

# **PANDUAN BELAJAR PEMROGRAMAN TERSTRUKTUR**

**ALGORITMA DAN  
PEMROGRAMAN C++**

Muhammad Sholeh

AKPRIND PRESS

Undang- Undang Nomor 7 Tahun 1987  
Tentang Hak Cipta  
Pasal 44

- (1) Barang siapa dengan sengaja mengumumkan atau memperbanyak suatu ciptaan atau memberi ijin untuk itu, dipidana dengan pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp. 100.000.000,00 (seratus juta rupiah).
- (2) Barang siapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta sebagaimana di maksud dalam ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp. 50.000.000,00 (lima puluh lima rupiah)

## **PANDUAN BELAJAR PEMROGRAMAN TERSTRUKTUR AGORITMA DAN PEMROGRAMAN C++**

Hak cipta 2013 pada penulis, dilarang keras mengutip, menjiplak,  
Memphoto copy baik sebagian atau keseluruhan isi buku ini  
Tanpa mendapat izin tertulis dari pengarang dan penerbit

Penulis : Muhammad Sholeh  
Page Make Up : Rochmat Haryanto  
Desain Cover : Rochmat Haryanto  
Dicetak Oleh : AKPRIND PRESS

ISBN : 978-602-7619-19-7

HAK CIPTA DILINDUNGI OLEH UNDANG-UNDANG

## KATA PENGANTAR

Banyak aplikasi komputer yang saat ini sudah digunakan untuk membantu manusia dalam kehidupan sehari-hari. Mulai dari aktivitas di Kelurahan dalam proses pengurusan KTP (e-KTP), sistem perbankan (ATM. Teller), supermarket, pengurusan surat ijin mengemudi (e-SIM) dan lainnya. Demikian juga kehidupan sosial, saat ini sudah tidak asing lagi dengan Facebook, Twitter, email.

Peran teknologi informasi komputer (TIK) sudah masuk di semua sektor kehidupan manusia. TIK tidak hanya sekedar seperangkat komputer tetapi juga harus didukung dengan perangkat lunak. Dukungan programmer dalam mengembangkan aplikasi sangat berperan.

Agar dapat menjadi programmer yang handal, perlu adanya pemahaman logika dan bahasa pemrograman. Pemahaman logika sangat diperlukan dalam pengembangan aplikasi. Apapun bahasa pemrograman yang digunakan, logika menjadi kunci utama dalam memindahkan persoalan ke dalam bentuk program.

Dalam buku ini, akan dibahas bagaimana memahami logika dan mengimplementasikan ke dalam bahasa C++. Pembahasan dikemas dengan menekankan pada contoh soal dan penyelesaian serta contoh-contoh bahasa C++ dengan penjelasan. Dengan cara ini dia diharapkan pembaca dapat memahami logika serta bagaimana aturan serta penyusunannya dalam bahasa C++.

Disamping materi memahami logika dan algoritma, buku ini juga mengupas dasar-dasar pemrograman C++, mulai dari dasar-dasar pemrograman, variabel, casting, pengulangan, perulangan, array dan structure.

Akhir kata, semoga buku ini dapat memberikan manfaat. Tak lupa penulis meminta masukan, saran dan kritik. Saran dapat dikirim ke muhash@akprind.ac.id

Yogyakarta, Maret 2013

Penulis



## DAFTAR ISI

Halaman Depan .....	i
Kata Pengantar .....	iii
Daftar isi .....	v

<b>Seluk Beluk Bahasa Pemrograman .....</b>	<b>1</b>
1.1. Pendahuluan .....	3
1.2. Apa itu program .....	4
1.3. Bahasa Pemrograman .....	5
1.4. Proses Kompilasi.....	6
1.5. Macam-macam Bahasa Pemrograman.....	7
1.6. Langkah-langkah Menyusun Program .....	9
1.7. Sejarah Bahasa C++ .....	13
1.8. Mengenal editor Bahasa C++.....	13
1.9. Proses instalasi C++.....	14
1.10. Membuat program C++.....	15
1.11. Memahami Kesalahan Aturan Sintaks Pada Bahasa C .....	16
1.12. Latihan .....	18
 <b>Struktur Bahasa C++ .....</b>	 <b>19</b>
2.1. Struktur Bahasa C++ .....	21
2.2. Perintah-perintah dasar .....	24
2.3. Latihan .....	30
 <b>Variabel dan, Tipe Data.....</b>	 <b>31</b>
3.1. Variabel .....	33
3.2. Penamaan Variabel.....	34
3.3. Kata kunci.....	35
3.4. Tipe Data .....	36
3.5. Deklarasi variabel .....	37
3.6. Menentukan tipe variabel .....	37
3.7. Memberikan Nilai ke variabel .....	38
3.8. Menampilkan isi variabel di monitor .....	39
3.9. Konstanta .....	41
3.10 Overflow Data .....	42
3.11 Konversi Tipe Data .....	44
3.12 Tipe Casting .....	46
3.13 Contoh kesalahan .....	50
3.14 Latihan .....	56
 <b>Operator dan Ungkapan.....</b>	 <b>57</b>
4.1. Operator penugasan .....	59
4.2. Operator aritmatika .....	60
4.3. Operator Penaik dan Penurun di c++ .....	62

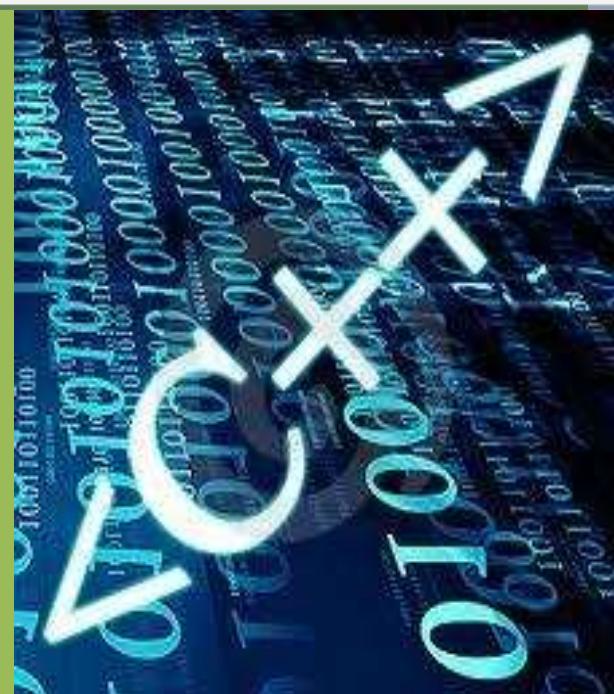
4.4. Operator Relasi .....	65
4.5. Operator Majemuk .....	66
4.6. Urutan Operasi.....	66
4.7. Latihan.....	70
<b>Perintah Masukan dan Keluaran.....</b>	<b>71</b>
5.1. Perintah Keluaran.....	73
5.2. Perintah Masukan.....	77
5.3. Fungsi Manipulator .....	79
5.4. Fungsi Matematika .....	85
5.5. String .....	88
5.6. Fungsi String .....	92
5.7. Latihan .....	100
<b>Operasi Penyeleksian Kondisi .....</b>	<b>103</b>
6.1. Pernyataan IF .....	110
6.2. Pernyataan IF - ELSE.....	110
6.3. Memahami keputusan dengan kondisi jamak .....	118
6.4. Urutan Operasi .....	122
6.5 . Pernyataan IF berkala...	124
6.6. Latihan .....	133
6.7. Pernyataan switch - case .....	135
6.8. Latihan.....	138
<b>Perulangan .....</b>	<b>139</b>
7.1. Memahami perintah for .....	141
7.2. Latihan.....	150
7.3. Memahami for di dalam for (kalang for) .....	151
7.4. Latihan.....	153
7.5. Memahami perintah while .....	154
7.6. Memahami perintah do - while .....	157
7.7. Pernyataan break .....	159
7.8. Pernyataan continue .....	161
7.9. Latihan .....	164
<b>Fungsi.....</b>	<b>165</b>
8.1. Memahami Fungsi.....	168
8.2. Fungsi Tanpa Nilai Balik .....	170
8.3. Fungsi dengan Nilai Parameter .....	171
8.4. Fungsi dengan Nilai Balik .....	175
8.5. Lingkup variabel .....	178
8.6. Latihan .....	187

<b>Array .....</b>	<b>189</b>
9.1. Array Dimensi Satu .....	191
9.2. Array Dimensi Dua.....	197
9.3. MelewatkAn Array sebagai parameter dalam suatu Fungsi .....	203
9.4 Latihan .....	208
 <b>Structure.....</b>	 <b>209</b>
10.1. Stucture .....	211
10.2. Stucture dengan Array.....	213
10.3. Latihan .....	215
 <b>Pointer .....</b>	 <b>217</b>
11.1. Pengertian Pointer .....	219
11.2. Mendefiniskan dan Mengisi variabel pointer .....	219
11.3. Pointer void .....	223
11.4. Mengubah isi variabel lewat pointer .....	225
11.5. Pointer dan fungsi .....	227
11.6. Latihan .....	230
 <b>INDEKS .....</b>	 231
<b>GLOSARIUM.....</b>	<b>233</b>
<b>DAFTAR PUSTAKA .....</b>	<b>235</b>



1

# SELUK BELUK BAHASA PEMROGRAMAN



## Tujuan Instruksional

Setelah membaca bab ini, diharapkan pembaca memahami bagaimana proses program bekerja mulai dari proses penyusunan program sampai proses kompilasi serta memahami sejarah bahasa C++, sejarah C++ serta proses pembuatan program dengan menggunakan bahasa C++ serta memahami editor turbo c untuk membuat program..

## Materi



Pengertian Program



Macam-macam Bahasa Pemrograman



Sejarah Bahasa C++



Proses instalasi C++

### 1.1. Pendahuluan

Penggunaan komputer saat ini sudah menjadi sesuatu yang biasa, permainan game, Facebook, aplikasi perkantoran (Microsoft office, open office) merupakan contoh sederhana pemakaian yang sering digunakan. Penggunaan game, Facebook tidak hanya anak muda tetapi kalangan orang tua pun sudah familiar dengan game atau pun permainan lainnya. Nah ..yang menjadi pertanyaan .bagaimana semua permainan yang ada di komputer dibuat?. Sebagai pengguna, tinggal buka komputer, klik permainan dan .....langsung asyik bermain. Bagaimana proses dibalik pembuatan permainan tersebut.

Apa yang dimainkan tersebut, tentunya membutuhkan proses untuk membuat atau mengembangkannya. Bahan utama dalam pembuatan permainan tersebut adalah PROGRAM.



Gambar 1.1 Aplikasi Microsoft Office  
[topspot-official.blogspot.com](http://topspot-official.blogspot.com)



Gambar 1.2 Aplikasi media sosial Facebook  
[www.facebook.com](http://www.facebook.com)



Gambar 1.3. Aplikasi untuk proses pembelajaran  
elearning.akprind.ac.id

## 1.2. Apa itu program

Pengertian program di sini adalah program komputer, karena jika hanya mengambil istilah program saja nanti menjadi rancu dengan istilah-istilah program yang lain, misal program akademik, program percepatan dan lainnya.

Pemrograman komputer merupakan suatu proses iteratif penulisan dan penyuntingan kode sumber sehingga membentuk sebuah program. Penyuntingan kode sumber meliputi proses pengetesan, analisis, pembetulan kesalahan, algoritma, normalisasi kode, dan kadang-kadang mengoordinasikan antara satu programmer dengan programmer lainnya jika sebuah program dikerjakan oleh beberapa orang dalam sebuah tim. Seorang praktisi yang memiliki keahlian untuk melakukan penulisan kode dalam bahasa pemrograman disebut sebagai programmer komputer atau programmer, pengembang perangkat lunak, Istilah **rekayasa perangkat lunak** (bahasa Inggris: *Software engineering*) seringkali digunakan karena proses penulisan program tersebut dipandang sebagai suatu disiplin ilmu perekayasaan. (wikipedia, 2013 [http://id.wikipedia.org/wiki/Program\\_komputer](http://id.wikipedia.org/wiki/Program_komputer)). Sedangkan software yang digunakan untuk membuat program sering disebut bahasa pemrograman,

Bagaimana dengan pengertian di atas, mumet ....pusing ...sabar nanti dengan ketekunan saya yakin ...kalian akan bisa membuat program walaupun masih sederhana .....

Jadi .... Komputer sebenarnya tidak bisa berbuat apa-apa tanpa adanya program. Demikian juga program tanpa komputer juga akan lumpuh ...sama-sama saling membutuhkan dan saling melengkapi..... Mirip dengan komputer yang banyak jenis dan merek nya, program juga mempunyai karakteristik yang bermacam-macam dan beraneka ragam jenisnya..

Bahasa pemrograman dipakai sejak komputer generasi pertama yaitu bahasa mesin atau pada komputer yang memaki bahasa biner. Nah selanjutnya, apa saja sih yang termasuk kedalam bahasa pemrograman?.

Hampir sama dengan pengertian bahasa manusia, bahasa pemrograman juga mempunyai jenis dan karakteristiknya yang sangat banyak. Dengan menggunakan bahasa pemrograman ini,

pengguna dapat melakukan proses ‘komunikasi’ dengan komputer. Dengan bahasa pemrograman ini memungkinkan seorang programmer dapat menentukan secara persis data mana yang akan diolah oleh komputer, bagaimana data ini akan disimpan/diteruskan, dan jenis langkah apa secara persis yang akan diambil dalam berbagai situasi. Jadi kalau misal kasir mengetikkan daftar barang yang dibeli, kemudian menghasilkan laporan total belanya dan seterusnya ....itu semua bisa terjadi karena programmer sudah membuat kode (dengan bahasa pemrograman tertentu) alur bagaimana proses program kasir tersebut bisa berjalan. Jadi program ATM, e-KTP, e-SIM merupakan hasil nyata dari suatu program komputer.



Gambar 1.4 Aplikasi ATM  
[agungmulyawan.blogspot.com](http://agungmulyawan.blogspot.com)

### 1.3. Bahasa Pemrograman

Bahasa pemrograman biasanya didasarkan menjadi 2 hal, yaitu :

1. Bahasa pemrograman beraras rendah (low level languages)

Bahasa ini cenderung berorientasi kepada mesin, yaitu bila memakai memberikan suatu perintah lebih banyak langsung ke dalam perintah yang sudah langsung dimengerti komputer. Yang termasuk dalam bahasa ini adalah :

- Bahasa mesin
- Bahasa rakitan

2. Bahasa pemrograman beraras tinggi (high level languages)

Dengan bahasa ini, seorang memakai tidak lagi harus menerjemahkan sendiri ke dalam bahasa yang dimengerti komputer, pemakai cukup memberikan suatu perintah (biasanya dalam bahasa Inggris) yang sudah dimengerti komputer. Bahasa inilah yang nantinya akan melakukan penerjemahan ke dalam bahasa yang dimengerti oleh komputer.

Bahasa-bahasa yang beraras tinggi, antara lain :

- Pascal
- Qbasic
- Cobol

Seperti diulas diatas, suatu bahas pemrograman yang beraras tinggi selalu melakukan proses penerjemahan, proses penerjemahan ini mengubah perintah yang diberikan (dalam bahasa inggris) ke dalam bahasa yang dimengerti oleh komputer ( 0 dan 1). Proses penerjemahan ini dapat dilaksanakan oleh

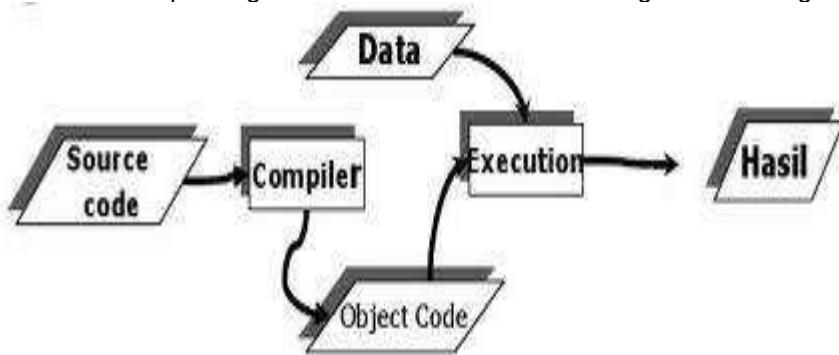
- Interpreter
- Kompiler

Perbedaan kedua penerjemah dapat diterangkan sbb:

Interpreter	Kompiler
<ul style="list-style-type: none"> <li>☞ Kesalahan kaidah akan segera diketahui saat program dijalankan</li> </ul>	<ul style="list-style-type: none"> <li>☞ Kesalahan program terdeteksi sewaktu kompilasi. Program harus terhindarkan dari segala kesalahan sewaktu program dijalankan</li> </ul>
<ul style="list-style-type: none"> <li>☞ Program tidak harus dijadikan file executabl / program langsung dapat dijalankan dari prompt</li> </ul>	<ul style="list-style-type: none"> <li>☞ Program dapat dikompilasi menjadi file executable</li> </ul>
<ul style="list-style-type: none"> <li>☞ Kecepatan eksekusi relatif pelan</li> </ul>	<ul style="list-style-type: none"> <li>☞ Kecepatan eksekusi tinggi</li> </ul>

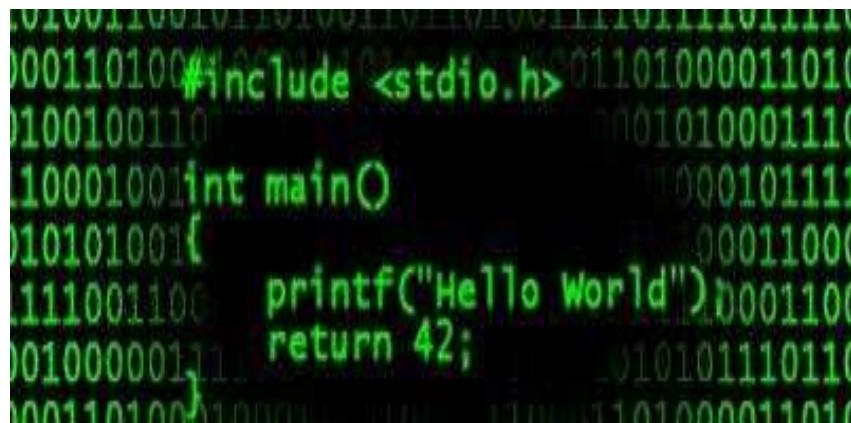
#### 1.4. Proses Kompilasi

Ada perbedaan antara bahasa manusia sehari-hari dengan bahasa komputer (bahasa mesin), manusia lebih suka dengan kata dalam bentuk abjad atau angka 1-9, sementara komputer hanya mengerti 1 dan 0. Nah .....dalam proses membuat program, programmer akan merasa sudah (apalagi bagi pemula) jika untuk membuat program harus menggunakan bahasa mesin (1 dan 0). Hal ini tidak menjadi masalah, para programmer dalam membuat program cukup mengetikkan perintah-perintah dalam bentuk kata dan proses untuk mengubah ke bahasa mesin akan diserahkan ke bahasa pemrograman tersebut. Proses ini sering disebut dengan compiler.



Gambar 1.5 Proses Kompilasi

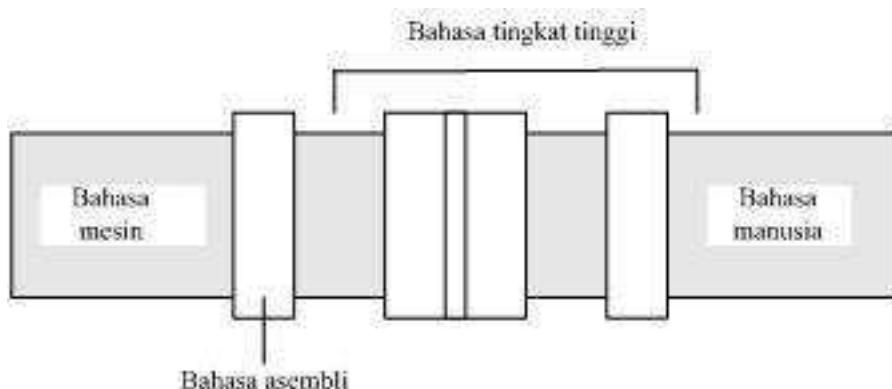
<http://rinacute14.blogspot.com/2012/10/compiler-dengan-interpreter-lagi-deh.html>



Gambar 1.6 Bahasa Pemrograman dan Bahasa Mesin  
<http://imanmauludin.wordpress.com/2012/10/01/bahasa-pemrograman-delphi/>

Menurut tingkat kedekatannya dengan mesin komputer, bahasa pemrograman terdiri dari:

1. Bahasa Mesin, yaitu memberikan perintah kepada komputer dengan memakai kode bahasa biner, contohnya 01100101100110
2. Bahasa Tingkat Rendah, atau dikenal dengan istilah bahasa rakitan (*Assembly*), yaitu memberikan perintah kepada komputer dengan memakai kode-kode singkat (kode *mnemonic*), contohnya MOV, SUB, CMP, JMP, JGE, JL, LOOP, dsb.
3. Bahasa Tingkat Menengah, yaitu bahasa komputer yang memakai campuran instruksi dalam kata-kata bahasa manusia (lihat contoh Bahasa Tingkat Tinggi di bawah) dan instruksi yang bersifat simbolik, contohnya {}, ?, <<, >>, &&, ||, dsb.
4. Bahasa Tingkat Tinggi, yaitu bahasa komputer yang memakai instruksi berasal dari unsur kata-kata bahasa manusia, contohnya begin, end, if, for, while, and, or, dsb.

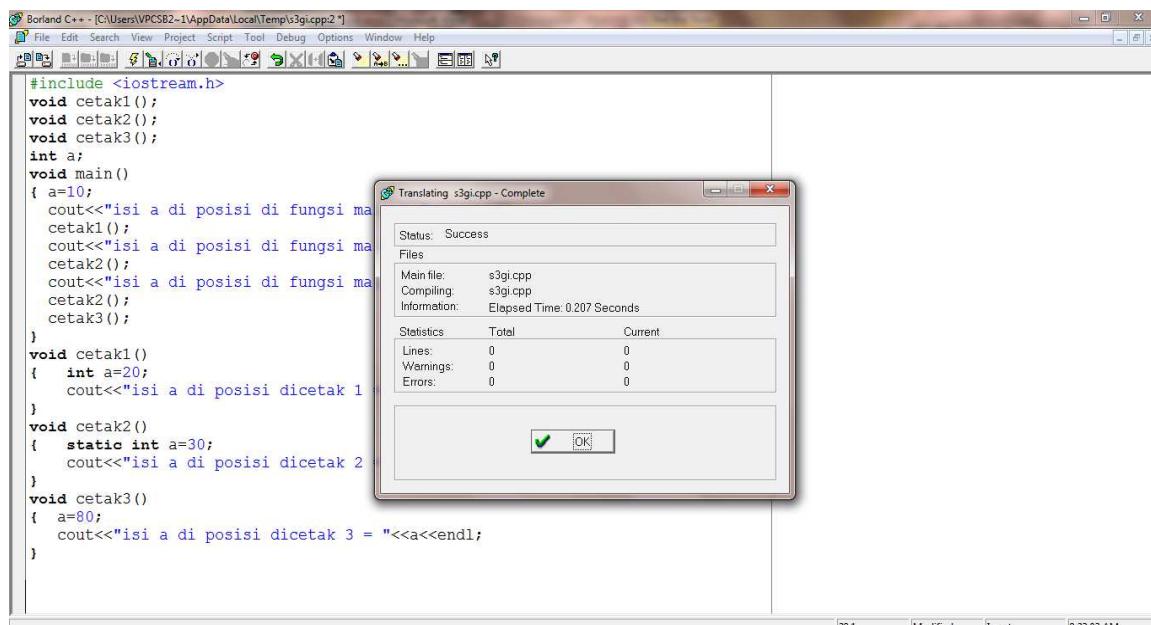


Gambar 1.7 Tingkatan Bahasa Pemrograman  
<http://desylvia.wordpress.com/2010/09/06/pemrograman-bahasa-c-pendahuluan/>

### 1.5. Macam-macam Bahasa Pemrograman

- o Bahasa pemrograman C (kategori tingkat tinggi)
- o Bahasa pemrograman JAVA (kategori tingkat tinggi)
- o Bahasa pemrograman PYTHON
- o Bahasa pemrograman SQL (kategori tingkat tinggi)

- o Bahasa pemrograman PHP (kategori tingkat tinggi)
- o Bahasa pemrograman HTML (kategori tingkat tinggi)
- o Bahasa pemrograman COBOL
- o Bahasa pemrograman MICROSOFT VISUAL BASIC (kategori tingkat tinggi)
- o Bahasa pemrograman DELPHI (kategori tingkat tinggi)
- o Bahasa pemrograman C++ (kategori tingkat tinggi)
- o Bahasa Pemrograman ASP
- o Bahasa Pemrograman PERL
- o Bahasa Pemrograman Javascript (kategori tingkat tinggi)



The screenshot shows the Borland C++ IDE interface. On the left is the code editor window containing the following C++ code:

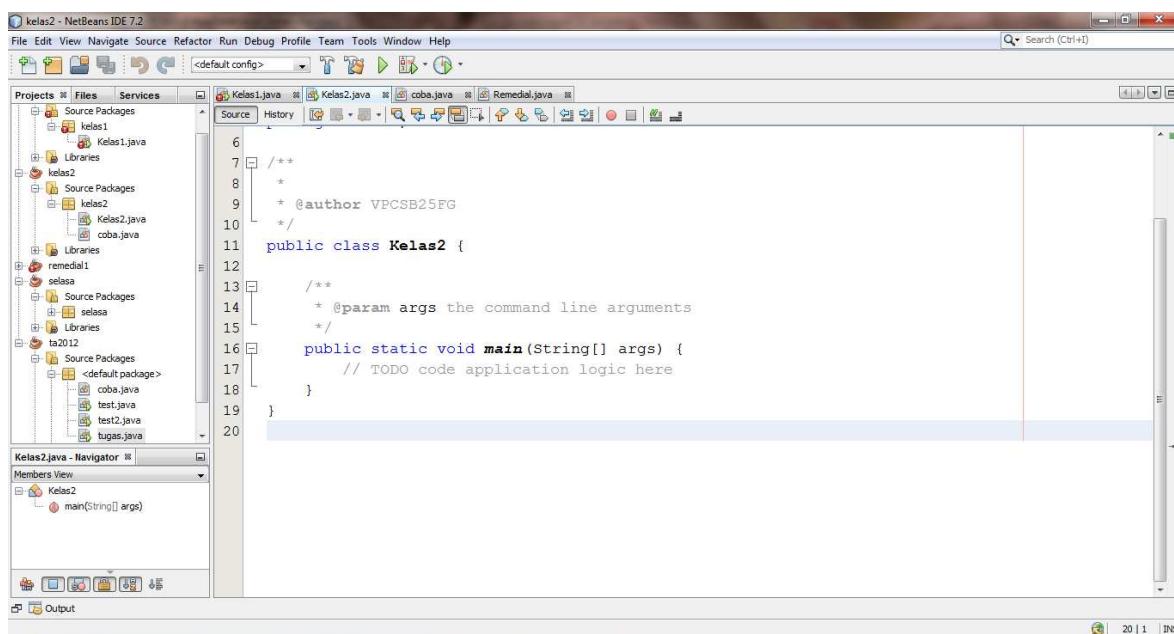
```
#include <iostream.h>
void cetak1();
void cetak2();
void cetak3();
int a;
void main()
{
    a=10;
    cout<<"isi a di posisi di fungsi main";
    cetak1();
    cout<<"isi a di posisi di fungsi main";
    cetak2();
    cout<<"isi a di posisi di fungsi main";
    cetak2();
    cetak3();
}
void cetak1()
{
    int a=20;
    cout<<"isi a di posisi dicetak 1";
}
void cetak2()
{
    static int a=30;
    cout<<"isi a di posisi dicetak 2";
}
void cetak3()
{
    a=80;
    cout<<"isi a di posisi dicetak 3 = "<<a<<endl;
}
```

A modal dialog box titled "Translating s3gi.cpp - Complete" is displayed in the center. It shows the following information:

Status:	Success	
Files:		
Main file:	s3gi.cpp	
Compiling:	s3gi.cpp	
Information:	Elapsed Time: 0.207 Seconds	
Statistics	Total	Current
Lines:	0	0
Warnings:	0	0
Errors:	0	0

At the bottom right of the dialog box are two buttons: a green checkmark icon labeled "OK" and a standard "OK" button.

Gambar 1.8 Tampilan Bahasa C



The screenshot shows the NetBeans IDE 7.2 interface. The left side features the "Projects" panel with several Java projects listed under "Source Packages". The central area is the code editor showing the following Java code:

```
6
7  /*
8   * @author VPCSB25FG
9   */
10 public class Kelas2 {
11
12     /**
13      * @param args the command line arguments
14     */
15     public static void main(String[] args) {
16         // TODO code application logic here
17     }
18 }
19 }
```

The code editor has line numbers on the left. The bottom left corner shows the "Navigator" and "Members View" panes, which are currently empty for this specific class. The bottom right corner shows the status bar with "20 | 1 | INS".

Gambar 1.9 Tampilan Bahasa Java (editor Netbeans)

C:\xampp\htdocs\indstri\counter.php - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

index.php petente.php metadname.php session.php embidata2.php loginct.php koneksi.php login.php loginact.php petadname.php autonim.inf counter.php

```
1 <?php
2     include "conf/fungsi_intodgl.php";
3     include "conf/fungsi_autolink.php";
4     include "conf/fungsi_combobox.php";
5     include "conf/library.php";
6     include "conf/fungsi_kalender.php";
7     include "conf/koneksi.php";
8
9         $ip = $_SERVER['REMOTE_ADDR']; // Mendapatkan IP komputer user
10        $tanggal = date("YmD"); // Mendapatkan tanggal sekarang
11        $waktu = time(); //
12
13        // Mencek berdasarkan IPnya, apakah user sudah pernah mengakses hari ini
14        $s = mysql_query("SELECT * FROM statistik WHERE ip='$ip' AND tanggal='$tanggal'");
15        // Kalau belum ada, simpan data user tersebut ke database
16        if(mysql_num_rows($s) == 0){
17            mysql_query("INSERT INTO statistik(ip, tanggal, hits, online) VALUES('$ip','$tanggal','1','$waktu')");
18        }
19        else{
20            mysql_query("UPDATE statistik SET hits=hits+1, online='$waktu' WHERE ip='$ip' AND tanggal='$tanggal'");
21        }
22
23        $pengunjung = mysql_num_rows(mysql_query("SELECT * FROM statistik WHERE tanggal='$tanggal' GROUP BY ip"));
24        $totalpengunjung = mysql_result(mysql_query("SELECT COUNT(hits) FROM statistik"), 0);
25        $hits = mysql_fetch_assoc(mysql_query("SELECT SUM(hits) as hitsday FROM statistik WHERE tanggal='".date('YmD')."' GROUP BY tanggal"));
26        $totalhits = mysql_result(mysql_query("SELECT SUM(hits) FROM statistik"), 0);
27        $bataswaktu = time() - 300;
28        $pengunjunganonline = mysql_num_rows(mysql_query("SELECT * FROM statistik WHERE online > '$bataswaktu'"));
29
30
31        $ext = ".png";
32
33        ?>
```

Gambar 1.10 Bahasa PHP (editor Notepad++)

## **1.6. Langkah-langkah Menyusun Program**

Hal pertama yang perlu diperhatikan dalam menyusun sebuah program adalah bagaimana algoritma dari program tersebut.. Dengan sebuah algoritma tentunya penyusunan sebuah program akan lebih mudah. Dalam memahami sebuah algoritma juga harus dibarengi dengan logika-logika yang akan membantu dalam pemahaman tersebut.

## Logika

Logika, dalam bahasa asing adalah *logic - science or method of reasoning* - yaitu suatu metode untuk mengambil keputusan dari suatu fakta atau pernyataan. Seringkali logika diselaraskan dengan nalar, yaitu proses berfikir manusia untuk menghubungkan fakta atau pernyataan sehingga sampai pada suatu kesimpulan. Data, fakta ataupun pernyataan yang akan dinalar boleh salah boleh benar. Di sinilah letak kerja penalaran.

## **Algoritma dan Pemrograman**

Menurut batasan umum, algoritma adalah : ***Sekumpulan langkah-langkah atau instruksi-instruksi yang terbatas untuk menyelesaikan suatu masalah.*** Asal kata algoritma - terjemahan dari istilah asing *algorithm* - diambil dari sebuah nama seorang astronom dan ahli matematika bangsa Arab, yaitu Al-Khowarizmi yang mengarang kitab Al-Jabr W'al Muqabala, dikenal sekarang sebagai ilmu aljabar. Semula, langkah-langkah seperti disebutkan pada batasan diatas dikaitkan dengan langkah-langkah untuk penyelesaian masalah numerik, tetapi pada akhirnya digunakan pada hal-hal yang lebih umum.

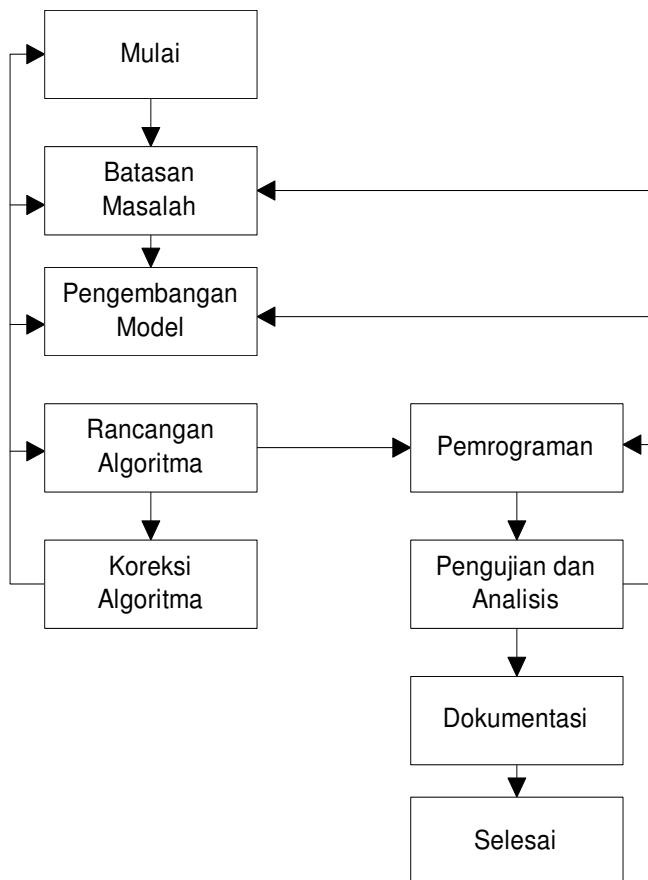
Sekumpulan langkah-langkah untuk penyelesaian suatu masalah dengan bantuan komputer disebut program, sedangkan proses membuat program disebut pemrograman atau dalam istilah bahasa asing disebut : *programming*, kemudian bahasa untuk menuliskan langkah-langkah dalam bentuk perintah dan pernyataan (*statement*) disebut bahasa pemrograman atau dalam istilah bahasa asing disebut : *programming language*.

### Tahap-tahap Pemrograman

Tahap-tahap yang diperlukan untuk menyelesaikan suatu masalah dengan komputer, dari saat masalah tersebut diberikan hingga akhir penyelesaian adalah :

1. Menentukan batasan masalah
2. Pengembangan Model
3. Rancangan Algoritma
4. Koreksi Algoritma
5. Pemrograman
6. Pengujian dan Analisis Program
7. Dokumentasi

Tahap-tahap tersebut diatas diantaranya ada yang memerlukan tinjau ulang seperti ditunjukkan dalam gambar berikut



Gambar 1.11 Tahap-tahap Algoritma dan Pemrograman

### Pembatasan Masalah

Seringkali masalah yang dihadapi masih belum begitu jelas, batasan masih kabur dan pada beberapa hal, inti permasalahan masih kabur. Untuk itulah diperlukan pemahaman atas masalah yang dihadapi, sehingga pembatasan masalah dapat dibuat lebih jelas, demikian juga ruang lingkup permasalahan bisa ditentukan dengan tegas. Namun demikian, pada tahap ini perlu difikirkan hal-hal yang dapat menampung perkembangan masalah di kemudian hari. Dapat

dikatakan bahwa pemahaman yang baik terhadap suatu permasalahan berarti setengah dari pekerjaan untuk menyelesaikan masalah tersebut telah dilalui.

### **Pengembangan Model**

Model adalah suatu sistem yang secara fisis ataupun matematis mengikuti kondisi-kondisi yang telah ditetapkan dalam sistem aslinya, dimana sifat yang diperoleh dari model tersebut dapat dipergunakan untuk memahami sistem asli tanpa harus berurusan secara langsung dengan sistem aslinya. Dengan mengembangkan model atas suatu permasalahan, maka diharapkan pemecahan masalah tersebut dapat dilaksanakan secara lebih sistematis.

### **Rancangan Algoritma**

Setelah suatu masalah dipahami dengan jelas yaitu melalui pembatasan masalah, kemudian model sudah dibuat, maka tahap selanjutnya adalah menyusun langkah-langkah untuk penyelesaian masalah, kebenaran langkah-langkah tersebut secara logika harus dapat ditelusuri. Walaupun langkah-langkah yang dibuat dalam rancangan algoritma ini masih merupakan langkah-langkah yang tidak terlalu rinci, tetapi secara keseluruhan ia harus dapat mengambarkan tahapan penyelesaian dengan jelas, untuk itulah dikembangkan teknik algoritma yang akan dijelaskan pada bab mendatang.

### **Koreksi Algoritma**

Koreksi dilakukan secara dini untuk memperbaiki alur logika yang dianggap kurang tepat, karena langkah-langkah pada rancangan algoritma belum melibatkan hal-hal yang terlalu rinci maka koreksi pada tahap ini lebih mudah dilaksanakan. Bila ada kerancuan dalam algoritma, maka yang perlu ditinjau ulang adalah model, besar kemungkinan model yang dibuat tidak mencerminkan secara tepat kondisi aslinya. Kemungkinan lain adalah meninjau ulang pembatasan masalah, karena model dapat keliru bila batasan masalah tidak jelas.

### **Pemrograman**

Pemilihan bahasa yang tepat untuk menyelesaikan suatu permasalahan sangat penting di sini karena ada beberapa bahasa yang khusus diarahkan pada suatu keperluan tertentu, seperti :

- Bahasa Cobol untuk keperluan bisnis
- Bahasa C, Modula-2 untuk pemrograman sistem
- Bahasa Simula untuk simulasi

Untuk hal-hal yang lebih umum bahasa-bahasa : Pascal, Basic, bahkan bahasa-bahasa yang disebutkan diatas dapat digunakan, tentunya dengan beberapa kendala yang akan dihadapi. Pada tahap pemrograman ini, selain penentuan bahasa pemrograman yang akan digunakan, seringkali harus ditentukan struktur data yang sesuai dengan model yang telah dibuat, untuk mencapai penyelesaian yang berdaya guna dan berhasil guna. Dalam kaitan ini pemilihan bahasa pemrograman harus lebih teliti karena tidak semua bahasa pemrograman mendukung struktur data yang diinginkan.

### **Pengujian dan Analisis**

Program yang telah selesai dibuat, kemudian diuji di laboratorium dan hasilnya dianalisis, untuk melihat : kecepatan, pemakaian memori, dsb. Seringkali program diuji di lapangan oleh calon

pemakai atau bila perlu oleh pihak lain yang lebih luas untuk memperoleh umpan balik yang lebih sempurna.

