1. **Apa itu algoritma**
2. Algoritma dalam kehidupan

Dalam sehari-hari kita tentu sering melakukan sebuah aktiviatas. Sejumlah langkah kita lakukan untuk menyelesaikan aktivitas tersebut. Dan uniknya, dua orang melakukan tujuan yang sama belum tentu melakukan cara yang sama.

Misalnya ada dua orang (A dan B) yang membuat mie. Berikut langkah-langkah yang mereka lakukan:

Yang A lakukan:

1. Rebus mie di air mendidih selama 3 menit sambil diaduk
2. Sementara mie direbus, campurkan bumbu bubuk dan minyak kedalam mangkuk
3. Tuangkan mi ke mangkok.
4. Taburkan mie dengan bawang goring
5. Mie siap dihidangkan

Yang B lakukan:

1. Rebus mie di air mendidih selama 3 menit sambil diaduk
2. Tuangkan mi ke mangkok
3. Campurkan bumbu bubuk dan minyak kedalam mangkok
4. Taburkan mie dengan bawang goring
5. Mie siap dihidangkan.

Sekilas yang dilakukan A dan B terlihat sama. Hanya pada urutan ke-2 dan 3 yang berbeda. Akan tetapi kedua langkah ini menghasilkan tujuan yang sama. Yang dilakukan A adalah menuangkan mie matang ke dalam mangkuk yang sudah berisi bumbu, sementara si B menaburkan bumbu setelah mie dituangkan ke dalam mangkuk.

Demikian gambaran sederhana tentang algoritma memasak mie. Terdapat target yang hendak dicapai (masak mie), terdapat input (mie dan bumbu), terdapat sarana (kompor dan panci) dan semuanya menghasilkan output yang sama (mie matang).

1. Algoritma di Pemrograman

Jadi apa itu algoritma? Secara bahasa dapat dikatakan bahwa algoritma adalah sejumlah langkah untuk menyelesaikan suatu pekerjaan. Lalu apa itu algoritma di dunia pemrograman? Di dalam ilmu komputer, algoritma adalah sejumlah langkah bagi komputer untuk menyelesaikan suatu tugas. Banyak solusi bagi permasalahan mulai dari yang sederhana hingga yang kompleks dapat diselesaikan oleh komputer dengan algoritma yang tepat. Tidak seperti algoritma memasak mie di atas yang relatif sederhana, ada banyak algoritma yang sudah ditemukan yang dapat digunakan untuk menyelesaikan suatu tugas dengan optimal. Selain itu dengan menggunakan komputer yang dapat memproses sesuatu dengan lebih cepat dibanding tanpa komputer, banyak tugas yang jenuh bila dilakukan secara manual dapat diselesaikan dengan cepat oleh komputer.

Berikut ini bentuk dasar algoritma:

* Algoritma Sekuensial (Sequence Algorithm)
* Algoritma Perulangan (Looping Algorithm)
* Algoritma Percabangan atau Bersyarat (Conditional Algorithm)

## Pengertian Algoritma Menurut Para Ahli

Agar lebih memahami apa itu algoritma, maka kita dapat merujuk pada pendapat para ahli berikut ini:

1. **Abu Ja’far Muhammad Ibnu Musa Al-Khawarizmi**

Menurut Abu Ja’far Muhammad Ibnu Musa Al-Khawarizmi  (ahli matematika dari Uzbekistan), pengertian algoritma adalah suatu metode khusus yang digunakan untuk menyelesaikan permasalahan.

1. **Donald Ervin Knuth**

Menurut Donald Ervin Knuth, definisi algoritma adalah sekumpulan aturan-aturan berhingga yang memberikan sederetan operasi-operasi untuk menyelesaikan suatu masalah tertentu.

1. **S. E. Goodman dan S.T. Hedetniemi**

Menurut Goodman dan Hedetniemi, pengertian algoritma adalah urutan terbatas dari operasi-operasi yang terdefinisi dengan baik, dimana masing-masing membutuhkan memori dan waktu yang terbatas untuk menyelesaikan suatu masalah.

