dengan 5 sehingga didapat kelompok dari curah hujan tersebut. Kelompok yang didapat sekaligus merupakan indeks array yang digunakan untuk menentukan kelompok suatu stasiun berdasarkan curah hujannya. Dengan demikian langkah 3.a dapat ditulis menjadi berikut ini :

3.a . [menentukan kelompok berdasarkan curah hujan]

J = (CURAH\_HUJAN div 5)

KELOMPOK[J] = KELOMPOK[J] + 1

Dengan demikian, algoritma KELOMPOK\_STASIUN direvisi menjadi algoritma KELOMPOK\_STASIUN\_revisi\_1 dan ilustrasi flowchart dari algoritma hasil revisi ini dapat dilihat pada gambar 5.11 di halaman berikutnya.

Algoritma KELOMPOK\_STASIUN\_revisi\_1

Algoritma untuk membaca curah hujan dari setiap stasiun pengamat cuaca, lalu mengelompokkannya berdasarkan tabel 5.2 dan menampilkan banyaknya stasiun pengamat cuaca yang ada dalam tiap kelompok. Variabel NAMA\_STASIUN adalah variabel string untuk menyimpan nama stasiun pengamat cuaca. Variabel CURAH\_HUJAN adalah variabel integer untuk menyimpan data curah hujan. Data nama stasiun dan curah hujan disimpan dalam file sehingga digunakan sentinel eof untuk mengakhiri pembacaan data. Variabel array KELOMPOK[9] bertipe integer untuk menyimpan banyaknya stasiun dalam suatu kelompok, indeksnya merupakan kelompok stasiun dan komponennya adalah banyaknya stasiun dalam suatu kelompok tersebut. J dan K adalah variabel integer. J adalah variabel yang menyimpan indeks yang didapat dari hasil pembagian CURAH\_HUJAN dengan angka 5.

[deklarasi array]

INTEGER KELOMPOK[10]

* 1. [inisialisasi array KELOMPOK, semua komponennya angka 0]

For ( K = 0 ; K <= 9 ; K++ )

{

KELOMPOK[K] =0

}

* 1. [membaca data nama stasiun dan curah hujan]

Read (NAMA\_STASIUN, CURAH\_HUJAN)

* 1. [membuat pengulangan untuk menentukan kelompok dan membaca data berikutnya]

While ( not eof )

{

3.a [menentukan kelompok berdasarkan curah hujan]

J = CURAH\_HUJAN div 5

KELOMPOK[J] = KELOMPOK[J] + 1

3.b. [membaca data berikutnya]

Read (NAMA\_STASIUN, CURAH­\_HUJAN)

}

* 1. [menuliskan judul tabel]

Write(‘Kelompok Jumlah Stasiun’)

* 1. [menuliskan hasil pengelompokan]

For ( K = 0 ; K <= 9 ; K++ )

{

Write (K+1,KELOMPOK[K])

}

* 1. [selesai]

Halt



Gambar 5.11 Flowchart dari Algoritma KELOMPOK\_STASIUN\_revisi\_1

# 5.3.5 Array dan Fungsi

Dalam beberapa kasus, array dapat digunakan sebagai parameter fungsi. Dengan demikian data yang disimpan dalam arrray akan “diserahkan” ke fungsi untuk diproses. Bentuk umum dari deklarasi fungsi yang menggunakan array adalah sebagai berikut:

**VOID nama\_fungsi ( tipe\_data\_array[ ], integer, parameter\_lain)**

**atau**

**FUNGSI nama\_fungsi (tipe\_data\_array[ ], integer, parameter\_lain)**

dengan :

- nama\_fungsi : nama dari fungsi yang dibuat, aturan untuk membuat nama fungsi sama dengan aturan untuk membuat *identifier* dan harus unik

- tipe\_data\_array [ ] : parameter yang menunjukkan tipe data dari komponen array. Tanda “[ ]” menunjukkan bahwa parameter ini adalah array

- integer : parameter yang menunjukkan banyaknya komponen array, tipe datanya integer

- parameter\_lain : parameter lain yang digunakan oleh fungsi untuk menerima data.

Contoh:

* FUNGSI RATA (REAL[ ], INTEGER)
* Nama fungsi: RATA, mempunyai 2 parameter, yaitu array 1 dimensi yang berisi bilangan real dan data integer
* FUNGSI STD\_DEV (REAL[ ], INTEGER, REAL)
* Nama fungsi: STD\_DEV, mempunyai 3 parameter, yaitu array 1 dimensi yang berisi bilangan real, data integer dan data real

Bentuk umum dari definisi fungsi yang menggunakan array sebagai parameternya adalah:

**VOID nama\_fungsi (nama\_array[ ], banyak, parameter\_lain)**

**atau**

**FUNGSI nama\_fungsi (nama\_array[ ], banyak, parameter\_lain)**

dengan :

- nama\_fungsi : nama dari fungsi yang dibuat, aturan untuk membuat nama fungsi sama dengan aturan untuk membuat *identifier* dan harus unik

- nama\_array [ ] : argumen yang menunjukkan nama dari array (*variable dummy*). Tanda “[ ]” menunjukkan bahwa parameter ini adalah array

- banyak: parameter yang menunjukkan banyaknya komponen array (*variable dummy*), tipe datanya integer

- parameter\_lain : parameter lain yang digunakan oleh fungsi untuk menerima data (*variable dummy*).

Sebagai contoh penggunaan array sebagai parameter dari fungsi, akan dihitung deviasi standar (*standard deviation*) dari sejumlah data yang disimpan dalam array. Rumus untuk menghitung deviasi standar adalah:



dengan: - N adalah banyaknya data,

- xi adalah nilai data ke i

- adalah nilai rata-rata dari seluruh data

Untuk menghitung deviasi standar ini digunakan 2 buah fungsi yang mengembalikan sebuah hasil yaitu fungsi RATA untuk menghitung rata-rata dari data yang disimpan dalam array dan fungsi STD\_DEV untuk menghitung deviasi standar. Banyaknya data yang dapat diproses oleh algoritma ini dibatasi untuk 1000 data, karena itu array didefinisikan untuk menyimpan 1000 komponen. Algoritma utama berisi statement untuk memasukkan data, memanggil fungsi dan mencetak hasil perhitungan.

Algoritma DEVIASI\_STANDAR

Algoritma untuk menghitung deviasi standar untuk maksimum 1000 data. Array DATA[1000] untuk menyimpan data yang bertipe real. Fungsi RATA digunakan untuk menghitung rata-rata dari data yang disimpan dalam array DATA. Fungsi STD\_DEV digunakan untuk menghitung deviasi standar dari data yang disimpan dalam array DATA. Variabel SIZE menunjukkan banyaknya data yang akan dihitung deviasi standarnya. Variabel K digunakan sebagai counter (pencacah) data. Kedua variabel ini bertipe integer.

[deklarasi array]

REAL DATA[1000]

[deklarasi fungsi]

FUNGSI RATA (REAL[ ], INTEGER)

FUNGSI STD\_DEV (REAL[ ], INTEGER, REAL)

1. [Memasukkan banyaknya data, maksimum 1000]

Read (SIZE)

1. [Membuat pengulangan untuk membaca data satu persatu lalu disimpan dalam array DATA]

For ( K = 0 ; K< SIZE ; K++ )

{

Read(DATA[K])

}

1. [Memanggil fungsi RATA untuk menghitung rata-rata dari data yang disimpan dalam array DATA]

AVERAGE = RATA(DATA, SIZE)

1. [Memanggil fungsi STD\_DEV untuk menghitung deviasi standar dari data yang disimpan dalam array DATA]

STDEV = STD\_DEV(DATA, SIZE, AVERAGE)

1. [Mencetak hasil perhitungan]

Write(‘Standar Deviasi = ‘,STDEV)

1. [Selesai]

Halt

Perhatikan deklarasi fungsi pada algoritma di atas, sepasang tanda [ ] pada parameter fungsi menunjukkan bahwa parameter dari fungsi tersebut adalah variabel array. Perhatikan pula pemanggilan fungsi RATA (langkah 3) dan fungsi STD\_DEV (langkah 4), nama array yang menjadi argumen tidak disertai dengan sepasang tanda [ ] seperti pada bagian deklarasinya.

Fungsi RATA(NILAI[ ], BK)

Fungsi untuk menghitung rata-rata dari nilai yang disimpan dalam array NILAI dengan banyaknya komponen BK. Variabel SUM dan K bertipe integer. Hasil perhitungan disimpan dalam variabel AVER yang bertipe real.

{

1. [Inisialisasi variabel SUM]

SUM = 0

1. [Membuat pengulangan untuk menghitung jumlah data]

For ( K = 0 ; K < BK; K++ )

{

SUM = SUM + NILAI[K]

}

1. [menghitung rata-rata]

AVER = SUM / BK

1. [mengembalikan hasil perhitungan]

Return(AVER)

}

Fungsi STD\_DEV(SAMPLE[ ], VAL, AVERAGE)

Fungsi untuk menghitung deviasi standar dari data yang disimpan dalam array SAMPLE dengan banyaknya komponen VAL dan rata-ratanya AVERAGE. Hasil perhitungan disimpan dalam variabel DS. Variabel K bertipe integer. Variabel SUM dan DS bertipe real.

