

DASAR PEMROGRAMAN

Komputer digunakan sebagai alat bantu penyelesaian suatu persoalan. Masalahnya, problematika itu tidak dapat "disodorkan" begitu saja ke depan komputer, dan komputer akan memberikan jawabannya. Ada "jarak" antara persoalan dengan komputer. Strategi pemecahan masalah masih harus ditanamkan ke komputer oleh manusia dalam bentuk program. Untuk menghasilkan suatu program, seseorang dapat memakai berbagai pendekatan yang dalam bidang pemrograman disebut sebagai paradigma. Namun demikian, semua pemrograman mempunyai dasar yang sama. Karena itu pada kuliah Dasar pemrograman, diajarkan semua komponen yang perlu dalam pemrograman apapun, walaupun implementasi dan cara konstruksinya akan sangat tergantung kepada paradigma dan bahasa pemrogramannya.

Paradigma adalah sudut pandang atau "sudut serang" tertentu yang diprioritaskan, terhadap kelompok problema, realitas, keadaan, dan sebagainya. Paradigma membatasi dan mengkondisikan jalan berpikir kita, mengarahkan kita terhadap beberapa atribut dan membuat kita mengabaikan atribut yang lain. Karena itu, jelas bahwa sebuah paradigma hanya memberikan pandangan yang terbatas terhadap sebuah realitas, karena itu fanatisme terhadap sebuah paradigma, mempersempit wawasan dan kadang berbahaya.

Dalam pemrograman pun ada beberapa paradigma, masing-masing mempunyai prioritas strategi analisa yang khusus untuk memecahkan persoalan, masing-masing menggiring kita ke suatu pendekatan khusus dari problematika keseluruhan. Beberapa jenis persoalan dapat dipecahkan dengan baik dengan menggunakan sebuah paradigma, sedangkan yang lain tidak cocok. Mengharuskan seseorang memecahkan persoalan hanya dengan melalui sebuah paradigma, berarti membatasi strateginya dalam pemrograman. Satu paradigma tidak akan cocok untuk semua kelas persoalan.

"Ilmu" pemrograman berkembang, menggantikan "seni" memprogram atau memprogram secara coba-coba (*"trial and error"*). Program harus dihasilkan dari proses pemahaman permasalahan, analisis, sintesis dan dituangkan menjadi kode dalam bahasa komputer secara sistematis dan metodologis.

Karena terbatasnya waktu, tentu saja tidak mungkin semua paradigma disatukan dalam sebuah mata kuliah. Pada matakuliah Dasar Pemrograman mahasiswa belajar memprogram dengan paradigma fungsional.

Berikut ini setiap paradigma akan dibahas secara ringkas. Sebenarnya seseorang dapat belajar memprogram dalam salah satu paradigma dan kemudian beranjak ke paradigma lain dengan memakai paradigma yang dipelajarinya sebagai "meta paradigma". Karena di Jurusan Informatika ITB paradigma pemrograman yang pertama kali diajarkan dalam kurikulum adalah paradigma prosedural, maka beberapa konsep yang sama pada paradigma prosedural (misalnya konsep objek dan type) akan dipakai sebagai acuan berpikir dalam konteks fungsional. Namun tanpa mata kuliah pemrograman prosedural sebagai prasyarat, buku ini dapat dipakai sebagai buku pegangan untuk matakuliah fungsional.

Paradigma tertua dan relatif banyak dipakai saat ini adalah paradigma prosedural. Bahasa fungsional banyak pula diwarnai fasilitas prosedural karena memang ternyata untuk beberapa hal kita belum bisa dari paradigma prosedural akibat mesin pengeksekusi (mesin Von Neumann) yang masih berlandaskan paradigma ini. Dengan alasan ini maka salah satu bagian dari buku ini akan mencakup aspek prosedural dalam paradigma pemrograman fungsional.

Beberapa paradigma dalam pemrograman :

1. Paradigma pemrograman **Prosedural atau imperatif**

Paradigma ini didasari oleh konsep mesin Von Neumann (stored program concept): sekelompok tempat penyimpanan (**memori**), yang dibedakan menjadi memori **instruksi** dan memori **data**; masing-masing dapat diberi nama dan harga. Instruksi akan dieksekusi satu per satu secara sekuensial oleh sebuah pemroses tunggal. Beberapa instruksi menentukan instruksi berikutnya yang akan dieksekusi (percabangan kondisional). Data diperiksa dan dimodifikasi secara sekuensial pula. Program dalam paradigma ini didasari pada **strukturasi informasi** di dalam memori dan **manipulasi** dari informasi yang disimpan tersebut. Kata kunci yang sering didengarkan dalam pendekatan ini adalah :

Algoritma + Struktur Data = Program.

