Fakultas Informatika
School of Computing
Telkom University

CCH1A4 / Dasar Algoritma & Pemrogramanan

Yuliant Sibaroni M.T, Abdurahman Baizal M.Kom

KK Modeling and Computational Experiment

Pengantar Algoritma

- Tentang Mata Kuliah
- Kasus-Kasus Pemrograman
- Paradigma Pemrograman
- Sejarah Bahasa Pemrograman
- Notasi Algoritmik
- Tipe Dasar dan Operasinya

Tentang Mata Kuliah

Hubungan dengan Mata Kuliah lain

- Mata kuliah dasar algoritma dan pemrograman mengajarkan konsep-konsep pemrograman secara prosedural/terstruktur
- Mata kuliah ini memiliki keterkaitan dengan MK pada tingkat atas
- Penguasaan yang baik terhadap MK ini diharapkan akan berguna sebagai sebagai bekal bagi mahasiswa dalam mengambil mata kuliah pemrograman pada tingkat yang lebih atas seperti : MK
  - **Pemrograman Berorientasi Objek,**
  - **Desain dan Analisis Algoritma**
  - **Pemrograman Web**
  - **Komputasi Paralel**
  - **Sistem Terdistribusi**
  - **Visualisasi Saintifik dan Komputasi Kinerja Tinggi**

Tentang Mata Kuliah

Titik Berat Pembelajaran

- Mata kuliah ini mengajarkan konsep pemrograman kepada mahasiswa, bukan Bahasa Pemrograman tertentu
- Notasi yang digunakan dalam kuliah ini adalah sebagian besar adalah notasi algoritmik yang mengacu kepada referensi [1].
- Titik berat: mahasiswa diajarkan untuk menyelesaikan suatu masalah dengan langkah-langkah yang sistematis menggunakan notasi standar (notasi algoritmik) yang telah ditetapkan

Kasus-kasus Pemrograman

Overview

- Kasus pemrograman menggunakan perintah-perintah Dasar
- Kasus pemrograman dengan Pencabangan
- Kasus pemrograman dengan Perulangan

Dalam sub-bab ini, akan diberikan kasus-kasus yang terjadi dalam dunia nyata beserta solusinya. Solusi yang diberikan berupa sederetan instruksi / langkah-langkah untuk menyelesaikan kasus tersebut. Beberapa komponen algoritma yang terkait dengan kasus tersebut kemudian disebutkan.

Kasus-kasus Pemrograman

Kasus-Kasus dengan Perintah- Perintah Dasar: assignment, I/O

Contoh 1.1

Dimiliki 2 gelas (A dan B) dengan bentuk dan ukuran yang sama, berisi teh dan kopi

Bagaimana cara mempertukarkan isi 2 gelas tersebut ?

Solusi

- Diperlukan 1 gelas kosong berukuran sama: C
- Tuangkan isi gelas A ke C (C sekarang berisi teh)
- Tuangkan isi gelas B ke A (A sekarang berisi kopi)
- Tuangkan isi gelas C ke B (B sekarang berisi teh)

Kasus-kasus Pemrograman

Kasus-Kasus dengan Perintah- Perintah Dasar: assignment, I/O

Contoh 1.2

Dimiliki Jerigen 3 liter dan 5 liter

Bagaimana cara mendapatkan air tepat 4 liter dari sumber air yang tak terhingga hanya dengan menggunakan kedua jerigen tersebut ?

Solusi

a. Isi jerigen 5 liter sampai penuh
b. Tuangkan air dari jerigen 5 liter ke jerigen 3 liter sampai penuh
   Saat ini : jerigen 5 liter berisi 2 liter air, jerigen 3 liter berisi 3 liter air
c. Buang semua air pada jerigen 3 liter
d. Tuangkan semua air dalam jerigen 5 liter ke jerigen 3 liter
   Saat ini : jerigen 5 liter kosong, jerigen 3 liter berisi 2 liter air
e. Isi penuh jerigen 5 liter
f. isi jerigen 3 liter sampai penuh menggunakan jerigen 5 liter
g. Sekarang jerigen 5 liter berisir tepat 4 liter air

Kasus-kasus Pemrograman

Kasus Pencabangan

Contoh 1.3

Menentukan mahasiswa yang tertinggi dari dua orang mahasiswa (A dan B) tanpa menggunakan pengukur

Solusi

a. Kedua mahasiswa diminta berdiri pada lantai yang datar

b. Dilihat posisi kepala kedua mahasiswa, jika A lebih tinggi maka A adalah mahasiswa tertinggi, jika tidak : B yang tertinggi

Kasus-kasus Pemrograman

Kasus Perulangan

Contoh 1.4

Mencari kupon undian yang bernama Budi dari 1000 kupon.