### Dokumentasi

Meskipun dalam gambar diatas, terlihat bahwa blok Dokumentasi berada pada bagian akhir, tetapi kegiatan dokumentasi dilakukan secara menyeluruh, yaitu dimulai dari tahap awal hingga pengujian dan analisis. Hal ini diperlukan karena setiap peninjauan ulang dan koreksi memerlukan dokumentasi yang lengkap. Dokumentasi diperlukan juga untuk kegiatan pemeliharaan dan pengembangan program.

### Contoh Perhitungan Upah Buruh

Perhitungan upah buruh dalam satu minggu (5 hari kerja), terdiri dari dua komponen

1. Upah Harian, ( $H$  rupiah / hari) bila ia bekerja 8 jam sehari
2. Upah Lembur, ( $L$  rupiah /jam) bila ia bekerja lebih dari 8 jam dalam sehari

Diinginkan untuk menghitung pendapatan seorang buruh dalam satu minggu, bila diketahui data jam kerja untuk setiap harinya.

### Algoritma Perhitungan Upah Buruh

1. Total Pendapatan  $\rightarrow 0$
2. Baca data jam kerja untuk hari ke 1.
3. Upah Hari Ini =  $H + (\text{jam kerja} - 8) \times L$ ,
4. Total Pendapatan =  $\square$  Total Pendapatan + Upah Hari Ini
5. Ulangi 2 s/d 4 hingga hari ke 5.

Misalkan Upah Harian  $H = 6000$  rupiah / hari

Upah Lembur  $L = 1000$  rupiah / jam

Jam kerja hari ke 1 s/ d 5 adalah : 10, 8, 12, 8 dan 8 jam

Pendapatan hari ke 1 =  $6000 + (10-8) \times 1000$

Pendapatan hari ke 2 = 6000

Pendapatan hari ke 3 =  $6000 + (12-8) \times 1000$

Pendapatan hari ke 4 = 6000

Pendapatan hari ke 5 = 6000

Total Pendapatan dalam seminggu (5 hari) kerja = 36.000 rupiah.

Contoh ini menunjukkan bagaimana pembatasan masalah masih belum begitu jelas, karena masih harus dipertanyakan lagi beberapa hal, misalnya :

- a. Bagaimana perhitungan bila jam kerja kurang dari 8 jam
- b. Apakah ada jumlah minimal dan maksimal jam lembur

Bila pertanyaan-pertanyaan tidak dapat dijawab, maka algoritma perlu ditinjau kembali, sehingga perhitungan menjadi benar, tidak merugikan buruh ataupun majikan.

### 1.7. Sejarah Bahasa C++

Tahun 1978, Brian W. Kerninghan & Dennis M. Ritchie dari AT & T Laboratories mengembangkan bahasa B menjadi bahasa C. Bahasa B yang diciptakan oleh Ken Thompson sebenarnya merupakan pengembangan dari bahasa BCPL ( Basic Combined Programming Language ) yang diciptakan oleh Martin Richard.

Sejak tahun 1980, bahasa C banyak digunakan pemrogram di Eropa yang sebelumnya menggunakan bahasa B dan BCPL. Dalam perkembangannya, bahasa C menjadi bahasa paling populer diantara bahasa lainnya, seperti PASCAL, BASIC, FORTRAN.

Tahun 1989, dunia pemrograman C mengalami peristiwa penting dengan dikeluarkannya standar bahasa C oleh American National Standards Institute (ANSI). Bahasa C yang diciptakan Kerninghan & Ritchie kemudian dikenal dengan nama ANSI C.

Mulai awal tahun 1980, Bjarne Stroustrup dari AT & T Bell Laboratories mulai mengembangkan bahasa C. Pada tahun 1985, lahirlah secara resmi bahasa baru hasil pengembangan C yang dikenal dengan nama C++. Sebenarnya bahasa C++ mengalami dua tahap evolusi. C++ yang pertama, dirilis oleh AT&T Laboratories, dinamakan cfront. C++ versi kuno ini hanya berupa kompiler yang menerjemahkan C++ menjadi bahasa C.

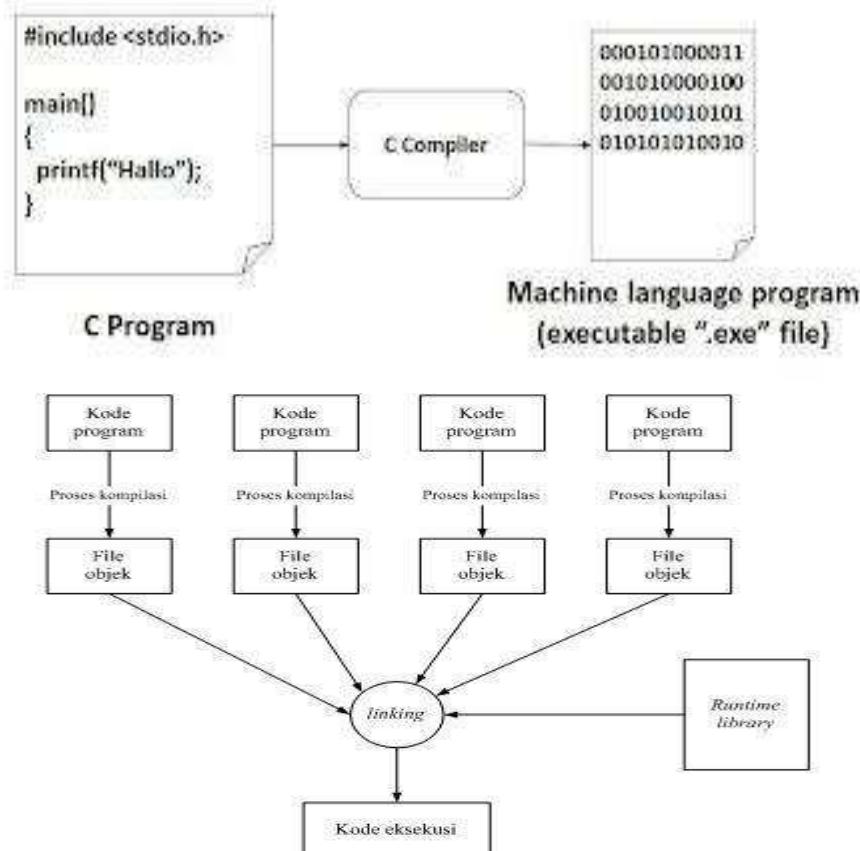
Pada evolusi selanjutnya, Borland International Inc. mengembangkan kompiler C++ menjadi sebuah kompiler yang mampu mengubah C++ langsung menjadi bahasa mesin (assembly). Sejak evolusi ini, mulai tahun 1990 C++ menjadi bahasa berorientasi objek yang digunakan oleh sebagian besar pemrogram profesional.

### 1.8. Mengenal editor Bahasa C++

Apapun bahasa pemrograman yang digunakan, untuk menulis perintah/ instruksi diperlukan tempat untuk mengetikkan perintah tersebut. Tempat yang digunakan untuk menulis perintah ini sering disebut dengan editor. Agar pembuatan program C++ menjadi mudah, dalam buku ini, editor yang digunakan adalah turbo C++. Tidak ada keharusan menggunakan editor turbo C++, masih ada beberapa editor yang masih dapat digunakan, diantaranya borland C++, GCC (untuk pengguna Linux).

Dengan menggunakan editor ini, proses pengetikan perintah C++, proses kompilasi dan proses untuk melihat hasilnya cukup menggunakan editor ini. Jadi dengan editor ini proses pembuatan program adalah :

1. Buat program
  2. Lakukan proses kompilasi
- Proses kompilasi akan berhasil jika susunan program yang diketik tidak mengandung kesalahan sintak (kesalahan aturan dalam tata tulis pemrograman).
3. Jalankan program / eksekusi

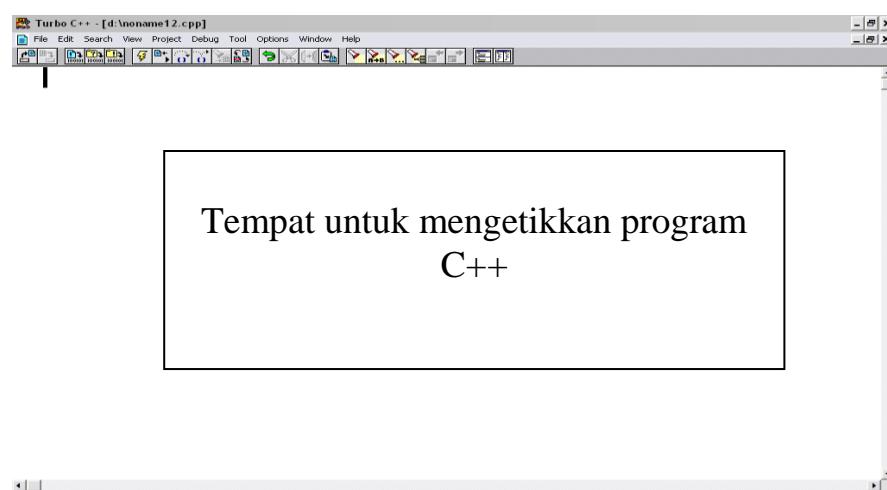


Gambar 1.12 Proses Kompilasi

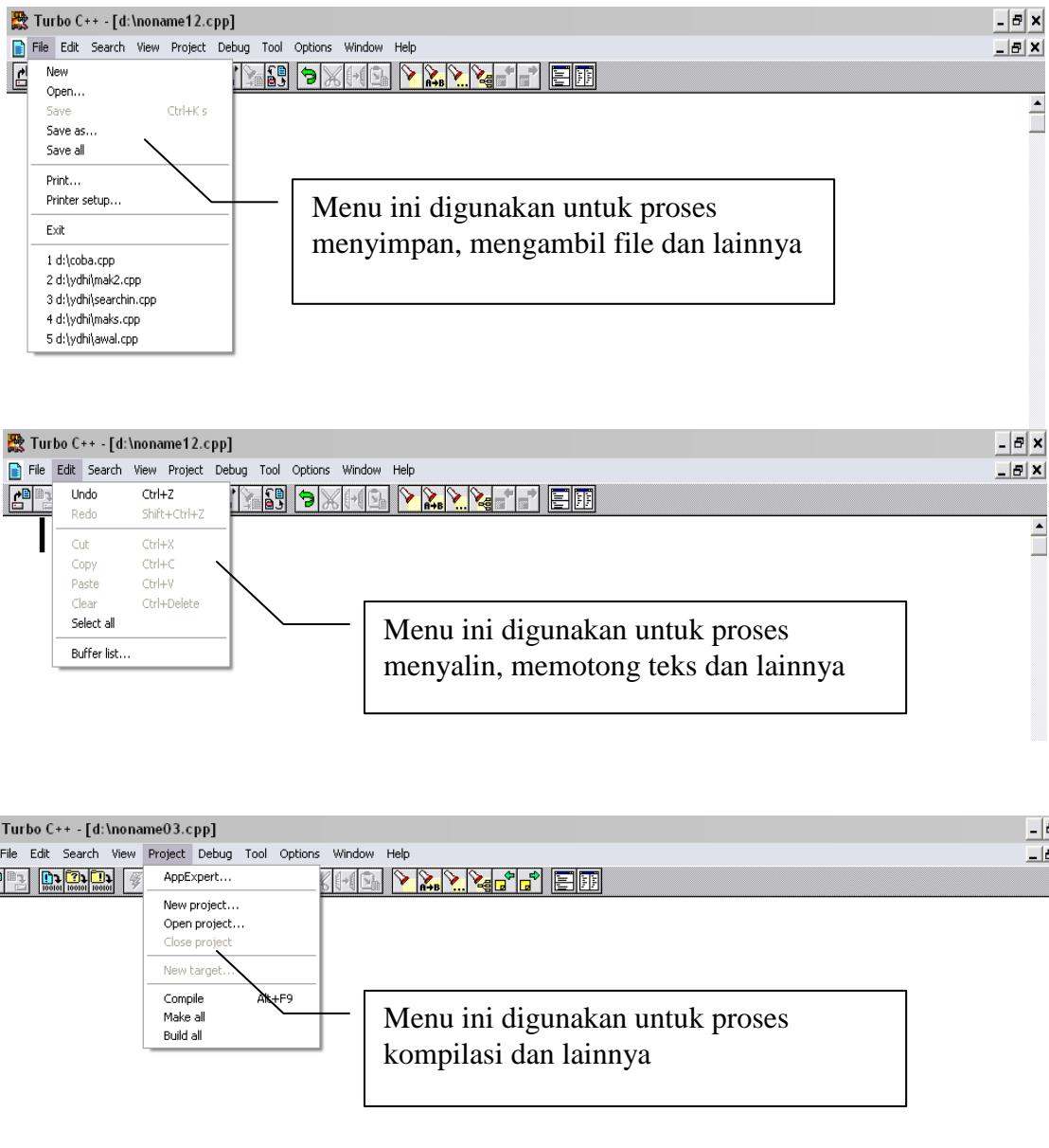
<http://desylvia.wordpress.com/2010/09/06/pemrograman-bahasa-c-pendahuluan/>

### 1.9. Proses instalasi C++

Agar bisa menggunakan C++, install C++ dan untuk memulai bekerja buka dan jalankan turbo C++. Hasil awal, akan tampil editor seperti di bawah ini :



Gambar 1.12 Editor C++



Gambar 1.13. Proses penyusunan program di C++

### 1.10. Membuat program C++

Setiap bahasa (tidak hanya bahasa manusia), pasti mempunyai aturan-aturan yang spesifik. Demikian juga bahasa C++, juga mempunyai aturan-aturan. Aturan ini harus benar-benar diperhatikan, salah penulisan, misal hanya kurang koma/ titik koma akan berakibat proses kompilasi gagal dan program tidak bisa dijalankan.

```
#include <iostream.h>
main()
{ cout<<"selamat datang di C++"; }
```

Ketikan program diatas di editor C++

```
#include <iostream.h>
main()
{
    cout<<"selamat datang di C++";
}
```

Klik icon ini untuk proses kompilasi

Proses kompilasi, jika tidak ada kesalahan akan ditampilkan hasil dari program tersebut

```
#include <iostream.h>
main()
{
    cout<<"selamat datang di C++";
}
```

Compile Status		
Status: Linking, pass ...		
Files	EXE file:	noname12.exe
	Linking:	c:\tcwin45\lib\import...
Statistics	Total	Link
Lines:	7	0
Warnings:	2	1
Errors:	0	0

Hasil program yang dibuat

```
#include <iostream.h>
main()
{
    cout<<"selamat datang di C+"
}
```

(Inactive D:\NONAME12.EXE)

selamat datang di C++

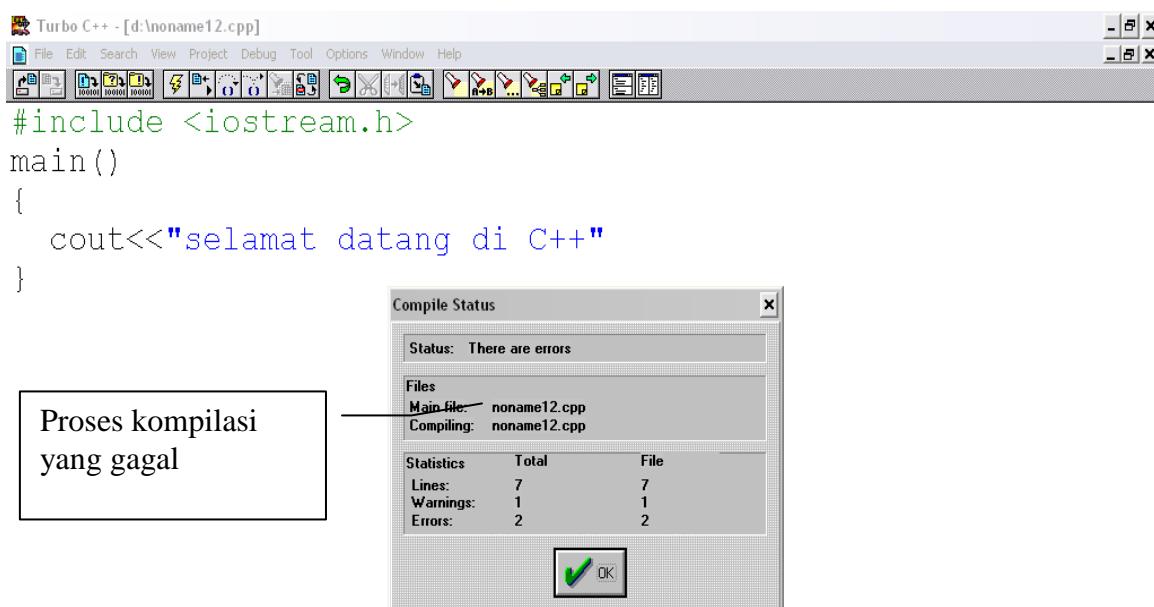
### 1.11. Memahami Kesalahan Aturan Sintaks Pada Bahasa C

Pengertian aturan sintak itu sendiri adalah sekumpulan aturan-aturan yang harus dipatuhi saat mengetikkan program. Aturan-aturan sederhana yang perlu diperhatikan diantaranya :

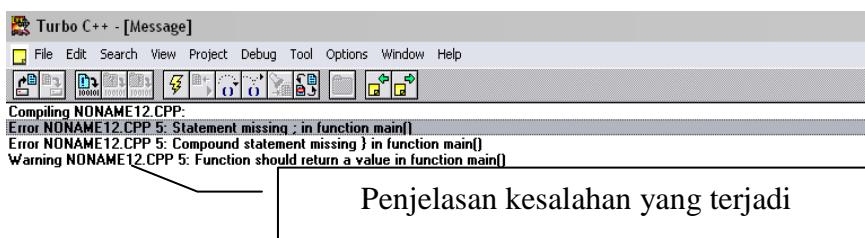
1. Bahasa c++ membedakan antara huruf besar dengan huruf kecil

2. Setiap akhir perintah diakhiri dengan tanda titik kom (;

```
#include <iostream.h>
main()
{
    cout<<"selamat datang di C++"
}
```



Kenapa program di atas terjadi kesalahan (gambar di atas), hal ini disebabkan ada kekurangan dalam penulisan `cout<<"selamat datang di C++"` kurang tanda titik koma (;). Letak kesalahan ini akan ditujukan Bahasa C++ dengan menginformasikan letak baris yang terjadi kesalahan serta penjelasan kenapa terjadi kesalahan. Jadi .... Agar tidak bingung saat terjadi kesalahan, lihat pesan kesalahan dan konsentrasi pada letak baris kesalahan dan lakukan perbaikan.



Seiring dengan bertambahnya materi yang dipelajari, kemungkinan muncul kesalahan juga makin kompleks, untuk itu benar-benar perhatikan aturan penulisan/ sintak setiap perintah

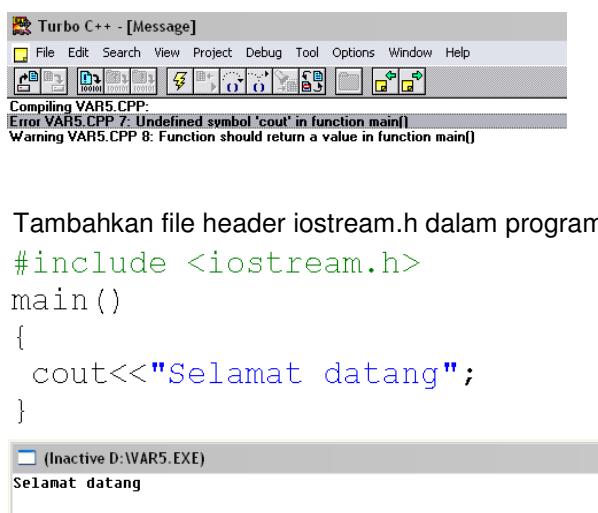
### contoh

- Perhatikan program di bawah ini

```
main()
{
    cout<<"Selamat datang";
}
```

#### Penjelasan

- Program di atas terjadi kesalahan, karena perintah cout belum dicantumkan file header iostream.h



- Tambahkan file header iostream.h dalam program tersebut

```
#include <iostream.h>
main()
{
    cout<<"Selamat datang";
}
```

- Hasil kompilasi akan menghasilkan file exe, cari file tersebut disimpan

- Jalankan file exe tersebut tanpa melalui editor C++, langsung dari folder
- Bila diinginkan melakukan modifikasi, bisakah hal tersebut dilakukan.
- Buka kembali file teks yang ada di editor C++, lakukan modifikasi, bisakah hal tersebut dilakukan.

### 1.12. Latihan

- Install bahasa C++ di Komputer
- Jalankan semua program di atas, pahami dan pelajari aturan bahasa C++

2

# STRUKTUR BAHASA C++



## Tujuan Instruksional

Setelah membaca bab ini, diharapkan pembaca memahami struktur bahasa C++, aturan dan proses kompilasi serta fungsi dan manfaat file header. Dengan memahami aturan dasar bahasa C++ diharapkan pembaca sudah bisa membuat program sederhana dengan C++

## Materi

Struktur Bahasa C++

Perintah-perintah Dasar Bahasa C++

## 2.1. Struktur Bahasa C++

Setiap bahasa (baik bahasa manusia maupun bahasa komputer) pasti mempunyai struktur yang mencirikan bahasa tersebut. Demikian juga C++, mempunyai ciri/ struktur yang khas.

```
// .... komentar  
#include <iostream.h>  
main ()  
{  
}
```

// komentar

- Baris ini adalah komentar. semua baris yang diawali dengan dua garis miring (//) akan dianggap sebagai komentar dan tidak akan berpengaruh terhadap program. Dapat digunakan oleh programmer untuk menyertakan penjelasan singkat atau observasi yang terkait dengan program tersebut.
- Jadi walaupun diberi baris ini ada penulisan yang tidak memenuhi sintak, tidak menjadi masalah. Hal ini dikarenakan komentar tidak akan dilakukan pengecekan aturan sintak
- // Komentar baris
- /\* Komentar Blok \*/
- Komentar baris, akan mengabaikan apapun mulai dari tanda (//) sampai akhir dari baris yang sama. Komentar Blok, akan mengabaikan apapun yang berada diantara tanda /\* dan \*/.

#include <iostream.h>

- Perintah yang ada dalam tanda siku tidak harus iostream, Kalimat yang diawali dengan tanda (#) adalah are pre-processor directive. Dalam contoh diatas perintah #include <iostream.h> memberitahukan pre-processor kompiler untuk menyertakan header file standard iostream. File spesifik ini juga termasuk library deklarasi standard I/O pada C++ dan file ini disertakan karena fungsi-fungsinya akan digunakan nanti dalam program.

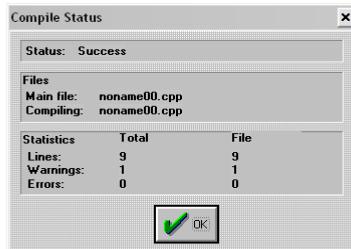
main ()

Dalam tubuh main inilah perintah-perintah bahasa c++ ditulis. fungsi main merupakan titik awal di mana seluruh program C++ akan mulai dieksekusi. Diletakkan di awal, di tengah atau di akhir program, isi dari fungsi main akan selalu dieksekusi pertama kali. Pada dasarnya, seluruh program C++ memiliki fungsi main.

main diikuti oleh sepasang tanda kurung () karena merupakan fungsi.

**Program di atas, bila dikompilasi tidak terjadi kesalahan tetapi tidak akan menghasilkan apa-apa.**

```
// .... komentar
#include <iostream.h>
main ()
{
}
```



Atau di tulis

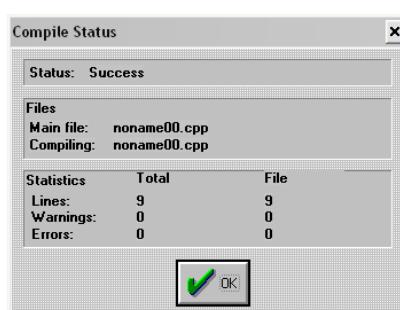
```
// .... komentar
#include <iostream.h>
int main ()
{
}
```

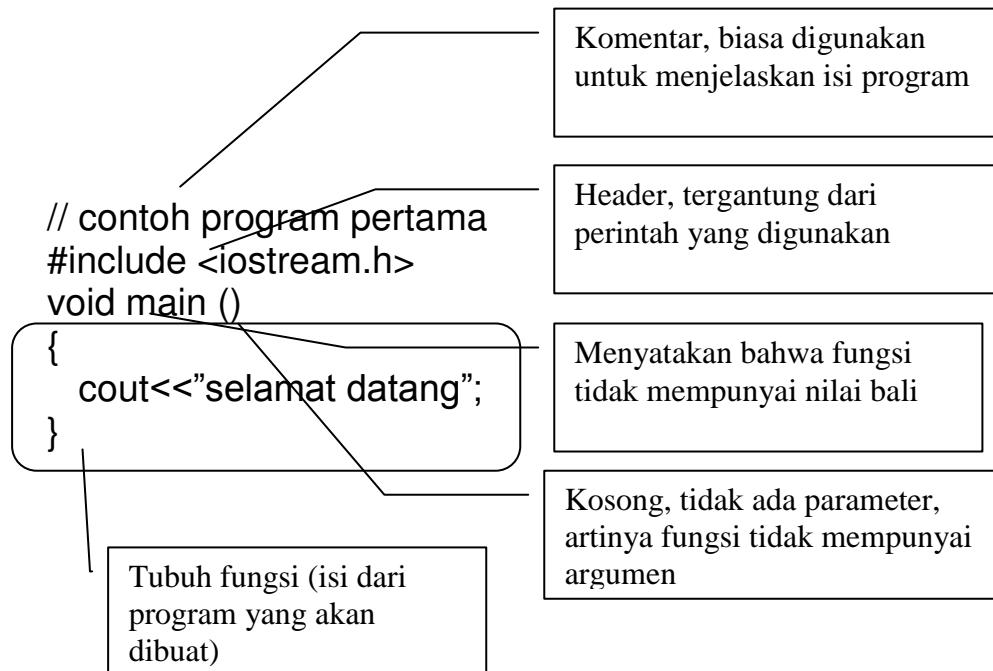
return 0;

Instruksi **return** menyebabkan fungsi **main()** berakhir dan mengembalikan kode yang mengikuti instruksi tersebut, dalam kasus ini 0. Ini merupakan cara yang paling sering digunakan untuk mengakhiri program.

```
// .... komentar
#include <iostream.h>
void main ()
{
}
```

**return** ;



**contoh****Hasil eksekusi :**

```

// contoh program pertama
#include <iostream.h>
void main ()
{
    cout<<"selamat datang";
}

```

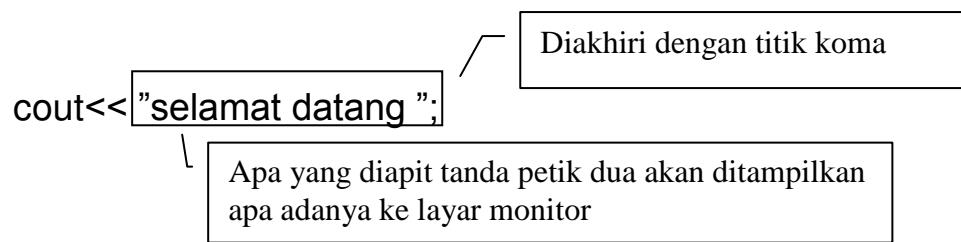
**Pernyataan**

```

{
    cout<<"selamat datang";
}

```

baris diatas sering disebut dengan pernyataan. Pada contoh diatas, pernyataan tersebut digunakan untuk menampilkan suatu kalimat di layar monitor. Kalimat atau string yang ditampilkan adalah kalimat/ string yang diapit dengan tanda petik dua (").



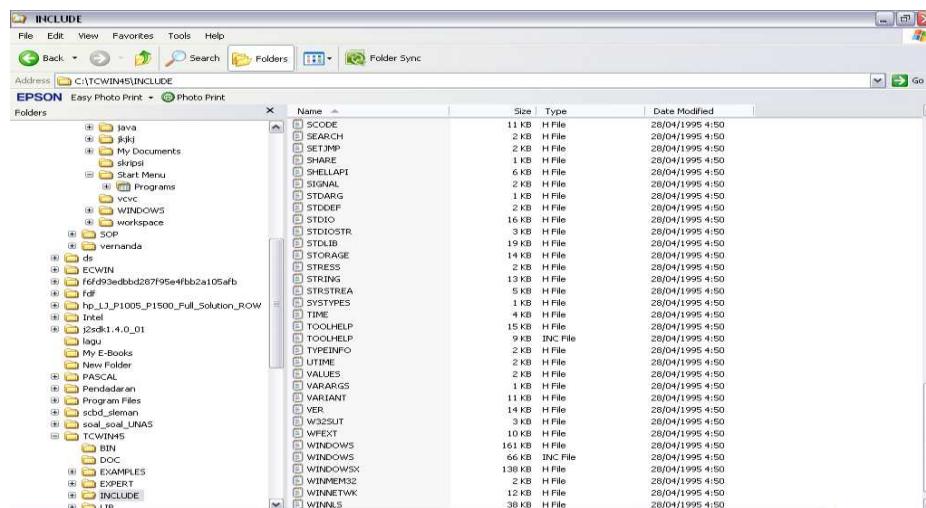
## 2.2. Perintah-perintah dasar

### Include

Setiap kali membuat bahasa C++, selalu di awal program harus dituliskan perintah include. Include merupakan suatu header yang . mempunyai kegunaan untuk 'menerjemahkan' kegunaan dari perintah-perintah yang akan digunakan. Semakin banyak ragam perintah kegunaan ada kemungkinan file header juga akan bertambah. Beberapa kegunaan file header yang biasa digunakan diantaranya:

1. Untuk manajemen memori
2. Untuk memanggil routines ROM BIOS
3. Untuk fungsi matematika kompleks
4. Untuk memanggil console DOS I/O (input output)

File header ini harus ada dalam folder C++. File header ini bisa dilihat bagaimana source tetapi jangan sekali-kali melakukan perubahan.



Gambar 2.1. File header

```

/*
 * iostream.h -- basic stream I/O declarations
 *
 * There are some inline functions here which generate a LOT of code
 * (as much as 300 bytes) but are available in the library because AT&T did
 * it this way. We have all made them true functions in the library
 * and conditionally deleted the inline code from this header.
 *
 * If you really want these big functions to be inline, #define the
 * macro name __BIG_INLINE__ before including this header.
 *
 * Programs will compile and link correctly even if some modules are
 * compiled with __BIG_INLINE__ and some are not.
 */

/*
 * C/C++ Run Time Library - Version 6.5
 * Copyright (c) 1990, 1994 by Borland International
 * All Rights Reserved
 */

#ifndef __CPLUSPLUS
#error Must use C++ for the type iostream.
#endif

#ifndef __TOSTREAM_H
#define __TOSTREAM_H
#if !defined(__DEFS_H)
#include <_defs.h>
#endif
#if !defined(__MEM_H) // to get memcpy and NULL
#include <mem.h>
#endif
#if !defined(__RC_INVOKED)
#pragma option -a- // byte packing
#if defined(__BCOPT__)
#if !defined(__RTL_ALLOW_PO) && !defined(__FLAT__)
#pragma option -po- // disable Object data calling convention
#endif
#endif
#endif
#endif

```

Gambar 2.2. Isi program File header

Beberapa file header yang sering digunakan :

iostream.h	diperlukan pada program yang melibatkan perintah input – output, misal cout, cin
conio.h	diperlukan bila melibatkan clrscr(), yaitu perintah untuk membersihkan layar.
iomanip.h	diperlukan bila melibatkan setw() yang bermanfaat untuk mengatur lebar dari suatu tampilan data.
math.h	diperlukan pada program yang menggunakan operasi matematika, misal sqrt untuk mencari akar, pow untuk mencari kuadrat
string.h	File header ini digunakan untuk memanipulasi sting dan array. Contoh fungsi : strcpy(), strlen(), strcat().
time.h	File header ini mengandung beberapa definisi fungsi untuk memanipulasi informasi tanggal dan waktu.
stdlib.h	File header stdlib.h menjabarkan beberapa fungsi umum dan marco termasuk manajemen memori dinamis, menjalin komunikasi dengan perangkat sekitar, membuat bilangan secara random, aritmetika bilangan integer, pencarian, pengurutan dan pengonversian.

### cout

Perintah ini hampir dipastikan ada di setiap program. Perintah ini digunakan untuk menampilkan kalimat atau string ke layar monitor dan header yang harus dicantumkan adalah IOSTREAM.

Aturan penulisan cout

cout<<" statement yang ditampilkan";

perintah diatas akan menampilkan statement dan setelah menampilkan statement tidak dilakukan proses ganti baris. Agar berganti baris penulisannya :

cout<<" statement yang ditampilkan"<<endl;

Instruksi untuk berganti baris setelah mencetak statement.

`cout<<" statement yang ditampilkan\n";`

Instruksi untuk berganti baris setelah mencetak statement.

Walaupun `\n` terletak dalam statement atau dalam tanda petik dua ("); perintah ini tidak ikut tercetak ke layar tetapi dianggap sebagai instruksi. Tanda `\` artinya bahwa setelah tanda `\` bukan termasuk karakter yang dicetak tetapi adalah instruksi yang harus dijalankan. `\n` artinya instruksi untuk berganti baris.

```
//contoh program cout tanpa menggunakan
//perintah ganti baris
#include <iostream.h>
void main()
{ cout<<"pertama";
  cout<<"kedua";
  cout<<"ketiga";
}
```

Tidak menggunakan instruksi untuk berganti baris



Program diatas hasilnya akan dicetak dalam satu baris, walaupun instruksi terdiri dari tiga perintah cout, hasilnya tetap satu baris. Hal ini setiap akhir instruksi cout tidak ada instruksi untuk ganti baris.

```
//contoh program cout dengan menggunakan
//perintah ganti baris endl
#include <iostream.h>
void main()
{ cout<<"pertama"<<endl;
  cout<<"kedua"<<endl;
  cout<<"ketiga"<<endl;
}
```

Menggunakan instruksi untuk berganti baris endl



Hasil program di atas menghasilkan tampilan di layar sebanyak tiga baris, hal ini dikarenakan dalam program terdapat tiga perintah cout dan di setiap akhir ada instruksi untuk ganti baris

```
//contoh program cout dengan menggunakan
//perintah ganti baris \n
#include <iostream.h>
void main()
{ cout<<"pertama\n";
  cout<<"kedua\n";
  cout<<"ketiga\n";
}
```

```
pertama
kedua
ketiga
```

Menggunakan instruksi untuk  
berganti baris \n

Dari ketiga contoh diatas terlihat bahwa banyaknya instruksi tidak mencerminkan banyaknya baris yang dihasilkan/ ditampilkan di layar monitor. Jadi untuk menampilkan hasil ke monitor tergantung dari ada tidaknya instruksi ganti baris.

```
//contoh program cout dengan campuran
//instruksi ganti baris
#include <iostream.h>
void main()
{ cout<<"pertama\n";
  cout<<"kedua";
  cout<<"ketiga"<<endl;
  cout<<"keempat";
}
```

```
pertama
kedua
ketiga
keempat
```

Menggunakan instruksi untuk  
berganti baris

Tidak menggunakan instruksi  
untuk berganti baris

Menggunakan instruksi untuk  
berganti baris endl

### Penjelasan

cout<<"pertama\n"; setelah mencetak tulisan pertama ada instruksi ganti baris  
 cout<<"kedua"; setelah mencetak tulisan kedua tidak ada instruksi ganti baris  
 cout<<"ketiga"<<endl; setelah mencetak tulisan ketiga ada instruksi ganti baris  
 cout<<"keempat\n"; setelah mencetak tulisan keempat tidak ada instruksi ganti baris

### contoh

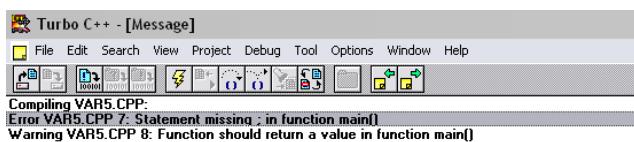
- Buat program dengan tampilan

```
Selamat datang di IST "AKPRIND" Yogyakarta
```

Jika dibuat seperti ini, hasilnya akan salah... kenapa ? :

```
#include <iostream.h>
main()
{
    cout<<"Selamat datang di IST "AKPRIND" Yogyakarta";
}
```

### Penjelasan



- Perintah petik dua ("") mempunyai fungsi untuk mengawali dan mengakhiri suatu teks/kalimat. Sehingga penulisan tersebut salah

**cout<<"Selamat datang di IST "AKPRIND" Yogyakarta";**

Awal teks/kalimat

Dianggap akhir teks/kalimat

- Akibat dari perintah " di akhir tulisan IST dianggap mengakhiri perintah untuk mencetak kalimat, berakibat kompiler akan menganggap tulisan AKPRIND menjadi perintah yang salah.
- Agar teks bisa menampilkan tanda petik, gunakan Escape Sequences.

