**Bab 3**

**Struktur DataDalam Algoritma**

**Pokok Bahasan**

1. Definisi struktur data dan tipe data serta jenis-jenisnya

2. Pengertian konstanta dan variabel

3. Pengertian Array

4. Pengertian Stack

5. Pengertian Queue

6. Pengertian Tree

7. Pengertian Graph

**Tujuan**

1. Memahami pengertian struktur data dan tipe data serta mengetahui jenis-jenisnya.

2. Memahami pengertian, cara mendeklarasikan dan memakai konstanta dan variabel.

3. Memahami pengertian, cara mendeklarasikan dan menggunakan array.

4. Memahami pengertian, cara mendeklarasikan, operasi dan menggunakan stack.

5. Memahami pengertian, cara mendeklarasikan, operasi dan menggunakan queue.

6. Memahami pengertian Tree dan contoh penggunaannya.

7. Memahami pengertian Pengertian Graph dan contoh penggunaannya.

Dalam istilah ilmu komputer, struktur data adalah cara penyimpanan, penyusunan dan pengaturan data di dalam media penyimpanan komputer sehingga data tersebut dapat digunakan secara efisien. Dengan kata lain struktur data adalah sebuah skema organisasi, seperti variabel dan array dan lain-lain, yang diterapkan pada data sehingga data dapat diinterprestasikan dan sehingga operasi-operasi spesifik dapat dilaksanakan pada data tersebut. Pemakaian struktur data yang tepat didalam proses pemrograman akan menghasilkan algoritma yang lebih jelas dan tepat, sehingga menjadikan program secara keseluruhan lebih efisien dan sederhana. Data adalah representasi dari fakta dunia nyata.

Fakta atau keterangan tentang kenyataan yang disimpan, direkam atau direpresentasikan dalam bentuk tulisan, suara, gambar, sinyal atau simbol. Data merupakan suatu nilai yang bisa dinyatakan dalam bentuk konstanta atau variabel. Konstanta menyatakan nilai yang tetap, sedangkan variabel menyatakan nilai yang dapat diubah-ubah selama eksekusi berlangsung.

**3.1 Tipe Data**

Setiap data memiliki tipe data, apakah merupakan angka bulat, angka pecahan, atau berupa karakter, dan sebagainya. Jadi, tipe data adalah pengelompokan data berdasarkan isi dan sifatnya. Dalam bidang informatika tipe data adalah jenis data yang dapat diolah oleh komputer untuk memenuhi kebutuhan dalam pemrograman komputer.

Setiap variabel atau konstanta yang ada dalam kode program, sebaiknya kita tentukan dengan pasti tipe datanya. Ketepatan pemilihan tipe data pada variabel atau konstanta akan sangat menentukan pemakaian sumberdaya komputer (terutama memori komputer). Salah satu tugas penting seorang programmer adalah memilih tipe data yang sesuai untuk menghasilkan program yang efisien dan berkinerja tinggi. Ada banyak tipe data yang tersedia, tergantung jenis bahasa pemrograman yang dipakai. Secara garis besar tipe data dapat dikategorikan menjadi tiga macam yaitu tipe data dasar (primitive data type) tipe data bentukan (composite data type) dan tipe data abstrak (abstract data type).

**3.1.1 Tipe dasar**

Tipe data dasar atau tipe data sederhana atau biasa juga disebut dengan tipe data primitif adalah tipe data yang sudah ada dan dijadikan standar dalam bahasa pemrograman tertentu. Isi dari tipe data sederhana ini adalah data-data tunggal. Tipe data dasar sudah disediakan oleh program sehingga programmer bisa langsung memakai.

1. Integer (Bilangan Bulat)

Yang dimaksud bilangan bulat adalah, -1, -2, -3, 0, 1, 2, 3, 4 dan lain lain yang bukan merupakan bilangan pecahan.

2. Float atau double (Bilangan Real)

Bilangan real adalah bilangan yang mengandung pecahan desimal. Contoh : 3.45, 6,233.

3. Char (Karakter)

Karakter adalah semua huruf yang ada di dalam alfabet, tanda baca maupun karakter spesial. Karakter ditulis diantara dua tanda petik tunggal. Contoh : 'A'.