{

1. [Inisialisasi variabel SUM]

SUM = 0

1. [Membuat pengulangan untuk menghitung jumlah variance]

For ( K = 0 ; K < VAL ; K++)

{

SUM = SUM + (SAMPLE[K] – AVERAGE)↑2

}

1. [menghitung deviasi standar]

DS = SQRT(SUM / (VAL-1))

1. [mengembalikan hasil perhitungan]

Return(DS)

}

Perhatikan definisi fungsi RATA dan fungsi STD\_DEV, salah satu parameter fungsi adalah array (lihat tanda sepasang kurung siku setelah nama parameter). Banyaknya komponen array tidak didefinisikan bersama parameter array, tetapi didefinisikan oleh parameter lain dalam fungsi tersebut, yaitu parameter BK untuk fungsi RATA dan parameter VAL untuk fungsi STD\_DEV

Ilustrasi flowchart untuk fungsi RATA dan fungsi STD\_DEV dapat dilihat pada gambar 5.12 sedang flowchart untuk algoritma DEVIASI\_STANDAR dapat dilihat pada gambar 5.13 di halaman berikut.

Gambar 5.12 Flowchart dari fungsi RATA dan fungsi STD\_DEV



Gambar 5.13 Flowchart dari algoritma DEVIASI\_STANDAR

# 5.4 String

String adalah struktur data array linier yang komponennya berjenis karakter (character) sehingga sering juga disebut sebagai deretan karakter (character string). Pada umumnya string digunakan untuk menyimpan data yang berbentuk teks (textual data) seperti sebuah kata, nama orang, judul lagu, atau paragraph. Dalam bahasa pemrograman C++ struktur data string didefinisikan sebagai deretan karakter yang disimpan dalam lokasi memori yang berurutan dan diakhiri oleh karakter ‘\0’ (null terminator). Operasi yang khusus didefinisikan untuk struktur data string dapat dilihat pada library function string.h.

Karakter-karakter yang membentuk string atau teks diambil dari suatu himpunan karakter seperti karakter dalam bahasa latin (alfabet), karakter dalam bahasa Yunani, karakter dalam braille, dan sebagainya. Dalam dunia komputer, himpunan karakter yang digunakan adalah ASCII, EBCDIC atau Unicode.

Konstanta string (*string constant* atau *string literal*) didefinisikan sebagai deretan karakter yang ditulis di antara tanda petik. Contohnya adalah sebagai berikut:

1. ”Siapa nama anda?”.
2. ”Fakultas Teknologi Informasi”
3. ”Kode pos: 11440”
4. ”Jakarta”

Konstanta string dalam program C++ disimpan dalam memori yang berurutan dan diakhiri dengan karakter **null** (**\0**) seperti gambar 5.14 di bawah ini.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| J | a | k | a | r | t | a | \0 |

Gambar 5.14 Representasi konstanta string dalam program C++

Beberapa operasi yang didefinisikan untuk struktur data string adalah:

1. mendeklarasikan string
2. menghitung panjang string (banyaknya karakter yang menyusun string)
3. menggabungkan 2 buah string menjadi sebuah string
4. menyalin string

**5.4.1 Mendeklarasikan string**

Mendeklarasikan string adalah mendefinisikan suatu variabel berjenis string untuk menyimpan konstanta string. Statement untuk mendeklarasikan string sama seperti statement untuk mendeklarasikan array 1 dimensi yang berjenis karakter. Sebagai contoh:

char NAMA[10] 🡺 NAMA adalah variabel string yang dapat menyimpan paling banyak 9 karakter. Tempat yang terakhir digunakan untuk menyimpan karakter ’\0’ yang menunjukkan akhir dari string. Ilustrasi pada gambar 5.15 di bawah ini menunjukkan variabel NAMA berisi string ”ADIANTO”. Tanda ’?’ setelah karakter ’\0’ menunjukkan bahwa pada indeks tersebut tidak digunakan untuk menyimpan data karena string telah berakhir di indeks 7.

**NAMA**:

?

?

\0

O

T

A

D

N

I

A

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] → indeks array

Gambar 5.15 Ilustrasi penyimpanan varibel string dalam array linier

Dari ilustrasi di atas, dapat disimpulkan bahwa perlu kehati-hatian dalam menentukan banyaknya lokasi memori yang akan digunakan untuk menyimpan string ketika mendeklarasikan variabel string. Jika terlalu banyak maka akan ada memori yang tidak terpakai, tetapi jika terlalu sedikit maka konstanta string tidak dapat disimpan dengan sempurna.

**5.4.2 Menghitung panjang string**

Untuk menghitung panjang string dapat digunakan *built in function* yang sudah disediakan yaitu fungsi **STRLEN(str).** Hasil dari fungsi ini adalah bilangan integer.

Sebagai contoh lihat ekspresi berikut ini. Variabel S dan T adalah variabel integer.

a. S = STRLEN(“Computer Science”) ⇒ S = 16

b. T = 2 + 3 + STRLEN(“Computer Science”) ⇒ T = 21

**5.4.3 Menggabungkan string**

Operasi menggabungkan atau menyambung 2 string menjadi sebuah string yang baru menggunakan *built in function* STRCAT(str1, str2). Hasil dari fungsi ini adalah str1 yang berisi str1 mula-mula yang disambung dengan str2. Sebagai contoh lihat ekspresi berikut ini. Variabel X dan Y adalah variabel string.

Contoh: X = “Mickey” Y = “ Mouse”

⇒ STRCAT(X,Y) ⇒ X = “Mickey Mouse”

**5.4.4 Menyalin String**

Operasi menyalin string dilakukan dengan menggunakan *built in function* STRCPY(str1, str2). Hasil dari fungsi ini adalah str1 akan berisi str2. Sebagai contoh lihat ekspresi berikut ini. Variabel X dan Y adalah variabel string.

Contoh: X = “ “ Y = “Mickey”

⇒ STRCPY(X,Y) ⇒ X = “Mickey”

**5.4.5. Contoh Aplikasi String**

Program ini menerima input berupa nama yang terdiri dari 2 kata yang dipisah dengan spasi, lalu program akan memisahkan nama tersebut menjadi nama depan dan nama keluarga. Sebagai contoh, nama yang diinput adalah “Joko Widodo” maka outputnya adalah:

Nama depan anda adalah Joko

Nama keluarga anda adalah Widodo

Karakter spasi digunakan sebagai tanda pemisah antara kata pertama dan kata kedua.

Algoritma PisahNama

Algoritma untuk mememisahkan nama yang terdiri dari 2 buah kata menjadi nama depan dan nama keluarga. NAMA adalah variabel string untuk menyimpan input. ND adalah variabel string yang menyimpan nama depan (kata pertama). NK adalah variabel string yang menyimpan nama keluarga (kata kedua). Diasumsikan panjang nama tidak lebih dari 40 karakter, serta panjang nama depan dan nama keluarga masing-masing tidak lebih dari 20 karakter.

[deklarasi variabel string]

char NAMA[41]

char ND[21]

char NK[21]

1. [menuliskan pesan]

Write (“Ketikkan nama depan dan nama keluarga anda”)

Write (“pisahkan dengan spasi”)

2. [membaca input]

Read (NAMA)

3. [inisialisasi variabel]

i = 0

4. [mencari lokasi karakter spasi untuk mendapatkan nama

depan]

While (NAMA[i] != ‘ ‘){

ND[i] = NAMA[i]

i++

}

5. [inisialisasi variabel, mulai dari kata yang kedua]

i = i+1

6. [mencari lokasi karakter null]

While (i<=STRLEN(NAMA))

{ NK[i] = NAMA[i]

i++

}

7. [mencetak hasil]

Write (“Nama depan anda adalah “, ND)

Write (“Nama keluarga anda adalah ”, NK)

8. [selesai]

halt

# Latihan Soal Subbab 5.4

1. Tuliskan deklarasi untuk array berikut ini:

a. DATA adalah array integer yang berisi 50 komponen.

b. HASIL adalah array real yang berisi 75 komponen.

c. NAMA\_MHS adalah array karakter yang berisi 60 komponen.

2. Untuk soal-soal berikut ini diketahui:

- A adalah array integer yang berisi 10 komponen.

- B adalah array real yang berisi 10 komponen.

- C adalah array karakter yang berisi 10 komponen.

- I adalah variabel integer

Buatlah trace table dan outputnya untuk potongan algoritma berikut ini:

a. 1. For ( I = 0 ; I < 10 ; I++ )

1.a. { A[I] = I/2

2.b. Write (A[I) }

b. 1. For ( I =0 ; I < 6 ; I ++ )

1.a. { A[I] = I \* I

1.b. Write(A[I]) }

2. For ( I = 6 ; I < 10 ; I ++ )

2.a. { A[I] = A[I – 5]

2.b. Write (A[I]) }

c. 1. I = 0

2. While ( I ≠ 10 )

2.a. { If (mod(I,3) = = 0)

2.a.i. { A[I] = 0 }

2.b Else

2.b.i. { A[I] = I }

2.c. Write (A[I])

2.d. I = I + 1 }

d. 1. A[0] = 3

2. I = 1

3. While ( I < 10 )

3.a. { A[I] = A[I-1] + 1

3.b. I = I + 1

3.c. Write (A[I]) }

e. 1. For ( I = 6 ; I < 10 ; I ++ )

1.a. { If (mod(I,2) = = 0)

1.a.i { B[I – 6] = 0

1.a.ii B[I] = 0 }

1.b Else

1.b.i. { B[I – 6] = 1.25 \* I

1.b.ii. B[I] = 2.75 \* I } }

f. 1. For ( I = 0 ; I < 10 ; I ++ )

1.a. { If (mod(I,4) = = 0)

1.a.i { C[I] =‘a’ }

Else

1.b. { If (mod(I,4) = = 1)

1.b.i { C[I] = ‘b’ }

Else

1.c. { If (mod(I,4) = = 2)

1.c.i { C[I] = ‘c’ }

Else

1.c.ii { C[I] = ‘d’ }

} } }

3. Buatlah deklarasi untuk array berikut lalu buat potongan algoritma untuk membuat array berikut ini:

a. Array integer yang indeksnya adalah 0 sampai 8 dan berisi data yang adalah urutan kebalikan dari indeksnya.

b. Array logika dengan banyaknya komponen 25 dan berisi data yang berupa konstanta logika TRUE jika indeksnya bilangan ganjil atau FALSE jika indeksnya bilangan genap.

c. Array real yang indeksnya adalah angka 0 sampai 359 dan komponennya berupa nilai dari fungsi sinus dari indeksnya (sudut 0°, 1°, 2°, ..., 359°).

d. Array integer yang berisi banyaknya hari dalam tiap bulan dalam satu tahun. Indeks array menyatakan angka bulan.

