Struktur Data

Diktat kuliah

Dr. Arya Adhyaksa Waskita

STMIK Eresha - 2017

Daftar Isi

Daftar Gambar

Daftar Program

Kata Pengantar

Diktat kuliah struktur data diperuntukkan bagi peserta mata kuliah struktur data STMIK Eresha semester II. Diktat ini menggunakan buku [Group, 2005] sebagai acuan utama.

Serpong,

Dr. Arya Adhyaksa Waskita

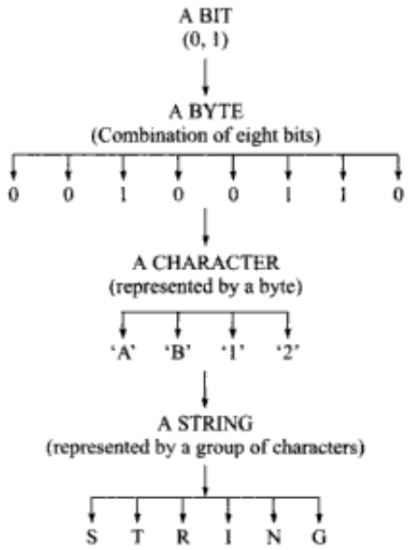
BAB 1

Pendahuluan

Memanfaatkan komputer dalam menyelesaikan suatu masalah menuntut pengetahuan tentang transformasi masalah tersebut agar dapat diselesaikan dengan komputer. Materi struktur data digunakan untuk memahami transfromasi data yang terlibat dalam suatu masalah serta operasi yang berlaku pada data sehingga dapat diselesaikan dengan komputer. Transformasi tersebut tentang bagaimana data diorganisasikan, dikendalikan serta struktur yang harus dirancang dan diterapkan. Targetnya adalah sebuah solusi yang sederhana dan efisien.

1.1 Representasi data

Dalam komputer, data dan instruksi dinyatakan dalam bentuk biner, berupa susunan angka 0 dan 1 dengan arti tertentu. Susunan 8 `bit` biner disebut dengan `byte` digunakan untuk merepresentasikan karakter. Sedangkan kumpulan karakter akan menjadi string. Ilustrasinya ditunjukkan pada Gambar 1.1.



Gambar 1.1: Representasi data dalam komputer

Acuan dalam merepresentasikan data ke dalam sistem biner antara adalah ASCII (*American Standard Code for Information Interchange*) dan BCD (*binary-coded decimal*). Dan setiap bahasa pemrograman telah mendefinisikan representasi untuk perintah dan data yang digunakannya seperti ilustrasi pada Gambar 1.1. Setiap jenis data memiliki skema representasi yang berbeda. Representasi data integer dilakukan menggunakan 8 bit (1 byte) biner. Misalnya, 27 dinyatakan dalam bilangan biner sebagai 00011011. Sementara untuk menyatakan bilangan negatifnya, dapat dilakukan dengan dua pendekatan. Yang pertama adalah melakukan operasi komplemen pada setiap bit biner, yaitu setiap bit 0 diubah menjadi 1, sedangkan bit 1 menjadi 0. Dengan demikian, angka -27 jika dinyatakan dalam biner dengan operasi komplemen menjadi 11100100. Sedangkan representasi yang kedua adalah menambahkan angka 1 pada hasil operasi komplemen pertama. Sehingga angka -27 dapat dinyatakan dalam bilangan biner dengan skenario kedua sebagai 11100101.

Selan itu, bilangan desimal dapat juga dinyatakan dalam bilangan biner dengan terlebih dahulu memisahkan bagian bilangan bulat (mantissa) dengan bagian pangkat (eksponen)nya. Sebagai contoh, bilangan 209.52 dapat dinyatakan sebagai 20952.10^{-2} dalam bilangan basis 10. 20952 adalah bagian mantissa, sedangkan -2 adalah bagian eksponennya. Dengan skenario sebelumnya, angka 20952 dinyatakan dalam biner sebagai 101000111011. Sedangkan bagian eksponennya dinyatakan dalam skenario komplemen sebagai 11111101. Sehingga keduanya dapat digunakan untuk menyatakan bilangan 20952.10^{-2} sebagai 101000111011.11111101.