4. Boolean (logika)

Boolean adalah tipe data logika yang terdiri dari dua pernyataan benar atau salah. Pernyataan benar biasanya ditulis True atau angka 1, sedangkan pernyataan salah ditulis dengan False atau angka 0. Sedangkan operasi aritmatika yang umum digunakan adalah or, not, and dan xor.

**3.1.2 Tipe data bentukan**

Tipe data bentukan atau tipe data komposit adalah tipe data yang dibentuk dari tipe data dasar dengan maksud mempermudah pekerjaan programer. Yang masuk dalam tipe data bentukan adalah array, string, record, union, struct, dan lain-lain. Tujuan dibuatnya tipe data bentukan adalah :

1. Mempermudah proses pemrograman

2. Mempermudah dalam penambahan variabel

3. Mempermudah pengelompokan data sehingga lebih teratur.

**3.1.3 Tipe data abstrak (Abstract Data Type)**

Tipe data abstrak atau yang dikenal sebagai Abstract Data Type adalah model matematika dari obyek data yang menyempurnakan tipe data dengan cara mengaitkannya dengan fungsi-fungsi yang beroperasi pada data yang bersangkutan. Tipe data abstrak adalah tipe data yang didefinisikan sendiri oleh pemrogram untuk suatu keperluan tertentu yang tidak memungkinkan untuk mendeklarasikan dari tipe data yang sudah ada. Contoh tipe data abstrak adalah stack, queue, list, tree, graph, dan lain-lain.

Harus dibedakan antara pengertian struktur data dan tipe data abstrak. Struktur data hanya memperlihatkan bagaimana data-data di organisir, sedangkan tipe data abstrak mengemas struktur data tertentu sekaligus dengan operasi-operasi yang dapat dilakukan pada struktur data tersebut. Dengan demikian, definisi umum tentang tipe data abstrak dapat dipahami bahwa tipe data abstrak adalah struktur data yang mengandung operasi-operasi atau aturan-aturan tertentu. Pada sub bab selanjutnya akan dibahas beberapa jenis tipe data dari tipe data – tipe data yang telah disebutkan sebelumnya.

**3.2 Konstanta dan Variabel**

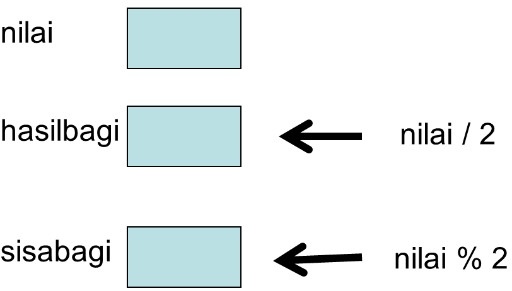
Konstanta dan variabel adalah suatu pengenal (identifier) yang digunakan untuk mewakili suatu nilai tertentu didalam proses program. Berbeda dengan konstanta yang nilainya tidak bisa diubah atau selalu tetap selang eksekusi berlangsung, nilai dari suatu variabel dapat berubah sesuai kebutuhan. Konstanta dan variabel merupakan tempat di memori komputer untuk menyimpan data berupa nilai dengan tipe data tertentu. Konstanta dan variabel harus diberi nama sebagai identifikasi.

Secara logika dapat dibayangkan sebuah konstanta atau variabel sebagai sebuah kotak kosong yang dapat diisi dengan sesuatu tipe data tertentu, misal kita membuat sebuah variabel berupa bilangan bulat, maka dalam logika, kita sedang membuat kotak kosong yang hanya dapat diisi dengan kertas bertuliskan bilangan bulat, tidak boleh jenis bilangan selain bilangan bulat. Ilustrasi logika ini dapat dilihat pada Gambar 3.1. Pada Gambar 3.1 dimisalkan membuat dua buah konstanta atau variabel dengan nama identifier **nilai** dan **X** yang masing-masing dapat digunakan untuk menyimpan suatu nilai dalam memori sesuai dengan tipe data yang telah ditentukan.