Pemrograman dengan paradigma ini sangat tidak "manusiawi" dan tidak "alamiah", karena harus berpikir dalam batasan mesin (komputer), bahkan kadang-kadang batasan ini lebih mengikat daripada batasan problematikanya sendiri.

Keuntungan pemrograman dengan paradigma ini adalah efisiensi eksekusi, karena dekat dengan mesin.

2. Paradigma pemrograman **Fungsional**

Paradigma ini didasari oleh konsep pemetaan dan **fungsi** pada matematika. Fungsi dapat berbentuk sebagai fungsi "primitif", atau komposisi dari fungsi-fungsi lain yang telah terdefinisi. Pemrogram mengasumsikan bahwa ada fungsi-fungsi dasar yang dapat dilakukan. Penyelesaian masalah didasari atas aplikasi dari fungsi-fungsi tersebut. Jadi dasar pemecahan persoalan adalah **transformasional**. Semua kelakuan program adalah suatu rantai transformasi dari sebuah keadaan awal menuju ke suatu rantai keadaan akhir, yang mungkin melalui keadaan antara, melalui aplikasi fungsi.

Paradigma fungsional tidak lagi mempernasalahkan memorisasi dan struktur data, tidak ada pemilahan antara data dan program, tidak ada lagi pengertian tentang "variabel". Pemrogram tidak perlu lagi mengetahui bagaimana mesin mengeksekusi atau bagaimana informasi disimpan dalam memori, setiap fungsi adalah "kotak hitam", yang menjadi perhatiannya hanya keadaan awal dan akhir. Dengan merakit kotak hitam ini, pemrogram akan menghasilkan program besar.

Berlainan sekali dengan paradigma prosedural, program fungsional harus diolah lebih dari program prosedural (oleh pemroses bahasanya), karena itu salah satu keberatan adalah kinerja dan efisiensinya.

3. Paradigma pemrograman **Deklaratif, predikatif atau logik**

Paradigma ini didasari oleh pendefinisian **relasi** antar individu yang dinyatakan sebagai **predikat**. Sebuah program logik adalah kumpulan aksioma (fakta dan aturan deduksi).

Pemrogram menguraikan sekumpulan fakta dan aturan-aturan (*inference rules*). Ketika program dieksekusi, pemakai mengajukan pertanyaan (*Query*), dan program akan menjawab apakah pernyataan itu dapat dideduksi dari aturan dan fakta yang ada. Program akan memakai aturan deduksi dan mencocokkan pertanyaan dengan fakta-fakta yang ada untuk menjawab pertanyaan.

4. Paradigma berorientasi Objek (**Object Oriented**)

Paradigma ini didasari oleh **Kelas dan objek**. Objek adalah instansiasi dari kelas. Objek mempunyai atribut (kumpulan sifat), dan mempunyai kelakuan (kumpulan

reaksi, metoda). Objek yang satu dapat berkomunikasi dengan objek yang lain lewat "pesan", dengan tetap terjaga integritasnya. Kelas mempunyai hirarki, anggota dari sebuah kelas juga mendapatkan turunan atribut dari kelas di atasnya. Paradigma ini menawarkan konsep modularitas, penggunaan kembali, kemudahan modifikasi.

Dalam paradigma ini, masih terkandung dari paradigma imperatif, karena mengkonstruksi program dari objek dan kelas adalah tidak berbeda dengan mengkonstruksi program dari struktur data dan algoritma, dengan cara enkapsulasi menjadi kelas. Kedekatan antara paradigma ini dengan paradigma lain dapat dilihat dari bahasa-bahasa bukan berorientasi obyek murni, yaitu bahasa prosedural atau fungsional yang ditambahi dengan ciri orientasi objek. Salah satu aspek penting dalam pemrograman berorientasi objek adalah aspek konkuren. Karena pada saat runtime akan lahir banyak (lebih dari satu objek), dengan kelakuan masing-masing, maka aspek konkuren tidak dapat dipisahkan dari pemrograman objek.

4. Paradigma Konkuren

Paradigma ini didasari oleh kenyataan bahwa dalam keadaan nyata, sebuah sistem komputer harus menangani beberapa program (task) yang harus dieksekusi bersama dalam sebuah lingkungan (baik mono atau multi prosesor). Pada pemrograman konkuren, kita tidak lagi berpikir sekuensial, melainkan harus menangani komunikasi dan sinkronisasi antar task. Pada jaman sekarang, aspek konkuren semakin memegang peranan penting.