### Contoh

```
#include <iostream.h>
main()
{
    cout<<"Selamat datang di IST \"AKPRIND\" Yogyakarta";
}
```

### Penjelasan

**cout<<"Selamat datang di IST \"AKPRIND\" Yogyakarta";**

Dengan adanya tanda \ ini berarti tanda petik dua ("") bukan berarti mengawali atau mengakhiri suatu kalimat, tetapi mempunyai arti bahwa tanda petik akan dianggap menjadi teks

Escape Sequences menggunakan notasi “\” ( back slash ) jika karakter terdapat notasi “\” ini sebagai karakter “escape” ( menghindar). Beberapa Escape Sequences lainnya antara lain :

ESCAPE SEQUENCES	PENGERTIAN
\b	Backspace
\f	Formfeed
\n	Baris Baru
\r	Carriage Return
\t	Tab ( default = 8 karakter )
\'	Tanda kutip tunggal ( ‘ )
\”	Tanda Kutip Ganda ( ” )
\\\	Backslash
\xaa	Kode ASCII dalam hexadecimal. ( aa menunjukkan angka ASCII ybs )
\aaa	Kode ASCII dalam octal. ( aaa menunjukkan angka ASCII ybs )

**Latihan**

**Program-program pendek ini, bila dijalankan hasilnya apa**

1. 

```
1. include <iostream.h>
void main(void)
{
    cout << "Bahasa C++";
    cout << "Bahasa Pascal";
    cout << "Bahasa Basic";
}
```
2. 

```
#include <iostream.h>
void main(void)
{
    cout << endl << "Bahasa C++";
    cout << endl << "Bahasa Pascal";
    cout << endl << "Bahasa Basic";
}
```
3. 

```
#include <iostream.h>
void main(void)
{
    cout << "Bahasa \nC++";
    cout << "Bahasa \n Pascal";
    cout << "\nBahasa Basic";
}
```
4. 

```
#include <iostream.h>
void main(void)
{
    cout << "Bahasa C++" << endl;
    cout << "Bahasa Pascal" << endl << "Bahasa Basic";
}
```
5. 

```
#include <iostream.h>
void main(void)
{
    cout << "Bahasa C++" << endl << "Bahasa Pascal" << endl << "Bahasa Basic";
}
```
6. 

```
#include <iostream.h>
void main(void)
{
    cout << "Bahasa C++\nBahasa Pascal\nBahasa Basic";
}
```

Buat program dengan tampilan

7. Selamat datang di IST “AKPRIND” Yogyakarta
8. Ini tanda \n

3

## VARIABEL DAN, TIPE DATA



## Tujuan Instruksional

Setelah membaca bab ini, diharapkan pembaca dapat memahami manfaat variabel, aturan dalam membuat variabel, perbedaan tipe data, konversi data, overflow serta casting. Dengan memahami bab ini pembaca diharapkan sudah mampu membuat program numerik serta program sederhana lainnya

## Materi

 Variabel dan aturannya

 Tipe data dan aturannya

 Konstanta

 Overflow Data

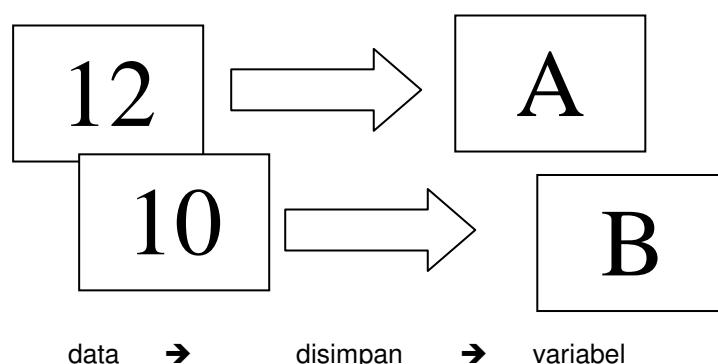
 Tipe Casting

### 3.1. Variabel

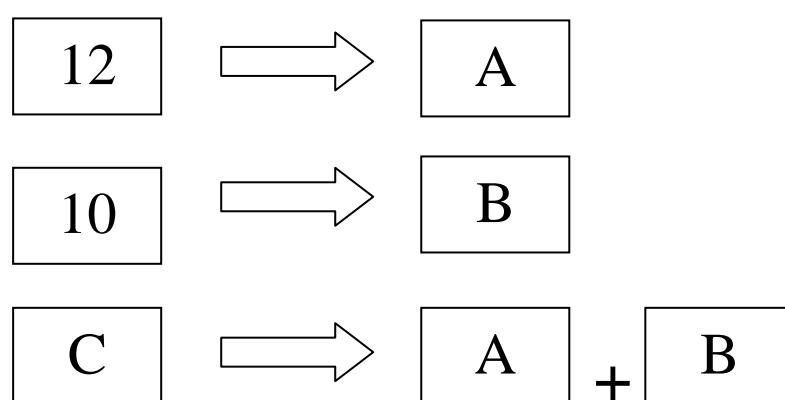
Unsur terpenting dalam pemrograman adalah bagaimana menyimpan data yang akan diolah disimpan oleh komputer. Tanpa ada data tentunya program tidak bisa dikerjakan. Bayangan program ATM tanpa adanya masukan data kartu ATM dan data PIN, tentunya program ATM tidak akan dapat bekerja. Demikian juga program kasir, tanpa ada data belanja, program tersebut tidak dapat bekerja.

Dalam pemrograman, data-data yang akan diolah atau dimasukan ke sistem sering disebut dengan variabel. Jadi, Variabel merupakan suatu tempat untuk menampung data atau konstanta di memori yang mempunyai nilai atau data yang dapat berubah – ubah selama proses program . Setiap data yang akan diolah program harus tersimpan dalam suatu variabel. Hal ini mirip dengan nomor HP teman yang dimiliki. Untuk mempermudah mengingat nomor tersebut milik siapa tentunya sudah menjadi kebiasaan sewaktu menyimpan nomor tersebut diberi identitas, Bayangkan jika ada puluhan nomor telepon tanpa ada nama identitas pemilik nomor tersebut.

Demikian juga program, setiap data yang akan dimasukan atau akan diolah komputer akan terlebih dahulu disimpan dalam suatu variabel. Sewaktu memasukan data PIN di ATM, program akan menyimpan data tersebut dalam suatu variabel.



Dari ilustrasi diatas, data 12 disimpan dalam variabel A dan data 10 disimpan dalam variabel B.



Variabel A berisi 12

Variabel B berisi 10

Variabel C merupakan penjumlahan antara variabel A dengan variabel B

Jadi jika ada operasi atau instruksi seperti ini :



Operasi/ perintah di atas salah, dikarenakan hasil penjumlahan antara A dengan B tidak disimpan dalam suatu variabel.

Seluruh proses ini dapat diekspresikan dalam C++ dengan serangkaian instruksi sbb :

```
A = 12;  
B = 10;  
C = A + B;
```

Jelas contoh di atas merupakan satu contoh yang sangat sederhana karena kita hanya menggunakan 2 nilai bilangan bulat (integer) yang kecil, tetapi komputer dapat menyimpan jutaan angka dalam waktu yang bersamaan dan dapat melakukan operasi matematika yang rumit.

Karena itu, kita dapat mendefinisikan variabel sebagai bagian dari memory untuk menyimpan nilai yang telah ditentukan. Setiap variabel memerlukan **identifier** yang dapat membedakannya dari variable yang lain, sebagai contoh dari kode diatas identifier variabelnya adalah **A**, **B** dan **C**, tetapi kita dapat membuat nama untuk variabel selama masih merupakan identifier yang benar.

### 3.2. Penamaan Variabel

Setiap variabel yang akan digunakan sebaiknya mencerminkan isi dari data yang akan disimpan serta mudah dibaca. Misal penamaan :

**Gaji\_pegawai=1500000;**

Yang menyatakan gaji pegawai. Penamaan ini tentunya lebih mudah dipahami daripada penamaan

**G=1500000;**

Aturan penamaan Variabel/ pengenal

- Tidak boleh diawali angka
- Tidak boleh diawali dengan karakter khusus (misal tanda matematika)
- Huruf besar dan huruf kecil berbeda
- Tidak boleh ada spasi, jika menggunakan 2 kata bisa menggunakan tanda hubung (misal underscore)

Contoh

Nama variabel	Keterangan	
Gaji2013	Benar	Walaupun ada angka tetapi angka dibelakang
2013Gaji	Salah	Angka tidak boleh di depan
A	Benar	Minimal 1 huruf
Gaji pegawai	Salah	Ada spasi, yang benar Gaji_pegawai
+A	Salah	Tanda operasi matematika tidak boleh ada di dalam pengenal
Nama-barang	Salah	Tidak boleh menggunakan karakter -

```
//contoh program mencari luas dan keliling segitiga
#include <iostream.h>
void main()
{ float gaji_pegawai;
  float gaji pegawai;
  gaji_pegawai=1500000;
  cout<<"Gaji pegawai "<<gaji_pegawai<<endl;
}
```

Program tersebut jika dijalankan akan terjadi kesalahan terutama pada penulisan perintah

**float gaji pegawai;**

Compiling VARS.CPP:  
Error VARS.CPP 7: Declaration syntax error in function main()  
Warning VARS.CPP 8: Constant is long in function main()  
Warning VARS.CPP 10: 'gaji' is declared but never used in function main()

Error. karena ada spasi  
dalam penamaan variabel

### 3.3. Kata kunci

Di samping aturan penamaan variabel/ pengenal yang disebutkan di atas, masih ada satu larangan dalam penamaan pengenal, yaitu tidak diperkenankan menggunakan kata kunci yang merupakan bagian dari instruksi di C++. Kata kunci ini tidak diperkenankan digunakan karena mempunyai arti khusus bagi kompiler. Kata kunci tersebut diantaranya :

Asm	auto	bool	break	case
Catch	char	class	const	const_cast
continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern
false	float	for	friend	goto
If	inline	int	long	mutable
namespace	new	operator	private	protected
public	register	reinterpret_cast	return	short
signed	sizeof	static	static_cast	struct
switch	template	this	throw	true
Try	typedef	typeid	typename	union
unsigned	using	virtual	void	volatile
wchar_t				

Sebagai tambahan, representasi alternatif dari operator, tidak dapat digunakan sebagai identifier.

Contoh :

**and, and\_eq, bitand, bitor, compl, not, not\_eq, or, or\_eq, xor, xor\_eq**

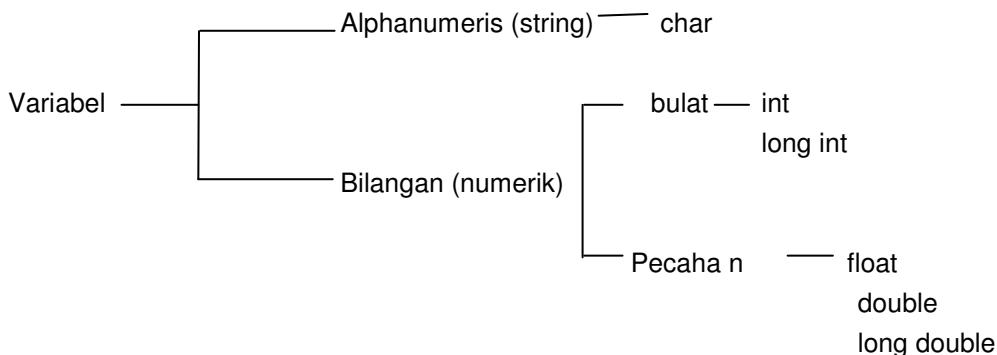
```
//contoh program mencari luas dan keliling segitiga
#include <iostream.h>
void main()
{
    float cout;
    float gaji pegawai;
    gaji_pegawai=1500000;
    cout<<"Gaji pegawai "
}
```

Mendeklarasikan variabel dengan nama variabel cout → salah. Hal ini disebabkan karena cout merupakan kata kunci/ sudah menjadi perintah di C++

```
Compiling VAR5.CPP:
Error VAR5.CPP 7: Declaration syntax error in function main()
Error VAR5.CPP 8: Undefined symbol 'gaji_pegawai' in function main()
Warning VAR5.CPP 8: Constant is long in function main()
Error VAR5.CPP 9: Illegal use of floating point in function main()
Warning VAR5.CPP 10: 'gaji' is declared but never used in function main()
Warning VAR5.CPP 10: 'cout' is declared but never used in function main()
```

### 3.4. Tipe Data

Hal lain yang perlu diperhatikan dalam membuat variabel adalah tipe data dari variabel tersebut. Tipe data akan mencerminkan isi dari variabel tersebut termasuk bilangan atau string serta jangkauan atau maksimal isi data dari variabel tersebut. Pada dasarnya variabel dibagi :



berikut nilai kisaran yang dapat direpresentasikan dalam bahasa c++ :

no	Tipe data	Ukuran memori	range	Keterangan
1	char	1 byte	-128 s/d 127	Karakter/string
2	int	2 byte	- 32768 s/d 32767	Integer/bilangan bulat
3	long	4 byte	-2.147.438.648 s/d 2.147.438.647	Integer/bilangan bulat
4	float	4 byte	- 3.4E-38 s/d 3.4E+38	Float/bilangan pecahan
5	double	8 byte	-1.7E-308 s/d 1.7E+308	Pecahan presisi ganda 15-16 digit
6	Long double	10 byte	-3.4E-4932 s/d 1.1E+4932	Pecahan presisi ganda 19 digit

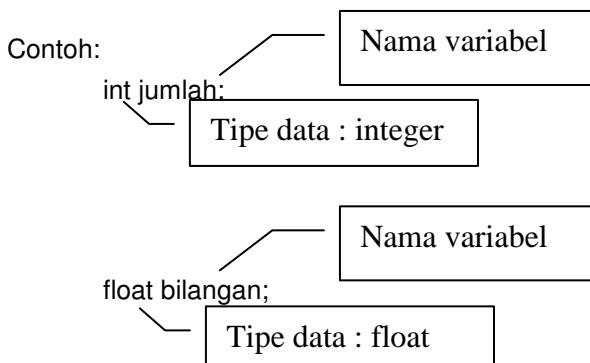
### 3.5. Deklarasi variabel

Dalam membuat program, variabel menjadi unsur terpenting terutama untuk menyimpan data. Variabel harus terlebih dahulu dideklarasikan/ dipesankan sebelum variabel tersebut diisi dengan data. Sintaks penulisan deklarasi variabel adalah dengan menuliskan tipe data yang akan digunakan diikuti dengan identifier yang benar, contoh :

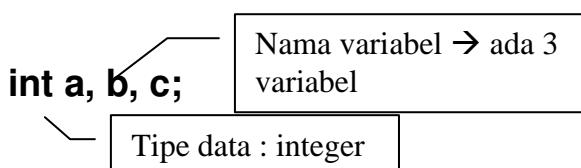
Bentuk pendeklarasian variabel :

#### Tipe data daftar\_variabel

Pada pendeklarasian variabel, daftar variabel dapat berupa sebuah variabel atau beberapa variabel yang dipisahkan dengan koma.



Jika akan menggunakan tipe data yang sama untuk beberapa identifier maka dapat dituliskan dengan menggunakan tanda koma, contoh :



### 3.6. Menentukan tipe variabel

Hal terpenting dalam membuat variabel adalah menentukan tipe data. Salah satu cara termudah untuk menentukan tipe data adalah menentukan apakah isi variabel tersebut berisi angka atau kalimat. Jika berisi bilangan tentukan bilangan bulat atau pecahan.

- int x; ➔ memesan variabel x yang mempunyai tipe data bilangan bulat (integer)
  - ➔ karena int maka kapasitasnya adalah -32768 hingga 32767
- float y; ➔ memesan variabel y yang mempunyai tipe data bilangan pecahan
  - ➔ karena float maka kapasitasnya adalah 3.4e + / - 38 (7 digits)
- long z ➔ memesan variabel y yang mempunyai tipe data bilangan bulat
  - ➔ karena long maka kapasitasnya adalah :-2147483648 hingga 2147483647

Jadi untuk memesan tipe data sebaiknya disesuaikan dengan kapasitas dari isi variabel.

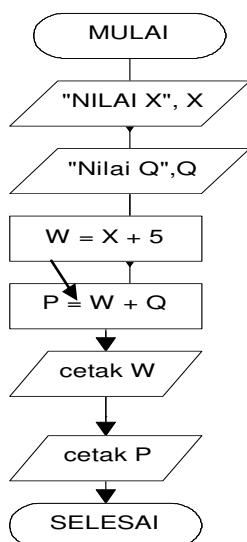
- |            |  |
|------------|--|
| A= 134     | ➔ sebaiknya tipe variabel adalah integer |
| B=10000000 | ➔ tipe variabel adalah integer           |

## Contoh

- Bila diketahui suatu rumus  
$$W = X + 5$$
  
$$P = W + Q$$

Bagaimana proses diagram alirnya :

- Tentukan nilai X dan Q
  - Hitung rumus  $W = X + 5$
  - Hitung rumus  $P = W + Q$  → Nilai W tidak dimasukkan dari keyboard. Nilai W diambil dari proses penjumlahan rumus  $W = X+5$



Hasil nilai W dari penjumlahan  $X + 5$ , digunakan untuk melakukan operasi  $W + Q$

dari contoh di atas, variabel yang harus dideklarasikan adalah :

- x,q,w,p → variabel tersebut dapat dideklarasikan dengan tipe data int.

### **3.7. Memberikan Nilai ke variabel**

Setelah dilakukan proses pendeklarasian, suatu variabel tentunya akan diisi dengan data. Proses pengisian nilai ke variabel dapat langsung diberikan pada variabel tersebut atau dimasukkan lewat program untuk meminta data. Proses memasukan data ke variabel secara langsung ke variabel adalah :

**Nama variabel = isi data**

Contoh :

```
int a;
```

Memesan variabel a yang bertipe integer

```
a=50;  
float b;
```

Mengisi variabel a dengan data 50

```
b=10.3;
a=60;
```

Mengisi variabel a dengan data 60, artinya isi a sebelumnya 50 diganti dengan 60;

50

60

a=50;  
mula-mula a diisi 50

a=60  
kemudian isi a diisi 60 sehingga isi a terakhir 60

### 3.8. Menampilkan isi variabel di monitor

Agar isi variabel dapat ditampilkan ke layar monitor, instruksi yang digunakan adalah perintah cout. Bentuknya adalah :

```
cout<<" isi statement"<<nama_variabel;
```

```
//contoh program variabel
#include <iostream.h>
void main()
{ int a;
  a=5;
  cout<<"isi variabel a "<<a;
}
```



#### Penjelasan :

int a; → memesan variabel a yang bertipe int (bilangan bulat)  
 a=5; → mengisi variabel a dengan data 5  
 cout<<"isi variabel a "<<a; → menampilkan isi a ke layar monitor

```
//contoh program variabel
#include <iostream.h>
void main()
{ int a;
  a=5;
  a=10;
  a=15;
  cout<<"isi variabel a "<<a;
}
```



#### Penjelasan :

Mula-mula variabel a diisi 5 (a=5), kemudian isinya diganti 10 dan a=15 artinya isi variabel a sekarang 15. Jadi yang disimpan dalam variabel a adalah isi data yang terakhir.

```
//contoh program variabel
#include <iostream.h>
void main()
{ int a,b,c;
  a=5;
  b=10;
  c=a+b;
  cout<<"isi variabel c "<<c;
}
```



### Penjelasan :

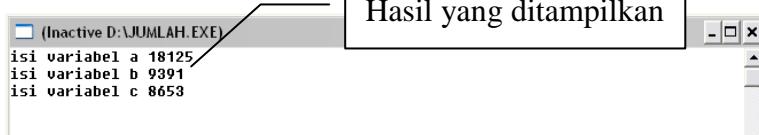
Variabel a diisi 5, variabel b diisi 10 kemudian variabel c menyimpan hasil penjumlahan antara isi variabel a dengan isi variabel b.

`cout<<"isi variabel c "<<c` → menampilkan isi variabel c ke layar monitor.

### Catatan :

Setiap variabel yang sudah dideklarasikan sebaiknya diisi dengan data. Jika tidak diisi data dan ditampilkan ke layar monitor hasilnya akan dilakukan pengacakan.

```
//contoh program variabel
#include <iostream.h>
void main()
{ int a,b,c;
  cout<<"isi variabel a "<<a<<endl;
  cout<<"isi variabel b "<<b<<endl;
  cout<<"isi variabel c "<<c<<endl;
}
```



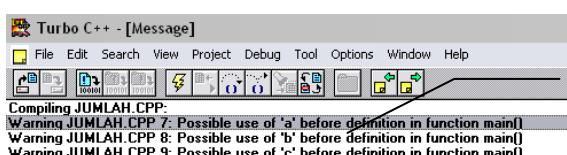
Menampilkan isi variabel  
a,b dan c ke monitor

Hasil yang ditampilkan

### Dari mana angka-angka tersebut diperoleh ?

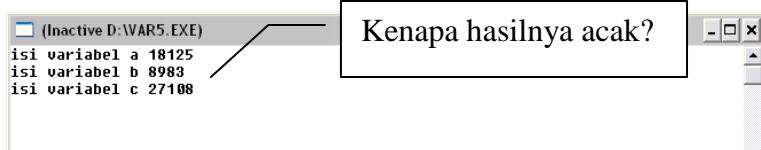
Jika variable tidak diisi atau diinisialisasi, dan ditampilkan maka nilai yang akan ditampilkan akan di acak oleh kompiler sehingga hasilnya nilai yang ditampilkan berbeda-beda tergantung dari jenis compiler nya.

Sebenarnya sewaktu dilakukan proses kompilasi, compiler memberi peringatan bahwa variabel tersebut belum di beri nilai awal.



Peringatan bahwa variabel belum diisi  
dengan data awal

```
#include <iostream.h>
void main()
{ int a,b,c;
c=a+b;
cout<<"isi variabel a "<<a<<endl;
cout<<"isi variabel b "<<b<<endl;
cout<<"isi variabel c "<<c<<endl;
}
```



### Penjelasan

- Memesan 3 variabel a,b,c dengan tipe integer
- Melakukan operasi penjumlahan antara isi variabel a dengan isi variabel b. Hasil penjumlahan ini disimpan di variabel c
- Karena variabel a dan b tidak diisi dengan data, maka isi variabel a dan variabel b diisi secara acak.

### 3.9. Konstanta

Konstanta sebenarnya juga termasuk dalam variabel. Perbedaannya kalau variabel nilai dapat berubah-ubah sementara konstanta nilainya bersifat tetap dan tidak bisa diubah/ diganti. Konstanta merupakan suatu nilai yang tidak dapat diubah selama proses program berlangsung. Konstanta harus didefinisikan terlebih dahulu di awal program.

Bentuk penulisannya adalah :

**#define pengenal nilai**  
Atau  
**const tipe\_data nama\_variabel**

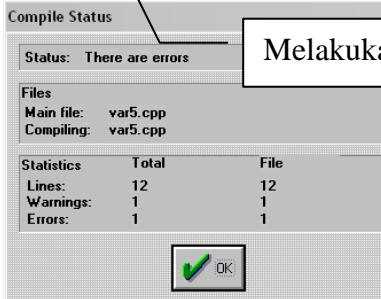
#### contoh

```
#include <iostream.h>
#define phi 3.14
main()
{
    cout<<"isi phi = "<<phi;
```

Pendeklarasian  
konstanta phi  
dengan nilai 3.14



Program di atas jika dilakukan perubahan nilai phi akan terjadi kesalahan

```
#include <iostream.h>
#define phi 3.14
main()
{ phi=3;
  cout<<"isi phi = "<<phi;
}


phi sebagai konstanta



Melakukan perubahan nilai phi


```

### contoh

```
#include <iostream.h>
void main()
{ int a;
  const float phi=3.14;
  a=9;
  cout<<"isi a = "<<a<<endl;
  cout<<"isi phi = "<<phi<<endl;
}


phi sebagai konstanta


```

### 3.10 Overflow Data

Penentuan tipe data variabel dapat menimbulkan masalah jika tidak tepat dalam penentuan tipe data.. Salah satu pembeda antara tipe data yang satu dengan yang lain adalah kapasitas dari tipe data tersebut. Tipe data int mempunyai batasan nilai dari -32.767 sampai 32.767. Persoalan akan muncul jika isi dari variabel tersebut diisi dengan nilai diluar batas kapasitasnya. Perbedaan antara tipe data dengan isinya tidak akan memunculkan masalah tetapi akan menghasilkan nilai yang berbeda.

```
int a;
```

a=40000; → melebihi **daya tampung tipe** data integer.

```
#include <iostream.h>
void main()
{ int a;
a=40000;
cout<<"isi variabel a "<<a<<endl;
}

```

(Inactive D:\WAR5.EXE)

isi variabel a -25536

Kenapa hasilnya tidak isi variabel a 40000 ?

**Penjelasan :**

Hasil diatas terjadi karena variabel a yang bertipe int diisi dengan data 40000. Hal ini tentunya melebihi batas maksimal tipe data int yang hanya bisa menampung nilai maksimal 32.767.

**Ilustrasi :**

Apa berbedaan dari gambar di bawah ini :



Gambar 3.1. Ilustrasi perbedaan tipe data

- Perbedaan dari keempat gambar tersebut adalah dari sisi kapasitas air yang bisa tertampung. Gelas tentunya isi lebih sedikit dari botol dan yang paling banyak kapasitasnya adalah tandon air.
- Ukuran tempat yang digunakan tentunya mengikuti kapasitas air yang akan digunakan, jika ingin menyimpan air  $\frac{1}{2}$  liter cukup menggunakan gelas. Tandon air digunakan jika air yang disimpan misal sampai 250 liter.
- Persoalan akan muncul jika diinginkan menyimpan air 1 liter tetapi disimpan di gelas yang hanya bisa menampung air  $\frac{1}{2}$  liter. Apa yang terjadi... tentunya air yang tertampung hanya  $\frac{1}{2}$  liter dan sisanya akan meluber.



Gambar 3.2. Ilustrasi terjadinya overflow

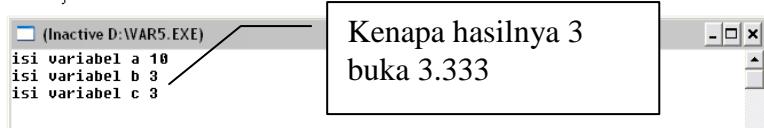
Ilustrasi tersebut mirip dengan casting di C++. Jika ada perbedaan tipe data maka akan dilakukan proses ‘penyesuaian’ bit sehingga hasilnya berbeda dengan nilai awalnya.

### 3.11 Konversi Tipe Data

Persoalan lain yang ada dalam perbedaan tipe data adalah adanya perbedaan hasil walaupun tipe data yang digunakan untuk menyimpan data sudah sesuai tipe datanya.

#### Contoh

```
#include <iostream.h>
void main()
{ int a,b,c;
a=10;
b=3;
c=a/b;
cout<<"isi variabel a "<<a<<endl;
cout<<"isi variabel b "<<b<<endl;
cout<<"isi variabel c "<<c<<endl;
}
```

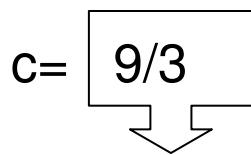


#### Penjelasan :

- Ketiga variabel tersebut mempunyai tipe data integer (bilangan bulat)
- $c=a/b$ ; → variabel c menyimpan hasil perhitungan  $a/9$  →  $9/3$
- Perhitungan manual tentunya menghasilkan 3.3333
- Dalam bahasa C++ → operasi  $a/b$  juga memperhatikan tipe data, karena a dan b bertipe integer maka proses pembagian juga menghasilkan bilangan integer (bulat)

$$C = \boxed{a/b}$$

Pembagian ini akan menghasilkan bilangan bulat, sehingga proses pembagian akan dilakukan pembulatan ke bawah.



semestinya menghasilkan 3.333 tetapi dilakukan pembulatan ke bawah sehingga hasilnya 3 dan disimpan di variabel c yang bertipe integer.

### Contoh

```
#include <iostream.h>
void main()
{   int a,b,c;
    a=9.1;
    b=9.5;
    c=9.8;
    cout<<"isi variabel a "<<a<<endl;
    cout<<"isi variabel b "<<b<<endl;
    cout<<"isi variabel c "<<c<<endl;
}
```



### Penjelasan

- Ketiga variabel tersebut mempunyai tipe data integer (bilangan bulat)
- Variabel a diisi 9.1 → hal ini tidak error tetapi nilai yang tersimpan di a adalah pembulatan dari 9.1 → 9
- Variabel c diisi 9.5 → hal ini tidak error tetapi nilai yang tersimpan di a adalah pembulatan ke bawah dari 9.5 → 9
- Variabel a diisi 9.8 → hal ini tidak error tetapi nilai yang tersimpan di a adalah pembulatan ke bawah dari 9.8 → 9

### Contoh

```
#include <iostream.h>
void main()
{   int a,b;
    float c;
    a=10;
    b=3;
    c=a/b;
    cout<<"isi variabel a "<<a<<endl;
    cout<<"isi variabel b "<<b<<endl;
    cout<<"isi variabel c "<<c<<endl;
}
```



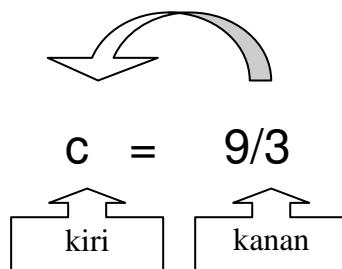
### Penjelasan

- Dua variabel mempunyai tipe data integer (bilangan bulat), yaitu variabel a dan b
- Variabel c bertipe float
- Dalam Operasi  $c=a/b$  menghasilkan 3 (disimpan di variabel c), kenapa 3 bukan 3.333 padahal c bertipe float (pecahan)?
- Hal terjadi karena hasil operasinya adalah 3 dan 3 inilah yang disimpan di c.

$$C = \boxed{9/3}$$

Pembagian ini akan menghasilkan bilangan bulat, sehingga proses pembagian akan dilakukan pembulatan ke bawah dan hasilnya disimpan di c. Walaupun variabel c bertipe float, hasil yang disimpan tetap 3 bukan 3.333

Jadi,..... hasil akhir adalah disebelah kanan dan bagian kiri hanya menampung hasil operasi



### Contoh

```
#include <iostream.h>
void main()
{   float a,b;
    int c;
    a=10;
    b=3;
    c=a/b;
    cout<<"isi variabel a "<<a<<endl;
    cout<<"isi variabel b "<<b<<endl;
    cout<<"isi variabel c "<<c<<endl;
}
```



### Penjelasan

- Dua variabel mempunyai tipe data float (bilangan pecahan), yaitu variabel a dan b
- Variabel c bertipe integer
- Dalam Operasi  $c=a/b$  menghasilkan 3 (disimpan di variabel c), kenapa 3 bukan 3.333 padahal c bertipe float (pecahan)?

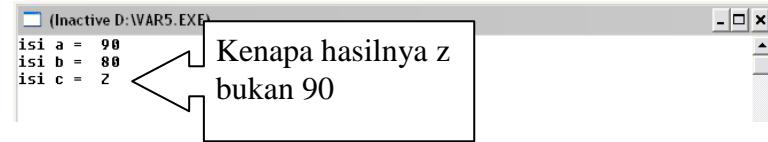
- Hal terjadi karena hasil operasinya sebenarnya 3.333 dan karena disimpan di variabel c yang bertipe integer maka hasilnya adalah 3 bukan 3.333

$$c = \boxed{9/3}$$

Pembagian ini sebenarnya menghasilkan bilangan pecahan (3.333), tetapi karena di simpan di c yang bertipe integer, maka hasilnya 3

### Contoh

```
#include <iostream.h>
main()
{ int a,b;
char c;
a=90;
b=80;
c=a;
cout<<"isi a = "<<a<<endl;
cout<<"isi b = "<<b<<endl;
cout<<"isi c = "<<c<<endl;
}
```



### Penjelasan

- Ada dua variabel bertipe integer, yaitu variabel a dan variabel b
- Satu variabel bertipe char, yaitu variabel c
- Variabel a diisi 90, variabel b diisi 80
- Variabel c diisi dari isi variabel a
- Di sini terjadi proses konversi → variabel c bertipe char sementara variabel a berisi integer
- Terjadi proses konversi variabel c tetapi menerima data dari variabel a (berisi 90) dan variabel c akan melakukan penyesuaian tipe, angka 90 akan diubah kebentuk bilangan ASCII). Bilangan ASCII 90 adalah z
- Sehingga variabel c akan menyimpan karakter z

### 3.12. Tipe Casting

Proses casting adalah merubah sementara tipe suatu data yang sudah didefinisikan. Dengan menggunakan tipe casting, tipe data yang sudah ada tidak perlu dilakukan modifikasi supaya menghasilkan data yang sesuai.

```
int x = 121;
float y;
y = x / 19;
```

maka y akan berisi nilai 6 karena operator pembagian akan menghasilkan pembulatan. Agar hasilnya sesuai yang diinginkan tanpa merubah tipe data digunakan typecasting, yaitu

mengubah suatu variabel/nilai sehingga menjadi type sesuai kemauan kita. Namun perubahan type ini hanya sementara.

```
int x = 121;
float y;
y = (float) x / (float) 19;
```

Sekarang y akan berisi 6.368421 karena operator pembagian' dilakukan pada 2 bilangan pecahan, maka hasilnya juga bilangan pecahan, tetapi variabel x tetap bertipe integer.

### Contoh

```
#include <iostream.h>
void main()
{   int a,b;
    float c;
    a=10;
    b=3;
    c=a/b;
    cout<<"isi variabel a "<<a<<endl;
    cout<<"isi variabel b "<<b<<endl;
    cout<<"isi variabel c "<<c<<endl;
}
```



Dari program di atas, hasil yang disimpan di variabel adalah 3 bukan 3.333 (lihat penjelasan sebelumnya). Persoalan bagaimana jika diinginkan hasilnya 3.333 bukan 3. Cara termudah adalah mengganti tipe data a dan b menjadi float.

### Contoh

```
#include <iostream.h>
void main()
{   float a,b;
    float c;
    a=10;
    b=3;
    c=a/b;
    cout<<"isi variabel a "<<a<<endl;
    cout<<"isi variabel b "<<b<<endl;
    cout<<"isi variabel c "<<c<<endl;
}
```

Semua tipe data variabel adalah float



Hasil di atas sudah menghasilkan hasil 3.333. Perbaikan di atas dilakukan dengan mengubah semua tipe data menjadi float.

```
float a,b;
float c;
```

Bagaimana solusinya, bila melakukan perubahan tipe data variabel di atas tidak diperkenankan. Agar permasalahan tersebut beres ...gunakan tipe casting. Bentuk umum perintah ini adalah

### Tipe\_data data

#### Contoh

```
#include <iostream.h>
void main()
{ int a,b;
float c;
a=10;
b=3;
c=float (a)/b;
cout<<"isi variabel a "<<a<<endl;
cout<<"isi variabel b "<<b<<endl;
cout<<"isi variabel c "<<c<<endl;
}
```



#### Penjelasan

- Variabel a dan b bertipe integer
- Variabel c bertipe c
- Variabel a diisi dengan 10, variabel b diisi 3
- c= float (a)/b → variabel a yang bertipe integer untuk sementara di ubah menjadi float, sehingga pada proses ini nilai a yang semula integer (3) diganti menjadi 3.000 (pecahan). Akibat perubahan ini jika a/b akan menghasilkan bilangan pecahan → 3.333 dan hasilnya disimpan di variabel c yang bertipe float, sehingga isi variabel c adalah 3.333

#### Contoh

```
#include <iostream.h>
void main()
{ int a,b;
float c;
a=10;
b=3;
c=(float) a/b;
cout<<"isi variabel a "<<a<<endl;
cout<<"isi variabel b "<<b<<endl;
cout<<"isi variabel c "<<c<<endl;
}
```



### Penjelasan

- Perintah  $c=(\text{float}) a/b$  sama artinya dengan  $c=\text{float}(a)/b$

### Contoh

```
#include <iostream.h>
void main()
{
    int a,b;
    float c;
    a=10;
    b=3;
    c=float (a/b);
    cout<<"isi variabel a "<<a<<endl;
    cout<<"isi variabel b "<<b<<endl;
    cout<<"isi variabel c "<<c<<endl;
}
```

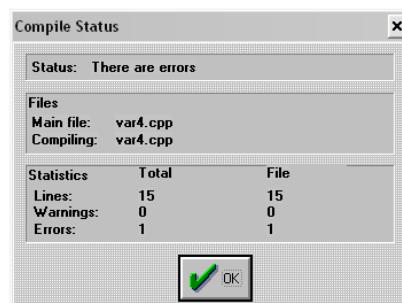


### Penjelasan

- Perintah  $\text{float } (a/b)$  mempunyai makna yang berbeda dengan contoh-contoh sebelumnya,  $\text{float } (a/b)$  artinya instruksi yang ada di dalam kurung akan diproses terlebih dahulu dan baru hasilnya dilakukan casting.
- Dalam contoh di atas, proses pertama adalah  $a/b \rightarrow 10/3$  karena a dan b bertipe integer maka hasilnya adalah 3. Hasil 3 inilah yang dilakukan proses casting sehingga hasilnya adalah 3 yang bertipe float. Hasil ini disimpan di variabel c yang juga bertipe pecahan.

