BAB 1
Mengenal algoritme

- Apa itu algoritme.?

Aloritme adalah suatu metode khusus yang tepat dan terdiri dari serangkaian langkah yang terstruktur dan dituliskan secara matematis, yang akan dikerjakan untuk menyelesaikan suatu masalah dengan bantuan komputer. Jadi berdasarkan definisi ini, dapat dikatakan algoritma merupakan langkah penyelesaian suatu masalah yang manghasilkan solusi dalam bentuk program komputer. Namun penting diketahui bahwa algoritma tidaklah tergantung oleh suatu bahasa pemrograman tertentu, artinya suatu algoritma harus dapat diwujudkan oleh bahasa pemrograman komputer apa apun.

- Mengapa algoritme penting.?

Mengapa algoritma sangat penting? Karena, jika kita memecahkan masalah dengan asal-asalan tanpa langkah-langkah (algoritma) yang tepat, maka sudah dipastikan masalah tersebut akan berantakan karena tidak tersusun dengan rapi. Misalnya kita ingin membuat kue, maka cara yang paling efektif dan efesien sesuai dengan pola dan abstraksi sebelumnya hingga tahap packing, diurutkan secara lengkap, terukur dan kreatif. Dan masih banyak permasalahan dalam kehidupan sehari-hari yang dapat di pecahkan dengann algoritma. Oleh karena itu dibutuhkan sebuah algoritma sangatlah penting, jika tidak ada algoritma mungkin masalah bisa diselesaikan tapi bisa juga cara menyelesaikannyalah yang akan rumit.

Algoritma itu adalah alur atau langkah-langkah yang disusun secara tertulis, sistematis juga berurutan yang digunakan untuk menyelesaikan masalah. Bisa juga algoritma tersusun secara tidak sengaja. Setiap jam bahkan setiap menit pun manusia bisa membuat satu algoritma tanpa disusun terlebih dahulu.

- Konsep dasar algoritme.?

Deskripsi Algoritma :

Aksi 1 : Tuangkan larutan dari bejana A ke dalam bejana B

Aksi 2 : Tuangkan larutan dari bejana B ke dalam bejana A.

• Algoritma TUKAR ISI BEJANA di atas tidak menghasilkan pertukaran yang benar. Langkah di atas

tidak logis, hasil pertukaran yang terjadi adalah percampuran kedua larutan tersebut.

• Untuk mempertukarkan isi duah bejana, diperlukan sebuah bejana tambahan sebagai tempat penampungan sementara, misalnya bejana C. Maka algoritma untuk menghasilkan pertukaran yang benar adalah sebagai berikut :

Deskripsi Algoritma 2:

Aksi 1 : Tuangkan larutan dari bejana A ke dalam bejana C.

Aksi 2 : Tuangkan larutan dari bejana B ke dalam bejana A.

Aksi 3 : Tuangkan larutan dari bejana C ke dalam bejana B.

Ciri penting algoritma:

• Algoritma harus berhenti setelah mengerjakan sejumlah langkah terbatas.

• Setiap langkah harus didefinisikan dengan tepat dan tidak berarti-dua (Ambiguitas).

• Algoritma memiliki nol atau lebih masukan (input).

• Algoritma memiliki nol atau lebih keluaran (output).

• Algoritma harus efektif (setiap langkah harus sederhana sehingga dapat dikerjakan dalam waktu yang efisien).

Struktur Dasar Algoritma
Langkah-langkah penyelesaian masalah bisa berupa :
a. Runtunan (sequence))
Sebuah runtunan terdiri dari satu atau lebih instruksi. Tiap instruksi dikerjakan berurutan sesuai aturan penulisannya. Urutan instruksi menentukan keadaan akhir algoritma, jika urutannya diubah maka hasil akhirnya mungkin akan berubah. Urutan instruksi menunjukkan cara berfikir penyusun algoritma dalam menyelesaikan masalah
Runtunan Instruksi:
Instruksi 1
Instruksi 2
Instruksi 3
b. Pemilihan (selection)
Adakalanya sebuah instruksi dikerjakan jika sebuah kondisi tertentu terpenuhi
Struktur umum :
If kondisi
then
Aksi
atau If kondisi then
Aksi 1
Else
Aksi 2
c. Pengulangan (repetition)
Komputer tidak pernah bosen dan lelah jika diminta untuk mengerjakan instruksi secara berulang-ulang.
Contoh :
• Menulis kalimat ”Saya harus lebih giat belajar” sebanyak 1000 kali
Ulangi :
– Tulis kalimat ” Saya harus lebih giat belajar”
Sampai jumlah_kalimat = 1000
• Mengupas 100 buah kentang
Selama kentang terkupas < 100 maka
– Kupas 1 kentang

- Struktur penulisan algoritme.

Berikut adalah beberapa contoh penulisan algoritma bahasa natural untuk kasus-kasus dalam kehidupan sehari-hari:

Kasus 1:
Menukar isi gelas berisi kopi dan gelas berisi teh.