### **Seymour Lipschutz dan Marc Lipson**

Menurut Seymour Lipschutz dan Marc Lipson (praktisi matematika dan komputer), pengertian algoritma adalah suatu daftar langkah demi langkah yang terhingga dari intruksi-intruksi yang terdefinisikan dengan jelas yang digunakan untuk memecahkan permasalahan tertentu.

1. **Marvin Minsky**

Menurut Marvin Minsky (pakar Artificial Intelligence), pengertian algoritma adalah seperangkat aturan yang memberitahukan kepada kita dari waktu ke waktu, tepatnya bagaimana untuk bertindak.

### **Andrey Andreyevich Markov**

Menurut Andrey Andreyevich Markov (ahli matematika dari Rusia), pengertian algoritma adalah hal umum untuk dipahami sebagai suatu keputusan yang tepat untuk mendefinisikan proses komputasi yang mengarahkan dari data awal hingga hasil yang diinginkan.

## Kriteria Algoritma

## Algoritma memiliki lima ciri utama yang saling berhubungan satu dengan lainnya. Menurut Donald E. Knuth, dapun kriteria algoritma adalah sebagai berikut:

1. **Ada Input**, yaitu permasalahan yang dihadapi dan akan dicarikan solusinya. Algoritma memiliki nol atau lebih input (masukan).
2. **Ada Proses**, yaitu rencana atau langkah-langkah yang harus dilakukan untuk mencapai tujuan akhir.
3. **Ada Output**, yaitu solusi atau tampilan akhir yang didapatkan dari suatu algoritma. Algoritma memiliki minimal satu output.
4. **Ada intruksi-intruksi yang jelas dan tidak ambigu**, yaitu instrukti yang jelas dalam algoritma sehingga tidak terjadi kesalahan dalam menghasilkan output.
5. **Ada tujuan akhir yang dicapai**, yaitu akhir dari program dimana program akan berhenti ketika tujuan akhir telah tercapai.

## Tujuan dan Fungsi Algoritma

Pada dasarnya tujuan dan fungsi utama dari algoritma adalah untuk memecahkan suatu masalah. Lebih jelasnya, adapun tujuan dan fungsi algoritma adalah sebagai berikut:

* Untuk membantu menyederhanakan suatu program yang rumit dan besar.
* Untuk memudahkan dalam membuat sebuah program untuk masalah tertentu.
* Algoritma dapat digunakan berkali-kali untuk menyelesaikan suatu permasalahan.
* Membantu memecahkan suatu permasalahan dengan logika dan sistematis.
* Untuk meminimalisir penulisan program secara berulang-ulang.
* Agar dapat melakukan pendekatan top-down dan divide and conquer.
* Untuk memudahkan membuat program yang lebih rapih dan terstruktur sehingga lebih mudah dipahami dan dikembangkan.
* Memudahkan proses modifikasi pada program karena bisa dilakukan hanya pada satu modul tanpa harus mengubah modiul lainnya.
* Ketika terjadi kesalahan, algoritma dapat membantu menemukannya karena alur kerja yang jelas.
* Memudahkan proses dokumentasi.

## Klasifikasi Algoritma

Algoritma dapat diklasifikasikan berdasarkan implementasinya. Mengacu pada pengertian algoritma di atas, adapun klasifikasi algoritma adalah sebagai berikut:

1. **Rekursi dan Iterasi**; Algoritma rekursi adalah algoritma yang memanggil dirinya sendiri secara berulang-ulang. Sedangkan algoritma iterasi adalah algoritma yang memakai konstrukti berulang dimana terkadang terdapat data tambahan pada struktur yang dibuat.
2. **Logical**; Algoritma logical adalah algoritma yang dapat memposisikan diri seperti logika deduksi yang terkontrol.
3. **Serial, Parallel, atau Terdistribusi**; Algoritma serial adalah algoritma yang menjalankan satu instruksi saja. Algoritam parallel adalah algoritma yang dapat mengerjakan suatu perintah dalam waktu yang sama. Sedangkan algoritma terdistribusi adalah algoritma yang memakai banyak mesin yang terkoneksi dengan jaringan.
4. **Deterministik atau Non-deterministik**; Algoritma deterministik adalah algoritma yang dapat memecahkan suatu masalah dengan keputusan yang tepat. Sedangkan algoritma Non-deterministik adalah algoritma yang memecahkan suatu masalah dengan metode penerkaan.
5. **Tepat atau Perkiraan**; Suatu algoritma mungkin saja memiliki solusi yang tepat, atau setidaknya mempunyai perkiraan yang mendekati solusi yang benar. Dalam merumuskannya dapat dilakukan dengan strategi deterministic ataupun secara acak.
6. **Algoritma Quantum**; Algoritma quantum adalah algoritma yang menggunakan model realistik dari komputasi quantum.