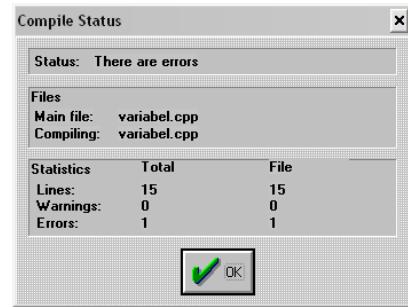
### 3.13. Contoh kesalahan

```
//contoh program variabel
#include <iostream.h>
void main()
{
    int a,b;
    a=10;
    b=15;
    c=5;
    cout<<"isi variabel a "<<a<<endl;
    cout<<"isi variabel b "<<b<<endl;
    cout<<"isi variabel c "<<c<<endl;
}
```



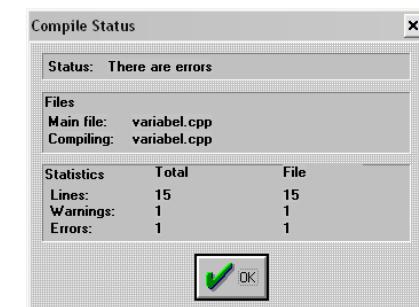
Program di atas terjadi kesalahan dikarenakan, dalam program terdapat pengisian variabel  $c=5$  sementara variabel c belum dideklarasikan.

```
//contoh program variabel
#include <iostream.h>
void main()
{ int a,b,c;
a=10;
b=5;
c=a+b
cout<<"isi variabel a "<<a<<endl;
cout<<"isi variabel b "<<b<<endl;
cout<<"isi variabel c "<<c<<endl;
}
```



```
//contoh program variabel
#include <iostream.h>
void main()
{ int a,b,c
a=10;
b=5;
a+b=c;
cout<<"isi variabel a "<<a<<endl;
cout<<"isi variabel b "<<b<<endl;
cout<<"isi variabel c "<<c<<endl;
}
```

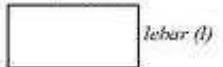
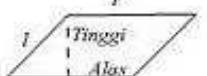
Operasi matematika harus  
disebelah kanan tanda =



Contoh program-program di bawah ini masih sederhana belum menggunakan perintah masukan (input). Program masih bersifat statis, program bila di jalankan akan selalu menghasilkan nilai yang sama dan bila menginginkan mengganti data maka harus dilakukan perubahan di dalam source program

### contoh

Buat program sederhana dari rumus di bawah ini

	<i>Nama Bangun</i>	<i>Rumus Keliling</i>	<i>Rumus Luas</i>
1	<i>Persegi</i> 	$4 \times \text{Sisi} = 4S$	$\text{Sisi} \times \text{Sisi} = S^2$
2	<i>PERSIPI PANJANG</i> 	$(2 \times P) + (2 \times l)$ $2 \times (P+l)$	$\text{Panjang} \times \text{lebar} = Pl$
3	<i>JAJARAN GENJANG</i> 	$(2 \times P) + (2 \times l)$ $2 \times (P+l)$	$\text{alas} \times \text{tinggi} = at$

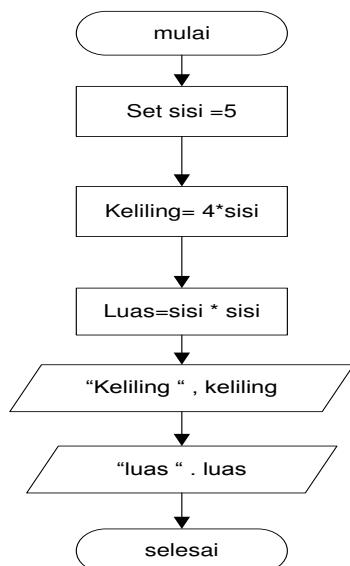
**Penyelesaian**

$$\text{Keliling} = 4 \times \text{sisi}$$

$$\text{Luas} = \text{sisi} \times \text{sisi}$$

**Algoritma**

1. Set variabel sisi
2. Hitung keliling =  $4 \times \text{sisi}$
3. Hitung luas =  $\text{sisi} \times \text{sisi}$
4. Tampilkan hasil perhitungan keliling
5. Tampilkan hasil perhitungan luas

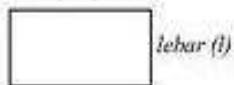
**Flowchart :****Program**

```
#include <iostream.h>
void main()
{
    int sisi, keliling, luas;
    sisi=6;
    keliling=4*sisi;
    luas= sisi*sisi;
    cout<<"Keliling = "<<keliling<<endl;
    cout<<"Luas      = "<<luas<<endl;
}
```

```

D:\ (Inactive D:\VAR5.EXE)
Keliling = 24
Luas      = 36
  
```

## 2 PERSEGI PANJANG

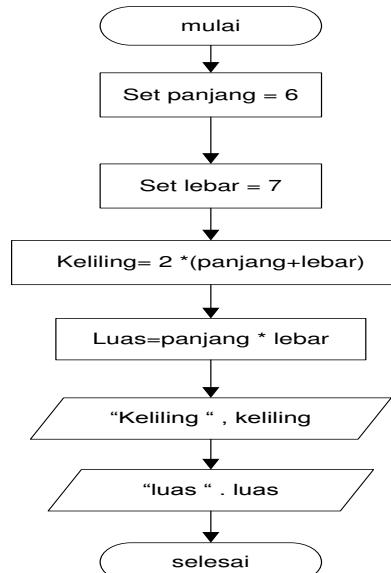
Panjang ( $P$ )

$$\text{Keliling} = 2 \times (P+l)$$

$$\text{Luas} = P \times l$$

**Algoritma**

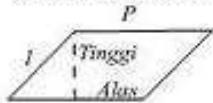
1. Set variabel panjang
2. Set variabel lebar
3. Hitung keliling=2 x (p+l)
4. Hitung luas=panjang \* lebar
5. Tampilkan hasil perhitungan keliling
6. Tampilkan hasil perhitungan luas

**Flowchart :****Program**

```

#include <iostream.h>
void main()
{
    int panjang, lebar, keliling, luas;
    panjang=6;
    lebar=7;
    keliling=2*(panjang+lebar);
    luas= panjang*lebar;
    cout<<"Keliling = "<<keliling<<endl;
    cout<<"Luas      = "<<luas<<endl;
}
  
```



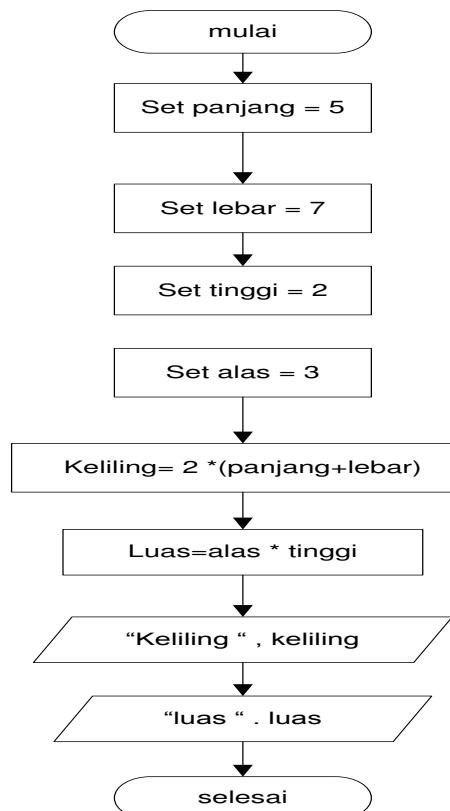
**3 JAJARAN GENJANG**

$$\text{Keliling} = 2 \times (\text{panjang} + \text{lebar})$$

$$\text{Luas} = \text{alas} \times \text{tinggi}$$

**Algoritma**

1. Set variabel panjang
2. Set variabel lebar
3. Set variabel tinggi
4. Set variabel alas
5. Hitung keliling =  $2 \times (\text{panjang} + \text{lebar})$
6. Hitung luas =  $\text{alas} \times \text{tinggi}$
7. Tampilkan hasil perhitungan keliling
8. Tampilkan hasil perhitungan luas

**Flowchart :**

```
#include <iostream.h>
void main()
{
    int panjang,lebar,tinggi,alas,keliling,luas;
    panjang=5;
    lebar=7;
    tinggi=2;
    alas=3;
    keliling=2*(panjang+lebar);
    luas= alas*tinggi;
    cout<<"Keliling = "<<keliling<<endl;
    cout<<"Luas      = "<<luas<<endl;
}
```



### 3.14 Latihan

1. Program di bawah ini, bila dijalankan apa hasilnya

```
#include <iostream.h>
#include <iomanip.h>
void main(void)
{ int a,b,c;
  float d,e,f;
  a=5.1;
  b=10.4;
  c=a+b;
  d=10.4;
  e=25.33;
  f=d+a;
  cout <<"ISI VARIABEL A = " <<a<<endl;
  cout <<"ISI VARIABEL B = " <<b<<endl;
  cout <<"ISI VARIABEL C = " <<c<<endl;
  cout <<"ISI VARIABEL D = " <<d<<endl;
  cout <<"ISI VARIABEL E = " <<e<<endl;
  cout <<"ISI VARIABEL F = " <<f<<endl; }
```

2. Buat diagram alir konversi dari Celcius ke fahrenheit dan reamur

Rumus

$$R = \frac{4}{5} * C$$

$$F = \frac{9}{5} * C$$

Dimana

$$R = \text{REAMUR}$$

$$C = \text{Celcius}$$

$$F = \text{fahreinheit}$$

3. Untuk menghitung luas trapesium sama kaki, Rumus-nya adalah

$$LT = \frac{1}{2} H * (AL + AT)$$

Dimana

$$AT = AL - 2.H \cotag S$$

$$LS = \frac{1}{2} H \cdot AL$$

Keterangan

$$LT = \text{Luas trapesium}$$

$$H = \text{Tinggi}$$

$$S = \text{Sudut alas}$$

$$AL = \text{Panjang alas}$$

$$AT = \text{Panjang atas}$$

$$LS = \text{Luas segitiga}$$

4. Hitung luas dan keliling lingkaran yang diketahui jari-jarinya.

RUMUS

$$K = 2 \pi R$$

$$L = \pi R$$

Dimana

$$K = \text{Keliling lingkaran}$$

$$R = \text{Jari-jari}$$

$$L = \text{Luas lingkaran}$$

$$\pi = 3.14$$

# 4

## OPERATOR DAN UNGKAPAN



## Tujuan Instruksional

Setelah membaca bab ini, diharapkan pembaca memahami macam-macam operator yang sering digunakan dalam pemrograman, seperti operator penugasan, operator aritmetika, operator relasi dan operator-operator lainnya serta dapat mengimplementasikan aplikasi-aplikasi yang terkait dengan operator dalam pemrograman C++

## Materi

Operator penugasan

Operator aritmatika

Operator Penaik dan Penurun di c++

Operator Relasi

Operator Majemuk

Urutan Operasi

#### 4.1. Operator penugasan

Operator merupakan simbol yang bisa digunakan untuk melakukan suatu operasi. Misal

- Melakukan proses matematika (penjumlahan dan lainnya)
- Memberikan nilai ke suatu variabel
- Membandingkan dua bahan nilai atau lebih.

#### Macam – macam operator:

1. **Operator Unary** adalah operator yang hanya membutuhkan satu operand saja.  
Contoh: tanda negative (-).
2. **Operator Binary** adalah operator yang membutuhkan dua operand, dan antara dua operand itu membutuhkan operator.  
Contoh operator tersebut di antaranya adalah tanda plus (+) atau tanda kali (\*).
3. **Operator Aritmatika** adalah operator yang sering kali kita jumpai dalam kalkulator. Contoh Operator tersebut di antaranya adalah tanda perkalian (\*), tanda pembagian (/), tanda plus yang berfungsi untuk penambahan (+), dan tanda kurang (-).
4. **Operator Relational** adalah operator yang digunakan untuk membandingkan antara dua buah nilai atau variable. Operator tersebut di antaranya adalah lebih besar (>), lebih kecil (<), lebih besar atau sama dengan (>=), lebih kecil atau sama dengan (<=), dan sama nilai dengan (==). Hasil dari operator ini adalah true atau false.
5. **Operator Logika** adalah operator yang digunakan untuk menghubungkan dua buah ungkapan kondisi menjadi satu ungkapan kondisi. Operator tersebut di antaranya adalah operator AND, operator OR, dan operator NOT. Sama halnya dengan operator relational, operator logika juga menghasilkan true atau false.

#### Assignment (=).

Operator *assignment* digunakan untuk memberikan nilai ke suatu variable. Operator Penugasan (Assignment operator) dalam bahasa C++ berupa tanda sama dengan “=”).

Contoh :

```
nilai = 80;  
A = x * y;
```

#### Penjelasan :

variable “nilai” diisi dengan 80 dan

variable “A” diisi dengan hasil perkalian antara x dan y.

Sisi kiri dari operator disebut *value* (left value) dan sisi kanan disebut *rvalue* (right value). *lvalue* harus selalu berupa variabel dan sisi kanan dapat berupa konstanta, variabel, hasil dari suatu operasi atau kombinasi dari semuanya.

**Contoh:**

```
int a, b;
a = 10;
b = 4;
a = b;
b = 7;
```

Hasil dari contoh diatas, **a** bernilai **4** dan **b** bernilai **7**.

**4.2. Operator aritmatika**

Operator	Deskripsi	Contoh
+	Penjumlahan ( Add )	$m + n$
-	Pengurangan ( Subtract )	$m - n$
*	Perkalian ( Multiply )	$m * n$
/	Pembagian ( Divide )	$m / n$
%	Sisa Pembagian Integer ( Modulus )	$m \% n$
-	Negasi ( Negate )	$-m$

```
//contoh program matematika sederhana
#include <iostream.h>
void main()
{ int a,b,c;
a=10;
b=15;
c=a+b;
c=a-b;
c=a*b;
c=a/b;
c=a%b;
cout<<"isi variabel c "<<c<<endl;
}
```

**Penjelasan :**

- Variabel a diisi dengan nilai 10
- Variabel a diisi dengan nilai 10
- $c=a+b \rightarrow$  artinya variabel c diisi dengan operasi  $a+b \rightarrow c=25$
- $c=a-b \rightarrow$  artinya variabel c diisi dengan operasi  $a-b \rightarrow c=-5$
- $c=a*b \rightarrow$  artinya variabel c diisi dengan operasi  $a*b \rightarrow c=150$
- $c=a/b \rightarrow$  artinya variabel c diisi dengan operasi  $a/b \rightarrow c=0$
- $c=a\%b \rightarrow$  artinya variabel c diisi dengan operasi  $a\%b \rightarrow c=10$
- Sehingga nilai c terakhir dan yang ditampilkan ke monitor adalah 10

Bandingkan dengan program di bawah ini, kenapa hasilnya berbeda

```
#include <iostream.h>
void main()
{ int a,b,c;
a=10;
b=15;
c=a+b;
cout<<"isi variabel c "<<c<<endl;
c=a-b;
cout<<"isi variabel c "<<c<<endl;
c=a*b;
cout<<"isi variabel c "<<c<<endl;
c=a/b;
cout<<"isi variabel c "<<c<<endl;
c=a%b;
cout<<"isi variabel c "<<c<<endl;
}
```

(Inactive D:\MAT1.EXE)  
isi variabel c 25  
isi variabel c -5  
isi variabel c 150  
isi variabel c 0  
isi variabel c 10

Operator sisa pembagian (%) digunakan untuk mencari sis pembagian dari suatu perhitungan. Operator ini diterapkan untuk operand bertipe integer. Contoh pembagian ini diantaranya.

9 % 2 ➔ 1	Sisa pembagian bilangan 9 dengan 2 adalah 1, artinya 9 dibagi 2 yang terdekat adalah 8 tetapi masih ada sisa 1
9 % 3 ➔ 0	Sisa pembagian bilangan 9 dengan 3 adalah 0, artinya 9 dibagi 3 yang terdekat adalah 9 dan tentunya sisa 0
25 % 3 ➔ 1	Sisa pembagian bilangan 25 dengan 3 adalah 1, artinya 25 dibagi 3 yang terdekat adalah 24 tetapi masih ada sisa 1
30%7 ➔ 2	Sisa pembagian bilangan 30 dengan 7 adalah 2, artinya 30 dibagi 7 yang terdekat adalah 28 tetapi masih ada sisa 2

```
//contoh program variabel
#include <iostream.h>
void main()
{ int a,b,c,d,e;
a=30;
b=a%2;
c=a%3;
d=a%4;
e=a%5;
cout<<"isi variabel a "<<a<<endl;
cout<<"isi variabel b "<<b<<endl;
cout<<"isi variabel c "<<c<<endl;
cout<<"isi variabel d "<<d<<endl;
cout<<"isi variabel e "<<e<<endl;
}
```

(Inactive D:\VARIABEL.... - □ x)  
isi variabel a 30  
isi variabel b 0  
isi variabel c 0  
isi variabel d 2  
isi variabel e 0

**Penjelasan :**

- $b=30\%2 \rightarrow 30$  dibagi 2 menghasilkan nilai bulat 15 dan tentunya sisanya 0
- $c=30\%3 \rightarrow 30$  dibagi 3 menghasilkan nilai bulat 10 dan tentunya sisanya 0
- $d=30\%4 \rightarrow 30$  dibagi 2 menghasilkan nilai pecahan, pembagian dengan 4 yang terdekat  $28/4 = 7$  dan tentunya sisanya 2
- $e=30\%5 \rightarrow 30$  dibagi 3 menghasilkan nilai bulat 6 dan tentunya sisanya 0

```
//contoh program variabel
#include <iostream.h>
void main()
{
    cout<<"27 % 2 = "<<27 % 2<<endl;
    cout<<"27 % 3 = "<<27 % 3<<endl;
    cout<<"27 % 4 = "<<27 % 4<<endl;
    cout<<"27 % 5 = "<<27 % 5<<endl;
    cout<<"27 % 6 = "<<27 % 6<<endl;
}
```

**Penjelasan :**

- $27 \% 2$  menghasilkan bilangan pecahan, pembagian dengan 2 yang terdekat adalah 26 (26 dibagi 2 =14 → bulat ) dan tentunya sisa 1 ( $27-26=1$ )
- $27 \% 3$  menghasilkan bilangan bulat  $27$  dibagi  $3 = 9 \rightarrow$  bulat dan tentunya sisa 0
- $27 \% 4$  menghasilkan bilangan pecahan, pembagian dengan 4 yang terdekat adalah 24 (24 dibagi 3 =8 → bulat ) dan tentunya sisa 3 ( $27-24=3$ )
- $27 \% 5$  menghasilkan bilangan pecahan, pembagian dengan 5 yang terdekat adalah 25 (25 dibagi 5 =5 → bulat ) dan tentunya sisa 2 ( $27-25=2$ )
- $27 \% 6$  menghasilkan bilangan pecahan, pembagian dengan 6 yang terdekat adalah 24 (24 dibagi 4 =6 → bulat ) dan tentunya sisa 3 ( $27-24=3$ )

**4.3. Operator Penaik dan Penurun di c++**

Masih berkaitan dengan operator pemberi nilai, C++ menyediakan operator penambah dan pengurang. Dari contoh penulisan operator pemberi nilai sebagai penyederhanaannya dapat digunakan operator penambah dan pengurang.

Operator	Contoh	Keterangan
<code>++</code>	<code>op++</code>	Op dinaikkan nilainya 1 <u>setelah</u> dilakukan operasi pada op
<code>++</code>	<code>++op</code>	Op dinaikkan nilainya 1 <u>sebelum</u> dilakukan operasi pada op
<code>--</code>	<code>op--</code>	Op diturunkan nilainya 1 <u>setelah</u> dilakukan operasi pada op
<code>--</code>	<code>--op</code>	Op diturunkan nilainya 1 <u>sebelum</u> dilakukan operasi pada op
<code>-</code>	<code>-op</code>	Menegaskan nilai op menjadi positif jika negatif atau sebaliknya

**Contoh :**

**A = A + 1** atau **A = A - 1**; disederhanakan menjadi **A ++** atau **A --**  
 Operator “ ++ ” atau “ -- ” dapat diletakkan didepan atau di belakang variabel.

**X=X+1**  
**y=y-1**  
 dapat ditulis

**X++**  
**y--**

atau

**++X**  
**--y**

```
//contoh program penugasan
#include <iostream.h>
void main()
{
    int x = 10;
    int y = 20;
    cout<<"isi variabel x = "<<x <<endl;
    cout<<"isi variabel y = "<<y <<endl;
    y=++x;
    x=y++;
    cout<<"isi variabel x++ = "<<x <<endl;
    cout<<"isi variabel ++y = "<<y <<endl;
}
```

**Penjelasan :**

- variabel x diset 10
- variabel y diset 20
- Perintah **y=++x**, berarti nilai y adalah penjumlahan dari nilai x yang terlebih dahulu ditambahkan
  - **x=x+1** sehingga isi x=11
  - **y=x** sehingga isi y=11
- Perintah **x=y++**, berarti nilai y adalah penjumlahan dari nilai x yang diisi dengan y
  - **x=y** sehingga isi x=11
  - **y=y+1** sehingga isi y=12

```
//contoh program penugasan
#include <iostream.h>
void main()
{
    int x = 10;
    int y;

    y= 10+ x++;

    cout<<"isi variabel x = "<<x <<endl;
    cout<<"isi variabel y = "<<y <<endl;
}
```



#### Penjelasan :

- variabel x diset 10
- Perintah  $y=10+ x++$ , berarti nilai y adalah penjumlahan dari nilai x terlebih dahulu ditambahkan setelah itu melakukan penjumlahan x
  - $y=10+x \rightarrow y=20$  kemudian variabel  $x=x+1 \rightarrow x=11$

```
//contoh program penugasan
#include <iostream.h>
void main()
{
    int x = 10;
    int y;

    y= 10+ ++x;

    cout<<"isi variabel x = "<<x <<endl;
    cout<<"isi variabel y = "<<y <<endl;
}
```



#### Penjelasan :

- variabel x diset 10
- Perintah  $y=10+ ++x$ , berarti nilai y adalah penjumlahan dari nilai x yang sudah ditambahkan terlebih
  - $y=10+ (x=+1) \rightarrow y=10+(11)$  sehingga  $y=21$  dan  $x=11$

#### 4.4. Operator Relasi

Operator Relasi digunakan untuk membandingkan dua buah nilai. Hasil perbandingan operator ini menghasilkan nilai numerik 1 (True) atau 0 (False).

Operator	Keterangan
==	Sama Dengan ( bukan pemberi nilai )
!=	Tidak Sama dengan
>	Lebih Dari
<	Kurang Dari
>=	Lebih Dari sama dengan
<=	Kurang Dari sama dengan

```
//contoh program penugasan
#include <iostream.h>
void main()
{ float a, b, c, d, e, f, x, y;
x=30;
y=40;
a = x == y;
b = x != y;
c = x > y;
d = x < y;
e = x >= y;
f = x <= y;
cout<<endl;
cout<<"Hasil dari "<<x<<" == "<<y<<" = "<<a<<endl;
cout<<"Hasil dari "<<x<<" != "<<y<<" = "<<b<<endl;
cout<<"Hasil dari "<<x<<" > "<<y<<" = "<<c<<endl;
cout<<"Hasil dari "<<x<<" < "<<y<<" = "<<d<<endl;
cout<<"Hasil dari "<<x<<" >= "<<y<<" = "<<e<<endl;
cout<<"Hasil dari "<<x<<" <= "<<y<<" = "<<f<<endl;
}
```

```
Hasil dari 30 == 40 = 0
Hasil dari 30 != 40 = 1
Hasil dari 30 > 40 = 0
Hasil dari 30 < 40 = 1
Hasil dari 30 >= 40 = 0
Hasil dari 30 <= 40 = 1
```

#### Penjelasan :

- variabel x diset 10 dan y diset 40
- a = x==y, artinya perintah menanyakan apakah nilai x sama dengan nilai y, karena memang nilai tidak sama maka hasilnya adalah salah yang direpresentasikan dengan angka 0
- a = x !=y, artinya perintah menanyakan apakah nilai x tidak sama dengan nilai y, karena memang nilai tidak sama maka hasilnya adalah benar yang direpresentasikan dengan angka 1

#### 4.5. Operator Majemuk

Operator majemuk digunakan untuk memendekkan penulisan operasi seperti :

```
x=x+2
y=y*2
z=z-2
```

Bisa disingkat menjadi

```
x+=2
y *=2
z-=2
```

```
//contoh program penugasan
#include <iostream.h>
void main()
{ int x=4;
    cout<<"Hasil dari = "<<x<<endl;
    x++;
    cout<<"Hasil dari = "<<x<<endl;
    x+=2;
    cout<<"Hasil dari = "<<x<<endl;
    x-=2;
    cout<<"Hasil dari = "<<x<<endl;
}
```



#### Penjelasan :

- variabel x diset 4
- x++ artinya nilai x ditambah 1 → x=x+1 sehingga x isi adalah 5
- x+=2 artinya nilai x ditambah 2 → x=x+2 sehingga x isi adalah 5+2 = 7 (bukan 4+2) → ingat nilai yang disimpan adalah nilai x terakhir (5)
- x-=2 artinya nilai x dikurangi 2 → x=x-2 sehingga x isi adalah 7-2 = 5 (bukan 4=-2) → ingat nilai yang disimpan adalah nilai x terakhir (7)

#### 4.6. Urutan Operasi

Dalam proses penulisan rumus matematika perlu diperhatikan penulisan notasi dari rumus dasar menjadi rumus matematika.

Contoh

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Bagaimana menulis notasi dalam program

Beberapa alternatif :

$$x = (-b + \text{pow}(b^b - 4*a*c, 0.5)) / (2*a)$$

atau dibuat bagian pembagian, misal mencari pembilang dan penyebut sendiri-sendiri

$$p = -b + \text{pow}(b^b - 4*a*c, 0.5)$$

$$q = 2*a$$

$$x = p/q$$

atau

$$p = \text{pow}(b^b - 4*a*c, 0.5)$$

$$q = 2*a$$

$$x = (-b*p)/q$$

**contoh :**

$$\frac{A}{B} C \quad \text{bukan} \quad \frac{A}{BC} \quad \text{tapi} \quad \frac{AC}{B} \quad \rightarrow \text{program} \quad A*C/B$$

$$A-B/C \quad \text{bukan} \quad \frac{A-B}{C} \quad \text{tapi} \quad A - \frac{B}{C} \quad \rightarrow \text{program} \quad A - B/C$$

$$\frac{(A-B)}{(C-D)*E} \quad \rightarrow \quad (A-B)/((C-D)*E)$$

$$\frac{(X-Y)}{2} + \frac{(L+M)}{N} \quad \rightarrow \quad ((X-Y)/2) + ((L+M)/N)$$

Yang perlu diperhatikan, operasi matematika mempunyai urutan prioritas, sehingga penulisan seperti ini akan mempunyai arti yang berbeda. Urutan yang harus dikerjakan adalah:

1. Tanda kurung
2. Perkalian
3. Pembagian
4. Modulus
5. Penjumlahan
6. Pengurangan

### Contoh

Jika ada rumus sebagai berikut

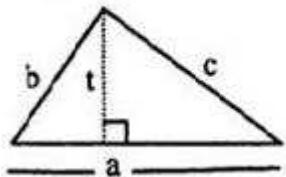
$$\frac{(X-Y)}{2} + \frac{(L+M)}{N}$$

Bagaimana programnya

```
#include <conio.h>
void main()
{ int x,y,l,m,n,hasil;
  x=10;
  y=5;
  l=2;
  m=2;
  n=4;
  hasil=((x-y)/2) + ((l+m)/n);
  cout<<"hasil = "<<hasil;
  getch();
}
```

### Contoh

#### Segitiga Siku-Siku



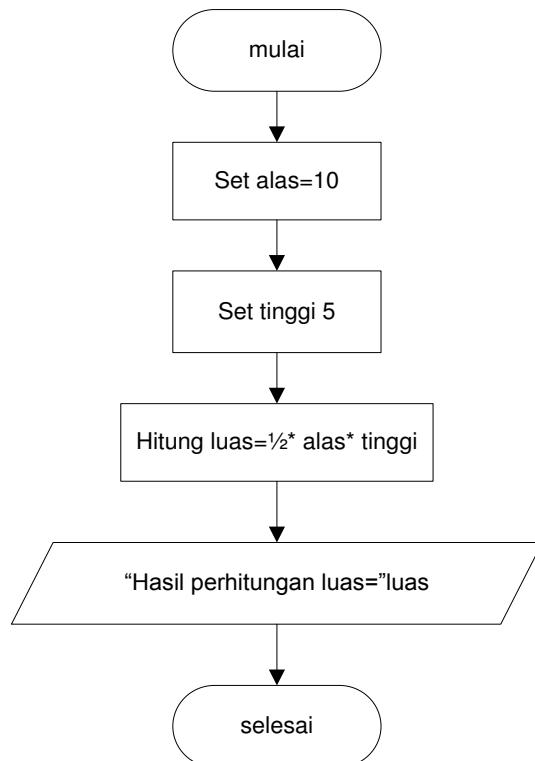
- ◎ Luas =  $\frac{1}{2} \times a \times t$
- ◎ Keliling =  $a + b + c$

Bagaimana untuk membuat program yang dapat mencari Luas dan keliling suatu segitiga

Sebelum membuat program sebaiknya membuat suatu alur dari langkah-langkah untuk menyelesaikan suatu persoalan. Langkah-langkah untuk membuat program bisa dituangkan dalam bentuk cerita (algoritma) atau menggunakan simbol (flowchart)

Algoritma mencari luas dan keliling segitiga

1. Masukan/minta data alas
2. Masukan/minta data tinggi
3. Car luas →  $\frac{1}{2} \times \text{alas} \times \text{tinggi}$
4. Tampilkan hasil perhitungan tersebut

**Flowchart**

```
//contoh program matematika sederhana
#include <iostream.h>
void main()
{ float luas,alas,tinggi;
  alas=10;
  tinggi=5;
  luas=1/2.0*alas*tinggi;
  cout<<"Hasil perhitungan luas "<<luas<<endl;
}
```



#### 4.7. Latihan

1. Hitung luas permukaan serta isi bola yang diketahui jari-jarinya

RUMUS

$$L = 4 \pi R$$

$$I = (4 \pi R^3) / 3$$

Dimana

$L$  = Luas bola

$R$  = Jari-jari bola

$I$  = isi bola

$\pi = 3.14$

2. Menentukan kecepatan serta jarak yang ditempuh pada gerak lurus berubah beraturan, bila diketahui kecepatan awal, percepatan dan waktunya.

RUMUS

$$V_t = V_0 + a t$$

$$S_t = V_0 t + 1/2 a t^2$$

Dimana

$V_t$  = Kecepatan saat  $t$

$V_0$  = kecepatan awal

$a$  = Percepatan

$t$  = waktu

$s_t$  = jarak sesudah  $t$

3. Buat rumus -rumus yang terkait dengan bidang untuk Kerucut

$$\text{Luas alas} = \pi r^2$$

$$\text{Luas selimut} = \pi r s$$

$$\text{Luas permukaan} = \pi r^2 + \pi r s$$

$$\text{Volume} = \frac{1}{3} \pi r^2 t$$

5

## PERINTAH MASUKAN DAN KELUARAN



## Tujuan Instruksional

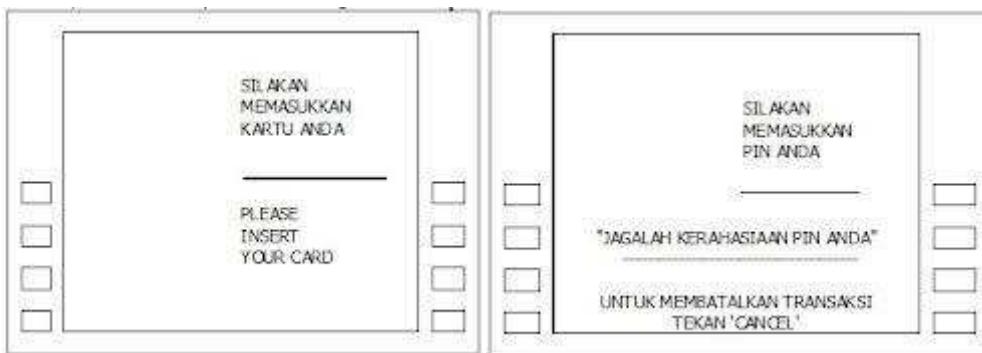
Setelah membaca bab ini, diharapkan pembaca memahami perintah masukan dan keluaran. Dengan memahami perintah ini diharapkan pembaca sudah mampu membuat program interaktif yang melibatkan pengguna dalam memasukkan data serta dapat menampilkan informasi kepada pengguna. Dalam bab ini juga dibahas fungsi manipulator yang digunakan untuk merapikan tampilan, fungsi matematika untuk membuat aplikasi matematika serta tipe data string yang digunakan untuk melakukan permintaan data yang berupa string

## Materi

-  Perintah Keluaran
-  Perintah Masukan
-  Fungsi Manipulator
-  Fungsi Matematika
-  String

### 5.1. Perintah Keluaran

Salah satu kriteria program yang baik adalah adanya komunikasi antara program dengan pengguna. Dengan adanya komunikasi tentunya antara pengguna dan program dapat saling berinteraksi sehingga tidak ada kesalahpahaman. Sebagai contoh program ATM, di sana ada dialog (komunikasi) antara program dengan pengguna. Program ATM dengan kalimatnya dapat ‘menyuruh’ pengguna untuk memasukan kartu, nomor PIN, jenis transaksi dan lainnya. Demikian juga program dapat menampilkan informasi saldo serta informasi-informasi lainnya. Dalam membangun suatu program proses komunikasi ini dapat menggunakan perintah I/O.



Gambar contoh tampilan masukan/ keluaran

Dalam contoh program di atas, pengguna akan memahami komunikasi yang ada dalam program tersebut. Bisa terbayangkan bila dalam program tidak ada komunikasi apapun. Proses yang digunakan untuk melakukan komunikasi ini dapat menggunakan perintah input maupun perintah output dan proses memasukan data dapat menggunakan keyboard serta proses untuk menampilkan hasil proses dapat ditampilkan di layar monitor.

#### Keluaran ( cout )

Penggunaan cout stream dihubungkan dengan operator overloaded << (Sepasang tanda "less than"). Contoh :

<code>cout &lt;&lt; "Selamat datang";</code>	Menampilkan tulisan Selamat datang
<code>cout &lt;&lt; 120;</code>	Menampilkan angka 120
<code>cout &lt;&lt; x;</code>	Menampilkan isi variabel x
<code>Cout&lt;&lt;1+2</code>	Menampilkan hasil penjumlahan 1+2

Operator << dikenal sebagai insertion operator, dimana berfungsi untuk menginput data yang mengikutinya. Jika berupa string, maka harus diapit dengan kutip ganda ("), sehingga membedakannya dari variable. Contoh :

<code>cout &lt;&lt; "Hello";</code>	Menampilkan tulisan Hello
<code>cout &lt;&lt; Hello;</code>	Menampilkan isi variabel Hello

```
//contoh program cout sederhana
#include <iostream.h>
void main()
{ int a,b,c;
a=10;
b=15;
cout<<a+b<<endl;
cout<<"a"<<endl;
cout<<a<<endl;
cout<<10*12;
}
```



### Penjelasan :

cout<<a+b<<endl;	➔	mencetak hasil penjumlahan antara variabel a dengan variabel b
cout<<"a" <<endl;	➔	mencetak string a (apa yang diapit tanda petik dua (" ")) akan ditampilkan apa adanya
cout<<a <<endl;	➔	mencetak isi dari variabel a
cout<<10*12 <<endl;	➔	mencetak hasil perkalian 10 dengan 12

Operator insertion (<<) dapat digunakan lebih dari 1 kali dalam kalimat yang sama, Contoh :

```
cout << "Hello, " << "I am " << "a C++ sentence";
```

Contoh diatas akan menampilkan Hello, I am a C++ sentence pada layar monitor. Manfaat dari pengulangan penggunaan operator insertion (<<) adalah untuk menampilkan kombinasi dari satu variabel dan konstanta atau lebih

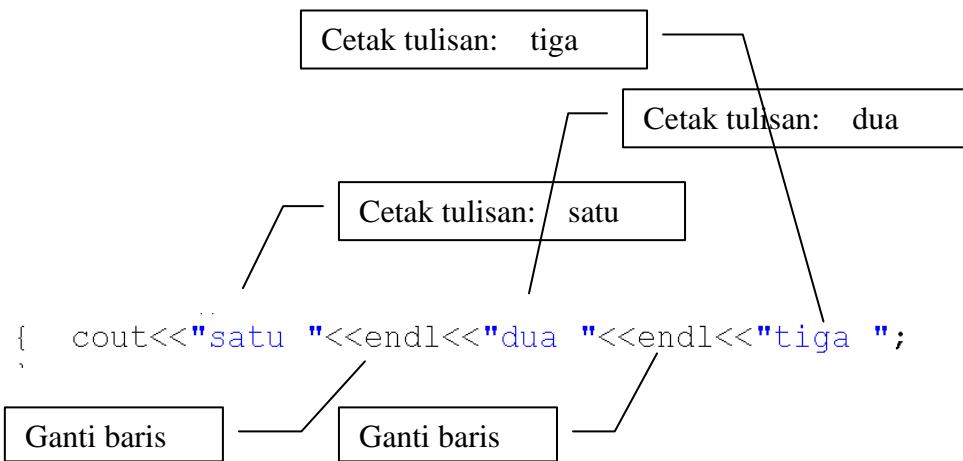
```
//contoh program cout sederhana
#include <iostream.h>
void main()
{ cout<<"satu "<<"dua "<<"tiga ";
```



### Penjelasan

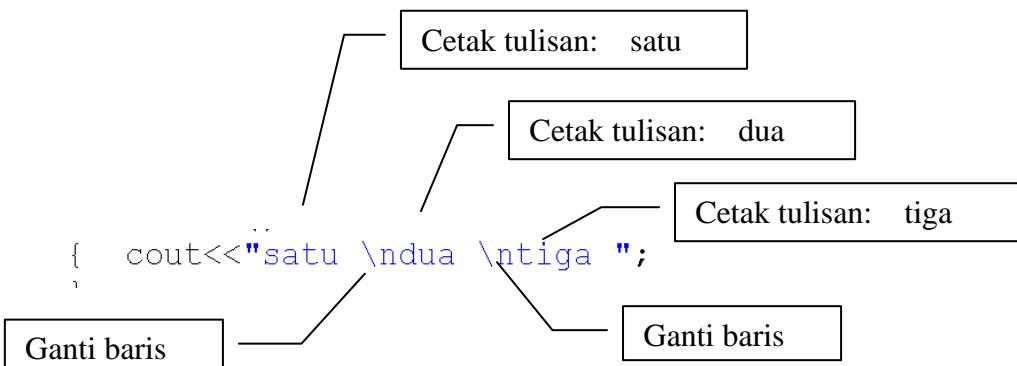
Perintah cout<<"satu "<<"dua "<<"tiga " sama saja dengan perintah cout<<"satu dua tiga ";

```
//contoh program cout sederhana
#include <iostream.h>
void main()
{   cout<<"satu "<<endl<<"dua "<<endl<<"tiga ";
}
```

**Penjelasan**

Atau → cetak **satu** kemudian ganti baris cetak **dua** kemudian ganti baris dan cetak **tiga**, perintah lain untuk ganti baris adalah \n

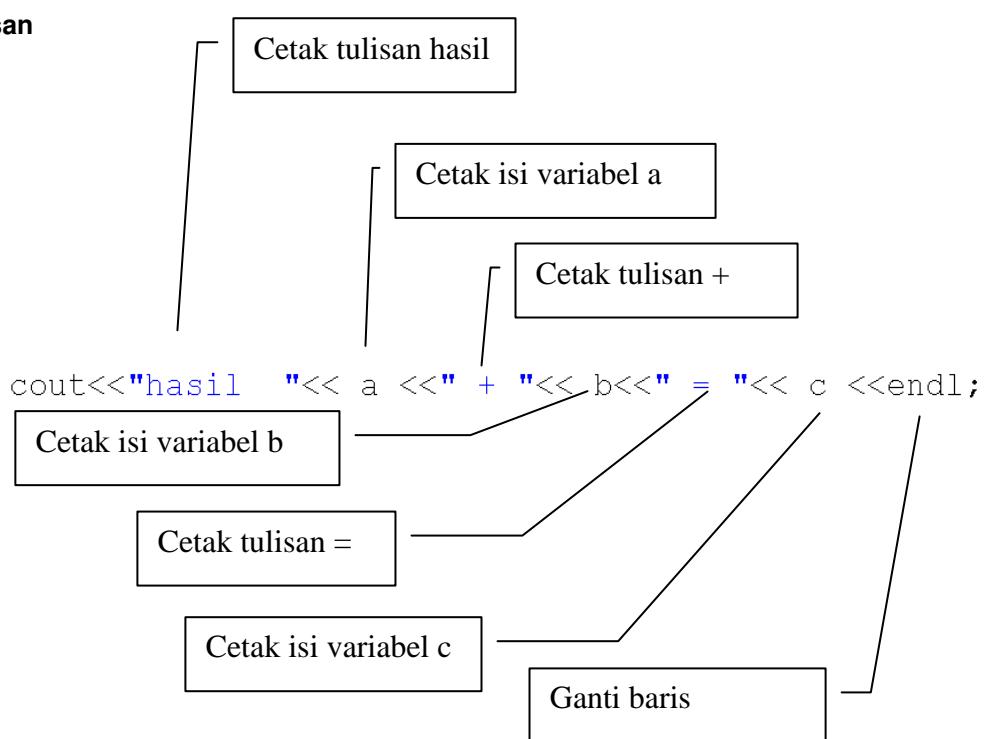
```
//contoh program cout sederhana
#include <iostream.h>
void main()
{   cout<<"satu \n dua \n tiga ";
}
```

**Penjelasan :**

```
//contoh program cout sederhana
#include <iostream.h>
void main()
{ int a,b,c;
a=10;
b=15;
c=a+b;
cout<<"hasil " << a << " + " << b << " = " << c << endl;
```



### Penjelasan



```
//contoh program cout sederhana
#include <iostream.h>
void main()
{ int a,b,c;
a=10;
b=15;
c=a+b;
cout<<"hasil " << a << " + " << b << " = " << c << endl;
```



## 5.2. Perintah Masukan

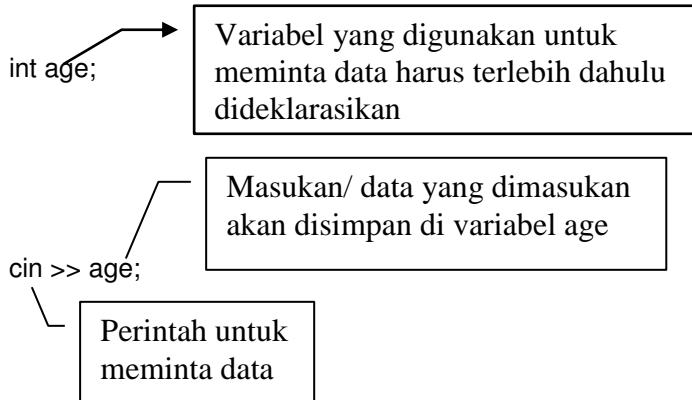
### Input ( cin ).

Menangani standard input pada C++ dengan menambahkan overloaded operator *extraction* (>>) pada **cin** stream. Harus diikuti dengan variable yang akan menyimpan data. Contoh :

Dalam C++, perintah **cin** digunakan untuk menginput suatu nilai dari suatu piranti masukan (*keyboard*) untuk selanjutnya di proses oleh program.