Solusi

a. Ambil sebuah kupon

b. Baca nama yang tertulis, apakah ‘Budi’ ataukah tidak

c. Ulangi langkah a dan b sampai ketemu kupon dengan nama ‘Budi’ atau kupon telah habis (sampai kupon ke-1000)

Kasus-kasus Pemrograman

Kasus Pencabangan dan Perulangan

Contoh 1.5

Mencari rute terpendek dari kota A ke kota G, dengan asumsi : kita tidak memiliki peta, hanya tahu jarak dari satu kota ke kota lainnya?

Solusi

a. Mulai dari kota A

b. Coba buat rute sembarang ke kota lainnya sampai berakhir ke G.

   jika tidak berhasil, ulangi langkah a. Jika berhasil hitung total jaraknya!

c. Ulangi langkah a dan b, sampai tidak ada rute yang sama terulang

d. Pilih rute dengan jarak yang terkecil

Paradigma Pemrograman

Definisi Paradigma pemrograman [Norman , Kurt]

Sebuah pola yang berfungsi sebagai panduan dalam pemrograman komputer

Paradigma Pemrograman dapat diKlasifikasi-kan menjadi [Norman , Kurt] **:**

A. Paradigma Pemprograman Utama

- **Prosedural/Imperative**: instruksi akan dieksekusi satu per satu secara sekuensial
- **Fungsional**: konsep pemetaan dan fungsi pada matematika
- **Logik**: pendefinisian relasi antar individu yang dinyatakan sebagai predikat
- **Object-Oriented**: didasari oleh objek yang mempunyai sifat-sifat dan kelakuan

B. Paradigma Pemrograman Lainnya yang Mungkin

- **Visual**: membuat program dengan memanipulasi elemen-elemen secara grafik daripada secara teks.
- **Parallel**: banyak perhitungan yang dilakukan secara bersamaan.
- **Constraint Based**: relasi antar variabel dinyatakan dalam bentuk konstrain.

Beberapa ahli berpendapat, klasifikasi paradigma seperti ini sulit diimplementasikan. Banyak bahasa pemrograman yang tidak bisa digolongkan secara tepat dalam paradigma tertentu tetapi merupakan **gabungan** dari beberapa paradigma (**Multiple Paradigm**).

Paradigma Pemrograman

Overview Paradigma Imperative

- Disebut juga paradigma **prosedural**
- Kata kunci [Norman , Kurt]]: First **do this** and next **do that**
- Dalam paradigma ini, penyelesaian kasus dilakukan dengan menyusun instruksi-intruksi pemrograman secara terurut, dalam masalah nyata hal ini dapat dianalogikan seperti prosedur-prosedur administrasi yang ada di suatu perusahaan, prosedur pemasangan alat, prosedur penyetelan dll
- Bahasa pemrograman yang menggunakan : Pascal, C

Contoh 1.6

Pencarian Nilai Terbesar antara a,b, dan c

- Langkah-langkahnya:
  - Asumsikan: nilai terbesar (Maks) adalah a
  - Bandingkan Maks dengan b, jika b lebih besar dari Maks, maka nilai Maks = b
  - Bandingkan Maks dengan c, jika c lebih besar dari Maks, maka nilai Maks = c
  - Tampilkan Maks

Paradigma Pemrograman

Overview Paradigma Fungsional

- Idenya berasal dari konsep matematis tentang fungsi, kata kunci [Norman , Kurt]]:

  Evaluate an expression and use the resulting value for something

- Dalam paradigma ini , model komputasinya adalah sebagai evaluasi ekspresi . Pemrograman fungsional memerlukan fungsi adalah sebagai **first-class,** yaitu fungsi diperlakukan sebagai nilai yang dapat dikirim sebagai argumen ke fungsi lainnya atau hasil dari suatu fungsi [5].