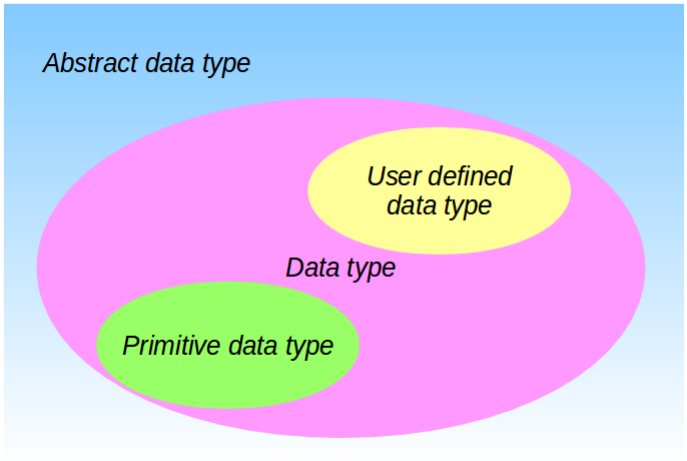
1.2 Tipe data abstrak

Pada kondisi tertentu, kita tidak dapat (sulit) merepresentasikan data atau obyek ke dalam komputer. Pada kondisi tersebut, bukan saja data yang perlu disimpan, tetapi operasi matematika yang berlaku pada data tersebut. Datanyapun seringkali berupa data jamak dari tipe yang berbeda. Bahasa pemorgraman modern menyebut tipe data ini sebagai tipa data abstrak (ADT/*Abstract Data Type*) yang muncul dalam bahasa pemrograman berorientasi obyek.

Untuk kasus yang lebih sederhana, di mana fokus utamanya justru pada sejumlah data yang perlu diacu bersama di saat yang sama, kita mengenal tipe data `structure`. Di sini, kita dapat mendefinisikan satu tipe data baru yang terdiri dari sejumlah tipe data yang telah didefinisikan dalam bahasa pemrograman yang kita gunakan.

Terdapat dua istilah terkait tipe data ini. Selain data struktur yang merupakan data yang tersusun dari sejumlah elemen data, terdapat juga tipe struktur *structured type*. *Structure type* merupakan hubungan yang terdapat pada data (elemen). Jika data struktur fokus pada data dengan elemen lebih dari satu, *structured type* fokus pada hubungan antar data.

Dapat disimpulkan bahwa ADT merupakan spesifikasi, sementara tipe data adalah penerapan dari ADT. ADT dapat juga dipandang sebagai merupakan bentuk umum dari tipe data. Jika telah didefinisikan dalam bahasa pemrograman tertentu sering disebut sebagai *primitive data type*, sedangkan jika didefinisikan oleh pengguna disebut sebagai *user defined data type*. Sementara data struktur adalah kumpulan data dengan tipe apapun. Hubungannya dapat diilustrasikan dalam Gambar 1.2 berikut.



Gambar 1.2: Hubungan tipe data

1.3 Prinsip dasar pemrograman

Prinsip dasar dalam pemrograman tahapan yang harus dilakukan dalam menyelesaikan masalah menggunakan bantuan komputer. Salah satu tahapan dalam pemrograman terstruktur dikenal sebagai *system development life cycle* (SDLC). Tahapan SDLC adalah sebagai berikut.

1. Analisis masalah: apa yang menjadi masalah, apa saja solusinya, apa dan siapa (pemrograman berorientasi obyek) yang terlibat, tahapan dalam penyelelesaian, indikasi keberhasilan, dll.
2. Membuat *prototype*, umunya dilakukan dalam bentuk *pseduo code* atau diagram alir.
3. Membangun algoritma:
   - merancang
   - verifikasi
   - analisis
   - memprogram
     - menguji
     - evaluasi
     - perbaikan (jika perlu)
     - optimasi
   - Merawat

Dalam merancang algoritma (termasuk juga program yang menerapkan algoritma) seperti disebutkan dalam tahapan ke-3 dari SDLC, diperlukan tahapan berikut.

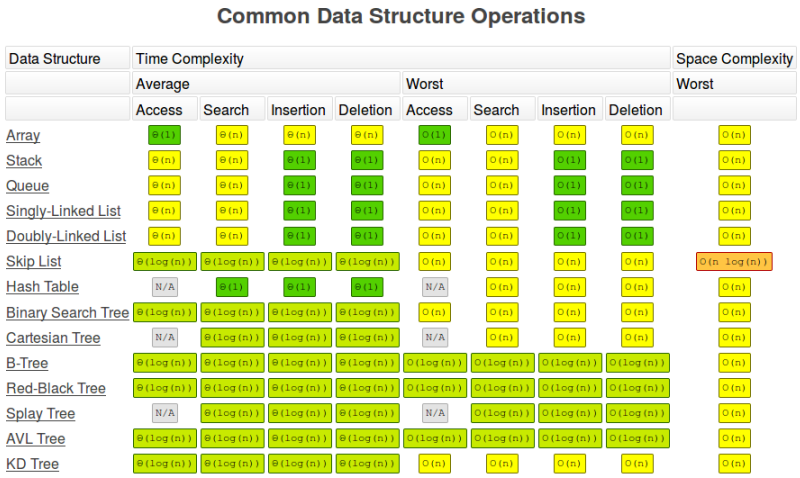
1. Program harus sejalan dengan masalah yang dihadapi, apakah itu prosedur, data maupun struktur datanya.

