Algoritma dan Pemrograman

Pertemuan Ke-2
Dasar-dasar Algoritma

Disusun Oleh :
Wilis Kaswidjanti, S.Si.,M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Pembangunan Nasional "Veteran"
Yogyakarta

Algoritma dan Pemrograman

Judul Materi : Dasar-Dasar Algoritma

Deskripsi Materi : Materi ini membahas dasar-dasar algoritma.

Tujuan Instruksional Khusus :

1. Menjelaskan struktur dasar algoritma
2. Menyusun algoritma untuk menyelesaikan masalah
3. Merumuskan algoritma dengan membuat flowchart dan psedo language
4. Mengenal *top-down design*

BAB II
DASAR-DASAR ALGORITMA

1. PENDAHULUAN

Bab ini memiliki kompetensi dasar untuk memahami dasar-dasar algoritma untuk diimplementasikan dalam menyelesaikan masalah. Sebelum mengimplementasikan suatu algoritma, terlebih dahulu memahami konsep pernyataan dan aksi, struktur dasar algoritma dan mengenal suatu strategi Perancangan Puncak Turun (*TopDown Design).*

2. PENYAJIAN

2.1. Pernyataan dan Aksi

Pada dasarnya, sebuah Algoritma merupakan deskripsi langkah-langkah pelaksana suatu proses. Sebuah proses dikerjakan oleh pemroses berdasarkan algoritma yang diberikan.

Setiap langkah penyelesaian dinyatakan dengan sebuah pernyataan (Statement). Sebuah pernyataan menggambarkan aksi (action) algoritmik yang dieksekusi, bila suatu aksi dieksekusi, maka sejumlah operasi yang bersesuaian dengan aksi itu dikerjakan oleh pemroses.

Contoh :

Misalkan didalam algoritma ada pernyataan :

Tulis “Hello, world”

Pernyataan tersebut menggambarkan aksi menuliskan “Hello, world” ke piranti keluaran (layer).

Efek dari aksi ini dapat diamati sebelum dan sesudah eksekusi. Maka setelah aksi tersebut dieksekusi, dilayar akan tertera tulisan

Hello, world

2. Struktur Dasar Algoritma

Sebuah algoritma dapat dibangun berdasarkan 3 buah struktur dasar :

A. Runtunan (Sequence)

Algoritma merupakan runtunan (sequence) satu atau lebih instruksi/pernyataan, dan setiap pernyataan dikerjakan secara berurutan sesuai dengan urutan penulisannya, yang berarti bahwa :

1. Tiap instruksi dikerjakan satu per satu
2. Tiap instruksi dilaksanakan tepat sekali (tidak ada instruksi yang diulang)
3. Tiap instruksi dilaksanakan dengan urutan yang sama antara pemroses dengan yang tertulis di dalam teks algoritmanya
4. Akhir dari instruksi terakhir merupakan akhir algoritma.

Caranya: sebuah instruksi dilaksanakan setelah instruksi sebelumnya selesai dilaksanakan. Urutan instruksi menentukan keadaan akhir algoritma.

Struktur Dasar Algoritma :

- Runtunan (sequence)
  - Algoritma merupakan runtunan (sequence) satu atau lebih instruksi / pernyataan
  - Setiap pernyataan dikerjakan secara berurutan sesuai dengan urutan penulisnya.

Contoh 1 :

Program Tukar_isi

Diberikan 2 buah ember, A dan B; ember A berisi air berwarna merah, ember B berisi air berwarna biru. Pertukarkan isi kedua ember itu sedemikian sehingga ember A berisi air berwarna biru dan ember B berisi air berwarna merah.

ALGORITMA:

1. Tuangkan air dari ember A kedalam ember C
2. tuangkan air dari ember B kedalam ember A
3. Tuangkan air dari ember C kedalam ember B

Algoritma diatas disusun oleh runtunan yang terdiri atas 3 buah pernyataan. Tiap pernyataan akan dieksekusi dalam urutan yang sama sebagaimana tertulis dalam algoritma diatas. Hasil akhir algoritma adalah: ember A berisi air dari ember B, dan ember B berisi air dari ember A semula.

Contoh 2 :

Misal nilai A=8, B=5. tukarkan nilai A dan B, sehingga menjadi A=5, B=8.