BAHASA PEMROGRAMAN

Berbicara mengenai bahasa pemrograman, ada banyak sekali bahasa pemrograman, mulai dari bahasa tingkat rendah (bahasa mesin dalam biner), bahasa assembler (dalam kode mnemonik), bahasa tingkat tinggi, sampai bahasa generasi ke empat (4GL).

Bahasa Pemrograman berkembang dengan cepat sejak tahun enam puluhan, seringkali dianalogikan dengan menara Babel yang berakibat manusia menjadi tidak lagi saling mengerti bahasa masing-masing. Untuk setiap paradigma, tersedia bahasa pemrograman yang mempermudah implementasi rancangan penyelesaian masalahnya. Contoh bahasa-bahasa pemrograman yang ada :

1. Prosedural : Algol, Pascal, Fortran, Basic, Cobol, C ...
2. Fungsional : LOGO, APL, LISP
3. Deklaratif : Prolog
4. Object oriented murni: Smalltalk, Eiffel.

Pemroses bahasa Pascal dan C versi terbaru dilengkapi dengan fasilitas orientasi objek, misalnya Turbo Pascal (mulai versi 5.5) pada PC dan C++.

BELAJAR MEMPROGRAM TIDAK SAMA DENGAN BELAJAR BAHASA PEMROGRAMAN

Belajar memprogram adalah belajar tentang strategi pemecahan masalah, metodologi dan sistematika pemecahan masalah tersebut kemudian menuangkannya dalam suatu notasi yang disepakati bersama. Beberapa masalah akan cocok kalau diselesaikan dengan suatu paradigma tertentu. Karena itu, pengetahuan tentang kelas persoalan penting adanya.

Pada hakekatnya, penggunaan komputer untuk memecahkan persoalan adalah untuk tidak mengulang-ulang kembali hal yang sama. Strategi pengenalan masalah melalui dekomposisi, pemakaian kembali modul yang ada, sintesa, selalu dipakai untuk semua

pendekatan, dan seharusnya mendasari semua pengajaran pemrograman. Karena itu perlu diajarkan metodologi, terutama karena sebagian besar pemrogram pada akhirnya memakai program yang sudah pernah ditulis orang lain dibandingkan dengan bongkar pasang program yang sudah ada.

Belajar memprogram lebih bersifat pemahaman persoalan, analisis, sintesis.

Belajar bahasa pemrograman adalah belajar memakai suatu bahasa, aturan sintaks (tatabahasa), setiap instruksi yang ada dan tata cara pengoperasian kompilator atau interpreter bahasa yang bersangkutan pada mesin tertentu. Lebih lanjut, belajar bahasa pemrograman adalah belajar untuk memanfaatkan instruksi-instruksi dan kiat yang dapat dipakai secara spesifik hanya pada bahasa itu. Belajar memprogram lebih bersifat keterampilan dari pada analisis dan sintesa.

Belajar memprogram dan belajar bahasa pemrograman mempunyai tingkatan dan kesulitan yang berbeda-beda. Mahasiswa seringkali dihadapkan pada kedua kesulitan itu sekaligus. Pemecahan persoalan dengan paradigma yang sama akan menghasilkan solusi yang "sejenis". Beberapa bahasa dapat termasuk dalam sebuah paradigma sama, pemecahan persoalan dalam satu paradigma dapat diterjemahkan ke dalam bahasa-bahasa yang berbeda. Untuk itulah diperlukan adanya suatu perjanjian, notasi yang disepakati supaya masalah itu dapat dengan mudah diterjemahkan ke dalam salah satu bahasa yang masih ada dalam lingkup paradigma yang sama. Pada pengajaran pemrograman fungsional ini dikembangkan suatu notasi yang disebut notasi fungsional, yang akan dipakai sebagai "bahasa ekspresi dan komunikasi" antara persoalan dengan pemrogram,

Pemilihan paradigma dan strategi pemecahan persoalan lebih penting daripada pemilihan bahasanya sendiri, walaupun pada akhirnya memang harus diputuskan bahasa yang dipakai. Bahasa pemrograman fungsional yang paling banyak dipakai saat ini adalah bahasa yang berinduk pada LISP. Karena itu pada bagian akhir buku ini diberikan contoh terjemahan dari notasi fungsional menjadi program dalam bahasa LISP atau salah satu dialeknya.

