P1 OPL

Minggu, 21 Februari 2021 15.12

Pokok Bahasan:
- Ilmu Komputer
- Programming
- Problem Solving (Pemecahan Masalah)

Computer Science (Ilmu Komputer)

- Jika berbicara tentang istilah "Computer Science" atau ilmu komputer, bukan berati kita membahas tentang komputer, akan tetapi komputer disini hanyalah sebagai alat untuk membantu memecahkan masalah.
- Ilmu komputer adalah studi/pelajaran tentang masalah, pemecahan masalah, dan solusi yang muncul dari proses pemecahan masalah.
- Ketika seorang ilmuwan komputer diberikan sebuah masalah, tujuan dari seorang ilmuwan komputer itu adalah untuk menciptakan sebuah algoritma
- Algoritma adalah daftar instruksi langkah demi langkah untuk memecahkan setiap masalah yang mungkin muncul.
- Algoritma dapat juga diartikan sebagai proses terbatas yang jika diikuti akan menyelesaikan masalah.
- Perlu dicatat bahwa tidak semua algoritma dapat menyelesaikan sebuah masalah.

- Ilmu abstraksi dibutuhkan dalam menyelesaikan masalah.
- Abstraksi diperlukan dalam menggambarkan permasalahan dan solusi.
- Teknik dalam melakukan abstraksi pemecahan masalah ada dua, yakni perspektif logikal dan perspektif fisikal
- Contoh
  Sebuah mobil digunakan untuk mengantarkan kita ke kampus. Supir mobil akan berinteraksi dengan mobil, dari memasukkan sebuah kunci mobil, menyalakan mesin mobil, kopling, rem, gas dan stir mobil. Dari pandangan/perspektif ini dikenal sebagai perspektif logikal dari sebuah mobil. Supir menggunakan fungsi/fitur yang telah disediakan pembuat mobil yang bertujuan untuk mengantarkan kita dari satu lokasi ke lokasi lain. Terkadang fungsi/fitur ini bisa disebut juga sebagai interface.

  Ketika mekanik/tukang servis mobil yang memperbaiki mobil mempunyai pandangan/perspektif yang berbeda. Seorang mekanik tidak hanya harus mengetahui cara berkendara, melainkan juga harus mengetahui detail-detail yang terdapat pada mobil. Seorang mekanik juga harus mengerti tentang bagaimana mesin mobil bekerja, bagaimana cara pergantian persneling mobil, batas-batas suhu mobil, dan detail lainnya. Pandangan/perspektif ini disebut dengan perspektif fisikal.

  Sama halnya ketika kita menggunakan komputer. Orang pada umumnya menggunakan komputer untuk menulis dokumen, mengirim dan menerima email, menjelajah web, mendengarkan musik atau bermain game tanpa harus mengetahui detail-detail apa saja yang diperlukan untuk menjalankan aplikasi-aplikasi yang dipakai. Mereka melihat komputer dari perspektif logikal atau perspektif pengguna. Seorang ilmuwan komputer, programmer, staff IT, dan administrator sebuah sistem melihatnya dengan sudut pandang/perspektif yang berbeda. Mereka harus mengetahui/paham detail-detail bagaimana pengoperasian sistem, bagaimana konfigurasi protokol jaringan/networknya, dan bagaimana code script pemrograman dapat mengendalikan fungsi dari aplikasi.

- Sebuah prosedur diperlukan dalam menghubungkan kedua perspektif tersebut, prosedur ini di sebut dengan prosedur abstraksi
- Contoh prosedur abstraksi:
  Kita ingin membuat sebuah program Python untuk menyelesaikan masalah perhitungan akar kuadrat. Python telah menyediakan modul bernama Math. Modul math mempunyai beberapa fungsi/metode di dalamnya termasuk fungsi perhitungan akar kuadrat. Kita dapat menggunakan fungsi tersebut seperti berikut:

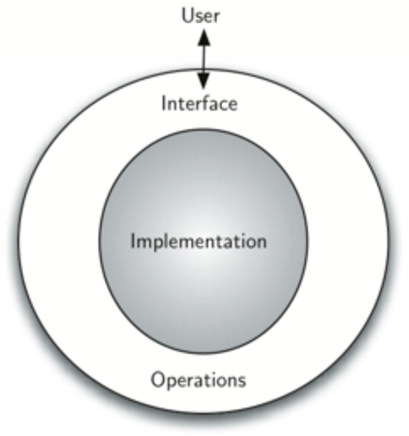
```
>>> import math
>>> math.sqrt(16)
4.0
```

  Kita tidak perlu mengetahui proses di dalam fungsi sqrt yang terdapat di dalam modul math, kita hanya perlu tahu prosedur penulisan penggunaan fungsi tersebut. Perspektif/pandangan ini biasanya disebut dengan black box view