Untuk kasus ini kita misalkan gelas berisi kopi adalah gelas A, sedangkan gelas isi teh adalah gelas B

Penulisan algoritma bahasa natural:

1. Mulai
2. Sediakan satu gelas kosong misal namanya gelas C.
3. Masukan isi gelas A (gelas berisi kopi)kedalam gelas C (Gelas kosong)
4. Masukan isi gelas B (gelas berisi teh) kedalam gelas C(gelas kosong yang sebelumnya berisi kopi)
5. Masukan isi gelas C (gelas kosong yang sudah diisi kopi) kedalam gelas B (gelas kosong yang sebelumnya berisi teh)
6 Selesai.

Kasus 2
Algoritma menyalakan motor

1. Mulai
2. Masukan kunci motor
3. Putar kunci motor hingga kontak aktif
4. Tekan tombol starter untuk menyalakan motor.
5. Jika motor tidak menyala gunakan cara manual.
5. Motor menyala
6. Selesai

Kasus 3

Algoritma untuk kasus menanak nasi

1. Mulai
2. Cuci beras sampai bersih
3. Masukan beras kedalam mejic com
4. Colokan mejicom ke listrik
4. Tekan tombol menanak nasi dan tunggu hingga tombol mati
5. Nasi masak
6. Selesai.

Pada intinya di kehidupan sehari-hari ada banyak sekali permasalahan permasalahan dari mulai sederhana sampai yang rumit yang bisa kita tuliskan langkah-langkah penyelesaian masalahnya menggunakan algoritma bahasa natural.

Algoritma juga bukan hanya bisa digunakan untuk kasus-kasus aktivitas harian saja, namun bisa juga digunakan untuk memecahkan masalah yang lebih rumit dari itu seperti masalah matematika dan masalah lainnya.

Penulisan algoritma dalam bahasa natural tidaklah sulit, yang terpenting anda tau urutan langkah penyelesainnya kemudian ditulis dengan bahasa yang mudah dipahami.

Itulah Beberapa Contoh Penulisan Algoritma menggunakan bahasa natural dalam kehidupan sehari

FLOWCHART

Flowchart adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program.

Ternyata langkah-langkah untuk menyelesaikan masalah dalam algoritma selain dapat menggunakan bahasa natural, anda juga bisa menggunakan flowchart, namun untuk menggunakan flowchart anda setidaknya harus memahami simbol-simbol yang digunakan oleh flowchart.

Berikut adalah daftar simbol-simbol flowchart yang harus anda pahami untuk membuat algoritma:

Kita bisa menggunakan simbol-simbol di atas untuk menyelesaikan sebuah masalah dalam algoritma.

Contoh:
Berikut adalah salah satu contoh menggunakan flowchart untuk menyelesaikan kasus menghitung luas segitiga:

Yang harus diperhatikan ketika menyajikan algoritma flowchart adalah harus dimulai dengan simbol Start / mulai dan diakhiri dengan simbol Finish / selesai, seperti terlihat pada contoh di atas, bisa menggunakan bahasa apapun yang mudah dipahami.

Ketika membuat algoritma pseudocode menggunakan pendekatan bahasa pemrograman pascal maka langkah-langkah penulisannya dibagi menjadi 3 bagian sebagai berikut:

a. Bagian Judul
Judul harus dimulai dengan kata ALGORITMA diikuti dengan nama judul.

Aturan penulisan nama judul:

- Tidak boleh mengandung spasi, spasi dapat diganti dengan karakter _ (underscore)
- Tidak boleh diawali dengan angka
- Tidak boleh menggunakan istilah-istilah yang sudah digunakan sebagai keyword di bahasa pemrograman.
- Bisa menggunakan huruf besar huruf kecil dan kombinasinya selama tidak menyalahi aturan diatas.

b. Bagian Deklarasi
Bagian deklarasi adalah bagian dalam algoritma yang digunakan untuk mendefinisikan jenis-jenis variable yang akan digunakan dalam proses algoritma. bagian ini dimulai dengan tulisan Deklarasi:

Jika anda pemula dan sulit memahami apa itu variable, sebaiknya anda membaca dulu mengenai konsep dasar algoritma yang sudah saya tulis sebeumnya.

c. Bagian Deskripsi
Bagian deskripsi adalah bagian yang berisi proses algoritma, pada bagian ini ditulis proses penyelesaian masalah. pada bagian ini diawali dengan tulisan Deskripsi:

d. Komentar
Komentar sifatnya opsional boleh dicantumkana tau tidak, komentar isinya untuk memberi penjelasan atau keterangan mengenai instruksi didalam algoritma penulisan komentar bisa diletakan dibaris mana saja didalam struktur algoritma, namun penulisan komentar harus di dalam tanda kurung kurawal {....}

contoh penulisan komentar:
{Ini komentar}

ALGORITME PSEUDOCODE
Selain memahami bagian penulisan algoritma di atas juga harus memahami instruksi atau syntax penulisan.