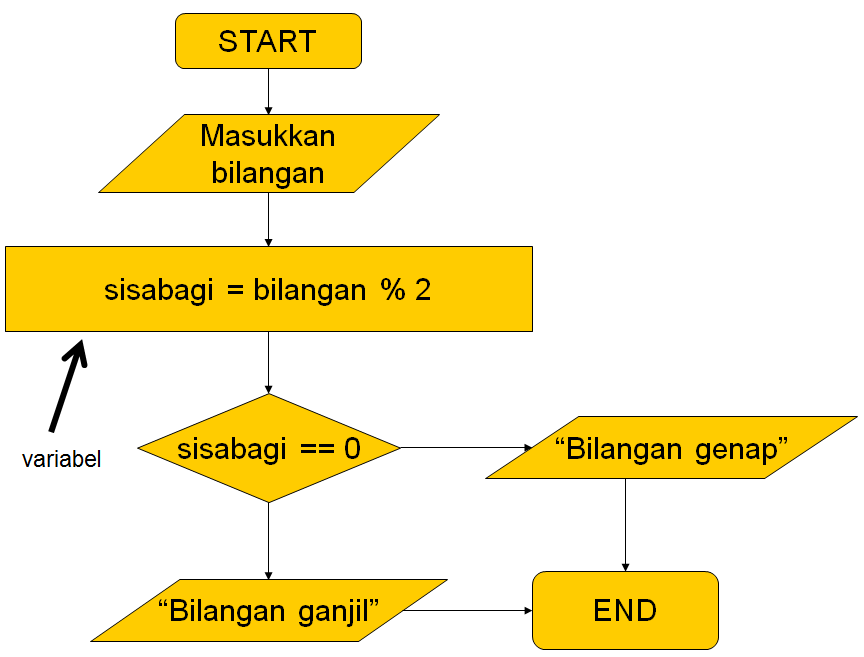


Gambar 3.1. Ilustrasi pembuatan konstanta atau variabel

Semisal **nilai** di Gambar 3.1 adalah sebuah variabel. Maka variabel nilai yang telah dibuat ini selanjutnya dapat digunakan dalam program, semisal dilakukan operasi aritmatika berupa operasi pembagian (/) atau modulo (%). Gambar 3.2 menunjukkan ilustrasi operasi yang terjadi pada variabel nilai. Dalam hal ini variabel nilai dibagi dengan angka 2, dan hasil operasi pembagian disimpan dalam variabel baru yang bernama **hasilbagi**. Operasi aritmatika lain yang terjadi pada Gambar 3.2 adalah variabel nilai dimodulo dengan angka 2 dan hasilnya disimpan dalam variabel baru yang bernama **sisabagi**. Operasi pada suatu konstanta atau variabel tidak hanya terbatas pada operasi aritmatika saja, tetapi juga dapat berupa operasi perbandingan. Misalnya nilai apakah lebih besar dari angka 10 (nilai>10) dan lain- lain. Contoh penggunaan variabel nilai dalam permasalahan sederhana adalah penentuan suatu bilangan termasuk dalam kategori ganjil atau genap seperti terlihat pada Gambar 3.3.



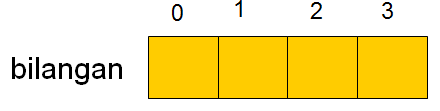
Gambar 3.2. Ilustrasi penggunaan variabel



Gambar 3.3. Contoh penggunaan variabel

**3.3 Array**

Array adalah suatu alokasi beberapa tempat di memori yang tersimpan secara berurutan yang digunakan untuk menyimpan beberapa nilai dengan tipe data yang homogen. Ukuran atau jumlah elemen maksimum array telah diketahui dari awal yaitu ketika array dibuat. Sekali ukuran array ditentukan maka tidak dapat diubah. Ukuran array adalah bilangan bulat positif. Array harus diberi nama sebagai identifikasi. Cara mengaksesnya adalah dengan menyebutkan nama array dan indeksnya. Indeks array dimulai dari 0 sampai dengan n-1 (n adalah ukuran array). Ilustrasi array dapat dilihat pada Gambar 3.4.