Algoritma :

Isikan nilai A kedalam B

Isikan nilai B kedalam A

Jika anda isikan seperti diatas maka algoritma anda salah. Yang benar adalah, kita harus menggunakan peubah bantu yaitu C, jadi algoritma yang benar adalah:

Algoritma :

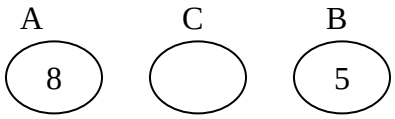
isikan nilai A kedalam C

isikan nilai B kedalam A

isikan nilai C kedalam B

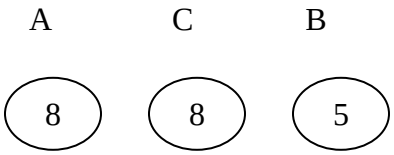
Ilustrasi :

Sebelum pertukaran:

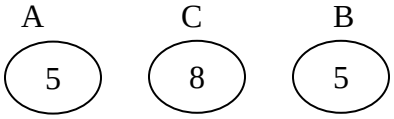


Proses pertukaran:

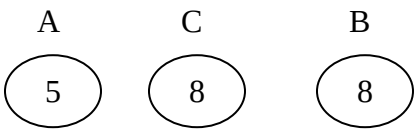
Isikan nilai A ke dalam C



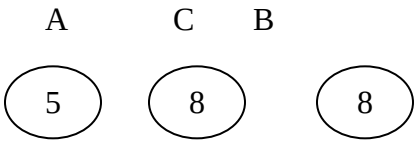
Isikan nilai B ke dalam A



Isikan nilai C ke dalam B



Setelah pertukaran



B. Pemilihan (*Selection*)

Dalam kasus Pemilihan adalakalanya sebuah aksi dikerjakan jika kondisi tertentu dipenuhi.

Misalnya : jika lampu traffic light sekarang berwarna merah, maka setiap kendaraan harus berhenti. Langkah seperti ini harus kita tulis dalam pernyataan :

Jika lampu traffic light berwarna merah, maka berhenti

Dan dalam algoritma ditulis dengan pernyataan:

If kondisi **then** aksi

If artinya jika, **then** artinya maka, **kondisi** adalah persyaratan yang dapat bernilai benar atau salah. **Aksi** sesudah kata then hanya dilaksanakan apabila kondisi bernilai benar, sebaliknya apabila kondisi bernilai salah maka aksi tidak dilaksanakan.

Bentuk lain :

If kondisi **then** aksi1
else aksi2

else artinya : kalau tidak, pernyataan ini diberikan jika kondisi salah, maka aksi yang kedua akan dikerjakan.

Contoh : Tentukan apakah bilangan bulat x merupakan bilangan ganjil atau genap

if x habis dibagi 2 then
    tulis x adalah bilangan genap
else
    tulis x adalah bilangan ganjil

C. Pengulangan (*Repetition)*

Struktur dasar Pengulangan, adalah kemampuan untuk dapat mengerjakan tulisan atau kalimat yang berulang-ulang.

Misalnya, kita ingin membuat kalimat "Saya sedang belajar Algo1" sebanyak 300 kali.

Kita tidak perlu menuliskan kalimat tersebut benar-benar sampai 300 kali, untuk mengatasi hal ini, maka kita bisa menggunakan struktur pengulangan

1) for-do

Struktur umum Pengulangan for-do

- **for pencacah pengulangan dari 1 sampai N do**
  aksi
- **for var ← awal to akhir do**
  aksi
  endfor

Penjelasan : aksi akan dilakukan sebanyak hitungan pencacah pengulangan, yaitu dari 1 sampai N.

Algoritma yang digunakan untuk misal diatas adalah:

PROGRAM Menulis_kalimat_300_kali
Menuliskan kalimat "saya sedang belajar Algo1"
Sebanyak 300 kali.

ALGORITMA:
for i dari 1 sampai 300 do
Tulis "saya sedang belajar Algo1"

2) repeat-until

Struktur pengulangan yang kedua adalah **repeat-until. repeat** artinya "ulangi", sedangkan **until** artinya "sampai" atau "hingga".

Struktur umum pengulangan repeat-until

repeat
aksi
until kondisi_stop

Penjelasan : aksi dikerjakan berulang sampai kondisi_stop benar.

3) while-do

Struktur pengulangan yang ketiga adalah **while-do**. **while** artinya "selagi" atau "selama" dan **do** artinya "lakukan".

Struktur umum pengulangan while-do

```
while kondisi_ulang do
        aksi
```

Penjelasan : selama kondisi_ulang masih benar, maka aksi dikerjakan.

Perbedaan dengan repeat-until, jika pada repeat-until kondisi pengulangan (kondisi_stop) dievaluasi di akhir, maka pada while-do kondisi pengulangan (kondisi_ulang) dievaluasi di awal pengulangan.

2.3. Strategi Perancangan Puncak Turun (*TopDown Design)*

Cara pendekatan ini sangat bermanfaat dalam membuat algoritma untuk masalah yang cukup rumit dan komplek. Strategi perancangan puncak turun dimulai dengan membuat algoritma secara global (garis besar) lebih dahulu, selanjutnya setiap langkah diuraikan lagi menjadi beberapa langkah yang lebih sederhana.

Contoh : urutkan data pada tabel berikut ini.