- Contoh bahasa pemrograman : Ocaml, Erlang, Haskell

Contoh 1.7

Perhitungan Fibonacci dengan Erlang [6]

```
-module(fibonacci).
-export([start/1]).

%% Fibonacci numbers in Erlang
start(N) -> do_fib(0,1,N).

do_fib(_,B,1) -> B;
do_fib(A,B,N) -> do_fib(B,A+B,N-1).
```

Paradigma Pemrograman

Overview Paradigma Logik

- Kata kunci [Norman , Kurt]]:

  Answer a question via search for a solution

- Paradigma ini sangat cocok bila diterapkan dalam masalah yang berhubungan dengan ekstraksi pengetahuan dari fakta-fakta dasar dan relasi.
- Bahasa Pemrograman : Prolog

Contoh 1.8

Pendefinisian Hubungan Keluarga dan Pemanfaatannya

- Langkah-langkahnya:
  - Didefinisikan relasi Ayah(A,B) { A Ayah B}
  - Didefinisikan relasi Ayah(A,C)
  - Didefinisikan relasi Ayah(D,A)
  - Didefinisikan relasi Kakek(X,Z) ← Ayah(X,Y) AND Ayah(Y,Z)
  - Dapat disimpulkan bahwa : D kakek B dan C

Paradigma Pemrograman

Overview Paradigma Object-Oriented

- Kata kunci [Norman , Kurt]]:

  Send messages between objects to simulate the temporal evolution of a set of real world phenomena

- Paradigma ini saat ini cukup luas digunakan dalam dunia pemrograman terutama dalam pemrograman yang menggunakan GUI (Graphical Unit Interface) didalamnya. Dalam paradigma ini, sebuah program komputer dipandang sebagai objek – objek yang saling berhubungan. Proses pembuatan program diawali dengan pendefinisian semua objek yang terkait beserta operator-operatornya.

- Bahasa Pemrograman : Java, C++

Contoh 1.9

Pembuatan Kalkulator Sederhana

- Langkah-langkahnya:
  - Didefinisikan objek **KotakInput** , dg operator utama : **DisplayKotak**, **ClearKotak, GetInput**
  - Didefinisikan objek **Kali**, dengan operator utama: **HitungKali**
  - Didefinisikan objek **Jumlah**, dengan operator utama: **HitungJumlah**
  - Didefinisikan objek **Bagi**, dengan operator utama: **HitungBagi**

Sejarah Bahasa Pemrograman

Overview

Bahasa pemrograman sudah berkembang cukup pesat sejak komputer pertama dibuat. Banyak sekali bahasa pemrograman yang telah dibuat sampai saat ini, biasanya disesuaikan dengan komputer yang digunakan pada saat itu.

Berikut adalah perkembangan bahasa pemrograman ditinjau dari sisi paradigma dan fokus/pembaharuan yang dilakukan pada periode waktu tersebut [Denis Sureau]

- **Tahun 40-an:**
  Bahasa Pertama, contoh : **ENIAC Coding System, ARC Assembly**
- **Tahun 50-an :**
  Pembuatan Bahasa Tingkat Tinggi (yang lebih dekat ke bahasa manusia).
  Contoh : **Autocode** , **IPL** (Information Processing Language)
- **Tahun 60-an:**
  Pengembangan ke arah bahasa dengan fungsi yang khusus
  Contoh : **LISP** (LISt Processing), **COBOL**(COmmon Business Oriented Language).

Sejarah Bahasa Pemrograman

Lanjutan Sejarah Bahasa Pemrograman

- **Tahun 70-an:**

  Pemrograman Terstruktur. Duel antara **Pascal** dan **C**. Dasar pengembangan PC sampai tahun 80s-an.

- **Tahun 80-an:**

  Eksperimen kearah paradigma lain termasuk objects.

  Contoh : **C++**, **Object Pascal**

- **Tahun 90-an dan 2000-an:**

  Generalisasi object-oriented programming dengan didukung kinerja microcomputers yang semakin cepat.

  Contoh : **Java**, **Python, Visual Basic, Borland Delphi**

  Internet Programming (dan inovasi lainnya).

  Contoh : **PHP**, **JavaScript, C#**

- **Tahun 2010-an:**

  Concurrency dan asynchronicity.