Sintaknya yaitu :

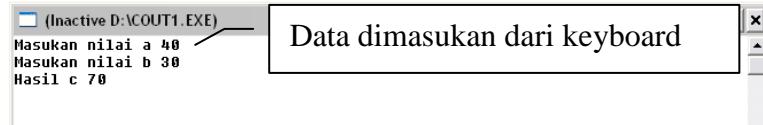
**cin>>variable;**



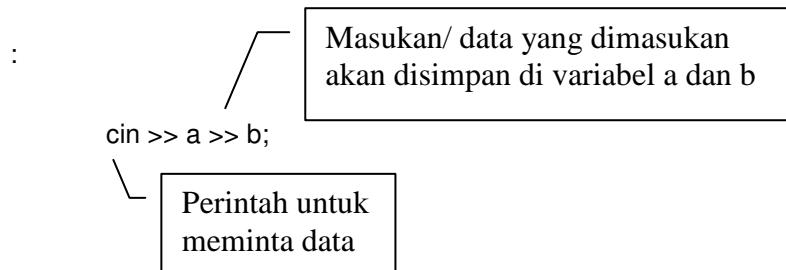
Contoh diatas mendeklarasikan variabel **age** dengan tipe **int** dan menunggu input dari **cin** (*keyboard*) untuk disimpan di variabel **age**.

**cin** akan memproses input dari keyboard sekali saja dan tombol ENTER harus ditekan.

```
//contoh program cin sederhana
#include <iostream.h>
void main()
{ int a,b,c;
    cout<<"Masukan nilai a ";
    cin>>a;
    cout<<"Masukan nilai b ";
    cin>>b;
    c=a+b;
    cout<<"Hasil c "<<c<<endl;;
}
```



**cin** juga dapat digunakan untuk lebih dari satu input



Ekuivalen dengan :

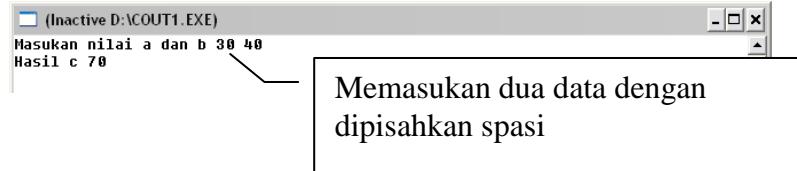
```

    cin >> a;
    cin >> b;
  
```

Dalam hal ini data yang di input harus 2, satu untuk variabel **a** dan lainnya untuk variabel **b** yang penulisannya dipisahkan dengan : spasi, tabular atau *newline*.

```

//contoh program cin sederhana
#include <iostream.h>
void main()
{   int a,b,c;
    cout<<"Masukan nilai a dan b ";
    cin>>a>>b;
    c=a+b;
    cout<<"Hasil c "<<c<<endl;;
}
  
```



Memasukan dua data dengan  
dipisahkan spasi

### 5.3. Fungsi Manipulator

Manipulator pada umumnya digunakan untuk mengatur tampilan layar, untuk menggunakan manipulator ini file header yang harus disertakan file header **iomanip.h**. Ada beberapa fungsi manipulator yang disediakan oleh C++, antara lain.

Fungsi	Keterangan	Jenis Manipulator
dec	Mengubah data numerik ke dalam desimal	Input & output
endl	Output sebuah karakter newline dan flush streamnya	Output
ends	Output sebuah null	Output
flush	Mengirim data langsung ke standard output	Output
hex	Mengubah data numerik ke dalam bentuk heksadesimal	Input & output
od	Mengubah data numerik ke dalam bentuk oktal	Input & output
setiosflags (long a)	Mematikan fungsi flags yang dispesifikasikan dalam a	Input & output
setbase (int a)	Memformat data ke basis a	Output
setfill (int ch)	Mengatur karakter pemenuh	Output
setprecision (int a)	Menetapkan presisi bilangan pecahan sebanyak n digit	Output
setw(int a)	Menetapkan lebar tampilan data sebesar a	Output
WS	Skip leading whitespace	Input

#### **setw**

Setw merupakan satu fungsi manipulator yang digunakan untuk mengatur lebar tampilan di layar dari suatu nilai variable. File header yang harus disertakan adalah file header **iomanip.h**. dimana bentuk penulisannya adalah:

**Setw (int n);**

Dimana n = merupakan nilai lebar tampilan data, integer.

### Contoh

```
# include <iostream.h>
# include <iomanip.h>

void main()
{long int a, b, c;
 a = 1050;
 b = 20500;
 c = 303450;
 cout<<"isi a = "<<a<<endl;
 cout<<"isi b = "<<b<<endl;
 cout<<"isi c = "<<c<<endl;
 cout<<"isi a = "<<setw(7)<<a<<endl;
 cout<<"isi b = "<<setw(7)<<b<<endl;
 cout<<"isi c = "<<setw(7)<<c<<endl;
```

}

Hasil tidak menggunakan setw

Hasil menggunakan setw

### Penjelasan

`setw(7)`, artinya disediakan tempat untuk menampung digit sebanyak 7 dan akan dibuat rata kanan.

1	2	3	4	5	6	7
			1	0	5	0
		2	0	5	0	0
	3	0	3	4	5	0

→ Lebar sebanyak 7 ←

### Contoh

```
# include <iostream.h>
# include <iomanip.h>
void main()
{long int a,b,c,d,e;
 a= 1050;
 b= 20500;
 c= 303450;
 d= 12345;
 e= 12000;
 cout<<"isi a = "<<a<<endl;
 cout<<"isi b = "<<b<<endl;
 cout<<"isi c = "<<c<<endl;
 cout<<"isi a = "<<setw(1)<<a<<endl;
 cout<<"isi b = "<<setw(2)<<b<<endl;
 cout<<"isi c = "<<setw(3)<<c<<endl;
}
```

```
isi a = 1050
isi b = 20500
isi c = 303450
isi a = 1050
isi b = 20500
isi c = 303450
```

### Penjelasan

Jika pendefinisian setw terlalu kecil maka setw tidak memberikan efek apapun. Dalam contoh di atas setw diset sebanyak 3 (untuk variabel c, cout<<"isi c = "<<setw(3)<<c<<endl;) dan bilangan yang ditampung ada 6 digit (303450) maka hasil setw tidak berpengaruh (hasilnya tetap rata kanan)

### setfill

Setfill merupakan suatu fungsi manipulator yang digunakan untuk menampilkan suatu karakter yang diletakkan di depan nilai yang diatur oleh fungsi setw(). File header yang harus disertakan adalah file header iomanip.h.

Bentuk penulisannya:

### Setfill (character);

```
# include <iostream.h>
# include <iomanip.h>
void main()
{long int a,b,c;
 a= 1050;
 b= 20500;
 c= 303400;

 cout<<"isi a = "<<a<<endl;
 cout<<"isi b = "<<setw(7)<<b<<endl;
 cout<<setfill('*');
 cout<<"isi b = "<<setw(7)<<b<<endl;
}
```

Hasil tidak menggunakan setfill
isi a = 1050 isi b = ***20500

Hasil menggunakan setfill
isi b = **20500

### Penjelasan

Penggunaan setfill akan memberikan tanda \* (tergantung dari pendefinisian di setfill) pada kolom yang tidak diisi dengan nilai digit. Jadi bila ada kelebihan kolom pada pendefinisian setw akan diisi dengan karakter yang sudah ditentukan.

Perintah setfill('\*) artinya jika ada kekosongan digit maka akan diisi dengan tanda \*

1	2	3	4	5	6	7
			1	0	5	0
		2	0	5	0	0
*	*	2	0	5	0	0

→ Lebar sebanyak 7 ←

### setiosflags()

Manipulator setiosflag() merupakan manipulator yang dapat dipakai untuk mengontrol sejumlah tanda format yang tercantum dalam tabel berikut

Tanda Format	Keterangan
ios::left	Menyetel rata kiri terhadap lebar field yang diatur melalui setw()
ios::right	Menyetel rata kanan terhadap lebar field yang diatur melalui setw()
ios::scientific	Memformat keluaran dalam notasi eksponensial
ios::fixed	Memformat keluaran dalam bentuk notasi desimal
ios::dec	Memformat keluaran dalam basis 10 (desimal)
ios::oct	Memformat keluaran basis 8 (oktal)
ios::hex	Memformat huruf dalam basis 16 (heksadesimal)
ios::uppercase	Memformat huruf pada notasi heksadesimal dalam bentuk huruf kapital
ios::showbase	Menampilkan awalan 0x untuk bilangan heksadesimal atau 0 (nol) untuk bilangan oktal
ios::showpoint	Menampilkan titik desimal pada bilangan pecahan yang tidak memiliki bagian pecahan
ios::showpos	Untuk menampilkan tanda + pada bilangan positif

### Tanda Format `ios::showpos`

Untuk menampilkan tanda + pada bilangan positif

```
# include <iostream.h>
# include <iomanip.h>
void main()
{long int a,b,c;
a= 1050;
b= 20500;
cout<<"isi a = "<<a<<endl;
cout<<"isi b = "<<setw(7)<<b<<endl;
cout<<setiosflags(ios::showpos) ;
cout<<"isi b = "<<setw(7)<<b<<endl;
}
```



### Penjelasan :

Pada baris `cout<<"isi b = "<<setw(7)<<b<<endl;` hasilnya tidak ada tanda + karena tidak ada instruksi `cout<<setiosflags(ios::showpos) ;` sedangkan pada baris `cout<<"isi b = "<<setw(7)<<b<<endl;` hasilnya terdapat tanda + karena sebelum baris ini sudah ada instruksi `setiosflags(ios::showpos)`

### `ios::scientific` dan `ios::fixed`

**ios::scientific** digunakan untuk memformat keluaran dalam notasi eksponensial sedangkan **ios::fixed** Memformat keluaran dalam bentuk notasi desimal

```
# include <iostream.h>
# include <iomanip.h>
void main()
{float a;
a= 1234567.1234;
cout<<"isi a = "<<a<<endl;
cout<<setiosflags(ios::scientific) ;
cout<<"isi a = "<<a<<endl;
cout<<setiosflags(ios::fixed) ;
cout<<"isi a = "<<a<<endl;
}
```



### Penjelasan :

<code>cout&lt;&lt;"isi a = "&lt;&lt;a&lt;&lt;endl;</code>	➔ ditampilkan dalam format eksponensial
<code>cout&lt;&lt;setiosflags(ios::scientific) ;</code>	➔ ditampilkan dalam format eksponensial
<code>cout&lt;&lt;"isi a = "&lt;&lt;a&lt;&lt;endl;</code>	➔ ditampilkan dalam format eksponensial
<code>cout&lt;&lt;setiosflags(ios::fixed) ;</code>	➔ ditampilkan dalam format fixed
<code>cout&lt;&lt;"isi a = "&lt;&lt;a&lt;&lt;endl;</code>	➔ ditampilkan dalam format fixed

**setprecision ()**

Fungsi setprecision() merupakan suatu fungsi manipulator yang digunakan untuk mengatur jumlah digit desimal yang ingin ditampilkan. Fungsi ini biasa pada fungsi cout(), file header yang harus disertakan adalah file header iomanip.h .

Bentuk penulisannya:

**setprecision (n)**

```
# include <conio.h>
# include <iostream.h>
# include <iomanip.h>

void main()
{float a, b, c;
 a = 30.45;
 b = 40.12;
 clrscr();
 c = a * b;
 cout<<setiosflags(ios::fixed);
 cout<<setprecision(1)<<c<<endl;
 cout<<setprecision(2)<<c<<endl;
 cout<<setprecision(3)<<c<<endl;
 cout<<setprecision(4)<<c<<endl;
 cout<<setprecision(5)<<c<<endl;
 getch();
}
```

```
D:\IRR.EXE
1221.7
1221.65
1221.654
1221.6541
1221.65405
```

**Penjelasan**

- Setprecision(1) ➔ akan menampilkan nilai di belakang koma sebanyak 1
- Setprecision(2) ➔ akan menampilkan nilai di belakang koma sebanyak 2
- Setprecision(3) ➔ akan menampilkan nilai di belakang koma sebanyak 3
- Setprecision(4) ➔ akan menampilkan nilai di belakang koma sebanyak 4
- Setprecision(5) ➔ akan menampilkan nilai di belakang koma sebanyak 5

#### 5.4. Fungsi Matematika

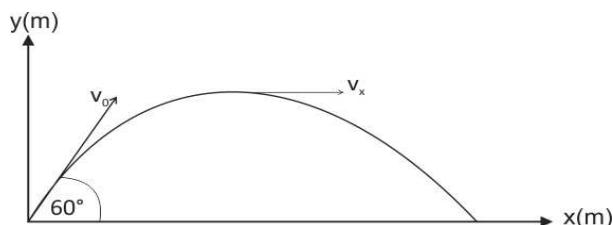
Dalam perhitungan rumus matematika atau fisika sering dijumpai perhitungan yang menggunakan sinus, cosines dan lainnya. Untuk melakukan perhitungan tersebut bahasa C sudah menyediakan beberapa fungsi bawaan. Function/fungsi adalah satu blok kode yang melakukan tugas tertentu atau satu blok instruksi yang di eksekusi ketika dipanggil dari bagian lain dalam suatu program. Header yang digunakan untuk perhitungan matematika adalah include <math.h>

##### Contoh

Hitung jarak peluru yang ditembakkan dari suatu lokasi dengan sudut penembakan ( $\alpha$ ) dan kecepatan ( $V_0$ ) dimasukan dari keyboard.

##### Rumus

$$X = \frac{V_0^2 \cdot \sin(\alpha) \cdot \cos(\alpha)}{g}$$



```
#include <iostream.h>
#include <math.h>
main()
{ float sudut,jarak,kecepatan,rad;
  const float PI=3.14;
  const float gravitasi=9.8;

  cout<<"Masukan nilai sudut penembakan ";
  cin>>sudut;
  cout<<"Masukan nilai kecepatan tembak ";
  cin>>kecepatan;
  rad=sudut*PI/180;
  jarak=2*pow(kecepatan,2)*sin(rad)*cos(rad)/gravitasi;
  cout<<"Jarak jatuh peluru = "<<jarak<<endl;
}
```



##### Penjelasan

- #include <math.h> → header ini digunakan karena ada fungsi untuk menghitung kuadrat (pow), sinus dan cosinus  
    **jarak=2\*pow(kecepatan,2)\*sin(rad)\*cos(rad)/gravitasi;**
- float sudut,jarak,kecepatan,rad;  
    → mendeklarasikan variabel-variabel yang akan digunakan dalam program
- const float PI=3.14;  
const float gravitasi=9.8;  
    → membuat 2 variabel sebagai nilai konstanta (tidak dapat diubah)

Beberapa contoh fungsi matematika diantaranya :

Fungsi	Keterangan	Bentuk umum
pow()	Digunakan untuk mencari nilai kuadrat $n^m \rightarrow \text{pow}(n,m)$	double pow(double x, double y)
abs()	Untuk memperoleh nilai absolut (mutlak) dari argumen x	int abs(int x)
acos()	Memperoleh nilai arc cosine	double acos(double x)
asin()	Memperoleh nilai arc sine	double asin(double x)
atan()	Memperoleh nilai arc tangen	double atan (double x)
cos()	Memperoleh nilai cos	double cos (double x)
rand()	Menghasilkan bilangan bulat acak yang terletak pada antara 0 sampai dengan nilai RAND_MAX Dimana RAND_MAX didefinisikan pada header stdlib.h	int rand(void)
random()	Menghasilkan bilangan bulat acak yang terletak pada antara 0 sampai dengan x-1	int random(int x)
sin()	Memperoleh nilai sin	double sin (double x)
sqrt()	Menghasilkan akar dari x	double sqrt(double x)
tan()	Memperoleh nilai tan	double tan (double x)

### Contoh

```
#include <iostream.h>
#include <math.h>
main()
{ float sudut,sin1,cos1,tan1;
cout<<"Masukan nilai sudut ";
cin>>sudut;
sin1=sin(sudut);
cout<<"sudut dalam sinus = "<<sin1<<endl;
cos1=cos(sudut);
cout<<"sudut dalam cosinus = "<<sin1<<endl;
tan1=tan(sudut);
cout<<"sudut dalam tangen = "<<tan1<<endl;
}
```



### Penjelasan

- `sin1=sin(sudut);`  
`cout<<"sudut dalam sinus = "<<sin1<<endl;`  
 $\rightarrow$  mencari sinus
- `cos1=cos(sudut);`  
`cout<<"sudut dalam cosinus = "<<sin1<<endl;`  
 $\rightarrow$  mencari cosinus
- `tan1=tan(sudut);`  
`cout<<"sudut dalam tangen = "<<tan1<<endl;`  
 $\rightarrow$  mencari tangen

**Contoh**

```
#include <iostream.h>
#include <math.h>
#include <stdlib.h>

main()
{ float acak,i;
  randomize();
  for(i=1;i<=5;i++)
  { acak=random(10);
    cout<<"hasil pengacakan = "<<acak<<endl;
  }
}
```

**Penjelasan**

- #include <stdlib.h> → digunakan untuk perintah randomize
  - randomize(); → digunakan untuk membangkitkan bilangan secara acak
  - for(i=1;i<=5;i++)
 { acak=random(10);
 cout<<"hasil pengacakan = "<<acak<<endl;
 }
- mengulang proses pengacakan 5 kali dengan menggunakan perulangan for. Mengacakan bilangan antara 0-10. Hasilnya adalah 1, 8, 9, 7, 5.
- Karena pengacakan dibatasi antara 0-10 maka hasilnya adalah menampilkan pengacakan antara 0-10
- Bandingkan dengan program di bawah ini

**contoh**

```
#include <iostream.h>
#include <math.h>
#include <stdlib.h>

main()
{ float acak,i;
  randomize();
  for(i=1;i<=5;i++)
  { acak=random(100);
    cout<<"hasil pengacakan = "<<acak<<endl;
  }
}
```



### Penjelasan

- ```
for(i=1;i<=5;i++)
{ acak=random(100);
    cout<<"hasil pengacakan = "<<acak<<endl;
}
```

mengacakan proses pengulangan dan melakukan pengacakan yang disimpan di variabel acak → { acak=random(100);  
Karena pengacakan dibatasi antara 0-100 maka hasilnya adalah menampilkan pengacakan antara 0-100

### contoh

```
#include <iostream.h>
#include <math.h>

main()
{ float kuadrat,i;
for(i=1;i<=5;i++)
{   kuadrat=pow(i,3);
    cout<<"hasil kuadrat = "<<i<<"^3 = "<<kuadrat<<endl;
}
}
```



### Penjelasan

- kuadrat=pow(i,3);**  
⇒ menghitung kuadrat 3 dari suatu bilangan (bilangan dari i)

## 5.5. String

Di samping variabel untuk menyimpan data bilangan, setiap bahasa pemrograman juga mengenal variabel String. String berarti deretan karakter. Sebuah string dapat mengandung sebuah karakter atau banyak mengandung karakter. Konstanta string ditulis dengan awalan dan akhiran tanpa tanda petik.

Contoh

| String         | Keterangan                                                           |
|----------------|----------------------------------------------------------------------|
| “AKPRIND”      | String dengan panjang 7 karakter berupa karakter huruf               |
| “28”           | String dengan panjang 2 karakter berupa karakter bilangan            |
| “Kalisahak 28” | String dengan panjang 12 karakter berupa karakter bilangan dan huruf |
| “”             | Tidak mengandung karakter                                            |

### Variabel String

Variabel string adalah variabel yang digunakan untuk menyimpan string. Deklarasi variabel string

```
char variabel_string[panjang_string]
```

Misal

```
char alamat[30];
```

Merupakan pernyataan yang mendefinisikan variabel string dengan panjang maksimal 30 karakter.

String pada C++ selalu diakhiri dengan karakter NULL(\0).sebagai contoh, jika terdapat string "YOGYAKARTA, maka representasi dalam memori dapat digambarkan

|   |   |   |   |   |   |   |   |   |   |    |
|---|---|---|---|---|---|---|---|---|---|----|
| Y | O | G | Y | A | K | A | R | T | A | \0 |
|---|---|---|---|---|---|---|---|---|---|----|

### Inisialisasi String

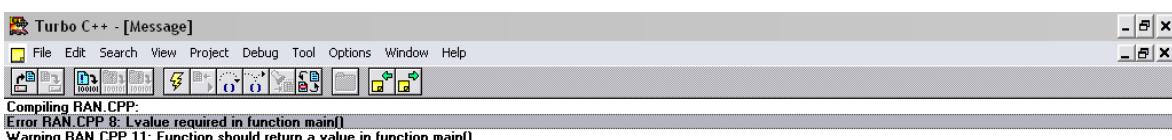
Pemberian nilai pada variabel string tidak bisa dilakukan seperti pemberian nilai pada variabel yang digunakan untuk menyimpan bilangan. Pemberian data secara langsung pada variabel string akan berakibat kesalahan. Agar variabel string mempunyai nilai awal, proses inisialisasinya dilakukan pada saat pendeklarasian variabel string tersebut.

**contoh :**

```
int a;
a=500;           ➔ benar
char nama[20];
nama="Raihan"; ➔ salah
```

```
char alamat[30]="Kalisahak 28"; ➔ benar;
```

```
#include <iostream.h>
main()
{ char nama[20];
  int a;
  a=70;
  nama="AKPRIND";
  cout<<"isi variabel a = "<<a<<endl;
  cout<<"isi variabel nama = "<<nama<<endl;
}
```



### Penjelasan :

Perintah nama="AKPRIND"; akan menimbulkan kesalahan, pemberian nilai pada variabel string tidak bisa dilakukan secara langsung seperti pada variabel yang digunakan untuk menyimpan bilangan.

### Contoh

```
#include <iostream.h>
main()
{   char nama[20] = "AKPRIND";
    int a;
    a=70;
    cout<<"isi variabel a = "<<a<<endl;
    cout<<"isi variabel nama = "<<nama<<endl;
}
```



### Penjelasan :

Proses pemberian data pada variabel string menggunakan proses inisialisasi pada waktu pendeklarasian variabel → char nama[20] = "AKPRIND";

### Membaca String dari Keyboard

cara lain untuk memberikan nilai pada variabel string dapat juga menggunakan perintah masukan (cin), tetapi proses penggunaan perintah cin berbeda dengan cin untuk meminta masukan data berupa bilangan.

```
#include <iostream.h>
main()
{   char nama[20] = "AKPRIND";
    int a;
    cout<<"Masukan data a = ";
    cin>>a;
    cout<<"Masukan data nama = ";
    cin>>nama;
    cout<<"isi variabel a = "<<a<<endl;
    cout<<"isi variabel nama = "<<nama<<endl;
}
```



### Penjelasan

Program di atas meminta data untuk variabel string (nama) dengan menggunakan perintah cin. Program di atas terlihat sudah benar, tetapi sebenarnya program akan menampilkan data yang tidak akan sama jika dimasukan data kalimat yang terdiri dari 2 kalimat atau lebih

### Contoh

```
#include <iostream.h>
main()
{ char nama[20] = "AKPRIND";
  int a;
  cout<<"Masukan data a = ";
  cin>>a;
  cout<<"Masukan data nama = ";
  cin>>nama;
  cout<<"isi variabel a = "<<a<<endl;
  cout<<"isi variabel nama = "<<nama<<endl;
}
```



### Penjelasan :

Hasil program di atas terjadi pemotongan data masukan yang disimpan pada variabel nama. Jadi bila menggunakan perintah cin untuk meminta data string, maka yang disimpan pada variabel tersebut adalah data sebelum tanda spas. Jika dimasukan akprind yogyakarta, maka yang disimpan adalah akprind

Agar kalimat yang digunakan tidak dilakukan pemotongan, perintah cin dapat dilengkapi dengan perintah **cin.getline(variabel\_string, sizeof(variabel\_string))**

contoh

```
#include <iostream.h>
#include <string.h>
int main()
{ char nama[20];
  char alamat[20];
  cout<<"Masukan nama pertama : ";
  cin.getline(nama, sizeof(nama));
  cout<<"Masukan nama kedua : ";
  cin.getline(alamat, sizeof(alamat));
  cout<<"Nama pertama : "<<nama<<endl;
  cout<<"Nama kedua : "<<alamat<<endl<<endl;
}
```

Perintah agar masukan lebih dari 1 kata tidak dipotong



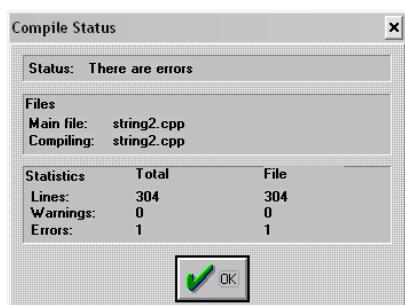
## 5.6 Fungsi String

Fungsi string biasa digunakan untuk melakukan proses manipulasi data-data string/kalimat. Dalam penggunaan fungsi string, header yang harus dipasang adalah string.h

| Fungsi   | Keterangan                                                                                                                                                                                                                                 | Bentuk umum                    |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| Strcat   | Fungsi ini digunakan untuk menggabungkan beberapa kata yang diinputkan. Untuk menggunakan fungsi ini anda harus menambahkan                                                                                                                | strcat(kata1,kata2);           |
| strcpy() | Dalam variabel string, proses penyalinan isi data tidak bisa dilakukan seperti dalam variabel yang digunakan untuk menyimpan data bilangan, perintah nama=alamat, tidak dikenal di variabel string. Agar proses penyalinan dapat dilakukan | strcpy(kata2, kata1);          |
| strlen() | Fungsi ini digunakan untuk memindahkan/menyalin data yang telah diinputkan oleh user dengan syarat tipe data dan besar karakternya harus sama. file header yang harus anda tambahkan:                                                      | strlen(variabel);              |
| strrev() | Fungsi ini digunakan untuk membalikkan string/karakter yang masuk.                                                                                                                                                                         | Input = ABC maka Outputnya CBA |
| strlwr() | Fungsi ini digunakan untuk mengubah huruf kapital pada string menjadi huruf kecil. file header yang harus anda tambahkan:                                                                                                                  | Strlwr(kata)                   |
| strupr() | Fungsi ini adalah kebalikan dari fungsi strlwr yaitu mengubah huruf kecil menjadi huruf kapital. file header yang harus anda tambahkan:                                                                                                    | strupr(kata);                  |

**Contoh:**

```
#include<iostream.h>
#include<string.h>
void main()
{
char nama1[20] = "AKPRIND";
char nama2[20];
nama2=nama1;
cout << "nama 1 = " << nama1 << endl;
cout << "nama 2 = " << nama2 << endl;
}
```



### Penjelasan

Program salah, karena ada perintah nama2=nama1;

### Contoh

```
#include<iostream.h>
#include<string.h>
void main()
{
char nama1[20] = "AKPRIND";
char nama2[20];
strcpy(nama2, nama1);
cout << "nama 1 = " << nama1 << endl;
cout << "nama 2 = " << nama2 << endl;
}
```



### Penjelasan

char nama1[20] = "AKPRIND";

char nama2[20];

→ Memesan 2 variabel string, nama1 diberi inisialisasi dan nama2 tidak  
strcpy(nama2, nama1);

→ digunakan untuk menyalin isi variabel nama1 ke variabel nama2

### Contoh:

```
#include<iostream.h>
#include<string.h>
void main()
{
char nama1[30] = "IST AKPRIND ";
char nama2[30] = "YOGYAKARTA";
strcat(nama1, nama2);
cout << "nama 1 = " << nama1 << endl;
cout << "nama 2 = " << nama2 << endl;
}
```



### Penjelasan

Perintah strcat(nama1,nama2) artinya menggabungkan isi variabel nama1 dengan nama2 dan hasilnya disimpan di variabel nama1

### Contoh

```
#include<iostream.h>
#include<string.h>
void main()
{
char nama1[30] = "ist akprind ";
char nama2[30] = "YOGYAKARTA";
strupr(nama1);
cout << "nama 1 = " << nama1 << endl;
cout << "nama 2 = " << nama2 << endl;
}
```



### Penjelasan

Perintah strupr(nama1) artinya mengubah isi variabel nama1 menjadi huruf besar semua dan hasilnya disimpan di variabel nama1

### contoh

```
#include<iostream.h>
#include<string.h>
void main()
{
char nama1[30] = "ist akprind ";
char nama2[30] = "YOGYAKARTA";
strlwr(nama2);
cout << "nama 1 = " << nama1 << endl;
cout << "nama 2 = " << nama2 << endl;
}
```

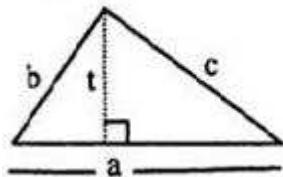


### Penjelasan

Perintah strlwr(nama1) artinya mengubah isi variabel nama1 menjadi huruf kecil semua dan hasilnya disimpan di variabel nama1

**Contoh**

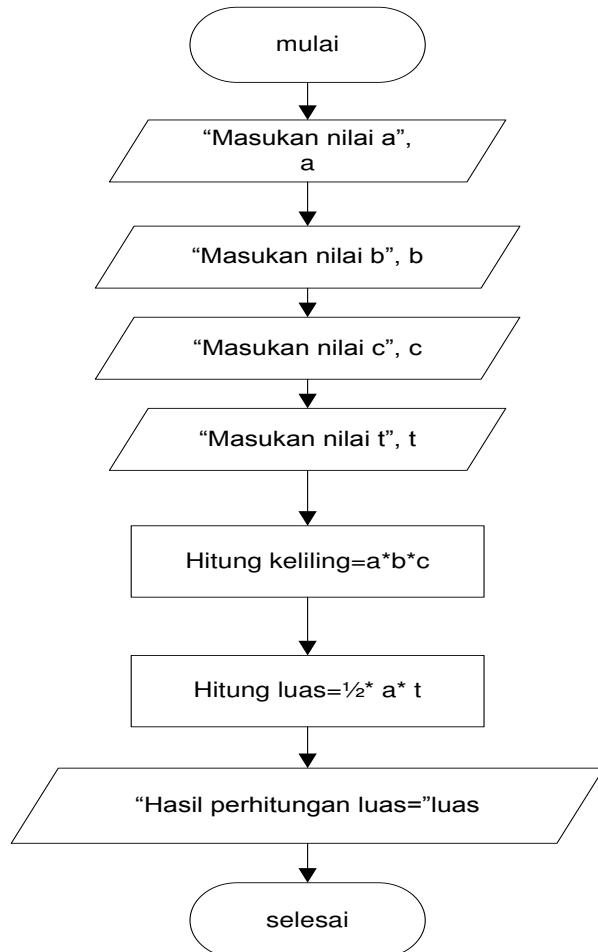
- Rumus –rumus Matematika

**Segitiga Siku-Siku**

• Luas =  $\frac{1}{2} \times a \times t$   
 • Keliling =  $a + b + c$

Algoritma mencari luas dan keliling segitiga

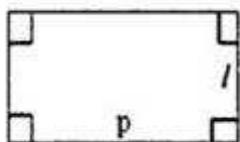
1. Masukan/minta data alas
2. Masukan/minta data tinggi
3. Cari luas  $\rightarrow \frac{1}{2} \times \text{alas} \times \text{tinggi}$
4. Cari keliling =  $a+b+c$
5. Tampilkan hasil perhitungan tersebut



```
//contoh program mencari luas dan keliling segitiga
#include <iostream.h>
void main()
{ float a,b,c,t,luas,keliling;
cout<<"Masukan nilai a ";
cin>>a;
cout<<"Masukan nilai b ";
cin>>b;
cout<<"Masukan nilai c ";
cin>>c;
cout<<"Masukan nilai tinggi ";
cin>>t;
luas=1/2.0*a*t;
keliling=a+b+c;
cout<<"Hasil luas "<<luas<<endl;
cout<<"Hasil keliling "<<keliling<<endl;
}
```

```
Masukan nilai a 4
Masukan nilai b 5
Masukan nilai c 6
Masukan nilai tinggi 2
Hasil luas 4
Hasil keliling 120
```

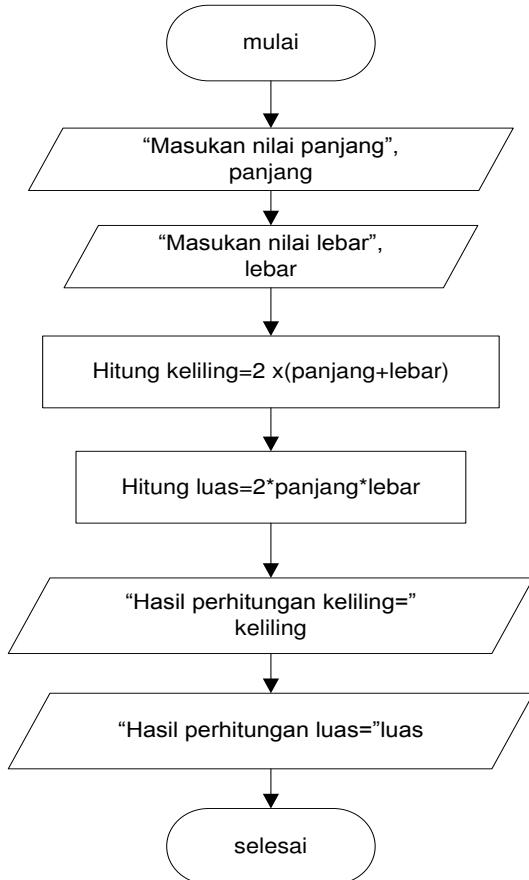
## Persegipanjang



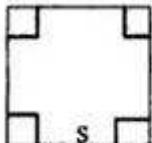
Luas =  $2 \times p \times l$   
 Keliling =  $2 \times (p + l)$

Algoritma mencari luas dan keliling segitiga

1. Masukan/minta data panjang
2. Masukan/minta data lebar
3. Cari luas →  $2 \times \text{panjang} \times \text{lebar}$
4. Cari keliling =  $2 \times (\text{panjang} + \text{lebar})$
5. Tampilkan hasil perhitungan tersebut



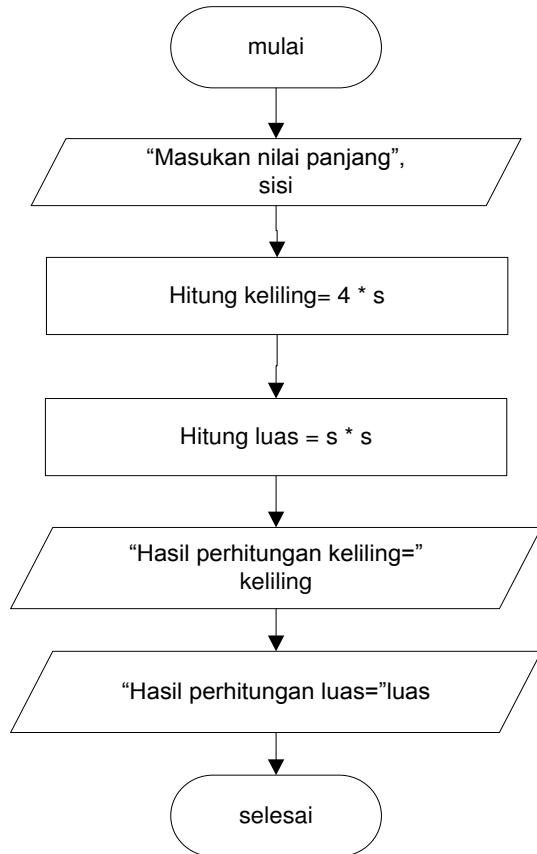
### Bujursangkar



|                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>● <math>\text{Luas} = s \times s</math></li> <li>● <math>\text{Keliling} = 4 \times s</math></li> </ul> |
|------------------------------------------------------------------------------------------------------------------------------------------------|

Algoritma mencari luas dan keliling segitiga

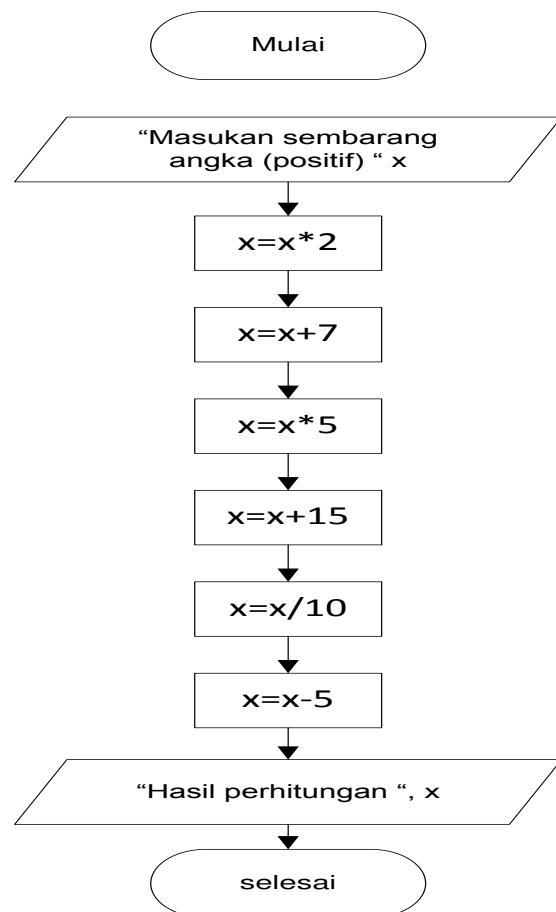
1. Masukan/minta data sisi
2. Cari luas → sisi \* sisi
3. Cari keliling = 4 \* sisi
4. Tampilkan hasil perhitungan tersebut



- Suatu bilangan akan menghasilkan bilangan yang sama jika dilakukan operasi perkalian dengan 2, kemudian hasilnya ditambah 7, lantas hasilnya dikalikan 5, kemudian tambahkan hasilnya dengan 15 , kemudian hasilnya bagi dengan 10 dan terakhir hasilnya kurangi 5 maka hasilnya terakhir adalah bilangan pertama yang dimasukan (khusus untuk bilangan positif):

## Logika

| Misal x = 17 | Bilangan awal (disimpan di variabel x) |
|--------------|----------------------------------------|
|              |                                        |
| 1 x=x*2      | x=17*2 → x sekarang berisi 34          |
| 2 x=x+7      | x=34+7 → x sekarang berisi 41          |
| 3 x=x*5      | x=41*5 → x sekarang berisi 205         |
| 4 x=x+15     | x=205+15 → x sekarang berisi 220       |
| 5 x=x/10     | x=220/10 → x sekarang berisi 22        |
| 6 x=x-5      | <b>x=22-5 → 17</b>                     |



### 5.7. Latihan

#### Buat flowchart dan program nya

1. Perhatikan permainan di bawah ini :

1. Sebutkan tanggal lahir Anda.
2. Kalikan 4 (empat),
3. Tambah 13 (tiga belas),
4. Kalikan 25 (dua puluh lima),
5. Kurangi 200 (dua ratus),
6. Tambah Bulan lahir Anda,
7. Kalikan 2 (dua),
8. Kurangi 40 (empat puluh),
9. Kalikan 50 (lima puluh),
10. Tambah dengan dua digit terakhir dari Tahun lahir Anda (Cth: 84),
11. Terakhir kurangi dengan 10.500 (sepuluh ribu lima ratus).

Dari langkah-langkah di atas, maka akan menghasilkan angka kelahiran

#### Logika

tgl = 10

bln = 12

th = 90

|    |               |        |
|----|---------------|--------|
| 1  | tgl=tgl*4     | 40     |
| 2  | tgl=tgl+13    | 53     |
| 3  | tgl=tgl*25    | 1325   |
| 4  | tgl=tgl-200   | 1125   |
| 5  | tgl=tgl+bln   | 1137   |
| 6  | tgl=tgl*25    | 2274   |
| 7  | tgl=tgl-40    | 2234   |
| 8  | tgl=tgl*50    | 111700 |
| 9  | tgl=tgl+th    | 111790 |
| 10 | tgl=tgl-10500 | 101290 |

2. Harga satuan saputangan adalah Rp. 3000 , jika membeli lusinan, harga 1 lusin Rp. 34.000.

Buat program yang dapat menghitung pembelian baik secara satuan maupun lusinan.