Untuk instruksi atau syntax dalam penulisan algoritma pseudocode umumnya akan menyesuaikan dengan pendekatan bahasa pemrograman yang akan digunakan.

Seperti misalnya jika bahasa pemrograman yang akan digunakan untuk mengkonversi algoritma tersebut adalah pascal, maka syntax algoritma pseudocode akan mengandung instruksi dan aturan penulisan yang digunalan dalam bahasa pemrograman pascal. begitu juga ketika menggunakan bahasa pemrograman yang lainnya.

Berikut beberapa syntax algoritma pseudocode bahasa pemrograman pascal yang sering digunakan untuk memahami algoritma tahap awal.

a. Penulisan variable
Penulisan variable ada dibagian deklarasi, aturan penulisannya kurang lebih seperti di bawah ini:

var1:var2,var3 : tipe_data

Untuk tipe_data, ada banyak sekali jenis nama tipe data yang bisa digunakan seperti integer, string, char, byte dll silahkan pelajari di artikel internet..

b. Masukan
untuk menulis instruksi masukan dari pengguna maka ditulis dengan instruksi:
read(variable_masukan)

c. Keluaran
untuk mencetak keluaran maka penulisannya adalah sebagai berikut:
write(variable_keluaran)

d. Instruksi lainnya menyesuaikan dengan instruksi bahasa pemrograman yang digunakan, akan dijelaskan lebih lanjut sesuai dengan kasus algoritma yang ingin dipecahkan.
Berikut adalah contoh algoritma pseudocode dengan pendekatan bahasa pemrograman pascal untuk kasus menghitung luas segitiga dengan ketentuan.

Luas dan alas diinput oleh pengguna dan hasil dari perhitungan disimpan dalam variable hasil kemudian hasilnya ditampilkan sebagai output.

OPRATOR DALAM ALGORITME

Dalam bahasa pemrograman, kita sudah pernah mendengar apa yang di maksud dengan Operator. Operator dalam bahasa pemrograman itu sendiri, terbagi dalam beberapa jenis, diantaranya :

- Operator Aritmatika
- Operator Penugasan/assigment
- Operator Logika
- Operator String

1. Operator Aritmatika

Operator aritmatika adalah operator yang digunakan hanya untuk melakukan operasi matematika, seperti : penjumblahan, pengurangan, perkalian, pembagian dan modulus (sisa hasil pembagian. Adapun simbol-simbol yang digunakan dalam operator ini antara lain :

- + : untuk penjumlahan
- - : untuk pengurangan
- * : untuk perkalian
- / : untuk pembagian
- % : modulus (sisa hasil bagi)

2. Operator Penugasan/assigment

Operator Penugasan/assigment adalah operator yang berfungsi untuk mengisi nilai suatu variabel, yang ditandai berupa (=) sama dengan. Adapun contohnya sebagai berikut :
B = 10
Pada variabel B akan terisi dengan nilai 10
A = 1 + B
Pada variabel A akan terisi nilai 1 yang ditambahkan dengan nilai B yaitu 10

3. Operator Logika

Operator Logika adalah operator yang hanya memiliki dua kondisi atau keadaan, yaitu true (1) dan false (0) yang biasanya digunakan untuk membandingkan hasil suatu keadaan. Operator logika terbagi menjadi 3 macam yaitu :

Operator Keterangan

&& Operator Logika AND
|| Operator Logika OR
! Operator Logika NOT

Adapun tabel kebenaran dari masing-masing operator adalah sebagai berikut :

- Operator AND (&&)

Nilai 1	Nilai 2	Nilai 1 && Nilai 2
False	False	False
False	True	False
True	False	False
True	True	True

- Operator OR (||)

Nilai 1	Nilai 2	Nilai 1 || Nilai 2
False	False	False
False	True	True
True	False	True
True	True	True

- Operator NOT (!)

Nilai	! Nilai
True	False
False	True

4. Operator String

Operator string adalah operator yang berfungsi untuk menggabungkan dua buah character dalam pemrograman. Adapun contohnya adalah sebagai berikut :
A = “Belajar”
B = “Bahasa C++”
C = A + B
Maka nilai C adalah “Belajar Bahasa C++”

Apa itu Algoritma Percabangan?
written by Teddy

Pengertian algoritma pemrograman seleksi kondisi, atau disebut juga algoritma percabangan (atau disebut juga dengan flow control dan algoritma pemilihan) adalah salah satu jenis perintah dalam algoritma yang digunakan sebagai cara untuk memberitahukan program tentang perintah apa yang harus dijalankan, dimana perintah tersebut disesuaikan dengan beberapa kondisi tertentu. Fungsi algoritma percabangan ini pada adalah untuk memproses keputusan yang tepat dan sesuai dengan yang keinginan pengguna sistem berdasarkan beberapa kondisi yang terjadi pada sistem yang digunakan tersebut.