## Contoh Penggunaan Algoritma

Pada dasarnya algoritma tidak hanya dalam ilmu komputer dan matematika saja, tetapi juga pada berbagai kegiatan manusia sehari-hari. Salah satu contohnya adalah ketika seseorang ingin membuat mie instan, maka algoritmanya adalah sebagai berikut:

* Siapkan 1 bungkus mie instan, 400 ml air (2 gelas), panci, mangkuk, sendok dan garpu
* Masukkan air ke dalam panic
* Masak air di atas kompor dengan api sedang
* Biarkan hingga air mendidih
* Masukkan mie instan ke dalam air mendidih
* Aduk perlahan hingga 3 menit
* Jika mie sudah matang masukkan bumbu
* Aduk mie hingga bumbu merata
* Sajikan mie instan

1. Mengapa Algoritma penting

Menurut saya, algoritma itu penting karena merupakan tahap - tahap dalam sebuah proses untuk menyelesaikan segala sesuatu. Sebelum melakukan sesuatu alangkah baiknya kita memikirkan dulu algoritmanya. Setidaknya dengan cara ini dapat meminimalisir logic error.

## Jenis-jenis bentuk Algoritma

Sebagai pendekatan untuk memahami intruksi bahasa pemrograman, maka algoritma mengenal 3 bentuk yaitu:

1. Algoritma Bahasa Natural

2. Algoritma Flow chart

3. Algoritma Pseudocode

### 1. Algoritma Bahasa Natural

Algoritma bahasa natural adalah jenis algoritma dimana penulisan langkah-langkah intruksi menggunakan bahasa yang paling dipahami (bahasa inggris, bahasa indonesia) atau bahasa apapun yang paling dipahami.

Algoritma jenis ini merupakan pendekatan awal untuk memahami algoritma, kasus-kasus atau permasalahan yang bisa digunakan untuk dapat dipecahkan menggunakan algoritma bahasa natural tidak harus kasus-kasus yang dapat diselesaikan secara terkomputerisasi.

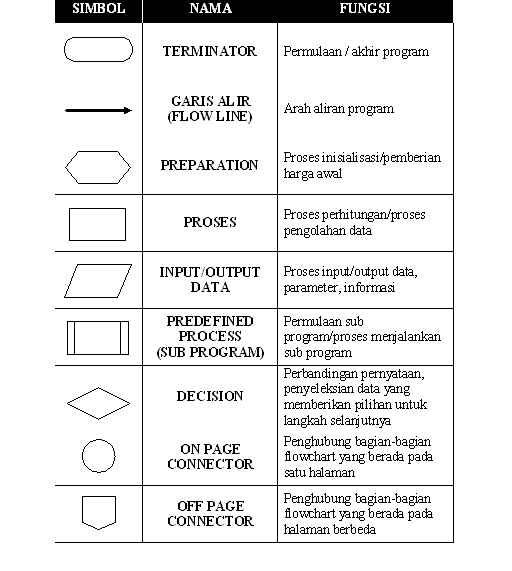
Algoritma bahasa natural hampir dapat menyelesaikan berbagai kasus, kasus-kasus dalam kehidupan sehari-hari, kasus matematika dan perhitungan dan kasus apapun bisa dipecahkan dengan menggunakan algoritma bahasa natural.