Misal membeli sebanyak 15 buah, berarti :

$$15 \text{ buah} = 1 \text{ lusin} * \text{Rp. } 34.000 + 3 \text{ satuan} * \text{Rp. } 3.000$$

**Logika :**

Berapa harga yang harus dibayar, jika ada pembelian sebanyak 25 saku tangan

1.  $1 \text{ lusin} = 12$
  2. 25 berarti 2 lusin dan lebih 1
  3. Untuk mencari banyaknya lusin, lakukan dengan pembagian bulat
  4. Mencari sisa → lakukan dengan pembagian sisa
  5.  $\text{Bayar\_lusin} = \text{jumlah\_lusin} * 3400$
  6.  $\text{Bayar\_biji} = \text{sisa\_biji} * 3000$
  7.  $\text{Total} = \text{bayar\_lusin} + \text{bayar\_biji}$
- 
3. Buat program yang dapat mengubah data detik menjadi jam, menit dan detik.

Misal ada data 4216 detik = 1 jam 10 menit 16 detik

$$3152 \text{ detik} = 0 \text{ jam } 53 \text{ menit } 32 \text{ detik}$$

**Logika :**

- Pahami rumus-rumus dasar
    - $1 \text{ jam} = 60 \text{ menit}$
    - $1 \text{ menit} = 60 \text{ detik}$
    - $1 \text{ jam} = 3600 \text{ detik}$
  - mencari jam berarti  $4216$  dibagi  $3600$  (pembagian bulat) =  $1.17$  tetapi ingat diambil bulatnya sehingga hasilnya  $1$  (artinya  $1$  jam)
  - kemudian cari menit, lakukan pembagian sisa  $4216$  dibagi  $3600$  (pembagian sisa) =  $616$
  - mencari menit berarti sisa menit dibagi  $60 \rightarrow 616$  dibagi  $60$  (pembagian bulat) =  $10.26$  tetapi ingat diambil bulatnya sehingga hasilnya  $10$  (artinya  $10$  menit)
  - mencari detik → Lakukan pembagian sisa  $616$  dibagi  $60$  (pembagian sisa) =  $16$
  - Ketemu  $1$  Jam,  $10$  menit dan  $16$  detik
- 
4. Gaji yang diterima pegawai terdiri dari gaji pokok, tunjangan dan upah lembur, besar tunjangan  $15\%$  dari gaji pokok, besar upah lembur perjam  $3\%$  dari gaji pokok. Buatlah program untuk menghitung gaji yang diterima pegawai. Masukan (Input). nomor pegawai, nama, gaji pegawai, jumlah jam lembur. Keluaran (Output): Gaji pokok, tunjangan, uang lembur, total gaji. Buat flowchart dan programnya

5. Buatlah program untuk membantu kasir swalayan untuk memisahkan pecahan uang kembalian menjadi 50.000, 20.000, 10.000, 5.000, 2000, 1000, 500,dan 100.

**Contoh tampilan:**

Jumlah Uang Kembali : Rp. 88800

Pecahan uang kembali:

1 Lembar 50.000

1 Lembar 20.000

1 Lembar 10.000

1 Lembar 5.000

1 Lembar 2000

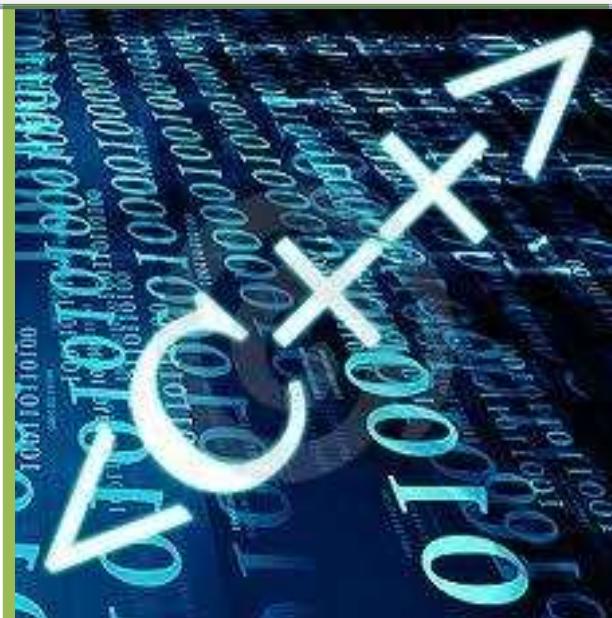
1 Lembar 1000

1 Lembar 500

1 Lembar 100

# 6

## OPERASI PENYELEKSIAN KONDISI



**B**ab ini membahas

- ❖ Struktur Bahasa C++

## Tujuan Instruksional

Setelah membaca bab ini, diharapkan pembaca memahami logika keputusan serta dapat mengimplementasikan persoalan keputusan dalam pemrogram. Pembaca diharapkan memahami berbagai macam bentuk keputusan mulai dari keputusan sederhana sampai keputusan bertingkat serta memahami berbagai perintah keputusan if, if-else serta case

## Materi

-  Pernyataan IF
-  Pernyataan IF - ELSE
-  Memahami keputusan dengan kondisi jamak
-  Pernyataan IF berkala
-  Pernyataan switch - case

Proses penyeleksian / pengambilan keputusan dalam kehidupan sehari-hari menjadi sesuatu yang sangat penting dalam menentukan suatu pilihan. Demikian juga dalam pemrograman pengambilan keputusan/ proses seleksi juga menjadi unsur yang sangat penting, sebagai contoh program untuk menentukan keabsahan password, menentukan saldo apakah masih mencukupi atau tidak. Demikian juga proses login dalam suatu aplikasi.

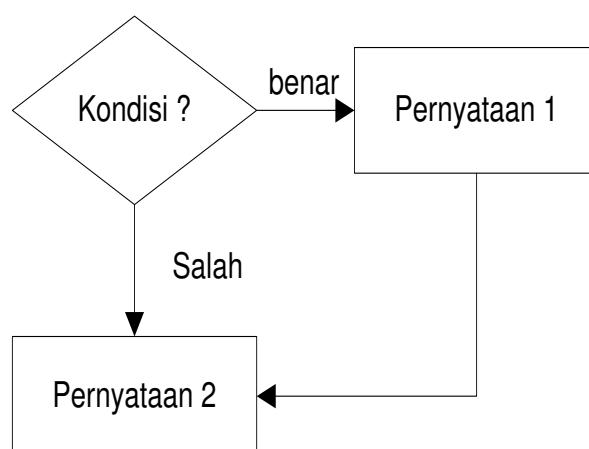


Gambar 6.1. Contoh program yang menggunakan keputusan

Pernyataan Percabangan digunakan untuk memecahkan persoalan untuk mengambil suatu keputusan diantara sekian pernyataan yang ada. Untuk keperluan pengambilan keputusan, C++ menyediakan beberapa perintah antara lain.

### 6.1. Pernyataan IF

Pernyataan if mempunyai pengertian, Jika kondisi bernilai benar, maka pernyataan 1 akan dikerjakan kemudian mengerjakan pernyataan 2 dan jika tidak memenuhi syarat (salah) maka akan mengerjakan pernyataan 2 Dari pengertian tersebut dapat dilihat dari diagram alir berikut:



Bentuk umum dari pernyataan if

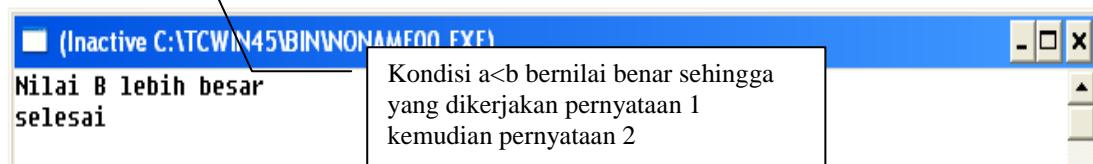
```
If (kondisi)
    Pernyataan 1
    Pernyataan 2
```

Atau

```
If (kondisi)
{
    Blok Pernyataan 1
}
Pernyataan 2
```

**Contoh 1 :**

```
#include <iostream.h>
void main(void)
{ int a=5,b=6;
if (a<b)
    cout<<"Nilai B lebih besar "<<endl;
    cout<<"selesai"<<endl;
}
```



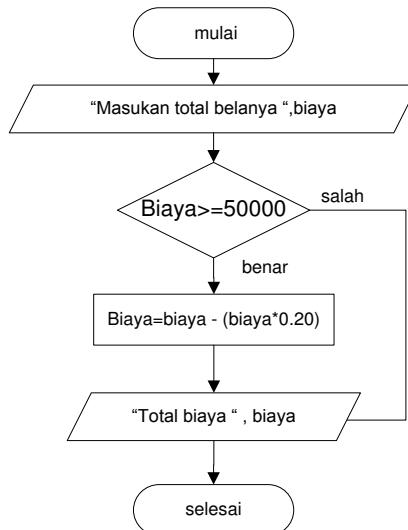
**Penjelasan :**

- Memberi nilai awal variabel a=5 dan variabel b=6
- Dalam program di atas
  - Kondisinya adalah apakah a<b
  - Pernyataan 1 adalah cout<<"nilai B lebih besar "<<endl
  - Pernyataan 2 adalah cout<<"selesai" <<endl
- If (a<b) artinya melakukan pengecekan apakah isi variabel a< b → benar
- Jika benar kerjakan pernyataan 1 dan kemudian kerjakan pernyataan 2
- Jika salah kerjakan pernyataan 2 .

### Contoh

Menentukan besarnya potongan dari pembelian barang yang diberikan seorang pembeli, dengan kriteria :

- Jika total pembelian lebih dari atau sama dengan Rp. 50.000,- potongan yang diterima sebesar 20% dari total pembelian.
- Tampilkan besar biaya yang dibayar



```

//contoh program keputusan
#include <iostream.h>
void main()
{ long potongan=0,biaya;
cout<<"Masukan total belanja= ";
cin>>biaya;
if (biaya>=50000)
    biaya=biaya-(biaya*0.20);
cout<<"Total bayar = "<<biaya<<endl;
}
  
```



### Penjelasan

- Variabel potongan diberi nilai awal dan variabel biaya tidak diberi nilai awal
- Dalam program di atas
  - Kondisinya adalah apakah  $biaya \geq 50000$
  - Pernyataan 1 adalah  $biaya = biaya - (biaya * 0.20)$ ;
  - Pernyataan 2 adalah  $cout \ll "Total bayar = " \ll biaya \ll endl;$
- Memasukan data biaya, misal dimasukan 40000
- Dilakukan pengecekan apakah  $biaya \geq 50000$ 
  - Apakah  $40000 \geq 50000 \rightarrow$  salah, sehingga langsung mengerjakan pernyataan 2

### Contoh

```
//contoh program keputusan
#include <iostream.h>
void main()
{ float a=10;
  if (a!=10)
    cout<<"satu"<<endl;
    cout<<"dua"<<endl;
    cout<<"tiga"<<endl;
}

```

Penyataan 1

Penyataan 2

dua

tiga

### Penjelasan

- Dalam program di atas
  - Kondisinya adalah apakah a tidak sama dengan 10
  - Pernyataan 1 adalah cout<<"satu"<<endl
  - Pernyataan 2 adalah cout<<"dua" <<endl; dan seterusnya
- Kondisi di atas menghasilkan nilai salah sehingga pernyataan yang dijalankan adalah pernyataan 2 dan baris berikutnya

### Contoh

```
//contoh program keputusan
#include <iostream.h>
void main()
{ float a=10;
  if (a==10)
    cout<<"satu"<<endl;
    cout<<"dua"<<endl;
    cout<<"tiga"<<endl;
}

```

Penyataan 1

Penyataan 2

satu

dua

tiga

### Penjelasan

- Dalam program di atas
  - Kondisinya adalah apakah a sama dengan 10
  - Pernyataan 1 adalah cout<<"satu"<<endl
  - Pernyataan 2 adalah cout<<"dua" <<endl; dan seterusnya
- Kondisi di atas menghasilkan nilai benar sehingga pernyataan yang dijalankan adalah pernyataan 1 kemudian pernyataan 2 dan baris berikutnya

Dari program di atas, bagaimana kalau diinginkan hasilnya hanya menampilkan tulisan **tiga** (jika kondisi salah) tetapi jika kondisi benar menampilkan :



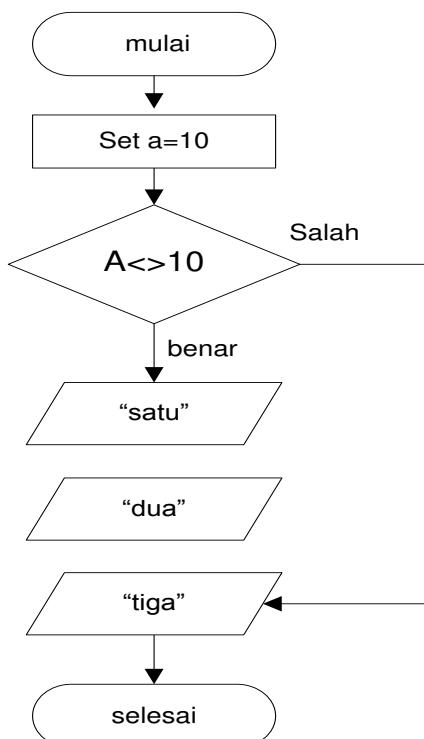
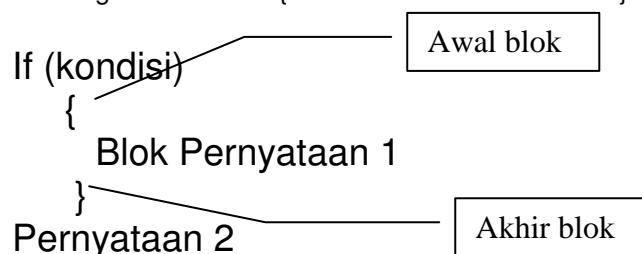
Dari masalah di atas, lakukan perubahan pada bagian pernyataan

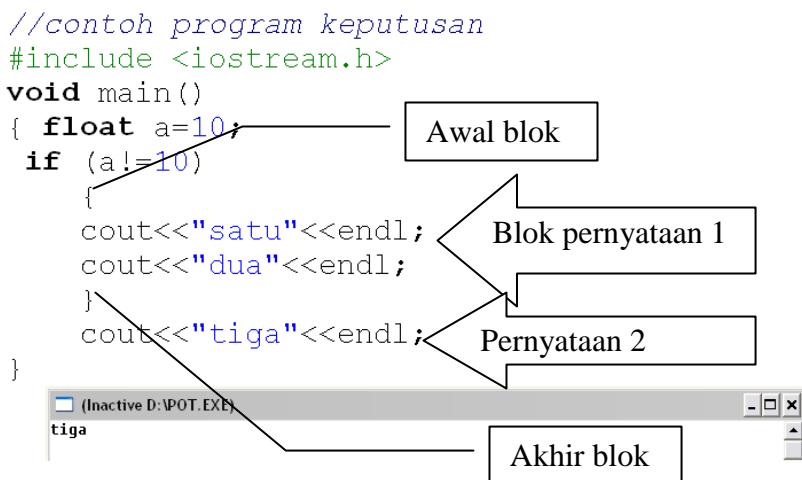
```
if (a==10)
    cout<<"satu"<<endl;
    cout<<"dua"<<endl;
    cout<<"tiga"<<endl;
```

menjadi pernyataan 1

menjadi pernyataan 2

Jika menginginkan suatu pernyataan lebih dari satu perintah gunakan perintah blok. Untuk menandai suatu blok gunakan tanda { untuk awal blok dan tanda } untuk akhir blok



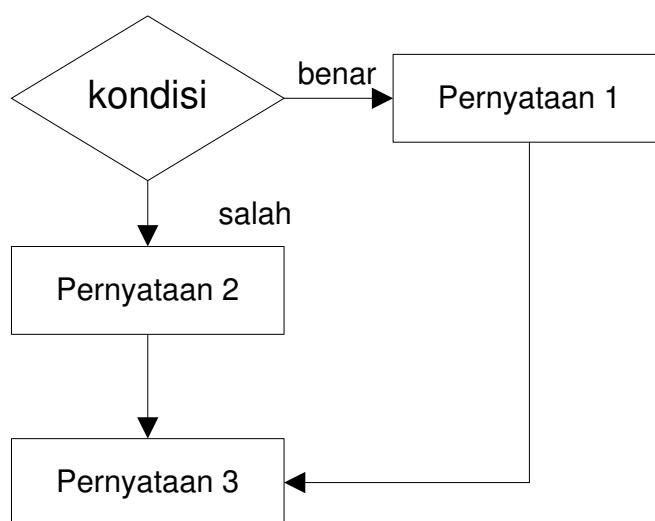


### Penjelasan

- Dalam program di atas
  - Kondisinya adalah apakah a tidak sama dengan 10
  - Pernyataan 1 adalah `cout<<"satu"<<endl` dan `cout<<"dua" <<endl`
  - Pernyataan 2 adalah `cout<<"tiga" <<endl;` dan seterusnya
- Kondisi di atas menghasilkan nilai salah sehingga pernyataan yang dijalankan adalah pernyataan 2 dan baris berikutnya

### 6.2. Pernyataan IF - ELSE

Pernyataan if mempunyai pengertian, Jika kondisi bernilai benar, maka pernyataan -1 akan dikerjakan kemudian kerjakan pernyataan 3 dan jika tidak memenuhi syarat maka akan mengerjakan pernyataan 2 kemudian kerjakan pernyataan 3 Dari pengertian tersebut dapat dilihat dari diagram alir berikut



Bentuk umum dari pernyataan if

```
If (kondisi)
    Pernyataan 1
else
    Pernyataan 2
Pernyataan 3
```

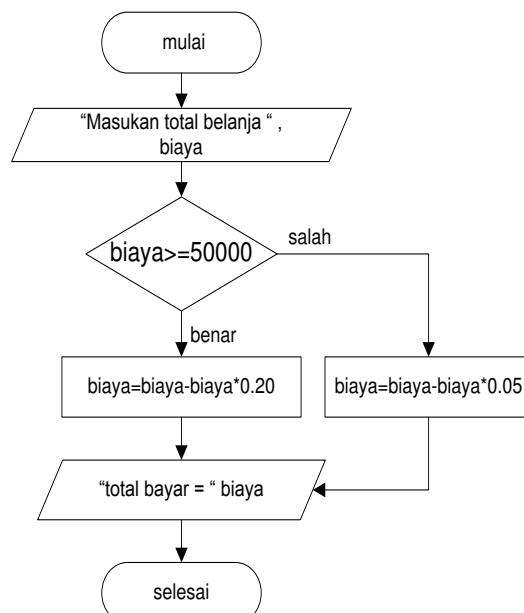
Atau

```
If (kondisi)
{
    Blok Pernyataan 1
}
else
{
    Blok Pernyataan 2
}
Pernyataan 3
```

#### Contoh

Menentukan besarnya potongan dari pembelian barang yang diberikan seorang pembeli, dengan kriteria :

- jika total pembelian kurang dari Rp. 50.000,- potongan yang diterima sebesar 5% dari total pembelian.
- Jika total pembelian lebih dari atau sama dengan Rp. 50.000,- potongan yang diterima sebesar 20% dari total pembelian.



```
//contoh program keputusan
#include <iostream.h>
void main()
{ float biaya;
cout<<"Masukan total belanja= ";
cin>>biaya;
if (biaya>=50000)
    biaya=biaya-(biaya*0.20);
else
    biaya=biaya-(biaya*0.20);
cout<<"Total bayar = "<<biaya<<endl;
}
```



### Penjelasan

- Dalam program di atas
  - Kondisinya adalah apakah biaya sama dengan atau lebih besar dari 50000
  - Pernyataan 1 adalah biaya=biaya-(biaya\*0.20);
  - Pernyataan 2 adalah biaya=biaya-(biaya\*0.20);
  - Pernyataan 3 adalah cout<<"Total bayar = "<<biaya<<endl; dan seterusnya
- Kondisi di atas menghasilkan nilai benar sehingga pernyataan yang dijalankan adalah pernyataan 1 kemudian pernyataan 3 dan baris berikutnya

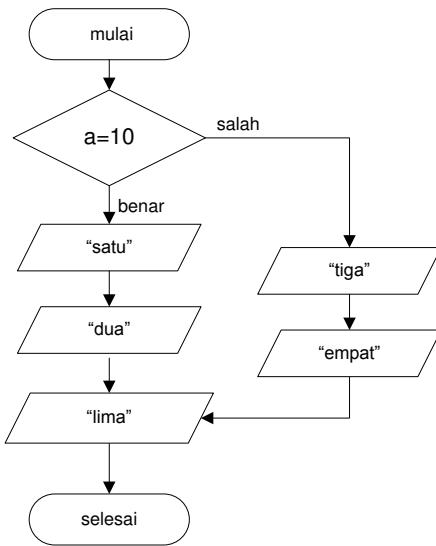
### Contoh

Bila ada teks/ perintah sebagai berikut :

```
cout<<"satu"<<endl;
cout<<"dua"<<endl;
cout<<"tiga"<<endl;
cout<<"empat"<<endl;
cout<<"lima"<<endl;
```

Bagaimana programnya bila diinginkan hasilnya

- Jika benar akan mencetak
  - satu
  - dua
  - lima
- Jika salah mencetak
  - tiga
  - empat
  - lima

**flowchart**

Dari flowchart di atas dapat ditentukan

Penyataan 1

```
cout<<"satu"<<endl;
cout<<"dua"<<endl;
```

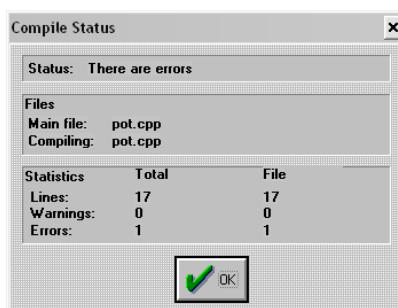
Penyataan 2

```
cout<<"tiga"<<endl;
cout<<"empat"<<endl;
```

Penyataan 3

```
cout<<"lima"<<endl;
```

```
//contoh program keputusan
#include <iostream.h>
void main()
{ int a=10;
  if (a==10)
    cout<<"satu"<<endl;
    cout<<"dua"<<endl;
  else
    cout<<"tiga"<<endl;
    cout<<"empat"<<endl;
  cout<<"lima"<<endl;
}
```



```
Compiling POT.CPP:
Error POT.CPP 9: Misplaced else in function main()
```

Program diatas masih ada kelemahan mendasar, yaitu penentuan blok belum jelas, mana yang pernyataan 1,2 dan 3. Agar kompiler mengetahui blok pernyataan maka gunakan tanda { sebagai awal blok dan } sebagai akhir blok

```
//contoh program keputusan
#include <iostream.h>
void main()
{ int a=10;
  if (a==10)
    cout<<"satu"<<endl;
    cout<<"dua"<<endl;
  else
    cout<<"tiga"<<endl;
    cout<<"empat"<<endl;
  cout<<"lima"<<endl;
}
```

pernyataan 1  
pernyataan 2  
pernyataan 3

Program di atas dimodifikasi dengan memberi tanda blok pernyataan 1 dan blok pernyataan 2 untuk pernyataan 3 tidak menggunakan blok.

```
//contoh program keputusan
#include <iostream.h>
void main()
{ int a=10;
  if (a==10)
  {
    cout<<"satu"<<endl;
    cout<<"dua"<<endl;
  }
  else
  {
    cout<<"tiga"<<endl;
    cout<<"empat"<<endl;
  }
  cout<<"lima"<<endl;
}
```

Kondisi bernilai benar  
Blok pernyataan 1  
Blok pernyataan 2  
Blok pernyataan 3



### Penjelasan

- Karena kondisi bernilai benar, maka pernyataan yang dikerjakan adalah blok pernyataan 1 kemudian mengerjakan blok pernyataan 3

### Contoh

```
//contoh program keputusan
#include <iostream.h>
void main()
{ int a=10;           Kondisi bernilai salah
  if (a!=10)
  {
    cout<<"satu"<<endl;
    cout<<"dua"<<endl;
  }
else
{
  cout<<"tiga"<<endl;
  cout<<"empat"<<endl;
}
cout<<"lima"<<endl;
}

```

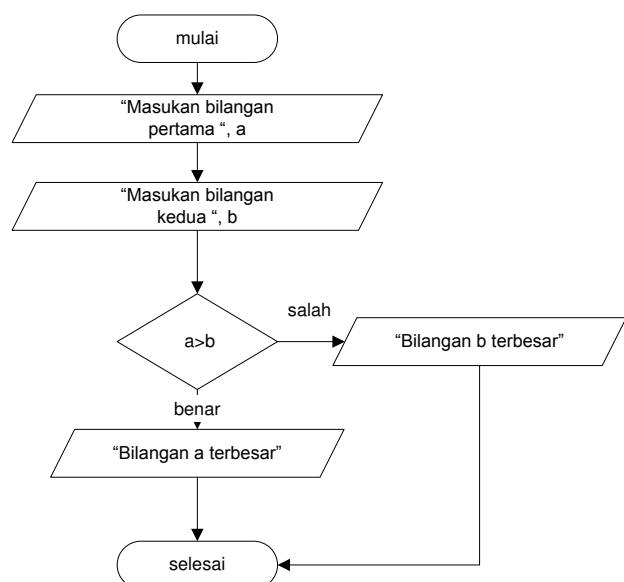


### Penjelasan

- Karena kondisi bernilai salah maka pernyataan yang dikerjakan adalah blok pernyataan 2 kemudian mengerjakan blok pernyataan 3

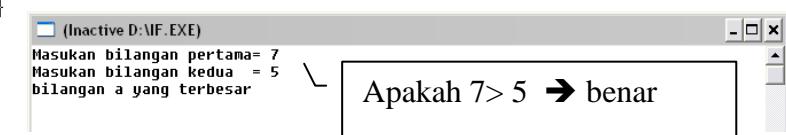
### Studi kasus

Diketahui 2 bilangan, bagaimana flowchart dan program untuk menentukan bilangan yang terbesar dari dua bilangan tersebut.



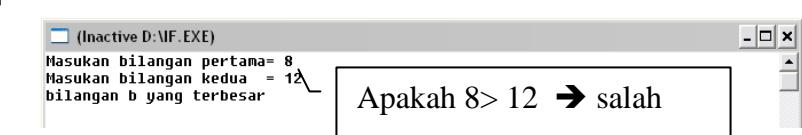
```
//contoh program keputusan
#include <iostream.h>
void main()
{ int a,b;
  cout<<"Masukan bilangan pertama= ";
  cin>>a;
  cout<<"Masukan bilangan kedua = ";
  cin>>b;

  if (a>b)
    cout<<"bilangan a yang terbesar"<<endl;
  else
    cout<<"bilangan b yang terbesar"<<endl;
}
```



```
//contoh program keputusan
#include <iostream.h>
void main()
{ int a,b;
  cout<<"Masukan bilangan pertama= ";
  cin>>a;
  cout<<"Masukan bilangan kedua = ";
  cin>>b;

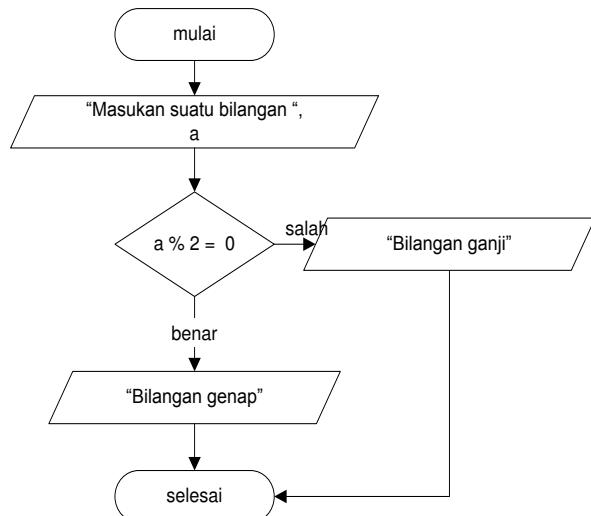
  if (a>b)
    cout<<"bilangan a yang terbesar"<<endl;
  else
    cout<<"bilangan b yang terbesar"<<endl;
}
```



Tentukan apakah suatu bilangan termasuk bilangan genap atau ganjil

**Logika :**

Untuk menentukan bilangan genap adalah lakukan pembagian dengan modulus 2 , jika hasil modulus 2 menghasilkan 0 maka bilangan tersebut ada genap dan bila menghasilkan 1 maka bilangan tersebut adalah bilangan ganjil



```

//contoh program keputusan
#include <iostream.h>
void main()
{ int a,b;
  cout<<"Masukan bilangan pertama= ";
  cin>>a;
  if (a%2==0)
    cout<<"bilangan genap"<<endl;
  else
    cout<<"bilangan ganjil"<<endl;
}
  
```



**Penjelasan :**

Jika dimasukan 80, maka dilakukan proses pembagian modulus.  $80 \% 2$  akan menghasilkan sisa pembagian 0, sehingga 80 termasuk bilangan genap

```
//contoh program keputusan
#include <iostream.h>
void main()
{ int a,b;
cout<<"Masukan bilangan pertama= ";
cin>>a;
if (a%2==0)
    cout<<"bilangan genap"<<endl;
else
    cout<<"bilangan ganjil"<<endl;
}
```



### Penjelasan :

Jika dimasukan 777, maka dilakukan proses pembagian modulus.  $777 \% 2$  akan menghasilkan sisa pembagian 1, sehingga 777 termasuk bilangan ganjil

### 6.3. Memahami keputusan dengan kondisi jamak

Yang dimaksud dengan keputusan dengan kondisi jamak adalah adanya suatu kondisi yang proses pengecekannya lebih dari satu (syaratnya lebih dari satu).

Misal keputusan jamak :

Syarat lulus adalah nilai mid > 50 dan nilai uas >40

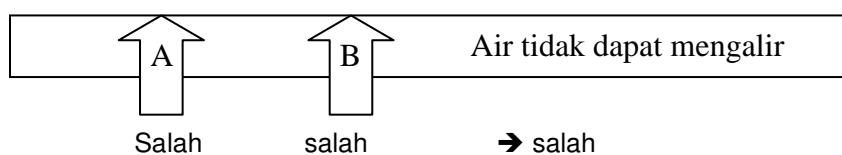
Syarat pengambilan kredit jika gaji\_pokok>1000000 atau gaji\_suami\_istri>=2000000

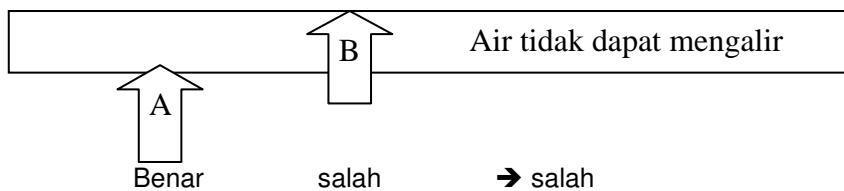
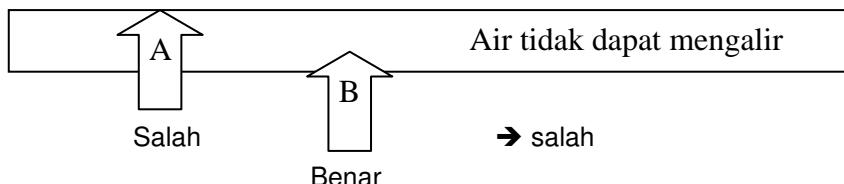
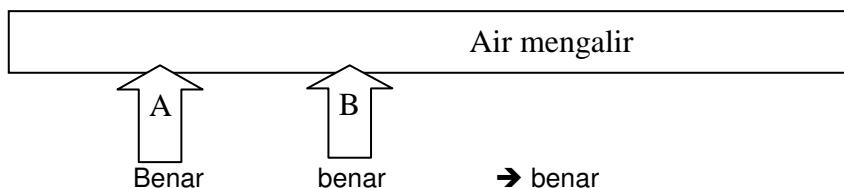
Untuk membuat keputusan yang jamak ini dapat dipakai operator logika

|     |       |    |
|-----|-------|----|
| AND | Dan   | && |
| OR  | Atau  |    |
| NOT | Tidak | !  |

**Kondisi AND** akan dikerjakan jika kedua syarat memenuhi persyaratan. Secara sederhana dapat digambarkan dengan ilustrasi di bawah ini. Ibarat suatu aliran air yang mengalir dalam satu pipa dan ternyata ada dua kran (A dan B), tentunya air hanya dapat mengalir jika kedua kran tersebut dibuka. Jika hanya kran A atau kran B yang terbuka maka aliran air akan berhenti

#### Kran A dan Kran B ditutup



**Kran A dibuka dan Kran B ditutup****Kran A ditutup dan Kran B dibuka****Kran A dibuka dan Kran B dibuka****contoh**

```
#include <iostream.h>
void main()
{ int a,b;
cout<<"Masukan nilai a ";
cin>>a;
cout<<"Masukan nilai b ";
cin>>b;
if (a<b && a>10)
    cout<<" nilai yang dimasukan a<b dan a>10"<<endl;
else
    cout<<"ada salah satu nilai yang salah"<<endl;
cout<<"selesai";
```

**Penjelasan**

Program di atas memasukan 2 bilangan dan dalam contoh dimasukan data 20 (disimpan di variabel a) dan 25 (disimpan di variabel b) kemudian dilakukan pengecekan apakah nilai  $a < b$  (benar) dan apakah  $a > 10$  (benar). Karena kedua kondisi tersebut dihubungkan dengan AND dan kedua kondisi bernilai benar maka hasil akhir benar sehingga bagian yang dikerjakan adalah pernyataan 1 kemudian ke pernyataan 3.

### Contoh

```
#include <iostream.h>
void main()
{ int a,b;
cout<<"Masukan nilai a ";
cin>>a;
cout<<"Masukan nilai b ";
cin>>b;
if (a<b && a>10)
    cout<<" nilai yang dimasukan a<b dan a>10"<<endl;
else
    cout<<"ada salah satu nilai yang salah"<<endl;
cout<<"selesai";
```

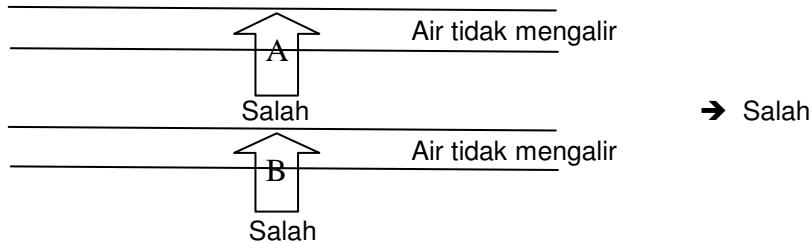


### Penjelasan

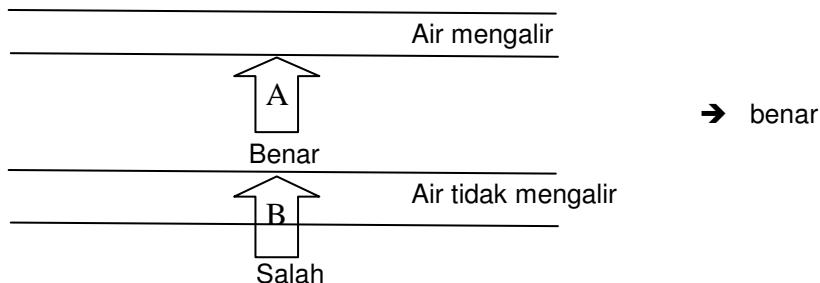
Program di atas memasukan 2 bilangan dan dalam contoh dimasukan data 10 (disimpan di variabel a) dan 15 (disimpan di variabel b) kemudian dilakukan pengecekan apakan nilai  $a < b$  (benar) dan apakah  $a > 10$  (salah). Karena salah satu kondisi ada yang salah dan dihubungkan dengan AND maka hasil akhir salah sehingga bagian yang dikerjakan adalah pernyataan 2 kemudian ke pernyataan 3.

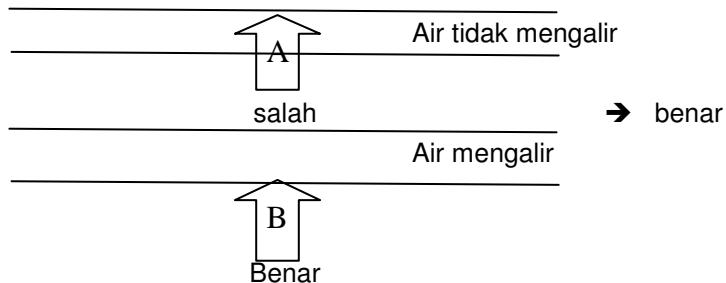
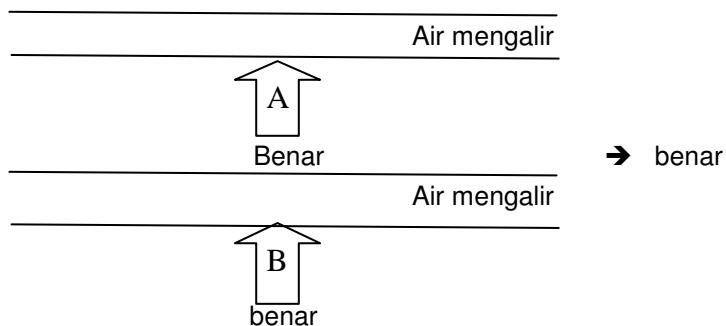
**Kondisi OR** akan dikerjakan jika salah satu syarat memenuhi persyaratan. Secara sederhana dapat digambarkan dengan ilustrasi di bawah ini. Ibarat suatu aliran air yang mengalir dalam dua pipa dan ada dua kran (A dan B), tentunya air dapat mengalir jika salah satu kran tersebut dibuka. Jika kran A atau kran B yang tertutup maka aliran air akan berhenti

### Kran A ditutup atau Kran B ditutup



### Kran A dibuka atau Kran B ditutup



**Kran A ditutup atau Kran B dibuka****Kran A dibuka atau Kran B dibuka****contoh**

```
#include <iostream.h>
void main()
{ int a,b;
cout<<"Masukan nilai a ";
cin>>a;
cout<<"Masukan nilai b ";
cin>>b;
if (a<b || a>10)
    cout<<" nilai yang dimasukan a<b atau a>10"<<endl;
else
    cout<<"ada salah satu nilai yang salah"<<endl;
cout<<"selesai";
```

**Penjelasan**

Program di atas memasukan 2 bilangan dan dalam contoh dimasukan data 10 (disimpan di variabel a) dan 15 (disimpan di variabel b) kemudian dilakukan pengecekan apakan nilai  $a < b$  (benar) atau apakah  $a > 10$  (salah). Karena salah satu kondisi ada yang benar dan dihubungkan dengan OR maka hasil akhir benar sehingga bagian yang dikerjakan adalah pernyataan 1 kemudian ke pernyataan 3.