  Contoh : **C++11, Go, Rust**

Notasi Algoritmik

Overview

- Didalam kuliah dasar algoritma pemrograman ini, kita tidak menggunakan bahasa pemrograman tertentu
- Notasi yang digunakan dalam kuliah ini: **Notasi Algoritmik**
- Tidak diperlukan software tertentu untuk menjalankan / me-running program yang dibuat dengan notasi algoritmik
- Kebenaran sebuah program dapat diperiksa berdasarkan ketepatan penggunaan notasi algoritmik dan kebenaran logika yang digunakan

Notasi Algoritmik

Komponen Utama

Ada 3 komponen utama penyusun sebuah program, yaitu:

- **Judul**

  Diawali dengan kata **Program** dilanjutkan dengan nama program, atau **FUNCTION** atau **PROCEDURE**

- **Kamus**

  Berisi pendefinisian Type, Konstanta, Variabel, fungsi, dan prosedur

- **Algoritma**

  Berisi instruksi-instruksi algoritmik untuk menyelesaikan suatu masalah

Selain 3 komponen utama tersebut, terdapat bagaian lain yang disebut **komentar.** Pembuatan komentar diawali dengan '{ 'dan diakhiri dengan '}'. Komentar berisi penjelasan dari sebuah program. Komentar sangat penting dalam pembuatan sebuah program (wajib ada), karena akan sangat membantu bagi kita dalam memahami alur logika sebuah program, terlebih untuk program yang cukup kompleks.

Notasi Algoritmik

Contoh 1.10

Membuat program kosong

Solusi

```
Program Kosong
Kamus

Algoritma

```

Notasi Algoritmik

Judul

Notasi : **Program** nama _Program

Pemberian nama program tidak memiliki aturan yang ketat, tetapi perlu diperhatikan hal-hal sebagai berikut:

- Diawali dengn alphabetical
- Tidak menggunakan nama yang sama dengan yang ada dalam notasi algoritmik
- Diusahakan nama program sesuai dengan apa yang dikerjakan program
- Kalau terdiri dari 2 kata atau lebih, jangan dipisah dengan spasi

Notasi Algoritmik

Kamus

Kamus digunakan untuk mendefinisikan segala sesuatu yang akan digunakan dalam program ini, meliputi :

- **Tipe bentukan**
  Didalam notasi algoritmik, ada 5 tipe dasar yang bisa digunakan (integer, real, character, boolean , string). Kita bisa mendefinisikan tipe lainnya dengan cara tertentu. Tipe ini disebut sebagai tipe bentukan

- **Konstanta**

  Jika didalam program, jika kita perlu menggunakan suatu nilai tertentu (terutama untuk nilai yang memiliki digit besar), kita bisa definisikan nilai terebut dalam bentuk konstanta

- **Variabel**

  Selama program dijalankan, biasanya kita perlu menyimpan nilai yang diolah program dalam sebuah wadah. Wadah yang dimaksud disebut sebagai variabel.

- **Fungsi** : spesifikasinya saja

- **Prosedur** : spesifikasinya saja

Notasi Algoritmik

Kamus: Tipe-tipe dasar

Ketika mendefinisikan sebuah variabel, maka juga harus diikuti dengan pendefinisian tipe datanya. Ada 5 tipe dasar yang bisa digunakan yaitu :

- **Real**

  Tipe ini digunakan bila datanya berupa bilangan desimal, seperti nilai rata-rata, data pengukuran dll

- **Integer**

  Tipe ini digunakan bila datanya berupa bilangan bulat, seperti jumlah orang, jumlah benda , mata uang dll

- **Character**

  Tipe ini digunakan bila datanya hanya terdiri dari satu karakter, seperti jenis kelamin (L/P) , jawaban multiple choice dll

- **Boolean**

  Tipe ini digunakan bila datanya bernilai true/false, misalnya dalam pencarian data, didefinisikan variabel found bertipe boolean

- **String** (kumpulan karakter)

  Tipe ini digunakan bila datanya terdiri dari banyak karakter , seperti nama,alamat dll

Notasi Algoritmik

Kamus

Contoh 1.11

Berikut cara mendefinsikan suatu kamus berisi tipe bentukan, konstanta dan variabel:

```
Kamus
  Type logika : boolean {mendefinsikan tipe bentukan dg nama logika}
  Constant  Phi : Real=3.14  {mendefinisikan konstanta bernama Phi}
  a,b : integer  {mendefinisikan variabel a dan b bertipe integer}
  c : String   { mendefinisikan c bertipe String }
  d : logika   { mendefinisikan  variabel d bertipe logika }
```

Catatan

Penulisan notasi algoritmik berupa tipe atau instruksi lainnya atau keyword menggunakan **underline,** contoh **integer** , **Type**, dll.