4. Diketahui A adalah array real yang terdiri dari 100 komponen dan diasumsikan komponen array bersifat unik.

a. Buatlah algoritma untuk menentukan komponen yang terkecil dan kedua terkecil dari array A ini.

b. Buatlah algoritma untuk menentukan selisih yang terbesar dari dua komponen yang berurutan.

5.a. Buat fungsi DUPLIKAT(s,n) untuk mengulang string s sebanyak n kali. Sebagai contoh: DUPLIKAT(”ha”, 3) akan menghasilkan : ”hahaha”. Buat pula algoritma utamanya.

5.b. Buat pula algoritma utamanya

6.a. Buat fungsi TENGAH untuk mencetak string input di tengah halaman jika diketahui lebar halaman adalah 120 karakter dan panjang string input < 120 karakter.

6.b. Buat pula algoritma utamanya.

7.a. Buat fungsi REVERSE untuk menerima string input dan menghasilkan string output yang isinya adalah kebalikan urutan karakter dalam string input. Sebagai contoh string input = “abcd”, maka string output = “dcba”. Buat pula algoritma utamanya.

7.b. Buat pula algoritma utamanya

8.a. Palindrome adalah string yang dapat dibaca dari depan maupun dari belakang. Contohnya adalah ”malam”, ”kodok” dan lain-lain. Buatlah fungsi untuk menentukan apakah sebuah string input termasuk palindron atau bukan. Diasumsikan string input berupa huruf kecil semua. Buat pula algoritma utamanya.

8.b. Buat pula algoritma utamanya.

9. Buatlah fungsi untuk rumus statistik berikut ini jika diketahui X adalah array real yang terdiri dari dari n komponen.

a. Mean deviation (MD) =  dengan 

b. Harmonic mean (HM) = 

c. Range (R) = maximum{x1, x2, ..., xn} - minimum{x1, x2, ..., xn}

d. Geometric mean (GM) = (x1 × x2 × ...× xn)1/n

10.a. Buat sub-algoritma untuk menghitung rata-rata, variansi dan deviasi standar dari sekelompok bilangan yang disimpan dalam array.

Diketahui:

 (x rata-rata)

Variansi =  Deviasi standar = 

10.b. Buat algoritma untuk membaca sekelompok bilangan, menyimpannya dalam array, memanggil sub algoritma 6.a untuk menghitung x rata-rata, variansi dan deviasi standar lalu menampilkan hasilnya.

11. Cara lain untuk menentukan nilai huruf (A, B, C, D atau E) untuk suatu nilai numerik adalah dengan menggunakan kurva nilai (*grading of the curve*) yang dihitung dari deviasi standar sekelompok nilai. Untuk menentukan nilai huruf dengan cara ini digunakan tabel 5.5 seperti di bawah ini.

Tabel 5.5 Menentukan nilai huruf dengan deviasi standar

x = nilai numerik nilai huruf

x <  E

 ≤ x <  D

 ≤ x <  C

 ≤ x <  B

x ≥  A

dengan m adalah nilai rata-rata dan σ adalah deviasi standar.

Buatlah algoritma untuk membaca sekelompok nilai, menyimpannya dalam array, menghitung rata-rata dan deviasi standarnya (gunakan sub algoritma pada no. 6.a), lalu tentukanlah nilai huruf dari masing-masing nilai numerik berdasarkan tabel tersebut.

12. Dalam sebuah percobaan, selalu ada faktor kesalahan pengukuran. Sebuah teknik yang disebut smoothing dapat digunakan untuk mengurangi efek dari kesalahan pengukuran tersebut. Diasumsikan data hasil pengukuran berupa N bilangan real dan disimpan dalam array B. Sebelum data hasil pengukuran tersebut dianalisis, dilakukan operasi smoothing untuk setiap data (kecuali untuk data yang pertama dan data yang terakhir atau V[0] = B[0] dan V[n] = B[n]) yaitu sebagai berikut:

V[i] = (B[i-1] + B[i] + B[i+1])/3

Dengan V adalah array hasil operasi smoothing.

Buatlah algoritma untuk membaca data hasil pengukuran, lalu lakukanlah operasi smoothing. Tampilkan data awal dan hasil operasi smoothingnya.

13. Sebuah perusahaan permen mempunyai 2 buah gudang, di Jakarta dan di Bandung, masing-masing menyimpan maksimum 25 jenis permen. Tiap jenis permen mempunyai nomor kode produksi yang unik. Buatlah algoritma untuk membaca nomor kode produksi permen yang ada di gudang Jakarta lalu menyimpannya dalam sebuah array. Ulangi proses ini untuk gudang di Bandung dan simpan dalam array yang lain. Bacalah kedua array itu untuk menemukan dan menampilkan potongan (*intersection*) dari kedua array, maksudnya semua nomor kode produksi yang sama di kedua array.

14. Ulangi soal nomor 13, tapi carilah gabungan (union) dari kedua array, maksudnya tampilkan semua nomor kode produksi yang ada sedikitnya di salah satu array.

15. Diketahui ada 3 array linier masing-masing terdiri dari 12 komponen. Array pertama berisi banyaknya hari dalam tiap bulan dalam 1 tahun, array yang ke dua berisi banyaknya hari libur dalam tiap bulan dalam 1 tahun. Array yang ketiga berisi banyaknya hari kerja dalam tiap bulan dalam 1 tahun (selisih dari array pertama dan array ke dua). Buatlah algoritma untuk membuat ke 3 array tersebut dan menampilkan hasilnya dalam bentuk tabel.

**5.5 Algoritma Sorting Sederhana**

Sorting adalah salah satu operasi yang penting dalam pengolahan data. Sorting didefinisikan sebagai suatu kegiatan untuk mengatur urutan sekelompok data sedemikian sehingga data pertama lebih kecil dari data kedua, data kedua lebih kecil dari data ketiga dan seterusnya atau data terurut dari kecil ke besar. Urutan data seperti ini disebut urut naik (*ascending*). Selain urut naik, data juga dapat diurutkan mulai dari besar ke kecil atau disebut urut turun(*decending*). Lihat gambar 5.16. Fakta menunjukkan bahwa data yang telah terurut akan memudahkan proses pencarian (*searching*) nantinya.

Data: 5 3 7 2 9 4 8 6

Ascending: 2 3 4 5 6 7 8 9

Descending: 9 8 7 6 5 4 3 2

Gambar 5.16 Data mula-mula dan hasil sorting

Algoritma sorting adalah algoritma yang berisi langkah-langkah untuk mengurutkan data. Ada banyak algortima sorting yang telah dibuat orang, beberapa diantaranya:

1. **Selection sort** adalah salah satu cara yang termudah untuk mengurutkan data. Sesuai namanya yang berarti pemilihan, prosesnya dimulai dengan memeriksa seluruh data sehingga dapat dipilih data yang terkecil, kemudian data tersebut ditempatkan di posisi paling awal. Selanjutnya pada data yang tersisa, dipilih lagi data yang terkecil, lalu tempatkan di posisi ke dua. Demikian seterusnya sampai seluruh data selesai diperiksa. Diasumsikan data disimpan dalam Arr[N] dan akan diurutkan secara ascending.

Void Selection\_sort (Arr[], N)

Fungsi untuk mengurutkan data dengan menggunakan algoritma Selection sort yang telah disempurnakan. Parameter Arr adalah array yang berisi data yang akan diurutkan. Paramenter N adalah panjang array.

{

1. For ( I = 0 ; I < N-1 ; I++ )

{

1.a. [ inisialisasi variabel ]

Min = I

A= Arr[I]

1.b. [ loop untuk mencari data minimum ]

For ( J = I + 1 ; J < N ; J++ )

{ if (Arr[J] < Min)

{ Min = J

A = Arr[J]

}

}

1.c. [tukar data minimum dengan data di posisi I] Arr[Min] = A[I]

A[I] = A

}

1. Return.

}

1. **Exchange sort** juga termasuk algoritma yang mudah dipahami. Sesuai namanya, Exchange yang berarti pertukaran, maka dalam algoritma ini banyak terjadi pertukaran data. Cara kerja algoritma ini mulai memeriksa array dari belakang (posisi N), bandingkan elemen array pada posisi N dengan N-1, jika data pada posisi N lebih besar dari data pada posisi N-1 maka lanjutkan perbandingan antara data pada posisi N-1 dengan N-2. Jika data pada posisi N lebih kecil dari data pada posisi N-1 maka tukarkan data dan lanjutkan perbandingan antara data pada posisi N-1 dengan N-2. Demikian seterusnya sampai mencapai posisi 1. Pada saat ini data terkecil sudah berada pada posisinya. Pemeriksaan elemen array dilakukan kembali mulai dari posisi N sampai posisi 1. Dan seterusnya sampai seluruh array selesai diperiksa.

Dengan cara ini, data terkecil akan langsung menempati posisinya pada putaran pertama, tetapi tidak demikian pada data yang terbesar. Pada setiap putaran, data yang terbesar perlahan-lahan bergeser ke posisi N. Hal ini seperti gelembung sabun yang bergerak perlahan-lahan ke atas sehingga sering kali Exchange sort ini disebut juga Bubble sort.