### Contoh

```
#include <iostream.h>
void main()
{ int a,b;
cout<<"Masukan nilai a ";
cin>>a;
cout<<"Masukan nilai b ";
cin>>b;
if (a<b || a>10)
    cout<<" nilai yang dimasukan a<b atau a>10"<<endl;
else
    cout<<"semua nilai salah"<<endl;
cout<<"selesai";
```



### Penjelasan

Program di atas memasukan 2 bilangan dan dalam contoh dimasukan data 10 (disimpan di variabel a) dan 5 (disimpan di variabel b) kemudian dilakukan pengecekan apakan nilai  $a < b$  (salah) atau apakah  $a > 10$  (salah). Karena semua kondisi salah dan dihubungkan dengan OR maka hasil akhir salah sehingga bagian yang dikerjakan adalah pernyataan 2 kemudian ke pernyataan 3.

#### 6.4. Urutan Operasi

Penggunaan kondisi jamak dapat diterapkan tidak hanya satu pembanding tetapi juga dapat digunakan lebih dari satu kondisi. Urutan prioritas harus yang harus diperhatikan, urutan prioritas adalah : NOT, AND, OR

misal

$LX = \text{true}$

$LY = \text{true}$

$LZ = \text{false}$

| Ungkapan                                                                                   | Hasil |
|--------------------------------------------------------------------------------------------|-------|
| ( LX or LZ ) and LY<br>1        2<br>( T or F) and T<br>T        and    T                  | TRUE  |
| (LX or LY) and (not LZ)<br>( T or T) and (not F)<br>2        1<br>T        and    T        | TRUE  |
| LX or LY and LZ<br>2        1<br>T        or    T    and    F<br>T        or        F<br>T | TRUE  |

**contoh**

```
#include <iostream.h>
void main()
{ int a,b,c;
cout<<"Masukan nilai a ";
cin>>a;
cout<<"Masukan nilai b ";
cin>>b;
cout<<"Masukan nilai c ";
cin>>c;
if ((a<b || a>10) && (b<5 || a<c))
    cout<<"Kondisi akhir bernilai benar "<<endl;
else
    cout<<"Kondisi akhir bernilai salah "<<endl;
cout<<"selesai";
```

**Penjelasan**

Nilai a =5, b=10 dan c=15 dihubungkan dengan kondisi

$$\begin{aligned} \text{if } & ((a < b \text{ || } a > 10) \text{ && } (b < 5 \text{ || } a < c)) \\ & ((5 < 10 \text{ or } 10 > 10) \text{ dan } (10 < 5 \text{ || } 5 < 15) \\ & ((\text{B or S}) \text{ dan } (\text{B or B})) \\ & ((\text{B}) \text{ dan } (\text{B})) \\ & \text{B} \end{aligned}$$

Hasil akhir adalah Benar

**Contoh**

```
#include <iostream.h>
void main()
{ int a,b,c;
cout<<"Masukan nilai a ";
cin>>a;
cout<<"Masukan nilai b ";
cin>>b;
cout<<"Masukan nilai c ";
cin>>c;
if ((a<b && a>10) && (b<5 || a<c))
    cout<<"Kondisi akhir bernilai benar "<<endl;
else
    cout<<"Kondisi akhir bernilai salah "<<endl;
cout<<"selesai";
```



**Penjelasan**

Nilai  $a = 5$ ,  $b = 10$  dan  $c = 15$  dihubungkan dengan kondisi

```

if (( a<b && a>10 ) && (b<5 || a<c))
    ((5<10 && 10>10) dan (10<5 || 5<15)
        (( B and S ) dan ( B or B )
            ( S ) dan ( B )
                S

```

Hasil akhir adalah salah

**6.5 . Pernyataan IF berkalang**

if berkalang merupakan pernyataan if berada di dalam pernyataan if yang lainnya. Pernyataan ini sering digunakan untuk menyelesaikan masalah keputusan yang bertingkat. Keputusan bertingkat ini sering terjadi pada persoalan yang memerlukan syarat yang lebih dari satu. Contoh :

- Bila salesman dapat menjual barang hingga Rp. 20.000 ,-, akan diberikan uang jasa sebesar Rp. 1.000 ditambah dengan uang komisi Rp. 10% dari pendapatan yang diperoleh hari itu.
- Bila salesman dapat menjual barang diatas Rp. 20.000 ,-, akan diberikan uang jasa sebesar Rp. 2.000 ditambah dengan uang komisi Rp. 15% dari pendapatan yang diperoleh hari itu.
- Bila salesman dapat menjual barang diatas Rp. 30.000 ,-, akan diberikan uang jasa sebesar Rp. 3.000 ditambah dengan uang komisi Rp. 20% dari pendapatan yang diperoleh hari itu.

Permasalahan di atas jika diselesaikan dengan menggunakan if – else tentunya tidak pas.

Bentuk penulisan pernyataan if berkalang adalah :

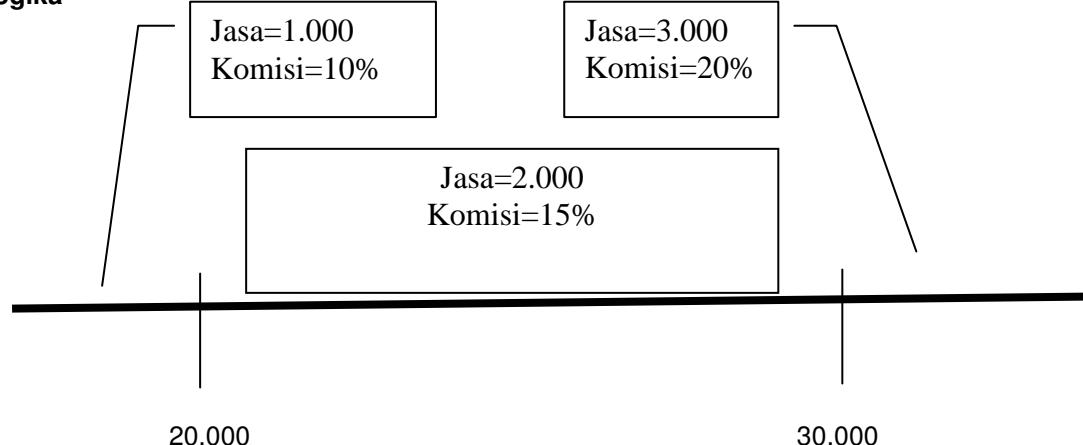
```

If (kondisi 1)
{
    Blok Pernyataan
}
else
    if (kondisi 2)
    {
        Blok Pernyataan
    }
.....
.....
```

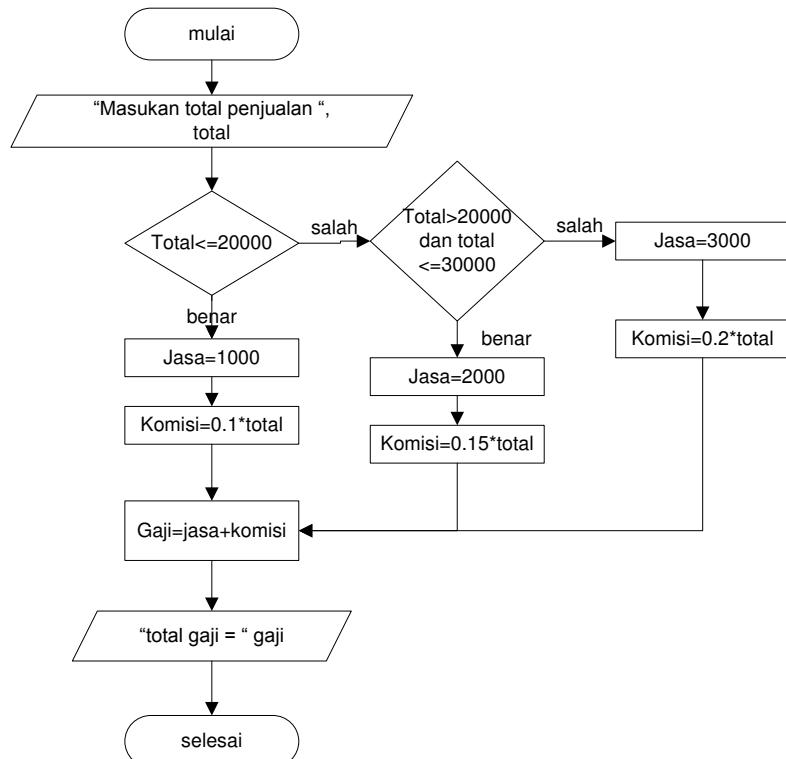
Dalam membuat if berkaliang harus hati-hati dalam menentukan pasangan dari masing-masing kondisi, jika tidak akan menghasilkan logika penyelesaian yang berbeda dengan hasil yang diinginkan.

Untuk mempermudah logika if berkaliang, dapat digambarkan dalam bentuk tabel diagram,

### Logika



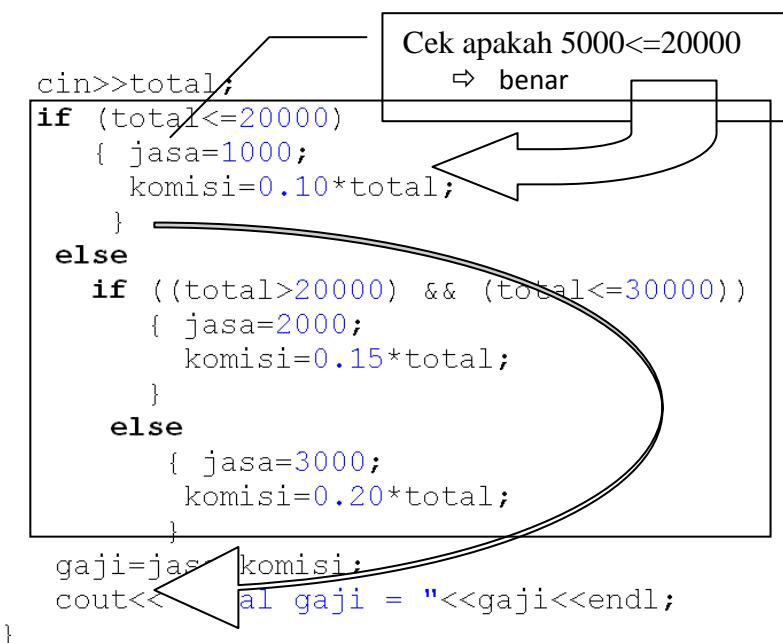
### Flowchart



```
//contoh program keputusan
#include <iostream.h>
void main()
{ float total,gaji, komisi,jasa;
cout<<"Masukan total penjualan = ";
cin>>total;
if (total<=20000)
{ jasa=1000;
komisi=0.10*total;
}
else
if ((total>20000) && (total<=30000))
{ jasa=2000;
komisi=0.15*total;
}
else
{ jasa=3000;
komisi=0.20*total;
}
gaji=jasa+komisi;
cout<<"total gaji = "<<gaji<<endl;
}
```

**Pemasukan data:**

- Jika dimasukan data 5000, maka prosesnya



### Penjelasan

- Cek apakah  $5000 \leq 20000$   
→ benar maka kerjakan pernyataan 1 ( satu bagian di bawah if )
- Hitung  $jasa = 1000$  dan  $komisi = 0.10 * total$
- Setelah mengerjakan perintah program langsung keluar dari kalang
- Menghitung gaji dan mencetak total gaji

Jika dimasukan data 25000  
`cin >> total;`

```

if (total <= 20000)
{
    jasa = 1000;
    komisi = 0.10 * total;
}
else
{
    if ((total > 20000) && (total <= 30000))
    {
        jasa = 2000;
        komisi = 0.15 * total;
    }
    else
    {
        jasa = 3000;
        komisi = 0.20 * total;
    }
}
gaji = jasa + komisi;
cout << "total gaji = " << gaji << endl;
}

```

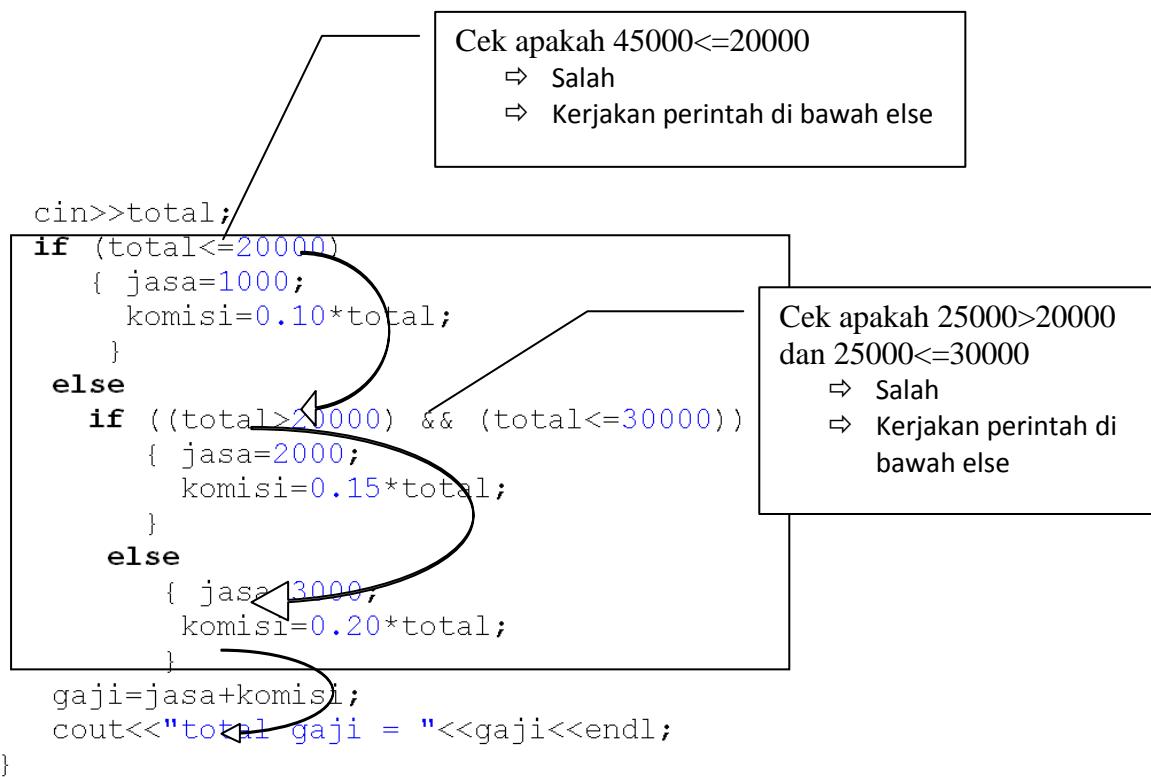
Cek apakah  $25000 > 20000$  dan  $25000 \leq 30000$   
⇒ benar

### Penjelasan

- Cek apakah  $25000 \leq 20000$   
→ salah maka kerjakan n if berikutnya
- Cek apakah  $25000 > 20000$  dan  $25000 \leq 30000$   
→ benar maka kerjakan pernyataan di bawah if
- Hitung  $jasa = 2000$  dan  $komisi = 0.15 * total$
- Setelah mengerjakan perintah program langsung keluar dari kalang
- Menghitung gaji dan mencetak total gaji

Masukan total penjualan = 25000  
total gaji = 5750

- Jika dimasukan data 45000



### Penjelasan

- Cek apakah  $25000 \leq 20000$   
 ➔ salah maka kerjakan if berikutnya
- Cek apakah  $45000 > 20000$  dan  $45000 \leq 30000$   
 ➔ salah maka kerjakan pernyataan di bawah else
- Hitung  $jasa = 3000$  dan  $komisi = 0.2 * total$
- Setelah mengerjakan perintah program langsung keluar dari kalang
- Menghitung gaji dan mencetak total gaji

```

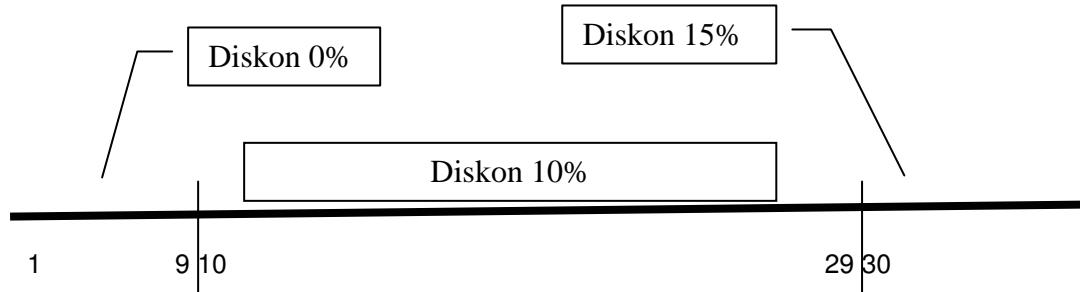
Masukan total penjualan = 45000
total gaji = 12000

```

**Contoh**

Sebuah toko buku memberikan diskon dengan ketentuan

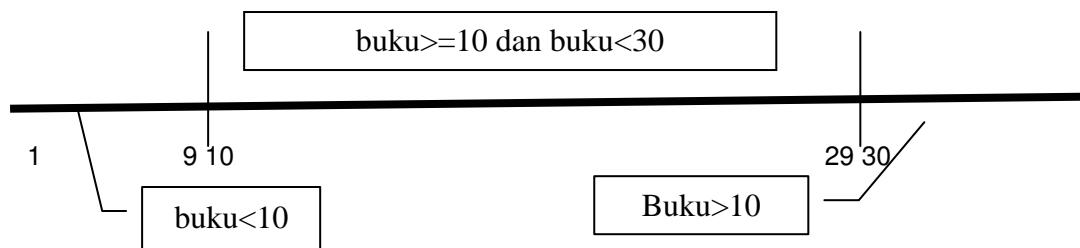
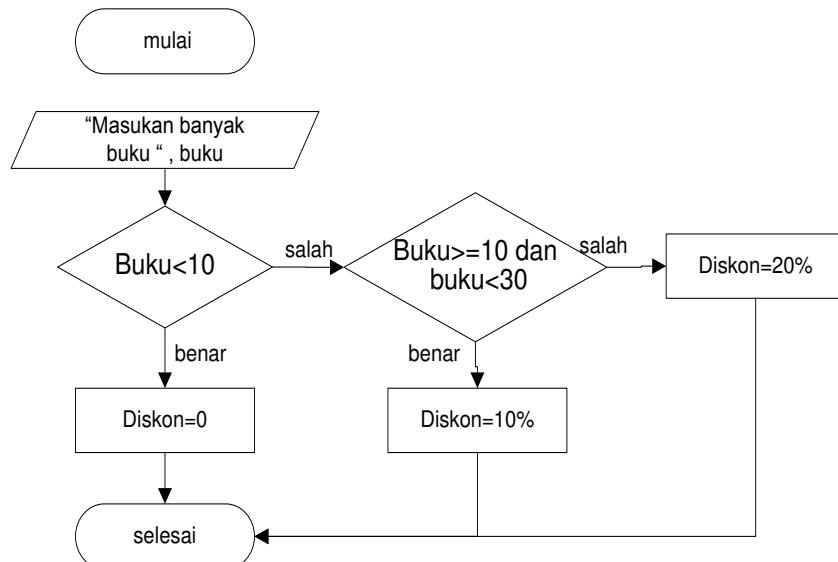
- Pembelian buku kurang dari 10 buku tidak mendapatkan diskon
- Pembelian buku di atas atau sama dengan 10 sampai 30 buku mendapatkan diskon 10%
- Pembelian buku di atas 30 mendapatkan diskon 15%

**Logika**

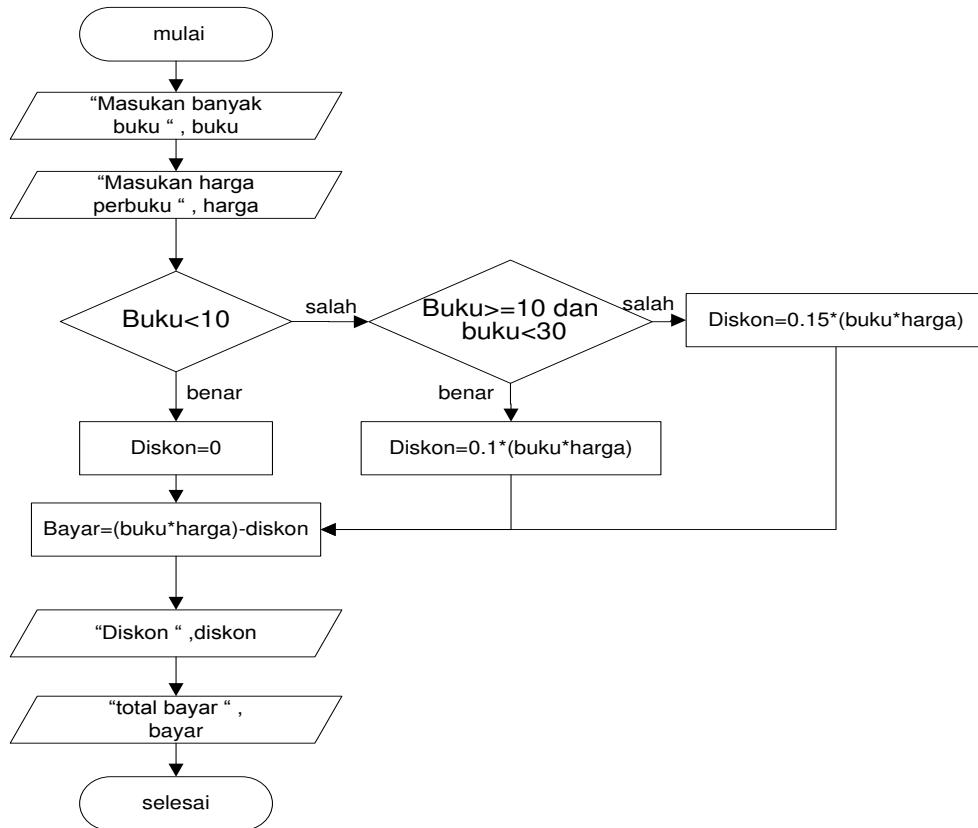
Dari gambar di atas, logikanya adalah :

1. Jika pembelian di area 1-9 tidak mendapatkan diskon
2. Jika pembelian di area 10-29 mendapatkan diskon 10 %
3. Jika pembelian di area 30 ke atas mendapatkan diskon 15 %

Proses penentuan keputusan :

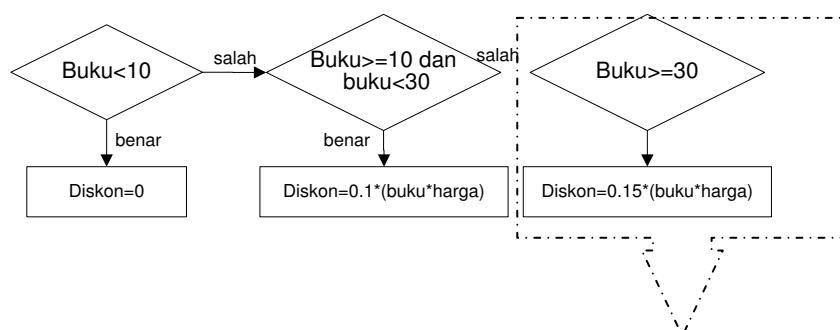
**flowchart**

Bagaimana jika flowchartnya dilengkapi dengan proses pembayaran



#### Catatan :

Dalam membuat sebuah keputusan tidak semua keputusan dibuat dalam bentuk if. Dari contoh di atas kenapa tidak ada perintah untuk melakukan seleksi apakah  $buku \geq 30$ .



Kenapa tidak ada seleksi ini ??

Dalam program keputusan di atas, cukup ditulis perintah keputusan  $buku \geq 10$  dan  $buku < 30$  saja, hal ini mengingat jika perintah tersebut salah pasti nilainya di atas 30. Jadi tidak perlu menanyakan apakah  $buku > 30$  ... Sebuah pertanyaan yang mubazir !!!!!!

```
//contoh program keputusan
#include <iostream.h>
void main()
{ float buku,harga,diskon,bayar;
cout<<"Masukan banyak pembelian buku = ";
cin>>buku;
cout<<"Masukan harga perbuku = ";
cin>>harga;
if (buku<=10)
    diskon=0;
else
    if ((buku>=10) && (buku<30))
        diskon=0.10*(buku*harga);
    else
        diskon=0.15*(buku*harga);

bayar=(buku*harga)-diskon;
cout<<"Mendapatkan diskon = "<<diskon<<endl;
cout<<"Total bayar      = "<<bayar<<endl;
}
```

### Logika program

```
(Inactive D:\IF3.EXE)
Masukan banyak pembelian buku = 5
Masukan harga perbuku = 10000
Mendapatkan diskon = 0
Total bayar      = 50000
```

Penjelasan :

```
void main()
{ float buku,harga,diskon,bayar;
cout<<"Masukan banyak pembelian buku = ";
cin>>buku;
cout<<"Masukan harga perbuku = ";
cin>>harga;

if (buku<=10)
    diskon=0;
else
    if ((buku>=10) && (buku<30))
        diskon=0.10*(buku*harga);
    else
        diskon=0.15*(buku*harga);

bayar=(buku*harga)-diskon;
cout<<"Mendapatkan diskon = "<<diskon<<endl;
cout<<"Total bayar      = "<<bayar<<endl;
}
```

Dimasukan 5

Cek apakah buku<=5  
 ⇒ Benar  
 ⇒ Kerjakan statemen di bawah if

```

[1] (Inactive D:\IF3.EXE)
Masukan banyak pembelian buku = 25
Masukan harga perbuku = 10000
Mendapatkan diskon = 25000
Total bayar      = 225000

```

**Penjelasan :**

```

void main()
{ float buku,harga,diskon,bayar;
  cout<<"Masukan banyak pembelian buku = ";
  cin>>buku;
  cout<<"Masukan harga perbuku = ";
  cin>>harga;

  if (buku<=10)
    diskon=0;
  else
    if ((buku>=10) && (buku<30))
      diskon=0.10*(buku*harga);
    else
      diskon=0.15*(buku*harga);

  bayar=(buku*harga)-diskon;
  cout<<"Mendapatkan diskon = "<<diskon<<endl;
  cout<<"Total bayar      = "<<bayar<<endl;
}

```

Dimasukan 25

Cek apakah buku<=10  
 ⇒ salah  
 ⇒ Kerjakan statemen di bawah else

Cek apakah buku>=10 dan buku<30  
 ⇒ ya  
 ⇒ Kerjakan statemen di bawah if

```

[1] (Inactive D:\IF3.EXE)
Masukan banyak pembelian buku = 40
Masukan harga perbuku = 10000
Mendapatkan diskon = 60000
Total bayar      = 340000

```

**Penjelasan :**

```

void main()
{ float buku,harga,diskon,bayar;
  cout<<"Masukan banyak pembelian buku = ";
  cin>>buku;
  cout<<"Masukan harga perbuku = ";
  cin>>harga;

  if (buku<=10)
    diskon=0;
  else
    if ((buku>=10) && (buku<30))
      diskon=0.10*(buku*harga);
    else
      diskon=0.15*(buku*harga);

  bayar=(buku*harga)-diskon;
  cout<<"Mendapatkan diskon = "<<diskon<<endl;
  cout<<"Total bayar      = "<<bayar<<endl;
}

```

Dimasukan 25

Cek apakah buku<=10  
 ⇒ salah  
 ⇒ Kerjakan statemen di bawah else

Cek apakah buku>=10 dan buku<30  
 ⇒ salah  
 ⇒ Kerjakan statemen di bawah else

### 6.5. Latihan

1. Suatu lembaga Pendidikan melakukan seleksi penerimaan mahasiswa baru. Syarat diterima menjadi mahasiswa jika memenuhi kriteria :

Nilai Ujian Bahasa Indonesia diatas 55

Nilai ujian Pancasila diatas 60

Nilai Matematika diatas 60

Jika ketiga syarat tersebut terpenuhi, maka dinyatakan lulus menjadi mahasiswa.

Buat diagram alir dan program dengan menggunakan proses keputusan

2. Sebuah toko buku dalam bulan promosinya memberikan potongan bagi pembeli. Tidak semua pembeli buku akan mendapatkan harga potongan. Hanya pembeli yang memenuhi yang mendapat potongan, yaitu :

Jika membeli 2 buah buku mendapat potongan 5 % dari total pembelian

Jika membeli antara 2 sampai 4 buku mendapat potongan 10 % dari total pembelian

Jika membeli diatas 4, mendapat potongan 15 % dari total pembelian.

Buat diagram alir dan program serta hitung berapa potongan yang didapat serta berapa biaya yang harus dibayarkan.

3. Sebuah perusahaan menerapkan sistem pengajian, dengan ketentuan

- gaji pokok
  - Pegawai Tetap Rp. 200.000
  - Pegawai honorer Rp. 100.000
- Bila kerja kembur mendapat tambahan honor lembur
  - Honor Lembur = jam Lembur \* 2 % dari gaji pokok
- Tunjangan 10 % dari gaji pokok
- Jumlah gaji = gaji pokok + honor lembur + tunjangan

Buat diagram alir, dan program data-data apa saja yang harus dimasukkan dari keyboard dan data-data apa saja yang dapat dihitung dari proses.

4. Buat diagram alir yang dapat menampilkan nama jurusan dari suatu kode yang dimasukkan.

Ketentuannya adalah :

- Kode 01 nama jurusan Teknik KIMIA
- Kode 02 nama jurusan Teknik INDUSTRI
- Kode 03 nama jurusan Teknik MESIN
- Kode 04 nama jurusan Teknik ELEKTRO
- Kode 05 nama jurusan Teknik INFORMATIKA

5. Nilai suatu ujian, biasanya dikonversikan ke Huruf. Buat diagram alir yang dapat melakukan konversi nilai dari nilai angka ke nilai huruf, dengan ketentuan  
Nilai A jika nilai angkanya  $100 > X \leq 80$   
Nilai B jika nilai angkanya  $80 > X \leq 70$   
Nilai C jika nilai angkanya  $70 > X \leq 60$   
Nilai D jika nilai angkanya  $60 > X \leq 50$   
Nilai E jika nilai angkanya  $50 > X$
6. Buat diagram alir yang dapat memberi suatu komentar :  
Bila NILAI yang dimasukkan dari keyboard GANJIL, maka komentarnya “Bilangan GANJIL”  
Bila NILAI yang dimasukkan dari keyboard GENAP, maka komentarnya “Bilangan GENAP”

### 6.6.Pernyataan switch - case

Bentuk dari switch - case merupakan pernyataan yang dirancang khusus untuk menangani pengambilan keputusan yang melibatkan sejumlah atau banyak alternatif penyelesaian. Pernyataan switch - case ini memiliki kegunaan sama seperti if – else bertingkat, tetapi penggunaannya untuk memeriksa data yang bertipe karakter atau integer. Bentuk penulisan perintah ini sebagai berikut :

```
switch (ekspresi integer atau karakter )
{
    case konstanta-1 :
        ... perintah;
        ... perintah;
        break;
    case konstanta-2 :
        ... perintah;
        ... perintah;
        break;
    .....
    .....
    default :
        ... perintah;
        ... perintah;
}
```

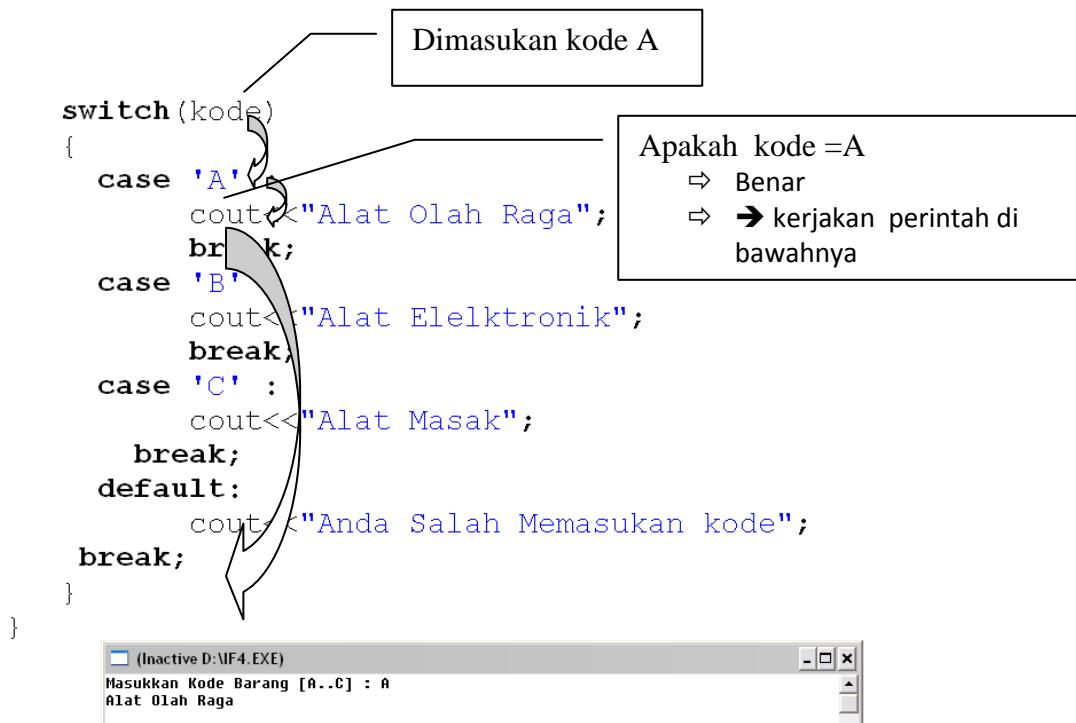
Setiap cabang akan dijalankan jika syarat nilai konstanta tersebut dipenuhi dan default akan dijalankan jika semua cabang di atasnya tidak terpenuhi.

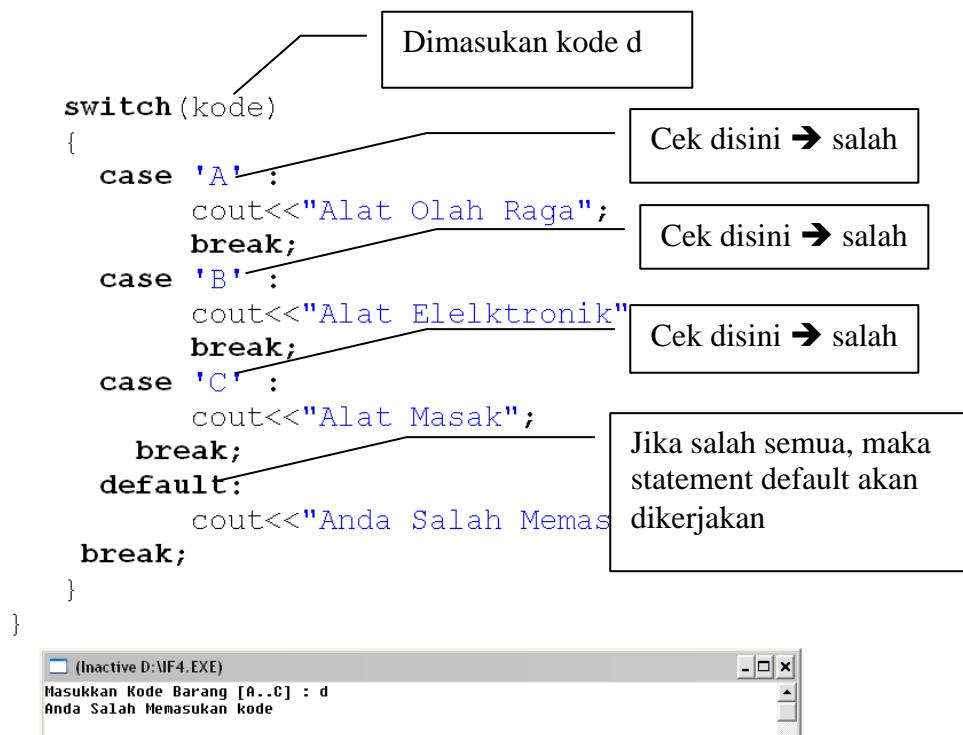
Pernyataan break menunjukkan bahwa perintah siap keluar dari switch. Jika pernyataan ini tidak ada, maka program akan diteruskan ke cabang – cabang yang lainnya.

```
//contoh program keputusan
#include <iostream.h>
main()
{
    char kode;
    cout<<"Masukkan Kode Barang [A..C] : ";
    cin>>kode;

    switch(kode)
    {
        case 'A':
            cout<<"Alat Olah Raga";
            break;
        case 'B':
            cout<<"Alat Elektronik";
            break;
        case 'C':
            cout<<"Alat Masak";
            break;
        default:
            cout<<"Anda Salah Memasukan kode";
            break;
    }
}
```

### Penjelasan





### 6.7.Latihan

- 1 Membaca sebuah bilangan bulat antara 1 sampai 4, lalu mencetak tulisan dari angka tersebut.  
Contoh: Jika diinput angka **1** maka output akan tercetak tulisan **SATU**  
jika di input selain dari **1 sampai 4** maka munculkan pesan "Anda Salah memasukan Input Kode"
- 2 Buatlah program untuk menentukan jumlah hari dalam bulan
- 3 Mencari hari kelahiran seseorang . Hari lahir kelahiran, sebagian orang pasti banyak yang tahu hari apa dia dilahirkan. Buat program untuk mencari hari kelahiran .

### Algoritma :

- a) Tentukan terlebih dahulu tanggal yang ingin dicari harinya.  
Sebagai contoh kita ambil tanggal lahir kemerdekaan negara saya Republik Indonesia, yaitu 17 Agustus 1945.
- b) Periksa terlebih dahulu tahun lahir tersebut apakah tahun kabisat atau tidak.  
Ketentuannya, jika habis dibagi 4 berarti tahun kabisat. Namun, untuk tahun kelipatan 100 harus habis juga dibagi 400 baru masuk golongan tahun kabisat. Misalnya, tahun 1900 adalah kelipatan 100 yang habis dibagi 4 tapi tidak habis dibagi 400. Maka, tahun 1900 bukanlah tahun kabisat. Tahun kabisat terdiri dari 366 hari karena bulan Februari pada tahun kabisat terdiri dari 29 hari. Karena tahun 1945 bukanlah tahun kabisat, maka bulan Februari 1945 terdiri dari 28 hari.
- c) Hitunglah nilai h  
 $h = \text{jumlah hari yang dilewati dari awal tahun (1 Januari) sampai pada tanggal itu jatuh.}$   
Maka, untuk contoh, nilai h adalah jumlah hari dari 1 januari 1945 – 17 Agustus 1945.  
 $h = 31+28+31+30+31+30+31+17=229.$   
Perlu diperhatikan bahwa untuk tahun kabisat jumlah hari bulan Februari = 29.
- d) Hitung juga nilai k  
 $k = (\text{tahun lahir} - 1) : 4$   
Maka pada contoh nilai  $k = (1945-1) : 4 = 486$  sisa 0. Proses pembagian yang Anda lakukan sebaiknya menggunakan teknik yang dipelajari ketika SD yaitu menggunakan pohon sisa. Nilai k yang diambil adalah nilai bulat hasil pembagian sedangkan sisanya diabaikan (dihilangkan). Maka nilai k = 486
- e) Cari nilai S  
 $S = (\text{tahun lahir} + h + k) : 7$   
Maka pada contoh nilai  $S = (1945 + 229 + 486) : 7 = 2660 : 7 = 380$  sisa 0.
- f) Hasil akhir yang kita gunakan untuk mengetahui hari dari tanggal tersebut adalah nilai SISA dari perhitungan S pada langkah 5 di atas.
  - o Jika sisa = 0 , maka tanggal tersebut adalah hari Jumat
  - o Jika sisa = 1 , maka tanggal tersebut adalah hari Sabtu
  - o Jika sisa = 2 , maka tanggal tersebut adalah hari Minggu
  - o Jika sisa = 3 , maka tanggal tersebut adalah hari Senin
  - o Jika sisa = 4 , maka tanggal tersebut adalah hari Selasa
  - o Jika sisa = 5 , maka tanggal tersebut adalah hari Rabu
  - o Jika sisa = 6 , maka tanggal tersebut adalah hari Kamis

Dari hasil perhitungan di atas bisa diketahui bahwa hari kemerdekaan Republik Indonesia adalah hari Jumat. (<http://blog.banditbatak.com/featured/mencari-tahu-hari-kelahiran-anda/>)

7

# PERULANGAN



## Tujuan Instruksional

Setelah membaca bab ini, diharapkan pembaca memahami logika perulangan, kapan perulangan digunakan, nilai awal, nilai akhir serta counter perulangan. Implementasi perulangan dapat menggunakan for, while dan do - while

## Materi

MeMemahami perintah whileahami perintah for

Memahami for di dalam for (kalang for)

Memahami perintah while

Proses instalasi Memahami perintah do - whileC++

Proses perulangan sering dijumpai dalam kasus pemrogram, misal saat bermain game dan kalah tentunya aka noda pertanyaan apakah akan melakukan permainan lagi atau tidak, Jika memilih main lagi berarti program melakukan proses perulangan.