2. Bekerja baik pada semua kondisi di mana masalah terjadi. Jika ada kondisi di mana solusi yang digunakan tidak dapat mencapai hasil yang diinginkan, kondisi tersebut menjadi pengecualian dan harus disebutkan secara eksplisit.

3. Dokumentasi tentang bagaimana rancangan tersebut dihasilkan (dokumentasi rancangan)

4. Tersusun atas sejumlah modul, fungsi, subrutin. Modularisasi menjadi penting ketika algoritma yang dibangun komplek dan besar. Modularisasi akan memberikan keuntungan dari kemudahan pengembangan.

5. Waktu eksekusi dan ruang penyimpanan yang diperlukan. Kedua hal ini adalah kriteria sebuah algoritma disebut baik. Kebutuhan media penyimpanan yang dimaksud di sini meliputi media yang bersifat *volatile* (*Random Access Memory*/RAM) atau *non-volatile* (*hard disk*). Sementara waktu eksekusi tentu akan berbeda ketika dijalankan di mesin yang berbeda, sehingga ukuran tepatnya adalah jumlah operasi aritmatika dan logika yang dijalankan sebuah algoritma. Nilainya dinotasikan sebagai Θ atau $\mathbf{O}$. Nilai notasi Θ untuk beragam struktur data dan algoritmanya diilustasikan pada Gambar 1.3[1].



Gambar 1.3: Perbandingan nilai Θ sejumlah struktur data dan algoritma

Algoritma yang dirancang selanjutnya dianalisis kinerjanya dalam bentuk waktu eksekusi. Selain dipengaruhi oleh mesin di mana algoritma dijalankan, waktu eksekusi juga dipengaruhi oleh masukannya. Sebagai ilustrasi, mengurutkan sejumlah nilai yang telah terurut sebelumnya tentu membutuhkan waktu yang lebih singkat dibanding dengan nilai-nilai yang belum terurut. Waktu eksekusi yang dipengaruhi masukan dapat dikelompokkan dalam 3 kategori.

1. *Best*: waktu minimum yang diperlukan sebuah algoritma.

2. *Average*: waktu rerata dari eksekusi sebuah algoritma dalam menyelesaikan masalah yang sama dengan masukan yang berbeda. Jika satu masalah memiliki kemungkinan masukan

[1] http://bigocheatsheet.com/

sebanyak n, maka waktu rerata adalah rata-rata dari waktu untuk menyelesaikan masalah dengan n masukan tersebut. Jika waktu untuk menyelesaikan algoritma untuk satu jenis masukan adalah T_i, maka waktu rerata (T_{avg}) dapat diformulasikan dalam persamaan (1.1).

3. *Worst*: waktu terburuk sebuah algoritma dalam mengolah satu jenis masukan.

$$T_{avg} = \frac{\sum_{i=1}^{n} T_i}{n} \tag{1.1}$$

Algoritma sendiri dapat didefinisikan sebagai urutan instruksi yang harus diikuti untuk menyelesaikan suatu masalah. Atau dapat juga didefinisikan sebagai prosedur untuk mengubah *input* menjadi *output*. Karakteristik algoritma yang baik adalah sebagai berikut.

1. Setiap instruksi di dalamnya harus unik dan tepat.
2. Setiap instruksi tidak boleh dijalankan secara berulang tanpa batas.
3. Perulangan tugas yang sama harus dihindari
4. Harus memberikan hasil yang tepat untuk masalah yang dihadapi
5. Efisien dalam menyelesaikan masalah. Faktor efisiensi seperti yang telah disebutkan terdiri dari efisiensi terkait media penyimpanan dan waktu eksekusi.