Void Exchange\_sort(Arr[],N)

Fungsi untuk mengurutkan data dengan menggunakan algoritma Exchange sort. Parameter Arr adalah array yang berisi data yang akan diurutkan. Paramenter N adalah panjang array. Variabel I dan J bertipe integer.

{

1. For ( I = 1 ; I <= N ; I++ )

{

For ( J = N ; J >= I ; J-- )

{

If (Arr[J-1] > Arr[J])

{

[tukar isi Arr[J] dengan Arr[J-1]]

Temp = Arr[J]

Arr[J] = Arr[J-1]

Arr[J-1] = Temp

}

}

}

1. Return.

}

1. **Insertion sort** berasal dari cara mengurutkan kartu bridge yang dilakukan oleh para pemain kartu. Sesuai namanya, algoritma ini akan mencari posisi yang sesuai untuk data yang akan diurutkan lalu menyisipkan (insert) data disitu. Jika hanya ada 1 data, maka data telah terurut. Jika ada 2 atau lebih data maka bandingkan data kedua dengan data pertama. Jika data kedua lebih kecil maka sisipkan data kedua di depan data pertama lalu lanjutkan dengan data ketiga. Jika data kedua lebih besar, lanjutkan dengan data ketiga. Dan seterusnya sampai seluruh data selesai diurutkan.

Void Insertion\_Sort(Arr[],N)

Fungsi untuk mengurutkan data dengan menggunakan algoritma sort. Parameter Arr adalah array yang berisi data yang akan diurutkan. Paramenter N adalah panjang array. Variabel i dan j bertipe integer

{

1. For (i = 1 ; i < N ; i++)

{

[inisialisasi variabel]

x = Arr[i]

j = i-1

[mencari tempat sesuai yang untuk x]

while (x < Arr[j])

{

Arr[j+1]= Arr[j]

j = j-1

}

[x menempati tempat yang sesuai]

Arr[j+1] = x

1. Return

}

Ketiga algoritma sorting yang telah dibahas di atas termasuk dalam kelompok algoritma sorting sederhana (*simple sort*) yaitu langkah-langkahnya sederhana, mudah dipahami dan ringkas, tetapi prosesnya membutuhkan waktu yang lama. Algoritma ini cocok untuk mengurutkan sedikit data (banyaknya data <10000). Selain itu ada kelompok algoritma sorting lanjut (*advanced sort*) yang langkah-langkahnya lebih rumit dibanding algoritma yang sederhana tetapi prosesnya membutuhkan waktu yang lebih singkat. Algoritma ini biasanya digunakan untuk mengurutkan data yang banyak. Ada banyak sekali algoritma sorting lanjut yang telah dibuat orang, contohnya Quick sort, Merge sort, Shell sort, Alpha sort, Radix sort dan sebagainya.

**5.6. Algoritma Searhing**

Selain sorting, operasi lain yang banyak dilakukan dalam pengolahan data dengan komputer adalah searching atau mencari suatu data dalam kumpulannya sampai ketemu atau tidak ketemu. Sebagai contoh ingin mencari data mahasiswa yang mempunyai NPM = k dalam file DATA. Diasumsikan semua mahasiswa memiliki NPM yang unik sehingga tidak ada 2 mahasiswa yang mempunyai NPM yang sama. Untuk kasus ini, k disebut sebagai key (kunci) yang menjadi dasar pencarian. Hasil pencarian ini terdiri dari 2 kemungkinan, yaitu pencarian sukses (diketemukan data mahasiswa dengan NPM = k) atau pencarian tidak sukses (tidak ada data mahasiswa yang mempunyai NPM = k).

Ada banyak teknik searching yang sudah dibuat, beberapa diantaranya adalah:

1. **Linear Search** merupakan algoritma pencari yang paling sederhana. Secara umum metode ini hanya terdiri dari 2 langkah yaitu membaca elemen dalam array lalu membandingkan elemen yang dibaca dengan key yang dicari. Jika sama maka pencarian selesai, jika tidak sama maka lanjutkan pembacaan dan perbandingan dengan elemen berikutnya dalam array sampai ditemukan atau tidak ditemukan key yang dicari. Jika tidak ditemukan elemen array yang sama dengan key maka dihasilkan 0, jika ditemukan maka dihasilkan posisi elemen dalam array. Linear Search ini juga disebut *sequential* search. Metode Linear Search ini dapat dilakukan pada kumpulan data yang sudah terurut maupun kumpulan data yang acak. Fungsi Linear Search dapat dilihat pada fungsi di bawah ini.

Function LINEAR\_SEARCH(ARR[],N,X)

# Fungsi untuk mencari apakah key X ada dalam array ARR dengan metode Linear Search. Fungsi akan menghasilkan indeks tempat X berada jika X ada dalam array ARR atau 0 jika X tidak ada. ARR adalah array 1 dimensi yang mengandung N elemen. X adalah elemen yang dicari. J adalah counter, berjenis integer.

{

1. [membuat loop untuk menelusuri array ARR]

For ( J = 1 ; J <= N ; J++ )

{

If ( ARR[J] == X ) [jika X ada dalam array ARR]

{ return (J)

}

}

2. [jika X tidak ada dalam array ARR]

Return(0)

}

1. **Binary Search**:

Jika data makin banyak, maka mencari data dengan Linear Search akan memerlukan waktu yang lama. Karena itu untuk data yang banyak yang disimpan dalam keadaan terurut dapat digunakan algoritma Binary search. Berbeda dengan Linear Search yang memeriksa semua data secara berurutan dari awal sampai akhir, Binary Search ini hanya memeriksa sebagian dari data sehingga waktu yang diperlukan untuk menemukan data pada umumnya lebih singkat dibanding Linear Search.

Binary Search memulai proses pencarian dari elemen yang berada di tengah array. Jika elemen yang dicari (X) lebih kecil dari elemen tengah, maka pencarian dilanjutkan pada bagian array di sebelah kiri elemen tengah. Sebaliknya jika lebih besar, maka pencarian dilanjutkan pada bagian array di sebelah kanan elemen tengah. Begitu seterusnya sampai X ditemukan atau tidak ditemukan.

Function BINARY\_SEARCH(ARR[],X,KIRI,KANAN)

# Fungsi untuk mencari apakah X ada dalam array ARR dengan metode Binary Search. Fungsi akan menghasilkan indeks tempat X berada jika X ada dalam array ARR atau 0 jika X tidak ada. ARR adalah array 1 dimensi yang mengandung N elemen yang sudah terurut. X adalah elemen yang dicari. KIRI adalah batas bawah dari array/subarray. KANAN adalah batas atas dari array/subarray. J adalah variabel untuk menyimpan indeks tempat X berada. KIRI, KANAN dan J adalah variabel integer. FOUND adalah flag untuk menghentikan pencarian jika X sudah ketemu, berjenis logika.

{

1. [inisialisasi flag]

FOUND = FALSE

2. [membuat loop untuk mencari X]

While (not FOUND && KIRI <= KANAN)

{

2.a [mulai dari indeks tengah]

TENGAH = (KIRI + KANAN) div 2

2.b If ( X == ARR[TENGAH])

{ [X ketemu pada indeks TENGAH]

2.b.i J = TENGAH

2.b.ii FOUND = TRUE

}

else [X belum ditemukan, X < ARR[TENGAH]]

2.b.iii if X < ARR[TENGAH]

{ [ubah batas atas]

KANAN = TENGAH – 1

}

else [X belum ditemukan, X > ARR[TENGAH]]

{ [ubah batas bawah]

KIRI = TENGAH + 1

}

[lanjutkan pencarian dengan batas

bawah/batas atas yang baru]

}

3. [X tidak ketemu]

If (not FOUND )

{ J = 0 }

4. Return(J)

}

Untuk menjelaskan proses pencarian data dengan fungsi Binary Search di atas akan dibuat loop behavior table untuk menemukan key = 19 dari array A berikut ini:

A[1] A[2] A[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10]

10 16 19 24 28 32 39 40 42 48

Tabel 5. 6 Loop behaviour table untuk menemukan X = 19 dengan fungsi Binary Search

Loop KIRI KANAN TENGAH KONDISI FOUND

1 1 10 5 X < A[5] false

2 1 4 2 X > A[2] false

3 3 4 3 X = A[3] true

Hasilnya: X ditemukan pada indeks = 3

Keterangan:

KIRI, KANAN, TENGAH : indeks array

KONDISI: perbandingan antara X dengan elemen tengah

Dari tabel di atas dapat diambil kesimpulan jika hasil perbandingan:

a. X < ARR[TENGAH] maka pencarian dilanjutkan pada bagian array ARR[1] … ARR[TENGAH-1].

Bagian array ARR[TENGAH] … ARR[N] tidak perlu diperiksa.

b. X = ARR[TENGAH] maka X ditemukan pada indeks = TENGAH, pencarian selesai.

c. X > ARR[TENGAH] maka pencarian dilanjutkan pada bagian array ARR[TENGAH + 1] … ARR[N].

Bagian array ARR[1] … ARR[TENGAH] tidak perlu diperiksa.

Dengan demikian bila dibandingkan dengan Linear Search, Binary Search lebih efisien dan lebih menghemat waktu, tetapi Binary Search hanya dapat dilakukan pada kumpulan data yang sudah terurut sehingga memerlukan waktu tambahan untuk mengurutkan data terlebih dulu.