Notasi Algoritmik

Kamus: Operasi-operasi pada tipe-tipe dasar

Beberapa tipe dasar yang siap digunakan memiliki beberapa operasi (operator) yang wajib diketahui :

Tipe : **Real**

Contoh : 2.3 10.0 4.1 25.6

Operator	Detail	Keterangan	Ekspresi (Tipe)
Aritmatika	+ , x, /, -	Penjumlahan, perkalian, pembagian, pengurangan	a + b (real) a x b (real) a / b (real)
Perbandingan	=, <,>,<>	Sama dengan, lebih kecil, lebih besar, tidak sama	a = b (boolean) a <> b (boolean)
	<=, >=	Lebih kecil atau sama dengan, lebih besar atau sama dengan	a <= b (boolean) a >= b (boolean)

Notasi Algoritmik

Kamus: Operasi-operasi pada tipe-tipe dasar

Tipe : **Integer**

Contoh : 2 10 25 21

Operator yang digunakan sama dengan tipe Real, ditambah

Operator	Detail	Keterangan	Ekspresi (tipe)
Aritmatika	Div	Hasil Pembagian Bulat	10 Div 3 (hasilnya 3: integer)
	Mod	Sisa Pembagian Bulat	10 Mod 3 (hasilnya 1: integer)

Notasi Algoritmik

Kamus: Operasi-operasi pada tipe-tipe dasar

Tipe : **Boolean**

Contoh : True False (nilainya hanya ini saja)

Operator	Detail	Keterangan	Ekspresi (tipe)
Logika	AND	Bernilai **true** jika kedua pernyataan (A,B) **true**	A AND B (boolean)
	OR	Bernilai **false** jika kedua pernyataan (A,B) **false**	A OR B (boolean)
	NOT	Negasi : Nilainya berlawanan dengan nilai pernyataan	NOT A (boolean)
	EQ	Ekuivalen: Bernilai **true** bila kedua pernyataan bernilai sama	A EQ B (boolean)
	XOR	Exlusive Or: bernilai **true** bila hanya terdapat satu pernyataan yang bernilai **true**	A XOR B (boolean)

Notasi Algoritmik

Kamus: Operasi-operasi pada tipe-tipe dasar

Tipe : **Character**

Contoh : 'A' 'B' '3'

Operator	Detail	Keterangan	Ekspresi (tipe)
Perbandingan	= , <>	Hanya untuk membandingkan apakah dua karakter sama ataukah tidak	A <> B (boolean)

Tipe : **String** (array of character)

Contoh : 'Apa' 'Bantal' 'k3k1'

Operator yang digunakan seperti Character, ditambah

Operator	Detail	Keterangan	Ekspresi (tipe: boolean)
Konstruksi	&	Konkatenasi: Menggabungkan 2 string	A & B 'rumah' & 'TUA' (Hasil: rumahTUA)

Referensi

1. Inggriani Liem, Diktat Kuliah IF223 Algoritma Dan Pemrograman, Jurusan Teknik Informatika Bandung, 1999
2. Rinaldi Munir, Algoritma dan Pemrograman, edisi ke-3, penerbit Informatika 2004
3. Normark, Kurt. *Overview of the four main programming* paradigms. http://people.cs.aau.dk/ ~normark diakses tanggal 12 September 2013.
4. http://www.haskell.org/haskellwiki/Functional_programming diakses tanggal 14/09/2013
5. http://en.wikipedia.org/wiki/Functional_programming , diakses tanggal 14/09/2013
6. Denis Sureau, *History and Evolution of Programming Languages,* http://www.scriptol.com/programming/history.php diakses tanggal 17/09/2013

Fakultas Informatika
School of Computing
Telkom University

THANK YOU