Gambar 3.4. Ilustrasi array

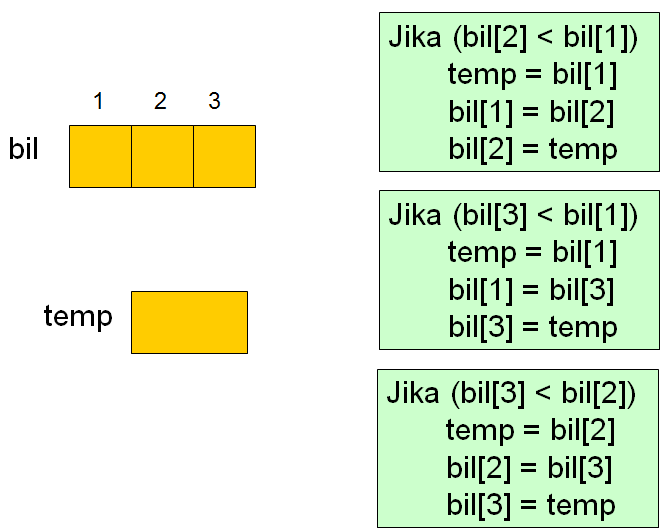
Operasi terhadap elemen array dilakukan dengan pengaksesan langsung. Artinya nilai di masing-masing posisi elemen dapat diambil dan nilai dapat disimpan tanpa melewati posisi-posisi lain. Dua operasi paling dasar terhadap satu elemen array adalah penyimpanan nilai elemen ke posisi tertentu di array dan pengambilan nilai elemen dari posisi tertentu di array. Biasanya bahasa pemrograman menyediakan sintaks tertentu untuk penyimpanan dan pengambilan nilai elemen pada posisi tertentu di array. Contohnya

NilaiMhs[7] =80, berarti menyimpan nilai 80 ke posisi ke-7 dari array NilaiMhs.

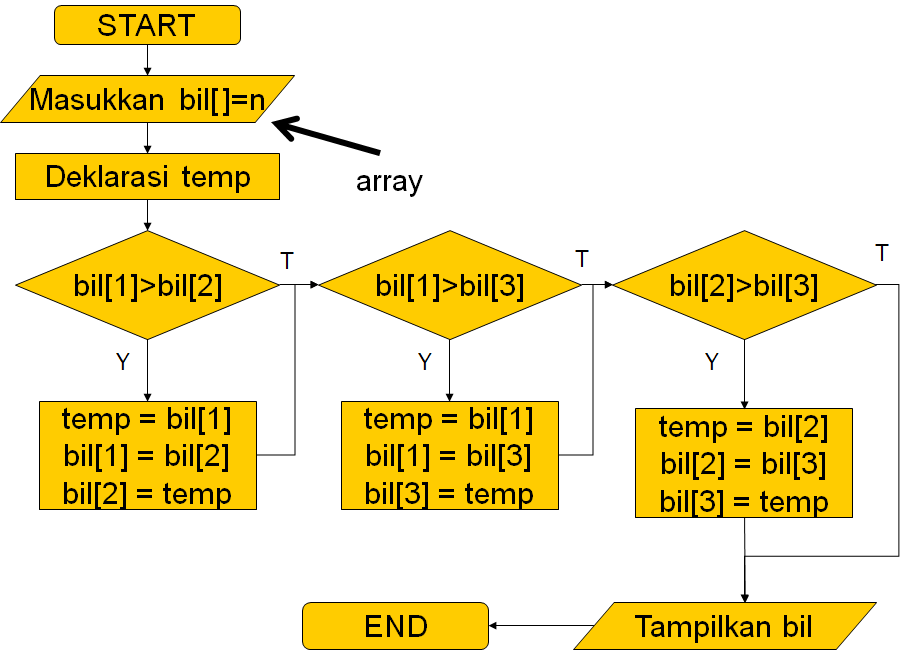
Nama = Mahasiswa[20], berarti mengambil nilai elemen posisi ke-20 dari array

Mahasiswa dan menyimpan nilai tersebut ke variabel yang bernama “Nama”.

Contoh cara mengakses elemen array yang lain dapat dilihat di Gambar 3.5. Contoh penggunaan array dalam permasalahan sederhana adalah pengurutan 3 buah bilangan seperti terlihat pada Gambar 3.6. Untuk mengurutkan tiga buah bilangan dibutuhkan operasi perbandingan yang menghasilkan kondisi benar atau salah (> atau <).