Programming

- Programming/pemrograman adalah proses mengambil algoritma dan mengkodekannya menjadi notasi bahasa pemrograman, sehingga dapat dieksekusi oleh komputer.
- Tanpa ada algoritma, tidak mungkin ada program.
- Ilmu komputer bukanlah mempelajari tentang programming, akan tetapi programming adalah bagian penting dari apa yang dilakukan ilmuwan komputer.
- Algoritma menggambarkan solusi untuk suatu masaah dalam hal data yang diperlukan untuk merepresentasikan/mewakilkan masalah tersebut serta serangkaian langkah yang diperlukan untuk membuat hasil/solusi yang diinginkan
- Oleh karena itu, bahasa pemrograman harus dapat menyediakan notasi-notasi untuk merepresentasikan proses dan data yang dibutuhkan algoritma. Contohnya notasi struktur kendali (if-else), tipe data (int, string, float), dll.

struktur data dan tipe data abstrak

- untuk mengelola kompleksitas dari suatu permasalahan dan proses pemecahannya, seorang ilmuwan komputer menggunakan abstraksi untuk fokus pada "gambaran besar" masalah tanpa harus kehilangan detail permasalahan.
- dengan membuat model masalah, kita dapat memanfaatkan proses pemecahan masalah yang lebih baik dan lebih efisien.
- model masalah memungkinkan kita mendeskripsikan data yang akan dimanipulasi/diproses oleh algoritma yang dibuat.
- Tipe data abstrak atau Abstract Data Type (ADT) adalah deskripsi logis tentang bagaimana kita melihat data dan operasi yang dijalankan tanpa melihat bagaimana data dan operasi tersebut di implementasikan. Kita hanya melihat dengan apa yang direpresentasikan oleh data dan bukan dengan bagaimana data itu nantiya akan dibuat.
- Dengan ADT, kita dapat membuat enkapsulasi di sekitar data. Berikut ilustrasi ADT:
- 

- Ide ADT adalah merangkum detail implementasi, menyembunyikan detail tersebut dari interface pengguna. Istilah ini disebut Information Hiding / Penyembunyian Informasi.

- Implementasi dari ADT biasa disebut dengan Struktur Data.
- Struktur Data mengharuskan kita menyediakan perspektif fisikal dari data menggunakan beberapa kumpulan konstruksi pemrograman dan tipe data primitif.

Mengapa belajar Algoritma?

- Kita belajar sebuah pemecahan masalah dengan melihat orang lain memecahkan masalah dan/atau dengan memecahkan masalah sendiri.
- Melihat perbedaan teknik-teknik pemecahan masalah dan bagaimana desain algoritmanya, dapat membantu kita dalam tantangan masalah yang kita hadapi.
- Dengan mempertimbangkan sejumlah algoritma yang berbeda, kita dapat memulai untuk mengembangkan pola penyelesaian sehingga ketika kita berjumpa dengan permasalahan yang sama, kita dapat menyelesaikannya dengan lebih baik.
- Algoritma seringkali sangat berbeda satu sama lain. Sebagai contoh penggunaan modul sqrt sebelumnya dalam mencari akar kuadrat. Sangat memungkinkan masih ada cara-cara lain dalam mengimplementasi fungsi dari akar kuadrat.
- Sebuah algoritma mungkin menggunakan banyak resource/sumber daya dibandingkan algoritma yang lain (resource/sumber daya yang dimaksud disini contohnya seperti memori RAM, proses CPU, kapasitas penyimpanan, dll). Algoritma yang lain mungkin membutuhkan waktu 10 kali lipat dalam menemukan hasil dibandingkan yang lain.
- Saat kita mempelajari algoritma, kita dapat mempelajari teknik analisis yang memungkin kita membandingkan dan membedakan solusi dari algoritma-algoritma hanya berdasarkan karakteristik algoritma itu sendiri, bukan karakteristik program atau komputer yang digunakan untuk mengimplementasikannya.
- Dalam sebuah skenario terburuk, kita mungkin menemukan sebuah masalah yang sama sekali tidak ada algoritma penyelesaiannya. Sangat penting bagi kita untuk dapat membedakan permasalahan yang mempunyai algoritma penyelesaian , permasalahan yang tidak memiliki algoritma penyelesaian dan permasalahan yang dimana algoritma penyelesaiannya ada tetapi membutuhkan banyak waktu atau resource.
- Kita perlu mengetahui dan memahami evolusi/perubahan teknik dalam menyelesaikan permasalahan.
- Akan ada banyak sekali algoritma dalam menyelesaikan permasalahan. Pencarian solusi dan memutuskan yang mana algoritma terbaik adalah tugas yang akan kita lakukan berulang kali.

Review Pemrogaman Dasar Python

- silahkan membaca dan mengerjakan modul praktikum satu untuk mengingat dan mengetahui apa saja dasar-dasar pemrograman yang dibutuhkan dalam mempelajari mata kuliah ini.