14
45
85
19
31

Langkah 1 : cari nilai terbesar diantara N buah data

Langkah ini masih terlalu global, kita harus menguraikan langkah pencarian menjadi langkah-langkah yang lebih sederhana sehingga bisa dikerjakan oleh komputer.

Mula-mula elemen ke-1 dari tabel dianggap sebagai elemen terbesar (maks), bandingkan maks dengan elemen ke-2, 3…N. selama membandingkan, bila ada yang lebih besar dari maks, maka element tersebut menjadi maks.

Diperinci dengan langkah sederhana:

```
1.1  asumsikan elemen ke-1 sebagai elemen terbesar sementara(maks)
1.2  while belum mencapai elemen ke-N do
       tinjau elemen berikutnya
           if elemen ini lebih besar dari maks then
             ganti maks dengan elemen tersebut.
```

Langkah 2 : tempatkan nilai terbesar tersebut pada posisi yang tepat.

Setelah elemen terbesar ditempatkan pada posisi elemen terbawah, maka elemen-elemen tabel yang belum terurut adalah dari elemen ke-1 sampai elemen ke(N-1)

Diperinci dengan langkah sederhana:

2.1 masukkan elemen ke-N didalam C

2.2 masukkan maks kedalam elemen ke-N

2.3 masukkan C kedalam tempat maks yang lama

Langkah 3 : ulangi dari langkah 1 untuk N-1 buah data yang lain.

Diperinci dengan langkah sederhana:

3.1 kurangi N dengan 1

3.2 ulangi dari langkah (1.1)

Kita menggunakan notasi while-do, untuk langkah 3.2 yang menyatakan pengulangan dari langkah 1.1 sampai 3.1.

Karena langkah 3.1 menyebabkan nilai N terus berkurang sehingga tersisa hanya satu elemen saja. Elemen terakhir ini tidak perlu kita urutkan karena sudah pasti terurut dengan sendirinya.

Algoritmanya menjadi:

PROGRAM Pengurutan

Program untuk mengurutkan N elemen tabel sehingga terurut membesar

ALGORITMA

1. cari nilai terbesar diantara N buah data
   1.1 asumsikan elemen ke-1 sebagai elemen terbesar sementara(maks)
   1.2 while belum mencapai elemen ke-N do
       tinjau elemen berikutnya
       if elemen ini lebih besar dari *maks* then
       ganti *maks* dengan elemen tersebut
2. tempatkan nilai terbesar tersebut pada posisi yang tepat.
   2.1 masukkan elemen ke-N didalam C (temporary)
   2.2 masukkan *maks* kedalam elemen ke-N
   2.3 masukkan C kedalam tempat *maks* yang lama
3. ulangi dari langkah 1 untuk N-1 buah data yang lain.
   3.1 kurangi N dengan 1
   3.2 ulangi dari langkah (1.1)

PENUTUP

1. Dasar-dasar setiap algoritma walaupun sekomplek apapun meliputi runtunan, pemilihan dan pengulangan.
2. Cara pendekatan *Top Down Design* sangat bermanfaat dalam membuat algoritma untuk masalah yang cukup rumit dan komplek.

SOAL-SOAL

1. algoritma dibawah ini membagi sekantung permen secara adil kepada 3 orang dengan cara memberikan satu kepada tiap anak secara berulang-ulang :

```
repeat
  berikan satu permen kepada anak pertama
  berikan satu permen kepada anak kedua
  berikan satu permen kepada anak ketiga
until kantung permen kosong
```

   pada keadaan bagaimana perulangan itu berhenti

2. Buat sintak perulangannya untuk menampilkan tulisan "Algoritma" sebanyak 5 kali.
3. Ubahlah penggalan algoritma di bawah ini ke dalam bahasa C++ (dengan menggunakan 3 macam sintaks pengulangan).

```
for p ← 1 to n do
    for q ← 1 to p do
        output (q)
    endfor
    output (p)
endfor
```

Referensi :

- Buku Teks
  1. Munir, Rinaldi (2005), *Algoritma dan Pemrograman dalam Bahasa Pascal dan C*, Buku 1, Edisi Ketiga, Penerbit Informatika Bandung.
  2. Charibaldi, N. (2004), *Modul Kuliah Algoritma Pemrograman I*, Edisi Kedua, Yogyakarta

- Buku Acuan/Referensi
  1. Brassard, Gilles (1999), *Fundamentals of algorithma*, PrinteceHall.
  2. Jarne, Stroustrup B. (1997), *C++ Programming language,* AT &T.
  3. Kristanto, Andri (2003), *Algoritma pemrograman C++*, Graha Ilmu.
  4. Schildt,Herbert (2000), *The Complete Reference C++*, McGraw-Hill.
  5. Sedgewick, R. (2000), *Algoritma Third edition In C part 5*, Addison Wesley.