Gambar 3.5. Cara mengakses elemen array



Gambar 3.6. Contoh penggunaan array

Keunggulan array adalah sebagai berikut:

1. Array sangat cocok untuk pengaksesan acak. Sembarang elemen di array dapat diacu secara langsung tanpa melalui elemen-elemen lain.

2. Jika telah berada di suatu lokasi elemen, maka sangat mudah menelusuri ke elemen- elemen tetangga, baik elemen pendahulu atau elemen penerus.

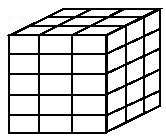
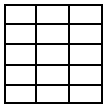
Kelemahan array adalah sebagai berikut:

1. Array mempunyai fleksibilitas rendah, karena array mempunyai batasan harus bertipe homogen. Kita tidak dapat mempunyai array dimana satu elemen adalah karakter, elemen lain bilangan, dan elemen lain adalah tipe-tipe lain

2. Kebanyakan bahasa pemrograman mengimplementasikan array dengan ukuran statik yang sulit diubah ukurannya di waktu eksekusi. Bila penambahan dan pengurangan terjadi terus-menerus, maka representasi statis ini bersifat tidak efisien dalam penggunaan memori.

Dalam bidang pemrograman, array dapat dibuat dengan berbagai macam dimensi. Semisal array dimensi satu akan mirip dengan pola linier seperti pada Gambar 3.4. Sedangkan array 2 dimensi akan tampak seperti tabel atau matrik. Cara mengaksesnya adalah

dengan menyebutkan nama matrik serta baris dan kolomnya. Sedangkan array 3 dimensi akan tampak seperti balok. Ilustrasi array 2 dan 3 dimensi dapat dilihat pada Gambar 3.7.



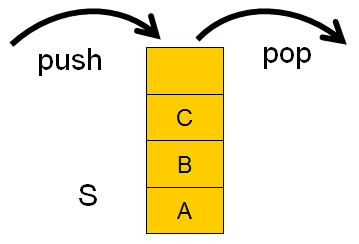
Gambar 3.7. Ilustrasi array 2 dimensi (kiri) dan array 3 dimensi (kanan)

**3.4 Stack**

Secara bahasa, stack berarti tumpukan. Jika dikaitkan dengan struktur data, stack berarti sekumpulan data yang strukturnya menyerupai tumpukan. Stack harus diberi nama sebagai identifikasi. Konsep penyimpanan data pada stack menganut sistem "yang terakhir masuk sebagai yang pertama keluar" (*Last In First Out* / LIFO). Dengan konsep ini, urutan pengambilan data akan berkebalikan dengan urutan penyimpanan data. Elemen yang terakhir disimpan akan menjadi yang pertama kali diambil. Dengan konsep ini maka kita tidak dapat mengambil data yang tersimpan dalam stack secara acak. Data dalam stack harus disimpan dan diambil dari satu sisi atau satu pintu saja. Contoh dalam kehidupan sehari-hari adalah tumpukan piring di sebuah restoran yang tumpukannya dapat ditambah pada bagian paling atas dan jika mengambilnya pun dari bagian paling atas pula.

Cara mengakses stack adalah dengan melakukan operasi dasar pada stack yaitu operasi **push** dan **pop**. Dimisalkan pintu keluar masuknya data pada stack disebut dengan TOP sebagaimana dilihat pada Gambar 3.8. Operasi push adalah proses memasukkan data baru ke stack melalui pintu TOP sehingga data terbaru akan terletak pada posisi teratas. Operasi pop adalah proses mengeluarkan atau mengambil data dari stack dan data yang diambil adalah data yang terletak di posisi teratas. Dengan konsep LIFO maka ketika operasi push dilakukan maka informasi yang diperlukan hanyalah isi atau nilai atau elemen yang akan disimpan atau diambil saja. Operasi push dan pop tidak memerlukan informasi posisi data. Sebagai ilustrasi, stack dapat dilihat seperti pada Gambar 3.8, tampak bahwa stack tersebut bernama S.