Proses memprogram adalah proses yang memerlukan kepakaran, proses koding lebih merupakan proses semi otomatis dengan aturan pengkodean. Proses memprogram memang berakhir secara konkrit dalam bentuk program yang ditulis dan dieksekusi dalam bahasa target. Karena itu memaksa mahasiswa hanya bekerja atas kertas, menganalisis dan membuat spesifikasi tanpa pernah me-run program adalah tidak benar. Sebaliknya, hanya mencetak pemrogram yang langsung "memainkan keyboard", mengetik program dan mengeksekusi tanpa analisis dan spesifikasi yang dapat dipertanggung jawabkan juga tidak benar (terutama untuk program skala besar dan harus dikerjakan banyak orang).

Produk yang dihasilkan oleh seorang pemrogram adalah program dengan rancangan yang baik (metodologis, sistematis), yang dapat dieksekusi oleh mesin, berfungsi dengan benar, sanggup melayani segala kemungkinan masukan, dan didukung dengan adanya dokumentasi. Pengajaran pemrograman titik beratnya adalah membentuk seorang perancang "*designer*" program, sedangkan pengajaran bahasa pemrograman titik beratnya adalah membentuk seorang "*coder*" (juru kode).

Pada prakteknya, suatu rancangan harus dapat dikode untuk dieksekusi dengan mesin. Karena itu, belajar pemrograman dan belajar bahasa pemrograman saling komplementer, tidak mungkin dipisahkan satu sama lain.

Metoda terbaik untuk belajar apapun adalah melalui contoh. Seorang yang sedang belajar harus belajar melalui contoh nyata. Berkat contoh nyata itu dia melihat, mengalami dan melakukan pula. Metoda pengajaran yang dipakai pada perkuliahan pemrograman fungsional ini adalah pengajaran melalui contoh tipikal. Contoh tipikal adalah contoh program yang merupakan "pola solusi" dari kelas-kelas persoalan yang dapat diselesaikan dengan paradigma pemrograman fungsional.

TUJUAN KULIAH PEMROGRAMAN FUNGSIONAL

Tujuan utama dari matakuliah ini adalah membekali mahasiswa cara berpikir dan pemecahan persoalan dalam paradigma pemrograman fungsional. Mahasiswa harus mampu membuat penyelesaian masalah pemrograman fungsional tanpa tergantung pada bahasa pemrograman apapun, dan kemudian ia mampu untuk mengeksekusi programnya dengan salah satu bahasa pemrograman fungsional LISP. Mahasiswa akan memakai bahasa pemrograman tersebut sebagai alat untuk mengeksekusi program dengan mesin yang tersedia.

Secara lebih spesifik, mahasiswa diharapkan mampu untuk :

1. Memecahkan masalah dengan paradigma fungsional tanpa tergantung pada bahasa pemrograman apapun.
2. Menulis program berdasarkan pemrograman fungsional menjadi salah satu bahasa pemrograman yang menjadi target (misalnya LISP).

Kenapa kuliah dasar pemrograman diisi dengan pemrograman fungsional ?

1. Pada hakekatnya, program dibuat untuk melaksanakan suatu fungsi tertentu sesuai dengan kebutuhan pemakai. Jika sebuah fungsi dapat kita umpamakan sebuah tombol khusus pada “mesin” komputer, maka mengaktifkan program untuk pemecahan persoalan idealnya adalah hanya dengan menekan sebuah tombol saja.
2. Pada pemrograman fungsional, kita dihadapkan kepada cara berpikir melalui fungsi (apa yang akan direalisasikan) tanpa mempedulikan bagaimana memori komputer dialokasi, diorganisasi, diimplementasi.
3. Pada paradigma fungsional, kita juga “terbebas” dari persoalan eksekusi program, karena eksekusi program hanyalah aplikasi terhadap sebuah fungsi.