BAB 2

Bahasa Pemrograman C++

2.1 Pendahuluan

Bahasa pemrograman `C++` dipilih dalam diktat ini karena `C++` secara eksplisit memanfaatkan *pointer* sebagai sarana membangun beragam struktur data. Sementara bahasa pemrograman lain yang lebih modern telah mengemas hal tersebut ke dalam class. Sebagai bahan ajar, `C++` dinilai lebih mampu memberikan pemahaman tentang merancang dan menerapkan rancangan struktur data ketimbang hanya menggunakan class yang telah terdefinisi dengan baik. Program 2.1 memperlihatkan program sederhana `C++`.

Program 2.1: hello.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5   cout << "Hello world\n";
6   return 0;
7 }
```

Penjelasannya adalah sebagai berikut.

1. Baris ke-1, merupakan pemanggilan pustaka di mana fungsi-fungsi yang digunakan dalam program diletakkan. Dalam contoh Program 2.1, digunakan fungsi `cout` yang digunakan untuk menampilkan data ke *standard output* (layar) dan fungsi tersebut didefinisikan di pustaka `iostream`.

2. Baris ke-2, digunakan untuk menyederhanakan penulisan fungsi. Karena `C++` adalah bahasa pemrograman berorientasi obyek, setiap fungsi tentu terdefinisi di dalam class tertentu, sehingga pemanggilan terhadap fungsi `cout` dilakukan apa adanya. Sebaliknya, tanpa deklarasi ini pemanggilan fungsi `cout` harus dilakukan dengan cara `std::cout`.

3. Baris ke-4, setiap program berbasis `C/C++` akan dieksekusi dari fugnsi yang bernama `main`. Tanpa fungsi ini, program `C/C++` tidak dapat dieksekusi (proses kompilasi tidak dapat menghasilkan *executable file*).

4. Baris ke-5, digunakan untuk menampilkan teks berupa "Hello World" ke layar.

5. Baris ke-6, digunakan untuk memenuhi syarat berupa *return value* dari fungsi main (Baris ke-4). Biasanya, fungsi yang mengembalikan nilai 0 disepakati sebagai fungsi yang menjalankan tugas dengan baik. Jika tidak ingin melakukan pengembalian fungsi, maka fungsi `main` harus diberi tanda *return value* sebagai `void`.

6. Setiap fungsi dalam `C/C++` dimulai oleh karakter "{" dan diakhiri oleh karakter "}", dan setiap instruksi di akhiri dengan karakter ";".

7. Di terminal, jalankan perintah `g++ -o hello hello.cpp`, dengan penjelasan.

   - `g++`: memanggil aplikasi kompilator `C++`
   - `-o`: opsi untuk melakukan kompilasi dan *linking* (meski dalam konteks Program 2.1 tidak diperlukan) sehingga dihasilkan *executable file*.
   - `hello`: nama executable file, jika dijalankan di Microsoft Windows dengan bantuan aplikasi MinGW, secara otomatis ditambahkan ekstensi `.exe`.
   - `hello.cpp`: kode sumber yang akan dikompilasi

2.2 IDE

IDE (*Integrated Development Environment*) adalah aplikasi yang digunakan dalam mengembangkan program. Untuk tujuan mempelajari materi struktur data, sebenarnya nyaris tidak diperlukan IDE. Kita hanya perlu memiliki kompilator *C++* sebagai bahasa pemrograman yang memiliki notasi eksplisit terkait *pointer* (dijelaskan dalam sub bab 2.3) serta editor teks untuk menulis sejumlah instruksi `C++`. Untuk pengguna sistem operasi GNU/Linux dalam berbagai variannya, nyaris semua jenis kompilator tersedia secara gratis. Saya sendiri menggunakan editor teks Geany[1] dengan kompilator `C++` dari GNU *Project*.

Namun, tidak demikian halnya dengan pengguna sistem Operasi Microsoft Windows. Umumnya, aplikasi di sistem operasi ini dikemas secara terintegrasi, bahkan dengan asumsi kita sedang berhadapan dengan sebuah proyek besar yang butuh cara mengelola proyek (dalam hal ini *software*) yang mumpuni. Padahal, di sisi lain, kita hanya memerlukan editor teks tempat di mana kita menyusun program serta kompilator yang akan membuat program tersebut dapat dijalankan. Beruntung, ada sejumlah aplikasi yang dapat memberikan kita lingkungan pengembangan yang sederhana seperti halnya di sistem operasi GNU/Linux. Misalnya dengan aplikasi MinGW (Minimalist GNU for Windows)[2] atau Cygwin[3]. Jangan lupa untuk memasang kompilator `C++` pada keduanya. Untuk yang masih kesulitan menggunakan aplikasi berbasis perintah baris seperti pada MinGW dan Cygwin, dapat menggunakan aplikasi IDE yang canggih seperti Visual Studio atau NetBeans dengan *plugin* kompilator `C/C++`.

[1] https://www.geany.org/
[2] http://www.mingw.org/
[3] https://www.cygwin.com/

2.3 Tipe data `pointer` dan `struct`

Tipe data *pointer* adalah salah satu tipe data yang didefinisikan di `C/C++`. Tugasnya untuk menunjuk lokasi memori yang menjadi perhatian. Tipe data ini, tidak untuk menyimpan nilai suatu variabel tetapi alamat memori secara fisik di mana suatu nilai disimpan. Meskipun hanya bertugas menunjuk, tetapi tipe datanya harus disesuaikan dengan nilai yang akan ditunjuk.

Sebagai contoh, untuk mendefinisikan sebuah variabel *integer*, `C/C++` memiliki sintaks `int a;`. Untuk membuat sebuah tipe data *pointer* yang akan menunjuk variabel bertipe *integer*, `C/C++` memiliki sintaks `int *p;`. Sedangkan untuk membuat *pointer* `p` menunjuk variabel `a`, `C/C++` memilki sintaks `p=&a;`. Proses *assignment* tersebut juga dapat dilakukan sekaligus pada saat deklarasi dengan sintaks `int *p=&a;`. Intinya, tipe data *pointer* dicirikan dengan karakter "`*`" di awal variabel dan mengikuti variabel yang ditunjuk. Jika akan digunakan menunjuk vairabel *integer*, maka *pointer* juga harus dideklarasikan sebagai *integer*.

Kemudian, ada tipe data lainnya yang bertugas untuk mengemas sejumlah tipe data yang telah terdefinisi dalam `C/C++` yang disebut sebagai `struct`. Tipe data ini memungkinkan kita membuat tipe data baru yang mirip dengan sebuah obyek tanpa fungsi, karena hanya memiliki atribut. Sekali diacu dalam program, maka kita dapat mengakses semua data yang dikemas dalam `struct`. Program 2.2 mengilustrasikan penggunaan variabel `pointer` dan `struct`.

Program 2.2: intro.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  typedef struct data {
5          int nilai;
6          struct data *pointer;
7  }DATA;
8
9  int main() {
10         int a;
11         a=10;
12         int *p;
13         p=&a;
14         DATA x;
15         x.nilai=7;
16         x.pointer=NULL;
17
18         cout << "Nilai variabel a adalah " << a << " dan berlokasi di alamat " << &a << endl;
19         cout << "Nilai variabel *p adalah " << *p << " dan berlokasi di alamat " << p << endl;
20         cout << "x.nilai=" << x.nilai << ", x.pointer merujuk ke alamat " << x.pointer << endl;
21         return 0;
22 }
```