Untuk lebih memahami algoritma bahasa natural, anda bisa baca: [Contoh algoritma bahasa natural dalam kehidupan sehari-hari](http://referensisiswa.blogspot.com/2018/07/penulisan-algoritma-bahasa-natural.html)

### 2. Algoritma Flow Chart

Algoritma flow chart merupakan salah satu bentuk penyajian algoritma yang menggunakan bentuk gambar dan simbol yang dapat dibaca dan dipahami secara logis.

Untuk dapat menyajikan pemecahan masalah dalam bentuk algoritma flowchart maka anda dituntut harus memahami terlebih dahulu simbol-simbol flow chart yang digunakan. seperti yang terlihat pada gambar di bawah ini:

[](https://4.bp.blogspot.com/-lIQsOQ5UuDE/W1uOFKrDW9I/AAAAAAAACAM/7iL3coLDl_sFdarXBzI7MdzWMGPQNfYNgCLcBGAs/s1600/flowchart.gif)

Untuk lebih memahami bagaimana pemecahan masalah menggunakan algoritma flowchart anda bisa baca :

* [contoh algoritma flowchart dalam kehidupan sehari-hari](http://referensisiswa.blogspot.com/2018/08/algoritma-dalam-kehidupan-sehari-hari.html)
* [5 contoh Algoritma flowchart dalam kehidupan sehari-hari](http://referensisiswa.blogspot.com/2018/08/5-contoh-algoritma-flowchart-sehari-hari.html)

### 3. Algoritma Pseudocode

Penyajian algoritma yang ke tiga adalah algoritma pesudocode, apa itu algoritma pseudocode ?

Secara sederhana algoritma pseudocode adalah penulisan langkah-langkah penyelesaian masalah menggunakan  pendekatan instruksi bahasa pemrograman, dengan tujuan agar lebih mudah dipahami ketika instruksi tersebut ditulis atau dikonversi kedalam bahasa pemrograman sebenarnya.

Penulisan algoritma pseudocode sebenarnya tidak mengenal aturan syntax (cara penulisan intruksi/kode) khusus, namun syntax atau instruksi yang digunakan biasanya disesuaikan dengan bahasa pemorgraman yang nanti akan digunakan.

Seperti misalnya jika algoritma nanti akan dikonvesi ke bahasa pemrograman pascal, maka instruksi algoritma pseudocode akan menggunakan syntax yang hampir mirip dengan kode bahasa pemrograman pascal, begitu juga ketika bahasa pemrograman yang akan digunakan menggunakan bahasa C, maka instruksi algoritma peudocode akan menggunakan pendekatan bahasa C, begitu juga untuk bahasa pemrograman lainnya.

**Aturan penulisan algoritma pseudocode**

Berikut adalah aturan penulisan algoritma pseudocode dengan pendekatan bahasa pemrograman pascal:

1. Struktur Algoritma pseudocode

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 {....}

#### 2. Instuksi / Syntax penulisan algoritma psedeocode

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.

Contoh:

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.

ALGORITMA hitung\_luas\_segitiga

{Algorima pseudocode untuk menghitung luas segitiga}

Deklarasi:

{Deklarasi variable yang digunakan diproses perhitungan}

hasil,alas,tinggi : Integer

Deskripsi :

{Input dari pengguna disimpan divariable alas dan tinggi}

read(alas,tinggi}

{perhitungan luas (1/2 alas x tinggi ) disimpan di variable hasil}

hasil <- 1/2 \* alas \* tinggi

{Menampilkan hasil perhitungan}

write(hasil)

## Konsep memahami algoritma?

Untuk lebih mudah memahami algoritma maka anda bisa memahami konsep dasarnya terlebih dahulu, konsep dasar yang banyak dijadikan contoh untuk memahami algoritma adalah pada kasus menukarkan 2 buah isi gelas.

jadi ada 2 gelas, gelas A berisi Air Susu dan gelas B berisi Air Kopi. bagaimana langkahnya agar Gelas A isinya menjadi Kopi dan gelas B isinya menjadi air susu.