IKHTISAR BUKU

Buku ini terdiri dari tiga bagian utama. Bagian pertama difokuskan kepada pemrograman fungsional dengan kasus-kasus tipikal serta primitif yang berguna untuk memprogram dalam skala lebih besar. Dapat dikatakan bahwa bagian pertama ini berisi nukleus dan atom pembentuk aplikasi nyata dalam program fungsional., sedangkan buku kedua pada penulisan program fungsional dalam bahasa LISP. Bagian ketiga berisi studi kasus terhadap dua kelas persoalan: kelas persoalan pertama adalah kelas persoalan yang cocok untuk diselesaikan dengan paradigma fungsional. Kelas persoalan kedua adalah kelas persoalan yang sebenarnya bersifat prosedural, namun dicoba untuk diselesaikan secara fungsional. Kelas persoalan kedua ini sekaligus merupakan studi banding antara paradigma pemrograman fungsional versus prosedural, yaitu paradigma yang masih banyak dipakai dan mendominasi aplikasi saat ini

Bagian pertama buku ini secara garis besar akan berisi : (akan diuraikan lebih detail)

1. Konsep dasar pemrograman fungsional
2. Ekspresi fungsional yang paling mendasar: ekspresi aritmatika
3. Ekspresi kondisional
4. Aspek eksekusi program fungsional (aplikasi, evaluasi, scope, binding,...) secara konseptual
5. Pengolahan type bentukan (objek) dalam konteks fungsional. Bagian ini akan diajarkan melalui contoh kasus pemrosesan type tanggal (date), pecahan, dan titik. Berdasarkan contoh tipikal pada tiga type yang sering dipakai dalam informatika

tersebut, diharapkan mahasiswa mampu memperluas aplikasinya menjadi type bentukan yang lain.

6. Pengertian dari **koleksi objek**, yang mendasari organisasi dan struktur data.
7. Pengolahan koleksi objek dalam bentuk **tabel**. Pada abagian ini akan diajarkan bagaimana realisasi tabel yang secara konsep banyak dipakai dalam informatika dalam konsep fungsional, yaitu sebagai tabulasi eksplisit atau berdasarkan fungsi
8. **Analisa rekurens**. Analisa rekurens adalah salah satu landasan berpikir dalam pemrograman fungsional yang sangat penting. Bagian ini menjadi dasar bagi bagian-bagian selanjutnya, dan kira-kira akan memakan porsi hampir dari separuh jam kuliah yang dialokasikan.
9. **Rekursifitas bilangan integer** dan kasus pemakaiannya.
10. **Type data rekursif dasar** : list dan kasus-kasus pemakaiannya. List adalah type data dasar yang banyak dipakai dalam informatika, dan khususnya menjadi struktur data dasar dalam bahasa LISP, sehingga hampir semua persoalan yang diselesaikan dalam LISP akan didasari oleh struktur data list ini.
11. **Type pohon** dan contoh kasusnya. Beberapa representasi pohon, khususnya pohon biner akan dibahas pada bagian ini yaitu prefix, infix dan postfix. Kasus yang dibahas adalah evaluasi ekspresi yang representasi internalnya adalah pohon.
12. **Aspek prosedural** dalam pemrograman fungsional. Beberapa persoalan tidak mungkin diselesaikan secara murni fungsional. Karena itu dibutuhkan aspek prosedural. Karena buku ini ditujukan untuk mahasiswa Teknik Informatika ITB, Bagian ini mengacu kepada notasi algoritmik yang diajarkan pada perkuliahan Pemrograman Prosedural yang memang diajarkan sebelumnya. Jika buku ini ingin dipakai dalam konteks lain, notasi algoritmik yang dipakai dapat digantikan dengan notasi dalam bahasa Pascal atau bahasa lain yang sudah dikenal dan dipakai mahasiswa sebelumnya. Jika pemrograman fungsional diajarkan sebagai paradigma pertama, maka bagian ini dapat dihilangkan tanpa mengubah esensi dari pemrograman fungsional.

Bagian kedua buku ini secara garis besar akan berisi :

1. Bahasa pemrograman fungsional
2. Bahasa LISP
3. Aplikasi fungsional dan evaluasi fungsi

Bagian ketiga akan memuat studi kasus mengenai :

1. Ekspresi aritmatika
2. Studi analisa rekurens versus iteratif
3. Studi kasus list dengan elemen list

DAFTAR PUSTAKA :

1. Abelson H., Sussman G.J., and Sussman J. : "Structure and Interpretation of Computer Programs", MIT Press, McGraw-Hill, 4th printing, 1986.
2. Bird R. and Wadler P. : "An Introduction to Functional Programming", Prentice-Hall International, 1988.
3. Scholl P.C., Fauvet M.C., Lagnier F., Maraninchi F. : "Cours d'informatique: langage et programmation", Masson (Paris), 1993.
4. Friedman D. and Felleisen M. : "The Little LISPer", Pergamon Pub.Co., 3rd edition, 1989.
5. Steele G.L. : "Common LISP", 1984.
6. Winston P.H. and Horn B.K.P. : "LISP", Addison-Wesley, 3rd edition, 1989.