**Latihan soal subbab 5.6**

1. Diketahui data yang disimpan dalam array A[0..8] berikut ini:

A[7, 5, 9, 3, 6, 1, 8, 2, 4]

Urutkan data dalam array A dengan algoritma berikut (buat trace table)

1.a. Selection sort

1.b. Exchange sort

1.c Insertion sort

2. Diketahui data yang disimpan dalam array A[1..16] berikut ini:

A[15, 3, 17, 10, 8, 20, 12, 4, 1, 5, 16, 7, 9, 11, 14, 18]

Carilah key: 5, 9, dan 13 dengan algoritma berikut (buatlah loop behavior table-nya)

1.a. Linear search

1.b. Binary search

**5.7. Array Multi Dimensi**

Array multi dimensi adalah array yang mempunyai lebih dari satu indeks/*subscript*. Jumlah maksimum indeks tergantung dari bahasa pemrograman yang digunakan, ada yang dapat menampung sampai 9 indeks atau array 9 dimensi. Dalam kuliah ini hanya dibahas array 2 dimensi yaitu bentuk yang paling sederhana dari array multi dimensi

# 5.7.1 Array 2 dimensi

Array 2 dimensi mempunyai 2 buah indeks. Indeks yang pertama mewakili banyaknya baris dan indeks yang kedua mewakili banyaknya kolom. Karena bentuknya seperti matriks maka array 2 dimensi juga disebut array matriks. Sama seperti array 1 dimensi, array 2 dimensi ini juga harus dideklarasikan di awal program. Bentuk umum deklarasi array 2 dimensi :

**tipe\_data nama\_array[limit\_1] [limit\_2]**

dengan:

- tipe data : tipe data dari komponen array

- nama\_array: nama dari array. Kriteria membuat nama array sama dengan kriteria untuk membuat identifier

- limit\_1: banyaknya baris dalam array, merupakan bilangan integer.

- limit\_2: banyaknya kolom dalam array, merupakan bilangan integer.

Contoh deklarasi array matriks:

a. REAL B[2] [3] → B adalah array 2 dimensi (matriks) yang terdiri dari 2 baris dan 3 kolom, yang komponennya bertipe real.

Contoh: Kolom 0 Kolom 1 Kolom 2

Baris 0 1.5 3.2 4.6 → matriks 2 x 3

Baris 1 2.9 5.7 6.1

b. INTEGER CODE [50] [3] → CODE adalah array 2 dimensi (matriks) yang terdiri dari 50 baris dan 3 kolom yang komponennya bertipe integer.

Contoh:

Kolom 0 Kolom 1 Kolom 2

Baris 0: 4 5 42 ⎫

Baris 1: 1 3 40 ⎪

… ... ... ... ⎬ 50 baris

… ... ... ... ⎪

Baris 49: 2 2 60 ⎭

# 5.7.2 Operasi input/output dengan array 2 dimensi

Untuk memasukkan atau menampilkan sebuah komponen array 2 dimensi cukup dengan menuliskan nama array dan indeksnya saja bersama perintah input atau output (I/O), seperti contoh berikut ini:

- Read(B [1] [1]) ⇒ membaca sebuah data lalu disimpan ke baris dengan indeks 1 dan kolom dengan indeks 1 dari array B

- Write(B [1] [2]) ⇒ menampilkan komponen yang berada pada baris dengan indeks 1 dan kolom dengan indeks 2 dari array B

Untuk memproses seluruh komponen array 2 dimensi, ada 2 cara yaitu :

1. Menggunakan nama array dalam perintah I/O
2. Menggunakan pengulangan bertingkat yang mengandung perintah I/O, tiap indeks mempunyai variabel pengulangan sendiri.

Sama seperti array 1 dimensi, pemrosesan/pembacaan/penulisan array dimulai dari komponen pertama, komponen kedua dan seterusnya sampai komponen terakhir. Karena array berbentuk 2 dimensi, maka dikenal 2 jenis urutan untuk memproses komponen array, yaitu:

1. *Rowwise*: proses berdasarkan baris, komponen-komponen pada baris pertama diproses lebih dulu, lalu baris berikutnya.
2. *Colomnwise*: proses berdasarkan kolom, komponen-komponen pada kolom pertama diproses lebih dulu, lalu kolom berikutnya.

Gambar 5.14 a dan b berikut menunjukkan ilustrasi dari kedua jenis urutan untuk memproses array A yang terdiri dari 3 baris dan 4 kolom (A[3] [4]). Perhatikan perbedaan urutan pemrosesan komponen array 2 dimensi dari kedua cara tersebut. Urutan komponen array yang diproses juga menggambarkan urutan penyimpanan komponen array di memori komputer. Pada umumnya, array matriks disimpan secara *rowwise*, kecuali jika disebutkan secara khusus cara penyimpanannya.

(a) *Rowwise*:

A[0][0]

A[0][0]

A[0][3]

A[0][2]

A[0][1]

A[0][1]

A[0][2]

A[0][3]

A[1][0]

A[1][3]

A[1][2]

A[1][1]

A[1][0]

A[1][1]

A[1][2]

A[1][3]

A[2][0]

A[2][2]

A[2][3]

A[2][0]

A[2][1]

A[2][1]

A[2][2]

A[2][3]

Gambar 5.14.a Urutan memroses komponen-komponen array A secara rowwise

(b) *Columnwise*:

A[0][0]

A[0][2]

A[0][3]

A[0][0]

A[0][1]

A[1][0]

A[2][0]

A[0][1]

A[1][3]

A[1][2]

A[1][1]

A[1][0]

A[1][1]

A[2][1]

A[0][2]

A[2][3]

A[2][2]

A[2][1]

A[2][0]

A[1][2]

A[2][2]

A[0][3]

A[1][3]

A[2][3]

Gambar 5.14.b Urutan memroses komponen-komponen array A secara column wise

Sebagai contoh, gunakan deklarasi array 2 dimensi yang terdiri dari 3 baris dan 2 kolom seperti berikut:

INTEGER X[3] [2]

Untuk membaca 6 buah data real lalu menyimpannya ke array matriks X, maka instruksi yang dapat diberikan adalah:

1. Menggunakan nama array dan indeksnya:

a. *Rowwise*:

READ (X[0] [0])

READ (X[0] [1])

READ (X[1] [0])

READ (X[1] [1])

READ (X[2] [0])

READ (X[2] [1])

Atau

READ (X[0] [0], X[0] [1], X[1] [0], X[1] [2], X[2] [0], X[2] [1],)

b. *Columnwise*:

READ (X[0] [0])

READ (X[1] [0])

READ (X[2] [0])

READ (X[0] [1])

READ (X[1] [1])

READ (X[2] [1])

Atau

READ (X[0] [0], X[1] [0], X[2] [0], X[0] [1], X[1] [1], X[2] [1])

2. Menggunakan pengulangan bertingkat:

a. *Rowwise*:

For ( I = 0 ; I < 2 ; I++ )

{ For ( J = 0 ; J < 3 ; J++ )

{ READ (X[I] [J])

}

}

b. *Columnwise*:

For ( J = 0 ; J < 3 ; J++ )

{ For ( I = 0 ; I < 2 ; I++ )

{ READ (X[I] [J])

}

}

# 5.7.3 Contoh aplikasi array 2 dimensi

Sebagai contoh aplikasi array 2 dimensi akan dibuat algoritma untuk melakukan perkalian antara matriks A [3,5] dengan vektor B[5] seperti berikut:

A11 A12 A13 A14 A15 B1

A21 A22 A23 A24 A25 B2

A31 A32 A33 A34 A35 B3

B4

B5

Hasilnya adalah vektor C[3]:

C1 = A11 B1  + A12 B2 + A13 B3 + A14 B4 + A15 B5

C2 = A21 B1 + A22 B2 + A23 B3 + A24 B4 + A25 B5

C3 = A31 B1 + A32 B2 + A33 B3 + A34 B4 + A35 B5

Matriks A disimpan dalam array 2 dimensi A[3][5] dan vektor B disimpan dalam array linier B[5], dan vektor C disimpan dalam array linier C[3].

ALGORITMA MATRIKS\_KALI\_VEKTOR

Algoritma untuk membaca data matriks A dan vektor B, kemudian melakukan perkalian matriks dengan vektor lalu menampilkan hasil perkalian. Matriks A disimpan dalam array A [3][5]. Vektor B disimpan dalam array B[5]. Hasil perkalian disimpan dalam vektor C[3]. Semua komponen array bertipe real. Variabel p dan q digunakan sebagai pencacah, bertipe integer.

[deklarasi array]

REAL A[3][5]

REAL B[5]

REAL C[3]

1. [membaca untuk data matriks A]

Write (“Masukkan data untuk matriks A[3,5]”)

For ( p = 0 ; p < 3 ; p++ )

{ For ( q = 0 ; q < 5 ; q++ )

{ Read ( A[p][q] )

}

}

1. [membaca data untuk vektor B]

Write (“Masukkan data untuk vektor B[5]”)

For ( p = 0 ; p < 5; p++ )

{ Read ( B[p] )

}

1. [melakukan perkalian matriks A dengan vektor B]

For ( p = 0; p < 3 ; p++ )

{ C[p] = 0

For ( q = 0 ; q < 5 ; q++ )

{ C[p] = C[p] + A[p][q] \* B[q]

}

}

1. [menampilkan hasil]

Write (“Hasil perkalian matriks A dengan Vektor B”)

For ( p = 0 ; p < 3 ; p++ )

{ Write ( C[p] )

}

1. [selesai]

Halt.

Ilustrasi flowchart untuk perkalian matriks dengan vektor dapat dilihat pada gambar 5.15 di halaman berikut.

Sebagai contoh berikutnya untuk penggunaan array 2 dimensi ini adalah untuk memproses data penjualan dari sebuah perusahaan yang mempunyai 50 orang salesman. Data penjualan dibuat per minggu sehingga untuk 1 bulan tiap salesman mempunyai 4 buah data. Setiap bulan data dianalisis untuk mengetahui siapa salesman yang penjualannya paling banyak. Untuk memudahkan pemrosesan, nama salesman disimpan dalam array linier bertipe char dan data penjualan disimpan dalam array 2 dimensi. Indeks baris menunjukkan salesman dan indeks kolom menunjukkan besarnya penjualan pada minggu ke 1, 2, 3 dan 4 serta total penjualan



Gambar 5.15 Ilustrasi flowchart untuk mengalikan matriks

selama 4 minggu. Dengan demikian diperlukan array 2 dimensi yang terdiri dari 50 baris dan 5 kolom. Contoh algoritma untuk memecahkan masalah ini dapat dilihat pada Algoritma Penjualan berikut.