Penjelasannya adalah sebagai berikut.

1. Baris ke-4 s/d 7: definisi tipe data baru berupa `struct data`. Pernyataan `typedef` di depannya menunjukkan bahwa tipe data `struct data` memiliki nama alias `DATA`. Itu sebabnya di baris ke-4 tertulis `DATA` yang berarti nama alias untuk `struct data`.

    - Baris ke-5: definisi penyusun pertama `struct data`, yaitu `int nilai;`.

- Baris ke-6: definisi penyusun kedua `struct data`, yaitu `struct data *pointer;`. Karena penyusun kedua ini ditugaskan untuk menujukan variabel dengan tipe `struct data`, maka ia harus didefinisikan dengan tipe yang sama.

2. Baris ke-10 dan 11: deklarasi dan *assignment* variabel *integer* dengan nama `a` dan dengan nilai `10`.

3. Baris ke-12 dan 13: deklarasi dan *assignment* variabel `pointer` dengan nama `*p` dan bernilai `&a` (alamat dari variabel `a` yang disimpan di RAM).

4. Baris ke-14: deklarasi variabel `DATA` dengan nama `x`. Perhatikan kembali cara mendeklarasi variabel di baris ke-10 dan 12.

5. Baris ke-15 dan 16: *assignment* nilai elemen penyusun variabel `x`, yaitu

   - `x.nilai=7`
   - `x.pointer=NULL`

6. Baris ke-18 s/d 20: menuliskan nilai-nilai yang sebelumnya di-*assign* ke variabel untuk ditampilkan di layar.

7. Baris tanpa penjelasan memiliki penjelasan yang sama dengan penjelasan Program 2.1.

Daftar Referensi

[Group, 2005] Group, I. (2005). *Data Structures Using C*. Ace series. McGraw-Hill Education (India) Pvt Limited.