TOP



BOTTOM

Gambar 3.8. Ilustrasi stack

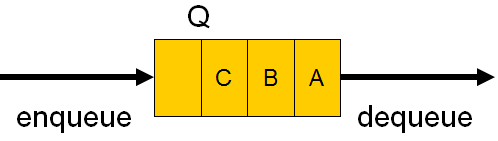
Bagian Top pada stack merupakan pintu untuk keluar masuknya data – data stack. A, B, dan C merupakan suatu koleksi. Dari ilustrasi dapat diketahui bahwa C merupakan data yang terakhir memasuki stack namun pertama keluar dari stack. Begitu sebaliknya dengan A, A merupakan data pertama yang memasuki tumpukan namun terakhir saat keluar dari tumpukan. Di dalam gambar juga terlihat urutan masuk dan keluar yang berkebalikan. Data yang masuk pertama akan keluar terakhir dan sebaliknya.

**3.5 Queue**

Secara bahasa queue adalah antrian. Queue adalah suatu kumpulan data dengan operasi pemasukan atau penyimpanan data hanya diperbolehkan pada salah satu sisi, yang disebut sisi belakang (rear) dan operasi pengambilan atau penghapusan hanya diperbolehkan pada sisi lainnya yang disebut sisi depan (front). Konsep ini dikenal dengan istilah Last In

First Out (LIFO). Ilustrasi queue dapat dilihat pada Gambar 3.9.

REAR



FRONT

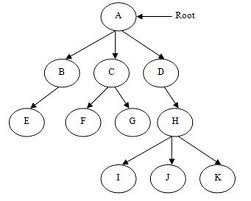
Gambar 3.9. Ilustrasi queue

Jenis struktur data queue sering digunakan untuk menstimulasikan keadaan dunia nyata. Antrian banyak dijumpai dalam kehidupan sehari-hari. Contoh yang paling populer untuk membayangkan sebuah queue adalah antrian pada kasir sebuah bank. Ketika seorang pelanggan datang, akan menuju ke belakang dari antrian. Setelah pelanggan dilayani, antrian yang berada di depan akan maju. Pada saat menempatkan data pada ujung *(rear)* dari queue disebut dengan *enqueue*, pada saat memindahkan elemen dari kepala (*front*) sebuah queue disebut dengan *dequeue*. Sama dengan stack, Dengan konsep FIFO maka ketika operasi enqueue dilakukan maka informasi yang diperlukan hanyalah isi atau nilai atau elemen yang akan disimpan atau diambil saja. Operasi enqueue dan dequeue tidak membutuhkan informasi posisi data.

Dari ilustrasi di Gambar 3.9 dapat diketahui bahwa C merupakan data yang terakhir memasuki queue Q dan akan menjadi yang paling akhir keluar dari queue. Begitu sebaliknya dengan A, A merupakan data pertama yang memasuki queue dan akan menjadi yang pertama saat keluar dari queue.

**3.6 Tree**

Tree merupakan salah satu bentuk struktur data tidak linear yang menggambarkan hubungan yang bersifat hirarki (hubungan one to many) antara elemen-elemen. Bentuk tree menyerupai sebuah pohon, yang terdiri dari serangkaian node (simpul) yang saling berhubungan. Node-node tersebut dihubungkan oleh sebuah vektor. Sehingga tree bisa didefinisikan sebagai kumpulan simpul atau node dengan elemen khusus yang disebut root atau akar. Ilustrasi tree dapat dilihat di Gambar 3.10. Contoh data yang dapat direpresentasikan dengan menggunakan tree adalah silsilah keluarga, hasil pertandingan yang berbentuk turnamen, atau struktur organisasi dari sebuah perusahaan.



Gambar 3.10. Ilustrasi tree

Dalam pemrograman, sebuah tree terdiri dari elemen-elemen yang dinamakan *node* (simpul) yang mana hubungan antar simpul bersifat hirarki. Contoh node pada Gambar 3.10 adalah node A, node B, node C dan seterusnya sampai dengan node K. Jadi Gambar 3.10 memiliki node sebanyak 11 node. Node yang paling atas dari hirarki dinamakan *root,* yaitu node A. Simpul yang berada di bawah root secara langsung, dinamakan anak dari root, yang mana biasanya juga mempunyai anak di bawahnya. Sehingga bisa disimpulkan, kecuali root, masing-masing simpul dalam hirarki mempunyai satu induk (parent). Jumlah anak sebuah simpul induk sangat bergantung pada jenis dari pohon.