ALGORITMA PENJUALAN

Algoritma untuk menyimpan data penjualan tiap minggu dari 50 salesman. Data penjualan disimpan dalam array matriks JUAL[50][5] yang bertipe integer. Kolom pertama sampai ke empat berisi data penjualan dari tiap salesman untuk minggu pertama sampai ke empat. Dan kolom yang ke lima berisi total penjualan dalam 4 minggu. Data nama salesman disimpan dalam array lininer NAMA[50] yang bertipe character. Perusahaan ingin mengetahui salesman yang mempunyai nilai penjualan tertinggi pada minggu 1 sampai 4 dan salesman yang nilai penjualannya tertinggi dalam 1 bulan. Variabel i, b dan j digunakan sebagai pencacah, bertipe integer.

[deklarasi array]

INTEGER JUAL[50][5]

CHARACTER NAMA[50]

1. [membaca data nama salesman]

For ( i = 0 ; i < 50 ; i++ )

{

Write (“Masukkan nama salesman ke ”, i+1)

Read ( NAMA[i] )

}

1. [membaca data penjualan]

For ( i = 0 ; i < 50 ; i++ )

{ Write (“Masukkan data penjualan minggu 1 sampai 4 untuk salesman ke ”, i+1)

For ( j = 0 ; j < 4 ; j++ )

{ Read ( JUAL[i][j] )

}

}

1. [menjumlahkan data penjualan untuk tiap salesman]

For ( i = 0 ; i < 50; i++ )

{ total = 0

For ( j = 0 ; j < 4 ; j++ )

{ total = total + JUAL[i][j] )

}

JUAL[i][4] = total

}

1. Write(”Penjualan terbesar tiap minggu: ”)
2. [mencari nilai penjualan yang terbesar tiap minggu ]

For ( i = 0; i < 4 ; i++ )

{ max = 0

b = 0

For ( j = 0 ; j < 50 ; j++ )

{ if (JUAL[j][i] > max)

{ max = JUAL[j][i]

b = j

}

}

Write(“Minggu ke “, i+1,” oleh salesman “, NAMA[b], “ sebesar “, max)

}

1. [inisialiasai variabel]

max = 0

b = 0

1. [Mencari total penjualan terbesar dalam 1 bulan]

For ( i = 0 ; i < 50 ; i++ )

{ if (JUAL[i][4] > max)

{ max = JUAL[i][4]

b = i

}

}

Write (“Penjualan terbesar pada bulan ini: ”, max)

Write (“oleh salesman “, NAMA[b])

1. [selesai]

Halt.

# 5.7.4 Array 2 dimensi dan Fungsi

Sama seperti array 1 dimensi, array dua dimensi dapat pula digunakan sebagai parameter fungsi. Dengan demikian data yang disimpan dalam arrray akan “diserahkan” ke fungsi untuk diproses. Bentuk umum dari deklarasi fungsi yang menggunakan array adalah sebagai berikut:

**VOID nama\_fungsi (tipe\_data\_array[ ] [ ], integer\_1, integer\_2, tipe\_data\_lain)**

**atau**

**FUNGSI nama\_fungsi (tipe\_data\_array[ ] [ ], integer\_1, integer\_2,**

**tipe\_data\_lain)**

dengan :

- nama\_fungsi : nama dari fungsi yang dibuat, aturan untuk membuat nama fungsi sama dengan aturan untuk membuat *identifier* dan harus unik

- tipe\_data\_array [ ] [ ]: parameter yang menunjukkan tipe data dari komponen array. Dua pasang tanda [ ] menunjukkan bahwa argumen ini adalah array dua dimensi

- integer\_1: parameter yang menunjukkan banyaknya baris dalam array, tipe datanya integer

- integer\_2: parameter yang menunjukkan banyaknya kolom dalam array, tipe datanya integer

- tipe\_data\_lain : tipe data dari parameter-parameter lain yang digunakan oleh fungsi untuk menerima data.

Bentuk umum dari definisi fungsi yang menggunakan array sebagai parameternya adalah:

**VOID nama\_fungsi (nama\_array[ ] [ ], baris, kolom, parameter\_lain)**

**atau**

**FUNGSI nama\_fungsi (nama\_array[ ] [ ], baris, kolom, parameter\_lain)**

dengan :

- nama\_fungsi : nama dari fungsi yang dibuat, aturan untuk membuat nama fungsi sama dengan aturan untuk membuat *identifier* dan harus unik

- nama\_array [ ] [ ]: menunjukkan nama dari array (*variable dummy*). Dua pasang tanda [ ] menunjukkan bahwa parameter ini adalah array dua dimensi

- baris: parameter yang menunjukkan banyaknya baris dari array (*variable dummy*), tipe datanya integer

- kolom: parameter yang menunjukkan banyaknya kolom dari array (*variable dummy*), tipe datanya integer

- parameter\_lain : parameter-parameter lain yang digunakan oleh fungsi untuk menerima data (*variable dummy*).

Sebagai contoh penggunaan array sebagai argumen dari fungsi, akan dibuat algoritma untuk membaca data 2 buah matriks yang berukuran 3 x 3, melakukan operasi tambah dan kali lalu menampilkan hasilnya. Seluruh proses dilakukan oleh fungsi (fungsi BACA\_DATA, fungsi TAMBAH, fungsi KALI dan fungsi CETAK\_MAT), sehingga algoritma utama (algoritma OPERASI\_MATRIKS) hanya berisi instruksi untuk memanggil fungsi. Hasil fungsi dikeluarkan melalui parameter output fungsi. Perhatikan deklarasi masing-masing fungsi.

VOID BACA\_DATA(MAT[ ] [ ], BARIS, KOLOM)

Fungsi untuk membaca data untuk matriks MAT yang berukuran BARIS X KOLOM. Matriks MAT bertipe real. Variabel BARIS, KOLOM, I dan J bertipe integer.

{

1. [membaca data matriks]

For ( I = 0 ; I < BARIS ; I++)

{ For ( J = 0 ; J < KOLOM ; J++ )

{ Read (MAT[I] [J])

}

}

}

VOID TAMBAH(MAT\_1[] [],MAT\_2[] [], MAT3& [] [], B, K)

Fungsi untuk melakukan operasi penambahan antara 2 buah matriks, MAT\_1 dan MAT\_2 yang masing-masing berukuran B X K. Hasil operasi disimpan dalam matriks MAT\_3. Matriks MAT\_1, MAT2 dan MAT\_3 bertipe real. Variabel I, J, B dan K bertipe integer.

{

1. [melakukan operasi penambahan]

For ( I = 0 ; I < B ; I++ )

{ For ( J = 0 ; J < K ; J++ )

{ MAT\_3[I] [J] = MAT\_1[I] [J] + MAT\_2[I] [J]

}

}

}

VOID KALI(MAT\_1[] [],MAT\_2[] [],MAT\_3&[] [],B1,K1,B2,K2)

Fungsi untuk melakukan operasi perkalian antara 2 buah matriks, yaitu MAT\_1 yang berukuran B1 X K1 dan MAT\_2 yang berukuran B2 X K2. Hasil operasi disimpan dalam matriks MAT\_3 yang berukuran B1 X K2. Variabel SUM, MAT\_1, MAT2 dan MAT\_3 bertipe real. Variabel I, J, K, B1, B2, K1, dan K2 bertipe integer. Syarat agar 2 buah matriks dapat dikalikan adalah K1 = B2.

{

1. [menguji apakah perkalian matriks dapat dilakukan]

If ( K1 ≠ B2 )

{ Write (‘Perkalian matriks tidak dapat dilakukan’)

}

Else

[melakukan operasi perkalian]

{ For ( I = 0 ; I < B1 ; I++ )

{ For ( J = 0 ; J < K2 ; J++ )

{ SUM = 0

For ( K = 0 ; K < K1 ; K++ )

{ SUM = SUM + MAT\_1[I] [K] \* MAT\_2[K] [J]

}

MAT\_3[I][J] = SUM

}

}

}

}

VOID CETAK\_MAT(MAT[ ] [ ], BARIS, KOLOM)

Fungsi untuk mencetak data dari matriks MAT yang berukuran BARIS X KOLOM. Matriks MAT bertipe real. Variabel I, J, BARIS dan KOLOM bertipe integer.

{

1. [mencetak data matriks]

For ( I = 0 ; I < BARIS ; I++)

{ For ( J = 0 ; J < KOLOM ; J++ )

{ Write (MAT[I] [J])

}

}

}

Algoritma OPERASI\_MATRIKS

Algoritma untuk membaca data matriks A dan B yang masing-masing berukuran 3 baris X 3 kolom. Kemudian melakukan operasi tambah dan kali. Hasil operasi disimpan dalam matriks C dan D yang berukuran 3 baris X 3 kolom. Seluruh operasi dilakukan oleh fungsi. Fungsi BACA\_DATA untuk membaca data matriks, fungsi TAMBAH untuk melakukan operasi penambahan, fungsi KALI untuk melakukan operasi perkalian dan fungsi CETAK\_MAT untuk menampilkan data matriks.