Dalam sebuah program atau sistem, ada saatnya sebuah instruksi atau perintah hanya bisa dilakukan jika memenuhi suatu kondisi atau persyaratan tertentu. Itu mengapa, algoritma percabangan ini bisa disebut juga dengan algoritma seleksi kondisi. Agar Anda paham maksudnya, kami berikan sebuah contoh. Misalkan, kita hendak menentukan apakah suatu

bilangan termasuk bilangan genap atau bilangan ganjil. Nah, algoritmanya dapat kita jelaskan sebagai berikut:

1. Mulai
2. Masukkan suatu bilangan, misalkan bilangan X)
3. Jika bilangan X habis dibagi dua, maka lanjut ke perintah keempat. Jika tidak lanjut ke perintah kelima.
4. Tuliskan “X adalah bilangan genap”. Lanjut ke perintah keenam.
5. Tuliskan “X adalah bilangan ganjil”
6. Selesai

Dari algoritma di atas, kita bisa lihat bahwa ada dua kemungkinan perintah yang akan dikerjakan setelah perintah ketiga dikerjakan. Perintah pertama, jika bilangan X habis dibagi dua maka selanjutnya perintah keempat yang dikerjakan, kemudian lompat ke perintah keenam dan perintah kelima tidak dikerjakan. Perintah kedua, jika bilangan X tidak habis dibagi dua maka melompat ke perintah kelima dan perintah keempat tidak dikerjakan. Kedua perintah tersebut sama-sama berakhir pada perintah keenam, yang menyatakan bahwa proses algoritma telah selesai.

Bagaimana, sudah paham mengenai maksud dari algoritma percabangan ini? Nah, ternyata algoritma percabangan ini banyak macamnya. Namun, inti dari algoritma ini sama, yaitu suatu program atau sistem akan mengerjakan sebuah perintah yang disesuaikan dengan kondisi atau syarat tertentu. Apa saja macam-macam algoritma percabangan? Artikel kali ini akan mengulasnya untuk Anda. Berikut ini pembahasannya:

1. Percabangan untuk 1 kondisi

Pada percabangan jenis ini, hanya ada satu kondisi yang menjadi syarat untuk melakukan satu buah atau satu blok instruksi. Format umum dari algoritma percabangan dengan satu kondisi adalah sebagai berikut:

```
IF kondisi THEN
instruksi
ENDIF
```

Arti dari format di atas, jika “kondisi” bernilai benar atau tercapai, maka aksi dikerjakan. Sedangkan jika bernilai salah, maka instruksi tidak dikerjakan dan proses langsung keluar dari percabangan dan kembali lagi ke kondisi awal.

Contoh dari penggunaan algoritma percabangan untuk satu kondisi adalah sebagai berikut:

```
if A > B then
write (A)
end if
```

Instruksi di atas artinya instruksi akan menampilkan nilai A hanya jika kondisi “A lebih besar daripada B” bernilai benar. Jika bernilai salah, maka tidak ada aksi yang akan dilakukan atau proses langsung keluar dari percabangan (end if).

Berikut ini kami berikan contoh beberapa contoh program algoritma percabangan untuk satu kondisi menggunakan macam-macam bahasa pemrograman. Berikut ini adalah contoh untuk program menggunakan bahasa Pascal adalah sebagai berikut:

```
uses crt;
var
jeniskelamin:char;
begin
clrscr;
```

```
writeln(‘Jenis Kelamin : ’);
writeln(‘L unutk laki-laki, P untuk perempuan’);
writeln(‘Jenis kelamin anda: ’);readln(jeniskelamin);
if(jeniskelamin = ‘l’) then writeln(‘Laki-laki’);
if(jeniskelamin = ‘p’) then writeln(‘Perempuan’);
readkey;
end
```

Contoh lainnya dari program percabangan untuk satu kondisi pada suatu program menggunakan bahasa C++ adalah sebagai berikut:

```
#include <iostream.h>
int main (){
int nilai;
char a;
cout<<“Masukkan Nilai Anda:”;
cin>>nilai;
if (nilai>60){
cout<<“Selamat Anda Lulus!!”;
}
cin>>a;
return 0;
}
```

2. Percabangan untuk 2 kondisi

Pada percabangan jenis ini, ada dua kondisi yang menjadi syarat untuk dikerjakannya salah satu dari dua instruksi. Kondisi ini bisa bernilai benar atau salah. Bentuk umum dari percabangan dengan dua kondisi adalah sebagai berikut:

```
IF kondisi THEN
instruksi 1
ELSE
instruksi 2
ENDIF
```

Arti dari format di atas, jika “kondisi” bernilai benar maka instruksi 1 yang akan dikerjakan. Sedangkan jika bernilai salah), maka instruksi 2 yang akan dikerjakan. Perbedaannya dengan percabangan untuk satu kondisi terletak pada adanya dua instruksi untuk dua kondisi, yaitu kondisi bernilai benar dan kondisi bernilai salah.