Ketika kita dihadapkan pada contoh kasus di atas, otak kita akan mencoba untuk menemukan cara/ logikanya.

berarti kita harus menyediakan 1 gelas Lagi yaitu gelas kosong sebut saja gelas C. dengan adanya gelas kosong tersebut, maka kita bisa menukarkan kedua isi gelas tersebut.

Semoga apa yang saya uraikan di atas membantu anda dalam memahami algoritma, strukturnya dan juga bisa memecahkan permasalahan menggunakan algoritma bahasa natural, algoritma flowchart dan algoritma pseudocode.

**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](https://dosenit.com/kuliah-it/pemrograman/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;  
}

Setelah Anda mempelajari algoritma percabangan dalam artikel ini, Anda selanjutnya dapat mengembangkannya ke dalam bentuk flowchart. Apa [fungsi flowchart dalam pemrograman](https://dosenit.com/kuliah-it/pemrograman/fungsi-flowchart-dalam-pemrograman)? Simak artikelnya di website kami ini. Sekian artikel kami kali ini mengenai algoritma percabangan.

Eitss, Anda juga harus mengerti dahulu [pengertian algoritma, flowchart, dan pseudocode](https://dosenit.com/kuliah-it/sistem-informasi/pengertian-algoritma-flowchart-dan-pseudocode) dalam mempelajari pemrograman. Anda juga bisa mengetahui semuanya di website kami ini. Semoga artikel kami ini bermanfaat bagi Anda untuk mengenal algoritma percabangan.

## latihan

1. Dibaca waktu tempuh seorang pelari marathon dalam jam-menit-detik (hh:mm:ss). Diminta mengkonversi waktu tempuh tersebut ke dalam detik. Tuliskan algoritmanya.

Ingatlah

1 menit = 60 detik

1 jam = 3600 detik

Misalnya waktu tempuh seorang pelari marathon adalah 1 jam, 5 menit, 40 detik. Dalam detik, waktu tempuh seluruhnya adalah ( 1 x 3600 ) + ( 5 x 60 ) + 40 = 3940 detik.

Penyelesaian :

Algoritma KONVERSI\_JAM\_KE\_DETIK

{ dibaca jam-menit-detik (hh:mm:ss). Nilai jam-menit-detik dikonversi ke dalam detik, lalu ditampilkan ke piranti keluaran }

DEKLARASI

Type jam : record <hh : integer {0..23}, {jam}

mm : integer {0..59}, {menit}

ss : integer {0..59}, {detik}

>

J : jam

TotalDetik : integer

DESKRIPSI

read(J.hh,J.mm,J.ss))

TotalDetik ← (J.hh\*3600) + (J.mm\*60) + J.ss

write(TotalDetik)

Jika anda mentranslasikan algoritma KONVERSI\_JAM\_KE\_DETIK ke dalam bahasa pascal, anda harus memperhatikan tipe bilangan bulat yang digunakan. Karena ranah nilai tipe integer terbatas, maka ada kemungkinan hasil pengubahan jam-menit-detik ke total detik bernilai negatif, sebab nilai (J.hh\*3600) + (J.mm\*60) + J.ss berada di luar rentang tipe integer. Tipe longint yang mempunyai ranah yang lebih besar dapat dipakai untuk masalah ini.

Jadi, program KONVERSI\_JAM\_KE\_DETIK dalam bahasa pascal adalah sebagai berikut :

program KONVERSI\_JAM\_KE\_DETIK;

{ dibaca jam-menit-detik (hh:mm:ss). Nilai jam-menit-detik dikonversi ke dalam detik, lalu ditampilkan ke piranti keluaran.}

uses wincrt;

(\* DEKLARASI \*)

type Jam = record

hh : longint; {jam}

mm : longint; {menit}

ss : longint; {detik}

end;

var

J : Jam;

TotalDetik : longint;

(\* deskripsi \*)

begin

write(‘Jam :’); readln(J.hh);

write(‘Menit:’); readln(J.mm);

write(‘Detik:’); readln(J.ss);

TotalDetik:= (J.hh\*3600) + (J.mm\*60) + J.ss;

writeln(‘Total detik = ‘, TotalDetik);

end.