Setiap node dapat memiliki 0 atau lebih node anak (child). Sebuah node yang memiliki node anak disebut node induk (parent). Sebuah node anak hanya memiliki satu node induk. Sesuai konvensi ilmu komputer, tree bertumbuh ke bawah, tidak seperti pohon di dunia nyata yang tumbuh ke atas. Dengan demikian node anak akan digambarkan berada di bawah node induknya. Node yang berada di pangkal tree disebut node root (akar), sedangkan node yang berada paling ujung tree disebut node leaf (daun).

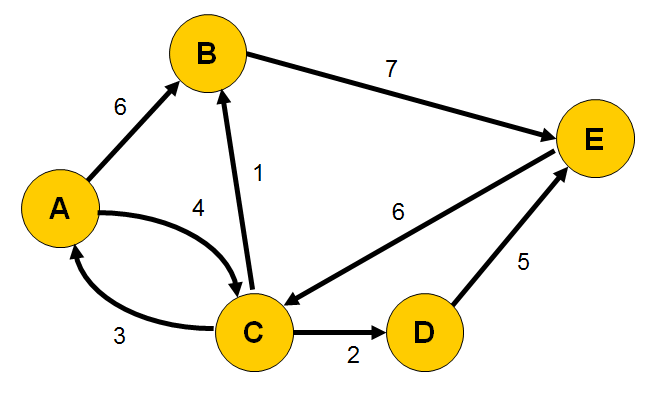
**3.7 Graph**

Dalam bidang matematika dan ilmu komputer, graph adalah struktur yang menggambarkan relasi antar obyek dari sebuah koleksi obyek. Jika struktur linear; misalnya array; memungkinkan pendefinisian keterhubungan sekuensial antara entitas data, struktur data tree memungkinkan pendefinisian keterhubungan hirarkis, maka struktur graph memungkinkan pendefinisian keterhubungan tak terbatas antara entitas data.

Banyak obyek dalam masalah-masalah nyata secara alamiah memiliki keterhubungan langsung secara tak terbatas. Contohnya informasi topologi dan jarak antar kota-kota di suatu pulau. Dalam masalah ini kota x bisa berhubungan langsung dengan hanya satu atau lima kota lainnya. Untuk memeriksa keterhubungan dan jarak tidak langsung antara dua kota dapat diperoleh berdasarkan data keterhubungan langsung dari kota-kota lainnya yang memperantarainya. Contoh lain penggunaan graph adalah untuk menggambarkan jaringan dan jalur kereta api, lintasan pesawat, sistem permipaan, saluran telepon, koneksi elektrik, ketergantungan diantara *task* pada sistem manufaktur dan lain-lain. Terdapat banyak hasil dan struktur penting yang didapatkan dari perhitungan dengan graph.

Representasi data dengan struktur data linear ataupun hirarkis pada masalah ini bisa digunakan namun membutuhkan operasi-operasi yang rumit sehingga kurang efisien. Struktur data graph secara eksplisit menyatakan keterhubungan ini sehingga pencariannya langsung dilakukan pada strukturnya sendiri.

Definisi dari suatu graph adalah himpunan obyek-obyek yang disebut node (atau vertek) yang terhubung oleh edge. Biasanya graph digambarkan secara grafis sebagai kumpulan lingkaran yang melambangkan node yang dihubungkan oleh garis yang melambangkan edge. Edge dalam suatu graph bisa berupa edge berarah atau tidak berarah. Ilustrasi graph dapat dilihat pada Gambar 3.11. Pada gambar tersebut terlihat bahwa graph memiliki 5 buah node. Pada ilustrasi ini dimisalkan node mewakili sebuah kota. Maka dapat dilihat bahwa dari kota A menuju kota E bisa dilalui melalui path A-B-E atau path A-C-D-E.



Gambar 3.11. Ilustrasi graph

Contoh lain masalah-masalah yang dapat diselesaikan dengan menggunakan struktur data graph adalah:

a. Masalah path minimum (Shortest path problem), yaitu mencari route dengan jarak terpendek dalam suatu jaringan transportasi.

b. Masalah aliran maksimum (maximum flow problem), yaitu menghitung volume aliran

BBM dari suatu reservoir ke suatu titik tujuan melalui jaringan pipa.

c. Masalah pencariah dalam graph (graph searching problem) yaitu mencari langkah- langkah terbaik dalam program permainan catur komputer.

d. Masalah pengurutan topologis (topological ordering problem), yaitu menentukan urutan pengambilan mata kuliah yang saling berkaitan dalam hubungan prasyarat.