3. Percabangan untuk 3 kondisi atau lebih

Algoritma percabangan untuk tiga kondisi atau lebih adalah bentuk pengembangan dari dua macam algoritma percabangan yang telah dibahas sebelumnya. Karena itu, percabangan jenis ini akan memiliki banyak variasi. Secara umum, format percabangannya dapat dituliskan sebagai berikut :

```
IF kondisi THEN
instruksi 1
ELSE IF kondisi 2 THEN
instruksi 2
ELSE
instruksi 3
ENDIF
```

Maksud dari algoritma di atas, instruksi 1 akan dikerjakan jika “kondisi 1” bernilai benar. Jika bernilai salah, pemeriksan dilanjutkan ke “kondisi 2”. Jika “kondisi 2” bernilai benar, maka instruksi 2 dikerjakan. Jika tidak, pemeriksaan dilanjutkan pada kondisi-kondisi lainnya. Pemeriksaan ini akan terus dilakukan terhadap semua kondisi yang ada. Jika tidak ada satu pun kondisi yang bernilai benar maka pernyataan yang dikerjakan adalah instruksi 3 atau instruksi (n+1) pada percabangan lebih dari 3 kondisi.

4. Percabangan “Case of….”

Selain menggunakan format yang dijelaskan pada poin 3, percabangan 3 kondisi atau lebih bisa juga menggunakan format “Case Of”. Format ini memiliki kegunaan yang sama, tetapi format ini digunakan untuk memeriksa data yang bertipe karakter atau integer. Secara umum format penulisannya adalah sebagai berikut:

```
switch (ekspresi) {
case kontanta-1:
instruksi 1 break;
case konstanta-2:
instruksi 2 break;
default:
instruksi 3
}
```

Contoh penerapan percabangan Case Of dalam sebuah program menggunakan bahasa Pascal adalah sebagai berikut:

```
uses wincrt;
var x : integer;
begin
write (‘Masukkan sebuah nilai [0…3] : ‘);
readln (x);
Case (x) of
0 : Writeln(‘X bernilai 0’);
1 : Writeln(‘x bernilai 1’);
2 : Writeln(‘X bernilai 2’);
3 : Writeln(‘X bernilai 3’);
else
Writeln(‘X tidak bernilai 0, 1, 2, ataupun 3’);
end;
end.
```

Contoh program percabangan Case Of menggunakan bahasa C++ :

```
void main() {
int nHari;
cout << “Masukkan No Hari [1..7] : “;
cin >> nHari;
cout << “Ini adalah hari “;
switch (nHari) {
case 1:
cout << “Ahad”;
break;
case 2:
```

```
cout << “Senin”;
break;
case 3:
cout << “Selasa”;
break;
case 4:
cout << “Rabu”;
break;
case 5:
cout << “Kamis”;
break;
default:
cout << “Jumat”;
}
getch();
}
```

5. Percabangan bersarang

Percabangan bersarang adalah instruksi yang terdiri dari adanya percabangan yang lain di dalam percabangan, atau di dalam percabangan ada percabangan lagi. Format penulisan untuk percabangan bersarang adalah sebagai berikut:

```
If <kondisi1> then
if <kondisi2> then
Instruksi1
Else
Instruksi2
Else
If <kondisi3>
Instruksi3
Else
Instruksi4
EndIf
```

Jika kondisi berjumlah lebih dari 3 kondisi, polanya tetap sama. Untuk kondisi ke 2 dan seterusnya, penulisannya menggunakan “ELSE IF kondisi THEN”, sedangkan untuk kondisi terakhir cukup menggunakan ELSE saja.

Mulanya, “kondisi1” dicek nilai kebenarannya. Jika benar, maka dicek nilai kebenaran “kondisi2”. Jika “kondisi2” benar, maka dikerjakan Instruksi1. Jika tidak, dikerjakan Instruksi2.

Sedangkan jika “kondisi1” tidak benar, maka akan dicek nilai kebenarannya. Jika “kondisi3” bernilai benar, maka dikerjakan Instruksi3. Jika tidak, maka akan dikerjakan Instruksi4.

Inilah salah satu contoh program percabangan bersarang (Nested If) menggunakan bahasa Pascal :

```
uses wincrt;
var x, y, z : real;
begin
write (‘Masukkan bilangan pertama : ‘);
readln (x);
```

```
write (‘Masukkan bilangan kedua : ‘);
readln (y);
write (‘Masukkan bilangan ketiga : ‘);
readln (z);
if x > y then
if x > z then
write (‘Bilangan terbesar : ‘,x:5:2)
else
write (‘Bilangan terbesar : ‘,z:5:2)
else
if y > z then
write (‘Bilangan terbesar :’,y:5:2)
else
write (‘Bilangan terbesar : ‘,z:5:2);
end.
```