[deklarasi array]

REAL A[3] [3]

REAL B[3] [3]

REAL C[3] [3]

REAL D[3] [3]

[deklarasi fungsi]

VOID BACA\_DATA (REAL [ ] [ ], INTEGER, INTEGER)

VOID TAMBAH (REAL [ ] [ ], REAL [ ] [ ], REAL& [ ] [ ], INTEGER, INTEGER)

VOID KALI (REAL [ ] [ ], REAL [ ] [ ], REAL& [ ] [ ], INTEGER, INTEGER, INTEGER, INTEGER)

VOID CETAK\_MAT (REAL [ ] [ ], INTEGER, INTEGER)

1. [Tulis pesan untuk memasukkan data matriks A]

Write (“Masukkan data untuk matriks A[3,3]”)

1. [panggil fungsi BACA\_DATA untuk membaca data matriks A]

BACA\_DATA(A,3,3)

1. [Tulis pesan untuk memasukkan data matriks B]

Write (“Masukkan data untuk matriks B[3,3]”)

1. [panggil fungsi BACA\_DATA untuk membaca data matriks B]

BACA\_DATA(B,3,3)

1. [panggil fungsi TAMBAH untuk operasi matriks: C = A + B]

TAMBAH(A,B,C,3,3)

1. [Tulis pesan untuk mencetak hasil penambahan matriks ]

Write (“Hasil penambahan matriks A dan matriks B adalah ”)

1. [panggil fungsi CETAK\_MAT untuk menampilkan isi matriks C]

CETAK\_MAT(C,3,3)

1. [panggil fungsi KALI untuk operasi matriks D = A X B]

KALI(A,B,D,3,3,3,3)

1. [Tulis pesan untuk mencetak hasil perkalian matriks A X B]

Write (“Hasil perkalian matriks A dan matriks B adalah ”)

1. [panggil fungsi CETAK\_MAT untuk menampilkan isi matriks D]

CETAK\_MAT (D,3,3)

1. [selesai]

Halt

Ilustrasi flowchart untuk fungsi BACA\_DATA dan fungsi TAMBAH dapat dilihat pada gambar 5.16, ilustrasi flowchart untuk fungsi CETAK\_MAT dan fungsi KALI dapat dilihat pada gambar 5.17 dan, ilustrasi flowchart algoritma OPERASI\_MATRIKS dapat dilihat pada gambar 5.18 di halaman berikut.





Gambar 5.16 Flowchart dari Fingsi BACA\_DATA dan fungsi TAMBAH



Gambar 5.17 Flowchart dari fungsi CETAK\_MAT dan fungsi KALI



Gambar 5.18 Flowchart dari algoritma OPERASI\_MATRIKS

Untuk contoh kedua dari aplikasi array 2 dimensi, akan dibuat algoritma untuk masalah berikut ini:

Sebuah toko yang menjual alat rumah tangga ingin membuat laporan penjualan bulanan berdasarkan jenis dan banyaknya alat yang dijual. Data hasil penjualan tiap alat per minggu selama satu bulan dapat dilihat pada tabel 5.6 di bawah ini.

Tabel 5.6 Hasil penjualan alat rumah tangga per minggu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Minggu ke- | Mesin Cuci | Blender | Lemari Es | Oven | Televisi |
| 1 | 6 | 4 | 8 | 9 | 3 |
| 2 | 7 | 7 | 10 | 3 | 5 |
| 3 | 5 | 3 | 7 | 8 | 2 |
| 4 | 8 | 10 | 15 | 12 | 5 |

Tabel 5.6 tadi akan disimpan dalam array 2 dimensi REPORT yang berukuran 4 baris dan 5 kolom. Baris menunjukkan minggu dan kolom menunjukkan jenis alat yang terjual. Perhitungan jumlah alat yang dijual akan dilakukan oleh fungsi sehingga algoritma utama hanya berisi instruksi untuk membaca data dan memanggil fungsi.

VOID WEEKLY\_TOTAL(REPORT[ ] [ ], BARIS, KOLOM)

Fungsi untuk menghitung jumlah barang yang dijual tiap minggu. REPORT adalah array 2 dimensi bertipe integer. BARIS menunjukkan minggu dan KOLOM menunjukkan tipe barang. Array REPORT berisi jumlah barang yang dijual per minggu per jenis. Variabel I, J, BARIS, KOLOM dan TOTAL adalah variabel integer.

{

1. [membuat loop untuk menghitung total penjualan per minggu]

For ( I = 0 ; I < BARIS ; I++)

{

1.a. [inisialisasi variabel TOTAL]

TOTAL = 0

1.b. [membuat loop untuk menghitung total penjualan semua barang]

For ( J = 0 ; J < KOLOM ; J++ )

{ TOTAL = TOTAL + REPORT[I] [J]

}

1.c. [mencetak output]

Write(‘Penjualan untuk minggu ke ‘, I ,’ adalah ’,TOTAL,’ unit’)

[kembali ke algoritma utama]

}

}

VOID UNIT\_TOTAL(REPORT[ ] [ ], BARIS, KOLOM)

Fungsi untuk menghitung jumlah unit yang dijual per jenis barang per bulan. REPORT adalah array 2 dimensi bertipe integer. BARIS menunjukkan minggu dan KOLOM menunjukkan jenis barang. Array REPORT berisi jumlah barang yang dijual per minggu per jenis. Variabel I, J, TOTAL dan APPLIANCE adalah variabel integer.

{

1. [membuat loop untuk menghitung total penjualan tiap jenis barang]

For ( J = 0 ; J < KOLOM; J++)

{

1.a. [inisialisasi variabel TOTAL]

TOTAL = 0

1.b. [membuat loop untuk menghitung total penjualan semua barang]

For ( I = 0 ; I < BARIS; I++ )

{ TOTAL = TOTAL + REPORT[I] [J]

}

1.c. [mencetak output]

Write(‘Penjualan untuk barang ‘, J ,’ adalah ’,TOTAL,’ unit’)

[kembali ke algoritma utama]

}

}

Algoritma LAPORAN

Algoritma untuk membuat laporan dari tiap jenis barang yang dijual per bulan dan jumlah barang yang dijual per minggu. Data jumlah barang yang dijual per jenis per minggu disimpan dalam array 2 dimensi REPORT yang berukuran 4 baris dan 5 kolom. BARIS menunjukkan minggu dan KOLOM menunjukkan jenis barang. REPORT, BARIS dan KOLOM bertipe integer. Fungsi WEEKLY\_TOTAL untuk menghitung jumlah barang yang dijual perminggu. Fungsi UNIT\_TOTAL untuk menghitung jumlah unit yang dijual per bulan untuk tiap jenis barang.

[deklarasi array]

REPORT[4][5]

[deklarasi fungsi]

VOID WEEKLY\_TOTAL (INTEGER[ ] [ ], INTEGER, INTEGER)

VOID UNIT\_TOTAL (INTEGER[ ] [ ], INTEGER, INTEGER)

1. [Tulis pesan untuk memasukkan data array REPORT]

Write (“Masukkan data untuk array REPORT[4,5]”)

1. [membaca data untuk array REPORT]

For ( I = 0 ; I < 4 ; I ++ )

{ For ( J = 0 ; J < 5 ; J++)

{ Read(REPORT[I][J])

}

}

1. [panggil fungsi WEEKLY\_TOTAL]

WEEKLY\_TOTAL(REPORT, 4, 5)

1. [panggil fungsi UNIT\_TOTAL]

UNIT\_TOTAL(REPORT, 4, 5)

1. [selesai]

Halt

Ilustrasi flowchart untuk fungsi WEEKLY\_TOTAL dan fungsi UNIT\_TOTAL dapat dilihat pada gambar 5.19 dan ilustrasi flowchart untuk algoritma LAPORAN dapat dilihat pada gambar 5.20 di bawah ini.



Gambar 5.17 Flowchart dari fungsi WEEKLY\_TOTAL dan UNIT\_TOTAL



Gambar 5.18 Flowchart dari algoritma LAPORAN

Sebagai latihan buatlah tabel telusur untuk fungsi WEEKLY\_TOTAL dan fungsi UNIT\_TOTAL dengan menggunakan data pada tabel 5.6.

### Latihan Soal Subbab 5.7

1. Untuk soal-soal berikut ini diasumsikan:

- A adalah array matriks yang berukuran 3 × 3 dan berisi bilangan integer.

- B adalah array linier yang berisi 6 bilangan integer.

- I dan J adalah variabel integer

Tuliskan isi array setelah potongan algoritma berikut ini:

a. For ( I = 0 ; I < 3 ; I++ )

{ For ( J = 0 ; J < 3 ; J++ )

{ A[I] [J] = I + J

Write (A[I] [J])}

}

b. For ( I =0 ; I < 3 ; I++ )

{ For ( J = 2 ; J >1 ; I -- )

{ If (I = = J)

{ A[I] [J] = 0 }

Else

{ A[I] [J] = 1 }

Write (A[I][J])}

}

c. For ( I = 0 ; I < 3 ; I++ )

{ For ( J = 0 ; J < 3 ; J++ )

{ If ( I < J )

{ A[I] [J] = – 1 }

Else

{ If (I = = J)

{ A[I] [J] = 0 }

Else

{ A[I] [J] = 1 }

Write (A[I][J]) }

}

}

d. For ( I = 0 ; I < 3 ; I++ )

{ For ( J = 0 ; J <= I ; J ++ )

{ A[I] [J] = 0

}

For ( J = I+1 ; J < 3 ; J++ )

{ A[I] [J] = 2

}

}

e. For ( I = 0 ; I < 6 : I++ )

{ B[I] = I

}

For ( I = 1 ; I <=3 ; I++ )

{ For (J = 0 ; J < 3 ; J++ )

{ A[I] [J] = B[I] + B[J]

}

}

f. For ( I = 0 ; I < 6 ; I++ )

{ B[I] = I

}

For { J = 0 ; J < 3 ; J++ )

{ A[1] [J] = J + 6

}

For ( I = 0 ; I < 2 ; I++ )

{ For ( J = 0 ; J < 3 ; J ++ )

{ A[B[I+1]] [B[J]] = B[I + J]

}

}

2. Diketahui sebuah matriks A dalam bentuk:



Transpose dari matriks A adalah:



Transpose matriks diperoleh dengan cara menukar kolom dengan baris. Buat fungsi yang mempunyai input argumen-nya adalah matriks dan output argumennya adalah transpose matriks. Variabel m dan n merupakan input argumen. Buat pula algoritma utamanya.