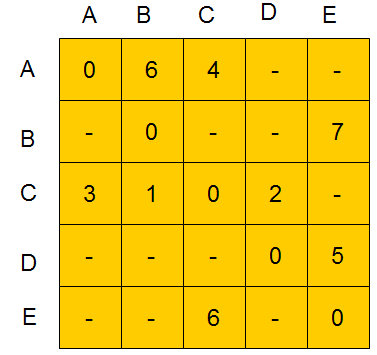
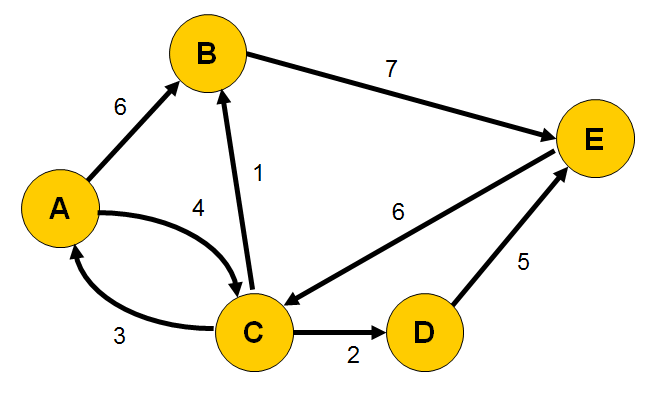
e. Masalah jaringan tugas (Task Network Problem), yaitu membuat penjadwalan pengerjaan suatu proyek yang memungkinkan waktu penyelesaian tersingkat.

f. Masalah pencarian pohon rentang minimum (Minimum Spanning Tree Problem), yaitu mencari rentangan kabel listrik yang totalnya adalah minimal untuk menghubungkan sejumlah kota.

g. Travelling Salesperson Problem, yaitu tukang pos mencari lintasan terpendek melalui semua alamat penerima pos tanpa harus mendatangi suatu tempat lebih dari satu kali.

h. Four-color proble, yaitu dalam menggambar peta, memberikan warna yang berbeda pada setiap propinsi yang saling bersebelahan.

Untuk menyelesaikan permasalahan jalur terpendek (sortest path) dari graph pada Gambar 3.11. Maka harus dilakukan penerjemahan dari bentuk graph ke bentuk matrik keterhubungan langsung (list adjancency) yang dibuat dengan menggunakan array dua dimensi. Hasil pembuatan list adjancency graph dapat dilihat pada Gambar 3.12. Nilai-nilai yang tertera dalam list tersebut adalah jarak antara dua kota. Misalnya nilai 6 yang terletak pada posisi baris 1 kolom 2. Artinya adalah ada jalur dari kota A ke kota B dan jaraknya 6 satuan. Kebalikannya dari kota B ke kota A (baris 2 kolom 1) tidak ada jalur sehingga bernilai -. Untuk mendapatkan jalur terpendek dan nilai jaraknya dapat diselesaikan dengan menggunakan beberapa algoritma sortest path, misalnya algoritma Dijkstra, yang akan dibahas di bab 11.



Gambar 3.12. List adjancency

**3.8 Latihan Soal**

1. Apakah yang dimaksud dengan tipe data?

2. Sebutkan dan jelaskan penggolongan tipe data!

3. Apakah yang dimaksud dengan struktur data?

4. Apakah yang dimaksud dengan konstanta dan variabel?

5. Apakah beda antara konstanta dan variabel?

6. Apakah yang dimaksud dengan array? Jelaskan!

7. Apakah yang dimaksud dengan stack? Jelaskan operasi pada stack dan beri contoh penerapannya dalam kehidupan sehari-hari!

8. Apakah yang dimaksud dengan queue? Jelaskan operasi pada stack dan beri contoh penerapannya dalam kehidupan sehari-hari!

9. Apakah yang dimaksud dengan tree? Berikan contoh penerapannya!

10. Apakah yang dimaksud dengan graph? Berikan contoh penerapannya!