Dan di bawah ini adalah satu contoh program percabangan bersarang lainnya menggunakan bahasa C++ :

```
#include <iostream.h>
void main() {
int A, B, C;
cout << “masukan angka 1 =  “;
cin >> A;
cout << “masukan angka 2 =  “;
cin >> B;
cout << “masukan angka 3 =  “;
cin >> C;
if(A<B){
if(A<C)
cout<< “angka terkecil adalah : ” << A;
else
cout<< “angka terkecil adalah : ” << C;
}
else if(B<C)
cout<< “angka terkecil adalah : ” << B;
else
cout<< “angka terkecil adalah : ” << C;}
```

Algoritma perulangan (iteration / looping), apa itu?
Part 3 — Perkenalan Struktur Algoritma

Mochamad Iqbal Dwi Cahyo
Follow
Nov 15, 2016 • 3 min read
Definisi Repetisi (looping)

Repetisi dalam bahasa inggris sering disebut loops, biasanya digunakan untuk mengulang kode yang sama berkali-kali. Jumlah repetisinya itu beragam sesuai yang diinginkan, biasanya berisi ekspresi true/false.

Analogi di kehidupan

Pernah ga kalian ketika terkena sanksi, dan diperintahkan guru untuk menulis “aku tidak akan melakukannya lagi” di papan tulis hingga 100x bahkan lebih? Cape? itu yang pertama kali terlintas. Membayangkannya saja sudah sangat lelah (whew). Di bahasa pemrograman, kamu tidak perlu melakukannya lagi secara manual. #thuglife

---

Struktur (Looping)

Ada 3 tipe struktur loop di Pascal, yaitu:

1. for loops
2. while loops
3. repeat-until loops

Ketiga tipe ini ga harus selalu ada, tapi menggunakannya dalam bentuk yang berbeda sesuai tujuannya sangat dianjurkan.

For loops

```
// for loop format
for (initialCondition; testExpression; iterativeStatement) {
   statement1;
   statement2;
   // ...
   statementN;
 }
```

How it works

initialCondition berjalan hanya sekali, ketika repetisi awal

Setelah itu periksa di testExpression. (bentuk ini mirip di while loops). Jika salah, berhenti. Jika benar, maka:

Menjalankan apa yang ada di badan loops, yaitu statement1 — statementN

Jalankan iterativeStatement, yaitu menambah nilai variabel hingga testExpression terlampaui.

Go back to the testExpression step and repeat

- Contoh gambar:

a.1 Flowchart — For loops

- Nah, ini ada problem set untuk kalian. Menurut kalian, apa keluaran dari pseudocode ini? :D

a.2 Pseudocode — For Loops

While-loop

```
 // while loop format
 var_number; // initialize condition or expression while (expression) // expression = var_number operator var_numberN
 {
   statement1;
   statement2;
```

```
    // ...

    statementN;
 }
```

How they work

Tentukan nilai awal var_number, untuk diperiksa pertama kali

Expression di sini untuk memeriksa kondisi yang ada, dan menentukan loop harus berhenti atau tidak.

expression = var1 operator var2

var1: adalah angka atau nilai awal

operator: bentuk operasi seperti <, >, =, !=, <=, >=, etc

var2: angka yang ingin dibandingkan oleh angka atau nilai awal

benar berarti tetap jalankan badan loop-nya.

salah berarti berhenti

Hal yang perlu diperhatikan. Dalam while loops, ekspresinya diperiksa pertama kali, berbeda dengan bentuk yang berikutnya yaitu repeat-until.

- Contoh gambar:

b.1 Flowchart — While loops

- Problem set lagi yippe! Apa keluaran dari pseudocode ini?

b.2 Pseudocode — While loops

Repeat-until

Sebenarnya bentuk while dan do — while (repeat — until) memiliki flowchart yang sama persis, hanya dengan pengecualian:

“badan” loop dijalankan terlebih dahulu, kemudian periksa kondisinya.

- Contoh gambar:

c.1 Flowchart — Repeat until

- Berikut problem-set untuk repeat-until:

c.2 Pseudocode — Repeat until

Studi Kasus

Nah, masih bersama dengan kami? Good :D. Teori di atas memang akan sangat membosankan ketika kita tidak segera mencobanya. Let’s get coding, guys! XD

For-loops

Berikut contoh kasus for loops:

While-loops

Berikut contoh kasus while-loops:

Repeat-until

Berikut contoh kasus repeat-until:

7 Contoh Algoritma Pemrograman Dasar untuk Latihan

written by Teddy

Dalam dunia programming, menguasai algoritma adalah hal penting. Karena algoritma adalah tumpuan untuk menyeleasikan sebuah persoalan. Lalu, apa sih pengertian algoritma pemrograman?

Algoritma pemrograman adalah urutan langkah logis tertentu untuk memecahkan suatu masalah. Hal ini ditekankan pada urutan langkah logis, yang artinya algoritma harus mengikuti suatu urutan tertentu, dan langkah-langkahnya tidak boleh diloncat. Pengertian lainnya dari algoritma adalah urutan langkah-langkah logis dalam penyelesaian masalah yang disusun secara sistematis.