3. Modifikasikan Algoritma OPERASI\_MATRIKS beserta fungsi-fungsinya sehingga hanya dapat menangani operasi tambah dan kali untuk matriks bujursangkar saja.

4. Diketahui 3 buah matriks berukuran 5 x 5. Buatlah algoritma untuk membaca data ketiga matriks lalu menghitung dan menampilkan rata-rata dari ketiga matriks tersebut.

5. Buatlah algoritma untuk membuat segitiga Pascal seperti berikut ini:

1

1. 1

1 2 1

1 3 3 1

1 4 6 4 1

... ... ... ... ... ...

Secara umum sebuah elemen pada suatu baris diperoleh dengan cara menjumlahkan dua elemen dari baris di atasnya. Buatlah algoritma untuk membuat 10 baris segitiga Pascal ini.

6. Diketahui matriks GRADE yang elemennya (GRADE[J,K]) adalah nilai dari siswa ke J untuk soal ke K dan sebuah vektor BOBOT yang elemennya (BOBOT[K]) menunjukkan bobot dari soal ke K. Nilai akhir yang diperoleh siswa ke J dihitung berdasarkan nilai siswa untuk masing-masing soal dari matriks GRADE dikali dengan bobotnya (vektor bobot) dibagi dengan jumlah bobot lalu simpan dalam vektor AKHIR. Berikut ini adalah contoh perhitungan nilai akhir untuk 5 siswa dan 3 buah soal:

BOBOT: 1 2 3

GRADE: 55 70 75 AKHIR: 70.0

65 75 85 78.3

75 70 90 80.8

70 65 70 68.3

80 60 80 73.3

AKHIR[i] = (GRADE[i] [1]\*BOBOT[1] + GRADE[i] [1]\*BOBOT[1] +

GRADE[i] [1]\*BOBOT[1])/(BOBOT[1] + BOBOT[2]

+BOBOT[3])

Contoh perhitungan untuk siswa yang pertama:

AKHIR[1] = (55\*1 + 70\*2 + 75\*3)/(1 + 2 + 3)

AKHIR[1] = 70.0

Buatlah algoritma untuk membaca nilai dari N siswa untuk M soal, membaca bobot untuk masing-masing soal, lalu menghitung nilai akhir yang diperoleh siswa.

7. Ujian akhir terdiri dari 100 soal pilihan ganda. Tiap soal mempunyai 5 pilihan dan hanya satu jawaban yang benar. Hasil ujian dan informasi mahasiswa dapat direpresentasikan sebagai berikut:

KUNCI:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | .... | 99 | 100 |
|  |  |  |  |  |  |  |

JAWABAN:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | .... | 99 | 100 |  | NAMA: |
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| ... |  |  |  |  |  |  |  |  |  |
| ... |  |  |  |  |  |  |  |  |  |
| N |  |  |  |  |  |  |  |  |  |

Array KUNCI adalah array integer, berisi jawaban yang benar dari soal ujian pilihan ganda (angka 1, 2, 3, 4, atau 5). Matriks JAWABAN[I,J] adalah array integer, berisi jawaban dari siswa ke I untuk soal ke J. Array NAMA adalah array string, berisi nama siswa yang mengikuti ujian. Jawaban ujian dikode sebagai angka 1, 2, 3, 4 atau 5. Jika ada lebih dari suatu jawaban, maka dikode sebagai angka 6. Jawaban yang benar bernilai 1 dan seorang siswa baru dapat dinyatakan lulus jika mendapat nilai minimal 60.

Buatlah algoritma untuk mengisi array KUNCI, JAWABAN dan NAMA. Telusurilah array KUNCI dan JAWABAN untuk menentukan nilai siswa, simpan hasilnya dalam array NILAI. Output algoritma adalah semua nama siswa dan nilainya yang mendapatkan predikat lulus.

8. Diketahui tabel ukuran diameter dan berat besi beton yang digunakan untuk membangun gedung tinggi seperti berikut:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ukuran | Diameter (in) | Berat (lb/ft) | Ukuran | Diameter (in) | Berat (lb/ft) |
| 2 | 0.250 | 0.167 | 10 | 1.1270 | 4.303 |
| 4 | 0.500 | 0.668 | 11 | 1.410 | 5.313 |
| 6 | 0.750 | 1.502 | 14 | 1.693 | 7.650 |
| 8 | 1.000 | 2.670 | 18 | 2.257 | 13.600 |

Sebuah kontraktor membutuhkan besi beton dengan ukuran dan panjang sebagai berikut: Ukuran Panjang (ft)

4 5000

8 3500

10 2000

14 1500

18 900

Buatlah algoritma untuk membaca data tabel ukuran diameter dan berat besi beton di atas, menyimpannya dalam array, membaca kebutuhan kontraktor lalu menghitung berat dari masing-masing besi beton dan berat total dari seluruh besi beton yang dibutuhkan.

9. Kontraktor di Indonesia biasa mengunakan satuan meter dan kg. Buat algoritma untuk mengkonversikan tabel pada no.8 di atas menjadi satuan sentimeter, meter dan kg jika diketahui

1 in = 2.54 cm 1 lb = 0.454 kg

1 ft = 12.0 inch 1 m = 100 cm

Lalu lanjutkan algoritma di atas untuk membaca dan menghitung berat besi beton (masing-masing dan total) seperti tabel sebagai berikut:

Ukuran Panjang (m)

4 2500

8 1500

10 900

14 600

18 400

10. Di dalam teori pengolahan gambar digital, terdapat sebuah operasi yang digunakan untuk mengabungkan gambar dengan cara menumpukkan satu gambar pada gambar yang lain. Apabila gambar yang berada di atas memiliki nilai transparansi = X maka gambar hasil gabungan yang tampak akan merupakan perpaduan gambar yang di atas dengan gambar di bawahnya yang akan bergantung pada nilai transparansi X tadi. Operasi ini dikenal sebagai BLENDING. Sebuah gambar hitam putih (monokromatik) dalam format digital akan memiliki nilai kemungkinan warna sebanyak 256 (dari 0 – 255) dengan 0 mewakili warna hitam dan 255 mewakili warna putih, sedangkan nilai di antaranya akan mewakili warna dari hitam ke putih (skala warna abu-abu – grey scale). Gambar hitam putih dalam format digital disimpan sebagai matriks yang masing-masing elemen mewakili komponen warna pada titik (disebut juga pixel) tersebut.

Misalnya diketahui gambar berukuran 4 x 4 titik/pixel dengan nilai dari masing-masing pixel seperti tabel berikut:

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 15 | 255 | 128 |
| 20 | 16 | 240 | 33 |
| 100 | 255 | 11 | 56 |
| 196 | 220 | 122 | 19 |

Anda diminta:

1. Buatlah sebuah fungsi BLENDING yang menerima input nilai transparansi dari gambar kedua (di atas) dan 3 buah array yaitu array pertama dan kedua mewakili gambar asal yang akan di-blend kemudian menghitung warna hasil gabungan/blending dan menyimpan hasilnya pada array ketiga. Fungsi juga menerima parameter banyaknya baris dan kolom yang mewakili ukuran gambar yang akan dioperasikan. (Pada soal ini diasumsikan bahwa ukuran kedua gambar sama).

Formula blending dengan P dan Q adalah gambar asal dan R adalah gambar hasil dan X adalah nilai transparansi gambar yang di atas (Q) dan nilai X adalah antara 0-100 diberikan sebagai berikut:

R[i][j] = X \* Q[i][j] + (1-X/100) \* P[i][j]

1. Buatlah algoritma yang membaca banyaknya baris dan kolom gambar. Kemudian sebanyak baris dan kolom tadi membaca input dari user dan menyimpannya dalam sebuah matriks yang mewakili gambar pertama. Lakukan pula untuk matriks kedua yang menjadi gambar kedua. Lalu gunakan fungsi BLENDING yang Anda buat sebelumnya kemudian tampilkan isi matriks hasil blending pada layar output. (Diasumsikan banyaknya baris dan kolom gambar 1 dan 2 sama).

11. Diketahui data gaji karyawan suatu perusahaan disimpan dalam array matriks. Baris menunjukkan karyawan ke i, kolom pertama menyimpan jumlah hari kerja dalam sebulan, kolom kedua menyimpan golongan karyawan dan kolom ketiga menyimpan gaji yang diterima dalam 1 bulan. Gaji karyawan dalam 1 bulan dihitung berdasarkan golongan seperti tabel berikut:

|  |  |  |
| --- | --- | --- |
| Golongan | Gaji pokok | Uang transport perhari |
| 1 | Rp. 2.000.000 | Rp. 50.000 |
| 2 | Rp. 2.500.000 | Rp. 60.000 |
| 3 | Rp. 2.750.000 | Rp. 75.000 |
| 4 | Rp. 3.000.000 | Rp. 100.000 |

11.a. Buatlah fungsi untuk menghitung gaji seorang karyawan berdasarkan tabel di atas

11.b. Buatlah algoritma untuk memasukkan data karyawan (jumlah hari kerja dan golongan) ke dalam ke array matriks lalu menghitung besar gaji untuk tiap karyawan dengan menggunakan fungsi pada 11.a dan menampilkan hasilnya dalam bentuk tabel. Banyaknya karyawan ditentukan sendiri. Hitung pula total gaji dari seluruh karyawan.