Asal kata algoritma sendiri berasal dari nama Abu Ja'far Mohammed Ibn Musa al-Khowarizmi, ilmuwan Persia yang menulis buku berjudul "Al Jabr W'Al-Muqabala" (Rules of Restoration and Reduction) yang diterbitkan pada tahun 825 M.

Dalam algoritma, alur pemikiran dalam menyelesaikan suatu persoalan dituangkan secara tertulis. Hal pertama yang ditekankan adalah alur pikiran, sehingga algoritma seseorang dapat juga berbeda dari algoritma orang lain. Sedangkan penekanan kedua adalah tertulis, yang artinya alur tersebut dapat berupa kalimat, gambar, atau tabel tertentu.

Algoritma sendiri memiliki beberapa ciri penting agar bisa digunakan untuk menyelesaikan masalah, diantaranya:

- Algoritma harus berhenti setelah mengerjakan sejumlah langkah tertentu
- Setiap langkah harus didefinisikan dengan tepat dan tidak ambigu
- Algoritma memiliki masukan berjumlah nol atau lebih.
- Algoritma memiliki keluaran berjumlah nol atau lebih.
- Algoritma harus efektif. Maksudnya setiap langkah yang tertulis harus sederhana sehingga dapat dikerjakan dalam waktu singkat dan masuk akal.

Dalam bidang komputer, fungsi algoritma sangat diperlukan untuk menyelesaikan berbagai masalah pemrograman, terutama dalam komputasi numerik. Tanpa algoritma yang dirancang dengan baik, proses pemrograman akan menjadi salah, rusak, lambat dan tidak efisien. Pelaksana langkah-langkah di dalam algoritma adalah sistem komputer. Agar manusia dan komputer dapat berkomunikasi, manusia memberikan perintah-perintah kepada komputer berupa kumpulan instruksi yang dikumpulkan di dalam program. Dalam menyelesaikan persoalan, komputer perlu merumuskan beberapa langkah penyelesaian persoalan dalam sekumpulan instruksi. Kumpulan instruksi yang dimengerti oleh komputer inilah yang disebut dengan program.

Untuk menerjamahkan bahasa manusia ke dalam bahasa komputer, diperlukan sebuah alat untuk menjembatani komunikasi di antara keduanya. Alat yang digunakan tersebut adalah bahasa pemrograman. Setiap bahasa pemrograman memiliki tingkatannya tersendiri, tergantung dari bagaimana bahasa tersebut bisa diterapkan langsung oleh manusia selaku pengguna. Tingkatan bahasa pemrograman dapat dikategorikan ke dalam 3 jenis, yaitu:

- Bahasa tingkat tinggi (High Level Language / HLL). Contohnya: Pascal, C, Java, PHP, ASP
- Bahasa tingkat menengah (Medium Level Language / MLL). Contohnya: Assembly
- Bahasa tingkat rendah (Low Level Language / LLL). Contohnya : Machine Code

Dari berbagai bahasa pemrograman, cara memberikan instruksinya berbeda-beda. Meskipun begitu, semuanya bertujuan untuk menghasilkan keluaran yang sama. Program yang ditulis dalam bahasa pemrograman akan dikonversi ke dalam bahasa mesin menggunakan penerjemah. Berikut ini metode menerjemahkan bahasa pemrograman ke dalam bahasa mesin dalam programming:

- Interpreter, yaitu menerjemahkan baris per baris instruksi. Bahasa Basic menggunakan metode ini.
- Compiler, yaitu menerjemahkan setelah seluruh instruksi yang ditulis. Bahasa Pascal, dan C adalah beberapa contoh bahasa pemrograman yang menggunakan metode ini.

Dalam mempelajari programming, Anda harus paham perbedaan antara belajar programming dengan belajar bahasa pemrograman. Belajar programming artinya Anda belajar tentang metode pemecahan masalah, kemudian menuangkannya dalam suatu notasi tertentu yang mudah dibaca dan dipahami. Sedangkan belajar bahasa pemrograman artinya Anda belajar memakai suatu bahasa, aturan tata bahasa, instruksi yang digunakan, serta tata cara pengoperasian compiler, untuk membuat program yang ditulis ke dalam bahasa tersebut.

Penulisan algoritma harus terdiri dari 3 bagian berikut ini:

- Judul algoritma;Bagian yang terdiri atas nama algoritma dan penjelasan (spesifikasi) tentang algoritma tersebut. Nama sebaiknya singkat dan menggambarkan apa yang dilakukan oleh algoritma tersebut.
- Deklarasi; Bagian untuk mendefinisikan semua nama yang digunakan di dalam program. Nama tersebut dapat berupa nama tetapan, peubah, tipe, prosedur dan fungsi.
- Deskripsi; Bagian ini berisi uraian langkah-langkah penyelesaian masalah yang ditulis dengan menggunakan notasi yang akan dijelaskan selanjutnya.

Penulisan algoritma sendiri tidak tergantung dari spesifikasi bahasa pemrograman dan kemampuan komputer yang mengeksekusinya. Notasi algoritma bukan notasi bahasa pemrograman, namun algoritma dapat diterjemahkan ke dalam berbagai bahasa pemrograman.

Lalu seperti apa contoh algoritma pemrograman dasar yang wajib Anda kuasai? Berikut ini contoh-contohnya:

1. Algoritma Narasi

Contoh: Algoritma Kelulusan_mhs

Persoalan: Diberikan data berupa nama dan nilai mahasiswa. Jika nilai mahasiswa lebih besar atau sama dengan 60 maka mahasiswa tersebut dinyatakan lulus. Sedangkan jika nilainya lebih kecil dari 60, maka mahasiswa tersebut dinyatakan tidak lulus.

Algoritmanya akan seperti berikut:

baca nama dan nilai mahasiswa.
jika nilai >= 60 maka
keterangan = lulus
tetapi jika
keterangan = tidak lulus.
tulis nama dan keterangan

2. Algoritma Pseudo Code

Contoh; Algoritma Kelulusan_mhs

Persoalan: Diberikan data berupa nama dan nilai mahasiswa. Jika nilai mahasiswa lebih besar atau sama dengan 60 maka mahasiswa tersebut dinyatakan lulus. Sedangkan jika nilainya lebih kecil dari 60, maka mahasiswa tersebut dinyatakan tidak lulus.

Deklarasi dari tipe datanya akan seperti berikut:

Nama = string
Nilai = integer
Keterangan = string

Algoritmanya akan seperti berikut:

read (nama, nilai)

if nilai >= 60 then
keterangan = ‘lulus’
else
keterangan = ‘tidak lulus’
write(nama, keterangan)

3. Algoritma Flowchart

Berikut ini adalah beberapa contoh dari algoritma flowchart. Fungsi flowchart pada pemrograman adalah untuk memudahkan programmer ketika merancang sebuah program komputer. Ini dia contoh-contohnya:

- Menentukan bilangan ganjil atau genap

Terdapat bilangan yang bernama “bilangan bulat” yaitu 0, 1, -1, 2, dan seterusnya serta bilangan asli 1, 2, 3, 4, 5, dst. Kedua jenis bilangan tersebut sering digunakan dalam perhitungan. Algoritma di bawha ini akan menentukan apakah bilangan tersebut ganjil atau genap> Berikut ini adalah flowchart dari algoritma tersebut:

Dari flowchart di atas, penjelasannya adalah sebagai berikut:

Bilangan genap adalah sebuah bilangan bulat yang akan habis atau tidak memiliki sisa jika dibagi 2 (dua). Sedangkan bilangan ganjil adalah sebuah bilangan bulat yang tidak akan habis apabila dibagi 2 (dua) atau ada sisa pembagian.

- Algoritma tahun kabisat

Tahun kabisat adalah sebuah tahun yang memiliki tambahan 1 hari yang bertujuan agar kalender dapat menyesuaikan dengan keadaan astronomi. Saat tahun kabisat, bulan Februari memiliki 29 hari. Tahun yang dapat untuk dibagi dengan 4 adalah tahun kabisat. Berikut ini adalah algoritma untuk menentukan tahun kabisat jika disajikan dengan flowchart seperti di bawah ini:

- Menghitung harga yang harus dibayar setelah mendapatkan sebuah diskon

Contoh flowchart kali ini adalah flowchart dari algoritma untuk menentukan biaya yang harus dibayar oleh pembeli sesudah mendapat diskon 10%. Syarat untuk mendapatkan diskon tersebut adalah jumlah dari total pembelian sebesar Rp.1.500.000,-. Berikut ini adalah algoritma flowchartnya:

Jumlah barang memiliki sifat yang dinamik sesuai dengan masukan dari pengguna. Intinya, jika total pembeliannya kurang dari 1500000, maka pembeli tidak akan mendapatkan diskon.

- Flowchart menghitung luas lingkaran

Algoritma flowchart di bawah ini adalah contoh flowchart untuk menghitung luas lingkaran dengan rumus L = πr2:

Algoritma:

- Program dimulai
- Tentukan nilai phi dan r
- Hitung L = phi x r2
- Cetak Hasil L
- Program Selesai

Flowchart:

- Flowchart menghitung luas segitiga

Berikut ini contoh flowchart program untuk menghitung luas segitiga, dimana diketahui nilai ALAS = 10 dan nilai TINGGI = 8. Pada flowchart ini, untuk mendapatkan nilai awal dari

“ALAS” dan “TINGGI” menggunakan kotak proses, karena nilai “ALAS” dan “TINGGI” sudah ditentukan sebelumnya.

Sekian artikel kami kali ini seputar contoh algoritma pemrograman dasar. Semoga artikel kami kali ini dapat menjadi bahan materi Anda untuk mempelajari pemrograman dasar.