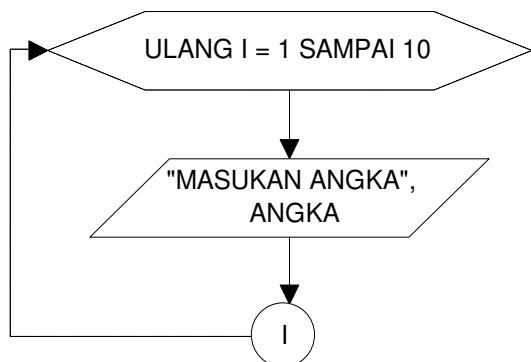
Gambar 7.1. Contoh Aplikasi yang berulang

Dalam bahasa C++ ada beberapa perintah perulangan, yaitu :

- FOR
- While

### 7.1. Memahami perintah for

Proses pengulangan di atas, dilakukan dengan memeriksa kondisi terlebih dahulu. Untuk suatu proses pengulangan yang sudah pasti dapat ditentukan berapa kali terjadi proses pengulangan, lebih baik menggunakan perintah for



Bentuk penulisan

**For (ekspresi1;ekspresi2,ekspresi3)**  
**Pernyataan 1;**  
**Pernyataan 2;**

Dimana

- Ekspresi 1 → dipakai untuk inisialisasi dari variabel yang dipakai untuk mengontrol ulangan eksekusi dari blok pernyataan yang ada di dalam pernyataan for → nilai awal dari pengulangan
- Ekspresi 2 → dipakai untuk menyatakan suatu kondisi, eksekusi dari blok pernyataan yang ada di dalam pernyataan for akan diulang bila kondisi menghasilkan nilai true
- Ekspresi 3 dipakai untuk menambah/mengurangi nilai variabel yang dipakai untuk mengontrol eksekusi dari blok pernyataan yang ada di dalam pernyataan for

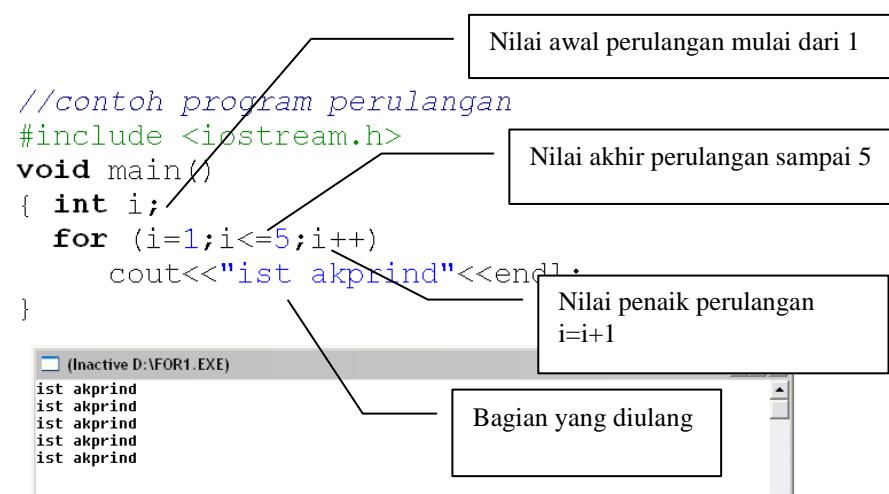
Untuk mempermudah dalam pembuatan perulangan, langkah awal harus menentukan :

- Nilai awal dari pengulangan (nilai awal pengulangan tidak harus selalu dari 1)
- Nilai akhir pengulangan
- Nilai penaik/ penurun dari pengulangan
- Blok pernyataan yang akan diulang
- Blok pernyataan setelah proses perulangan selesai

Contoh :

```
for (i=1;i<=5;i+
    cout<<"ist akprind"<<endl;
```

- nilai awal pengulangan 1
- nilai akhir pengulangan jika  $i \leq 5$
- nilai penaik adalah  $i = i + 1$
- bagian yang diulang adalah `cout<<"ist akprind"<<endl;`



### Penjelasan

- nilai awal perulangan adalah 1 ( $i=1$ )
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=1$ ) masih memenuhi  $i \leq 5$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  ditambah1 →  $i=2$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=2$ ) masih memenuhi  $i \leq 5$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  ditambah1 →  $i=3$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=3$ ) masih memenuhi  $i \leq 5$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  ditambah1 →  $i=4$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=4$ ) masih memenuhi  $i \leq 5$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  ditambah1 →  $i=5$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=5$ ) masih memenuhi  $i \leq 5$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  ditambah1 →  $i=6$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=6$ ) masih memenuhi  $i \leq 5$ 
  - tidak → kerjakan pernyataan 2 dan baris seterusnya

### Studi kasus

#### Contoh

```
//contoh program perulangan
#include <iostream.h>
void main()
{ int i;
  for (i=5;i>=1;i--)
    cout<<"ist akprind"<<endl;
}
```



### Penjelasan

- nilai awal perulangan adalah 5 ( $i=5$ )
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=5$ ) masih memenuhi  $i \geq 1$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  dikurangi 1 →  $i=4$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=4$ ) masih memenuhi  $i \geq 1$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  dikurangi 1 →  $i=3$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=3$ ) masih memenuhi  $i \geq 1$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  dikurangi 1 →  $i=2$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=2$ ) masih memenuhi  $i \geq 1$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  dikurangi 1 →  $i=1$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=1$ ) masih memenuhi  $i \geq 1$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai  $i$  dikurangi 1 →  $i=0$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=0$ ) masih memenuhi  $i \geq 1$ 
  - tidak → kerjakan pernyataan 2 dan baris seterusnya

- ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
- nilai i dikurangi 1 →  $i=1$
- lakukan proses pengecekan, apakah nilai i ( $i=1$ ) masih memenuhi  $i \geq 1$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak tulis IST AKPRIND
  - nilai i dikurangi 1 →  $i=0$
- lakukan proses pengecekan, apakah nilai i ( $i=0$ ) masih memenuhi  $i \geq 0$ 
  - tidak → kerjakan pernyataan 2 dan baris seterusnya

### Contoh

```
//contoh program perulangan
#include <iostream.h>
void main()
{ int i;
  for (i=5;i>=1;i++)
    cout<<"ist akprind"<<endl;
}
```

Program di atas bila dijalankan akan terus menerus mencetak pernyataan 1, hal ini dikarenakan nilai i akan selalu bertambah 1 sehingga nilai  $i \geq 1$  akan selalu bertambah besar

```
//contoh program perulangan
#include <iostream.h>
void main()
{ int i;
  for (i=0;i>=10;i++)
    cout<<"ist akprind"<<endl;
}
```

Program di atas bila dijalankan tidak ada hasilnya (tidak mengerjakan pernyataan 1), hal ini dikarenakan nilai awal  $i = 0$  sementara syarat perulangan  $i \geq 10$  sehingga akan bernilai salah dan program akan mengerjakan pernyataan 2 (penjelasan ada pada program di bawah ini)

```
//contoh program perulangan
#include <iostream.h>
void main()
{ int i;
  for (i=0;i>=10;i++)
    cout<<"ist akprind"<<endl;
  cout<<"selesa";
}
```

Pernyataan 1

Pernyataan 2

 (Inactive D:\FOR1.EXE)  
selesa

```
//contoh program perulangan
#include <iostream.h>
void main()
{ int i;
  for (i=1;i<=5;i++)
    cout<<"a"<<endl;
  cout<<"b"<<endl;
  cout<<"c"<<endl;
  cout<<"selesa";
}
```

```
(Inactive D:\FOR1.EXE)
a
a
a
a
a
b
c
selesa
```

Dari program di atas, bagaimana jika hasilnya diinginkan sebagai berikut :

```
(Inactive D:\FOR1.EXE)
a
b
c
a
b
c
a
b
c
a
b
c
a
b
c
selesai
```

### Logika

- buat blok pada bagian yang menjadi satu kesatuan pernyataan 1

```
//contoh program perulangan
#include <iostream.h>
void main()
{ int i;
  for (i=1;i<=5;i++)
    cout<<"a"<<endl;
    cout<<"b"<<endl;
    cout<<"c"<<endl;
    cout<<"selesa";
}
```

Untuk membuat agar beberapa perintah menjadi satu-kesatuan blok gunakan tanda { sebagai awal blok dan tanda } sebagai akhir blok

```
//contoh program perulangan
#include <iostream.h>
void main()
{ int i;
  for (i=1;i<=5;i++)
  { cout<<"a"<<endl;
    cout<<"b"<<endl;
    cout<<"c"<<endl;
  }
  cout<<"selesai";
}
```

```
//contoh program perulangan
#include <iostream.h>
void main()
{ int i;
  for (i=1;i<=10;i=i+2)
    cout<<i<<endl;
  cout<<"selesai";
}
```

(Inactive D:\FOR1.EXE)  
1  
3  
5  
7  
9  
selesai

Nilai awal i=1  
Nilai perulangan jika  $i \leq 10$   
Nilai penaik  $i = i + 2$

Pernyataan 1

Pernyataan 2

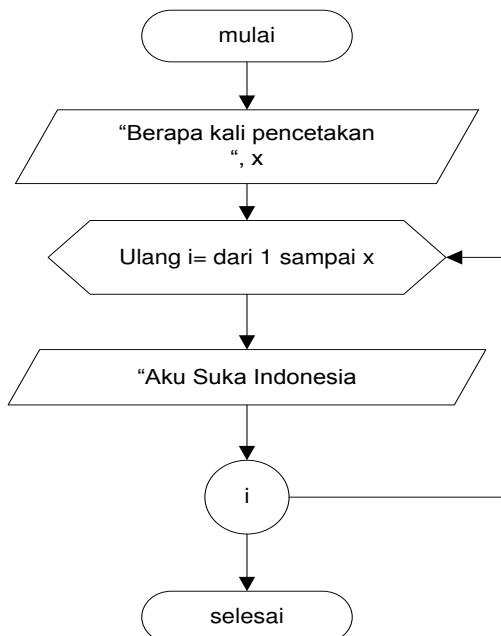
### Penjelasan

- nilai awal perulangan adalah 1 ( $i=i$ )
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=1$ ) masih memenuhi  $i \leq 10$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak isi variabel  $i \rightarrow 1$
  - nilai  $i$  ditambah 2 →  $i=3$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=3$ ) masih memenuhi  $i \leq 10$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak isi variabel  $i \rightarrow 3$
  - nilai  $i$  ditambah 2 →  $i=5$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=5$ ) masih memenuhi  $i \leq 10$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak isi variabel  $i \rightarrow 5$
  - nilai  $i$  ditambah 2 →  $i=7$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=7$ ) masih memenuhi  $i \leq 10$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak isi variabel  $i \rightarrow 7$
  - nilai  $i$  ditambah 2 →  $i=9$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=9$ ) masih memenuhi  $i \leq 10$ 
  - ya → kerjakan bagian yang diulang (pernyataan 1) → mencetak isi variabel  $i \rightarrow 9$
  - nilai  $i$  ditambah 2 →  $i=11$
- lakukan proses pengecekan, apakah nilai  $i$  ( $i=11$ ) masih memenuhi  $i \leq 11$ 
  - tidak → kerjakan pernyataan 2 dan baris seterusnya

Buatlah program untuk mengulang kalimat “Aku Suka Indonesia” sebanyak b kali

**Logika:**

Soal di atas akan menampilkan tulisan Aku Suka Indonesia sebanyak b kali, dimana nilai b dimasukan dari keyboard



Buatlah program untuk menampilkan bilangan ganjil kurang dari 12

**Logika**

- Tentukan nilai awal 1 (misal  $i=1$ ) → karena bilangan ganjil
- Tentukan nilai akhir perulangan jika nilai  $i \leq 12$
- Tentukan nilai penaik adalah 2  $\rightarrow i=i+2$  → agar menjadi nilai ganjil
- Tentukan bagian yang diulang (pernyataan 1) adalah mencetak isi  $i$

Buatlah program untuk menampilkan bilangan genap kurang dari 12!

**Logika**

- Tentukan nilai awal 0 (misal  $i=0$ ) → karena bilangan genap
- Tentukan nilai akhir perulangan jika nilai  $i \leq 12$
- Tentukan nilai penaik adalah 2  $\rightarrow i=i+2$  → agar menjadi nilai genap
- Tentukan bagian yang diulang (pernyataan 1) adalah mencetak isi  $i$

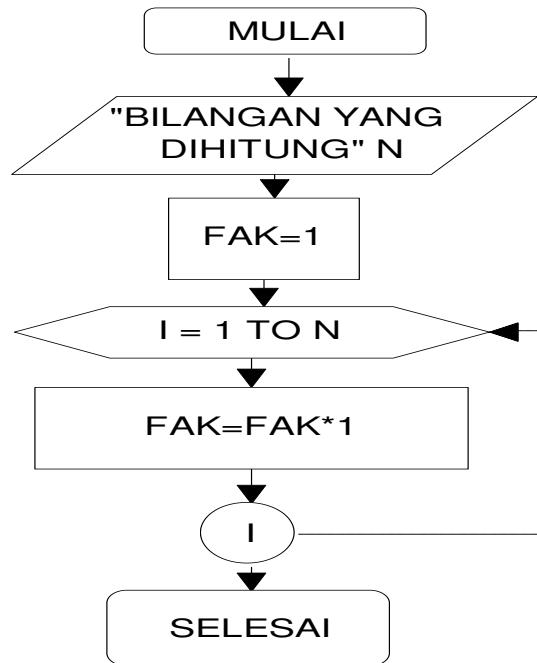
Buatlah program untuk Menghitung Faktorial  
 $N! \rightarrow N*(N-1)!$

$$5! \rightarrow 5*4!$$

$$\text{Sehingga } 5! \rightarrow 5*4*3*2*1$$

### Algoritma

1. Masukkan N data yang ingin dihitung
2. Ulang langkah 3 dari 1 sampai N ( $I \rightarrow 1$  sampai  $N$ )
3. HASIL  $\rightarrow$  HASIL\* $I$
4. Tampilkan hasilnya
5. Selesai

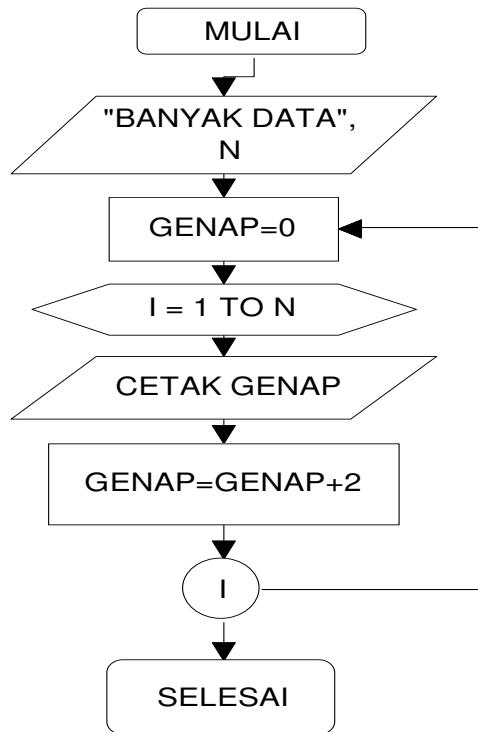


### Mencetak Bilangan BULAT

- Mencetak bilangan bulat dari 0 sampai N data
- Mencetak 5 data pertama  
 $\rightarrow 0,2,4,6,8$

### Algoritma

1. Masukkan banyaknya data yang di cari  $\rightarrow$  N data
2. Inisialisasi GENAP  $\rightarrow 0$
3. Ulang langkah 4 dan 5 dari 1 sampai N ( $I \rightarrow 1$  sampai N)
4. Cetak GENAP
5. Tambah GENAP dengan 2  $\rightarrow$  GENAP=GENAP+2
6. Selesai



## 7.2.Latihan

- Buatlah program untuk menampilkan bilangan ganjil antara 100 sampai 125
- Buatlah program untuk menampilkan bilangan ganjil antara 100 sampai 125 tetapi dengan menampilkan dari nilai terbesar sampai nilai terkecil
- Buatlah program untuk menampilkan bilangan ganjil antara x sampai y, di mana nilai x dan y dimasukan dari keyboard
- Bilangan ajaib (<http://djunaedird.wordpress.com/2007/12/28/bilangan-bilangan-ajaib/> )

Bilangan 37 kalau dikalikan dengan 3, atau kelipatannya,

Misal  $37 \times 3 = ???$

$37 \times 6 = ???$

$37 \times 9 = ???$

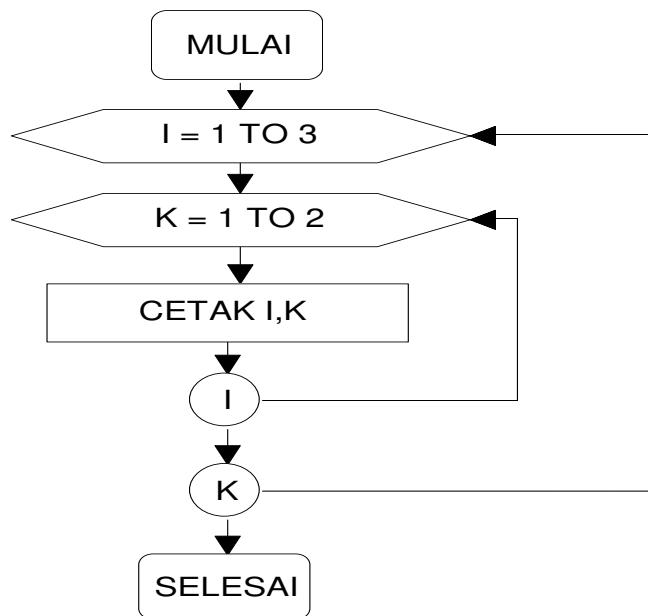
$37 \times 12 = ???$

```
111
222
333
444
555
666
777
888
selesai
```

- Bilangan 15.783 merupakan bilangan ajaib, kenapa?? karena bilangan ini akan memberi hasil yang istimewa bila dikalikan dengan kelipatan 7.
  1.  $15.873 \times 7 = 111.111$
  2.  $15.873 \times 14 = 222.222$
  3.  $15.873 \times 21 = 333.333$
  4. Dan seterusnya jika bilangan tersebut dikalikan dengan kelipatan 7.
- Bilangan 8547 merupakan bilangan ajaib, kenapa?? karena bilangan ini akan memberi hasil yang istimewa bila dikalikan dengan kelipatan 13.
  - $8547 \times 13 = 111.111$
  - $8547 \times 26 = 222.222$
  - $8547 \times 39 = 333.333$
  - Dan seterusnya jika bilangan tersebut dikalikan dengan kelipatan 13.
- Buat tabel suku-suku deret geometri 4, 12, 36, .... Sampai 10 suku berikutnya dan hitung jumlah derek tersebut
- Buat tabel deret kuadrat 1, 4, 9,... sampai 10 suku berikutnya dan hitung jumlah deret tersebut.
- Buat tabel deret  $\frac{1}{2}, \frac{3}{4}, \frac{5}{6}, \dots$  sampai 10 suku dan hitung pula hasil penjumlahannya.
- Buat tabel suku deret  $1, \frac{1}{3}, \frac{1}{5}$  sampai 17 kali. Hitung jumlahnya.
- Hitung deret kubik 1, 8, 27, ... sampai 10 suku. Jumlahkan hasilnya.

### 7.3. Memahami for di dalam for (kalang for)

Pada aplikasi tertentu, kadang diperlukan suatu proses yang berulang dan di dalam perulangan tersebut juga harus dilakukan proses perulangan. Contoh aplikasi yang banyak menggunakan kalang for adalah operasi matriks.



Flowchart di atas, dilakukan proses kalang dalam for.

Proses dilakukan dengan mengulang nilai  $I=1$  dan saat pengulangan nilai  $I=1$  ini dilakukan proses pengulangan  $K=1, K=2$  , Proses selanjutnya menambah nilai  $I=2$  dan dilakukan perulangan kembali  $K=1, K=2$ , kemudian nilai  $I=3$ , kalang  $K=1, K=2$

```

//contoh program perulangan
#include <iostream.h>
void main()
{ float i,j;

  for (i=1;i<=3;i++)
    for (j=1;j<=5;j++)
      cout<<"A"<<endl;
  cout<<"selesai";
}
  
```

The code above is a C++ program demonstrating nested loops. It uses two nested for-loops. The outer loop iterates over  $i$  from 1 to 3. For each value of  $i$ , the inner loop iterates over  $j$  from 1 to 5. Inside the inner loop, the character 'A' is printed 5 times, followed by a new line. After the inner loop completes for all values of  $i$ , the word 'selesai' is printed. The output window shows the character 'A' repeated 15 times (3 rows by 5 columns) followed by the word 'selesai'.

**Penjelasan**

- For pertama akan melakukan proses pengulangan dari 1 sampai 3
  - Bagian yang diulang ternyata juga proses perulangan
    - Proses perulangan mulai dari 1 sampai 5
  - Langkah-langkah pengulangan
    - Ulangan i=1 : benar
      - ⇒ Ulangan j=1 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=2 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=3 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=4 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=5 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=6 : salah
        - Kembali ke pengulangan i
    - Ulangan i=2 : benar
      - ⇒ Ulangan j=1 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=2 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=3 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=4 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=5 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=6 : salah
        - Kembali ke pengulangan i
    - Ulangan i=3 : benar
      - ⇒ Ulangan j=1 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=2 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=3 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=4 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=5 : benar
        - Kerjakan pernyataan 1 → cetak A
      - ⇒ Ulangan j=6 : salah
        - Kembali ke pengulangan i
    - Ulangan i=4 : salah, kerjakan pernyataan 2

**7.4.Latihan**

- Buatlah program untuk menampilkan bilangan dalam segitiga dengan tampilan

1 \*  
2 \*\*  
3 \*\*\*  
4 \*\*\*\*  
5 \*\*\*\*\*

- Buatlah program untuk menampilkan bilangan dalam segitiga dengan tampilan

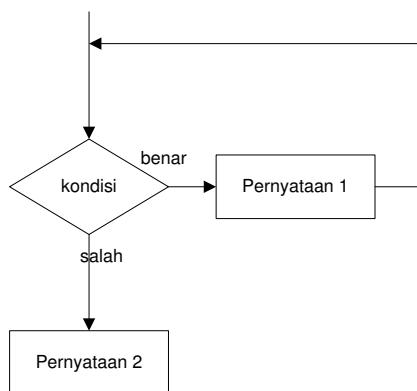
```
*  
* * *  
* * * * *  
* * * * * * *  
* * * * * * * * *
```

### 7.5. Memahami perintah while

Pernyataan ini dipakai untuk mengulang eksekusi dari suatu blok pernyataan yang jumlah ulangannya tergantung dari kondisi yang diberikan, sejauh kondisi true, maka ulangan eksekusi dari blok tersebut terus dilakukan. Format pernyataan while adalah :

## While kondisi Blok pernyataan 1 Blok pernyataan 2

Blok pernyataan 1 akan terus dikerjakan bila hasil kondisi bernilai TRUE, jika kondisi bernilai salah maka akan mengerjakan baris di bawah blok pernyataan 1, blok pernyataan 2. Lain dengan pengulangan for, pengulangan while harus hat-hati, hal ini karena nilai awal dan nilai pengulangan harus didefinisikan sendiri dan tempat nilai awal dan nilai pengulangan tidak bersifat tetap dan bisa berbeda-beda setiap program. Dalam format di atas, nilai awal dan nilai pengulangan tidak secara pasti didefinisikan.



```
# include <iostream.h>
void main()
{int awal;
awal=1;
while (awal<=3)
{ awal=awal+1;
cout<<"IST AKPRIND "<<endl;
}
}
```

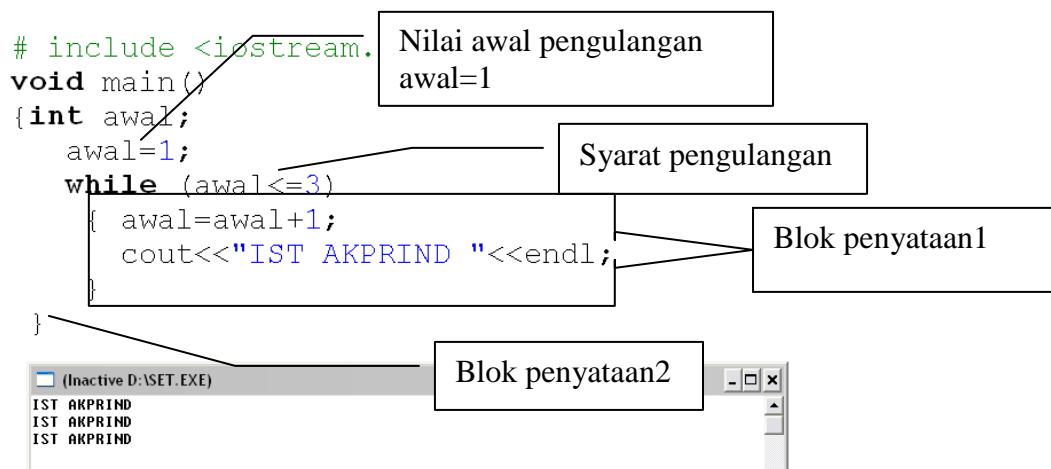


#### Penjelasan

- Nilai awal =1
- Proses pengulangan akan dilakukan selama nilai awal<=3
- Nilai penaik adalah awal=awal+1
- Bagian yang diulang adalah

```
{ awal=awal+1;
cout<<"IST AKPRIND "<<endl; }
```

- Proses pengulangan
  - Awal =1
  - Cek apakah awal<=3
    - ⇒ Benar → kerjakan blok pengulangan
      - awal=awal+1 → awal=2
      - cetak IST AKPRIND
  - Cek apakah awal<=3
    - ⇒ Benar → kerjakan blok pengulangan
      - awal=awal+1 → awal=3
      - cetak IST AKPRIND
  - Cek apakah awal<=3
    - ⇒ Benar → kerjakan blok pengulangan
      - awal=awal+1 → awal=4
      - cetak IST AKPRIND
  - Cek apakah awal<=3
    - ⇒ salah → keluar dari blok pengulangan



### Contoh

```

#include <iostream.h>
void main()
{int awal;
awal=5;
while (awal<=3)
{
    awal=awal+1;
    cout<<"IST AKPRIND "<<endl;
}

```

### Penjelasan

- Program di atas jika dijalankan tidak ada hasilnya, hal ini mengingat nilai awal=5
- Kemudian dilakukan pengecekan apakah awal<=5 → 5<= 3
  - Salah, sehingga blok yang dikerjakan adalah blok pernyataan 2

```
# include <iostream.h>
void main()
{int awal;
awal=1;
while (awal<=3)
{ //awal=awal+1;
cout<<"IST AKPRIND "<<endl;
}
}
```

Dijadikan komentar (tidak diproses)

### Penjelasan

- Proses pengulangan
  - Awal =1
  - Cek apakah awal<=3
    - ⇒ Benar → kerjakan blok pengulangan
      - cetak IST AKPRIND
  - Cek apakah awal<=3
    - ⇒ Benar → kerjakan blok pengulangan
      - cetak IST AKPRIND
  - Cek apakah awal<=3
    - ⇒ Benar → kerjakan blok pengulangan
      - cetak IST AKPRIND
  - proses di atas akan berulang terus-menerus, hal ini dikarenakan nilai awal tidak pernah bertambah sehingga nilai akan selalu bernilai benar.

### Contoh

```
# include <iostream.h>
void main()
{int awal;
awal=1;
while (awal<=3)
{ awal=awal+1;
cout<<"IST AKPRIND "<<endl;
}
cout<<"Yogyakarta "<<endl;
cout<<"selesai "<<endl;
}
```

Blok pernyataan 1

Blok pernyataan 12



### Penjelasan

- Proses pengulangan
  - Awal =1
  - Cek apakah awal<=3
    - ⇒ Benar → kerjakan blok pengulangan
      - awal=awal+1 → awal=2
      - cetak IST AKPRIND

- o Cek apakah awal<=3
  - ⇒ Benar → kerjakan blok pengulangan
    - awal=awal+1 → awal=3
    - cetak IST AKPRIND
- o Cek apakah awal<=3
  - ⇒ Benar → kerjakan blok pengulangan
    - awal=awal+1 → awal=4
    - cetak IST AKPRIND
- o Cek apakah awal<=3
  - ⇒ salah → keluar dari blok pengulangan
  - ⇒ cetak Yogyakarta dan selesai

### contoh

Bagaimana kalau diinginkan mencetak

```
IST AKPRIND
Yogyakarta
i IST AKPRIND
i Yogyakarta
i IST AKPRIND
Yogyakarta
r selesai
```

Ubah blok pernyataan 1 menjadi

```
{ awal=awal+1;
cout<<"IST AKPRIND "<<endl;
cout<<"Yogyakarta "<<endl;
}
```

dan blok pernyataan 2 menjadi

```
cout<<"selesai "<<endl;
```

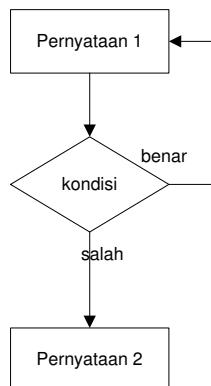
```
# include <iostream.h>
void main()
{int awal;
 awal=1;
 while (awal<=3)
 { awal=awal+1;
 cout<<"IST AKPRIND "<<endl;
 cout<<"Yogyakarta "<<endl;
 }
 cout<<"selesai "<<endl;
}
```

### 7.6.Memahami perintah do - while

Pernyataan ini dipakai untuk mengulang eksekusi dari suatu blok pernyataan yang jumlah ulangannya tergantung dari kondisi yang diberikan, sejauh kondisi true, maka ulangan eksekusi dari blok tersebut terus dilakukan. Format pernyataan while adalah :

**Blok pernyataan 1**  
**While kondisi**  
**Blok pernyataan 2**

Blok pernyataan 1 akan dikerjakan terlebih dahulu dan baru memeriksa hasil kondisi. Jika kondisi bernilai TRUE, maka blok pernyataan 1 akan dikerjakan lagi dan bila kondisi salah, program akan keluar dari proses perputaran dan keluar dari while



### Contoh

```
#include <iostream.h>
void main()
{
    int i=0,a=5;
    do
    {
        cout<<"nilai = "<<i<<endl;
        i++;
    }
    while (i<=10);
}
```

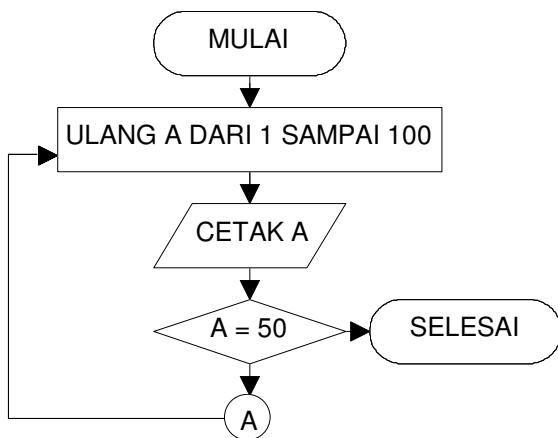
Pernyataan 1

Pernyataan 2

```
nilai = 0
nilai = 1
nilai = 2
nilai = 3
nilai = 4
nilai = 5
nilai = 6
nilai = 7
nilai = 8
nilai = 9
nilai = 10
```

### Penjelasan

Proses akan melakukan pengecekan pengulangan di belakang. Proses akan mengerjakan pernyataan 1 terlebih dahulu dan akan melakukan pengecekan.



### Contoh

```

#include <iostream.h>
void main()
{
    int i=10, a=5;
    do
    {
        cout<<"nilai = "<<i<<endl;
        i++;
    }
    while (i<=10);
}
  
```



### Penjelasan

- Mendeklarasikan variabel I dan a serta memberi nilai awal i=10 dan a=5
- Do ➔ awal pengulangan
  - Cetak nilai = 10
  - Nilai ditambah 1 sehingga i=11
  - Cek apakah i<=10 (apakah 11<=10)
    - ⇒ Salah, program keluar dari while dan pernyataan 2

### 7.7. Pernyataan break

Pernyataan break dapat digunakan untuk memaksa suatu perulangan keluar dari proses perulangan walaupun sebenarnya proses tersebut masih berlangsung. Dalam pembahasan di atas sudah diimplementasikan dalam perintah switch. Dengan penggunaan break ini, proses switch akan selesai.

```
#include <iostream.h>
void main()
{
    int i,a=5;
    for(i=0;i<=10;i++)
        { cout<<"nilai i = "<<i<<endl;
          if (a==5)
              break;
        }
    cout<<"selesai";
}
```

Jika kondisi benar, maka akan mengerjakan perintah break, artinya keluar dari perulangan

**contoh**

```
#include <iostream.h>
void main()
{
    int i,a=5;
    for(i=0;i<=10;i++)
        cout<<"nilai i = "<<i<<endl;

    cout<<"selesai";
}
```

```
nilai i = 0
nilai i = 1
nilai i = 2
nilai i = 3
nilai i = 4
nilai i = 5
nilai i = 6
nilai i = 7
nilai i = 8
nilai i = 9
nilai i = 10
selesai
```

Program di atas belum ada perintah break sehingga hasilnya akan melakukan proses pengulangan sampai 11 kali (dari 0 sampai 10). Bandingkan dengan contoh di bawah ini

```
#include <iostream.h>
void main()
{
    int i,a=5;
    for(i=0;i<=10;i++)
        { cout<<"nilai i = "<<i<<endl;
          if (a==5)
              break;
        }
    cout<<"selesai";
}
```

```
nilai i = 0
selesai
```

**Penjelasan :**

- Dilakukan proses mendeklarasikan variabel i dan a sekaligus diisi dengan 5
- Melakukan proses pengulangan

- $i=0$ 
  - $\Rightarrow$  Cek apakah  $i \leq 10$  ya  $\rightarrow$  cetak Nilai  $i = 0$
  - $\Rightarrow$  If ( $a == 5$ ) cek apakah  $a = 5$  ya  $\rightarrow$  break, artinya program keluar dari proses pengulangan dan mengerjakan perintah diluar pengulangan, mencetak Selesai
- Walaupun nilai  $i$  baru 0 dan belum sampai 10, proses perulangan tetap dihentikan

### Contoh

```
#include <iostream.h>
void main()
{   int i=0;
    while (i<=10)
        { cout<<"isi i = "<<i<<endl;
          if (i%5==1)
              break;
          i++;
        }

    cout<<"selesai";
}
```



### Penjelasan

- Nilai  $i$  diset 0
- Proses pengulangan, pengecekan apakah  $i \leq 10$ 
  - Ya  $\rightarrow$  dilakukan proses pengulangan
    - $\Rightarrow$  Cetak isi  $i = 0$
    - $\Rightarrow$  Cek apakah  $i \% 5 == 1$  (apakah  $0 == 1$ )
      - Tidak, lanjutkan proses pengulangan
      - Nilai  $i$  ditambah 1  $\rightarrow i = 1$
  - Cek apakah  $i \leq 10$  (apakah  $1 < 10$ )
    - Ya  $\rightarrow$  dilakukan proses pengulangan
      - $\Rightarrow$  Cetak isi  $i = 1$
      - $\Rightarrow$  Cek apakah  $i \% 5 == 1$  (apakah  $1 == 1$ )
        - Ya, kerjakan perintah break
        - Keluar dari proses pengulangan
        - Cetak selesai

### 7.8.Pernyataan continue

Pernyataan continue dapat digunakan untuk mengarahkan proses kembali ke pengulangan kembali. Dengan continue proses akan melanjutkan pengulangan untuk iterasi selanjutnya, Jadi, bila pengulangan sudah ke 10, maka proses continue akan melanjutkan ke pengulangan 11.

```
#include <iostream.h>
void main()
{
    for (int i=0; i<=10; i++)
    {
        cout<<"nilai = "<<i<<endl;
        continue;
        cout<<"di bawah continue"<<endl;
    }
}
```

**Penjelasan :**

- Perintah continue akan mengakibatkan proses akan kembali ke for. Jadi perintah cout<<" di bawah continue"<<endl tidak pernah dikerjakan.

```
#include <iostream.h>
void main()
{
    int i=0, a=5;
    while (i<=10)
    {
        cout<<"nilai = "<<i<<endl;
        if (i<=5)
        {
            i++;
            continue;
        }
        cout<<"di bawah continue"<<endl;
        i++;
    }
}
```

```
nilai = 0
nilai = 1
nilai = 2
nilai = 3
nilai = 4
nilai = 5
nilai = 6
di bawah continue
nilai = 7
di bawah continue
nilai = 8
di bawah continue
nilai = 9
di bawah continue
nilai = 10
di bawah continue
```

**Penjelasan**

- Mendeklarasikan variabel i dan a serta memberi nilai awal i=0 dan a=5
- Proses pengulangan, pengecekan apakah  $i \leq 10$  (while ( $i \leq 10$ ))
  - Ya → dilakukan proses pengulangan
    - ⇒ Cetak nilai = 0
    - ⇒ Cek apakah  $i \leq 5$  (apakah  $1 \leq 5$ )
      - Ya , Nilai i ditambah 1 →  $i=1$
      - lanjutkan proses pengulangan
- Cek apakah  $i \leq 10$  (apakah  $2 < 10$ )
  - Ya → dilakukan proses pengulangan
    - ⇒ Cetak nilai = 1
    - ⇒ Cek apakah  $i \leq 5$  (apakah  $1 \leq 5$ )
      - Ya , Nilai i ditambah 1 →  $i=2$
      - lanjutkan proses pengulangan
      - Cetak selesai

dan seterusnya pada langkah 6
- Cek apakah  $6 \leq 10$  (apakah  $6 < 10$ )
  - Ya → dilakukan proses pengulangan
    - ⇒ Cetak nilai = 6

- ⇒ Cek apakah  $i \leq 5$  (apakah  $6 \leq 5$ )
  - Tidak , keluar dari IF
  - Kerjakan  $\text{cout} \ll "di bawah continue" \ll \text{endl};$
  - Tambahkan  $i = i + 1;$
- Lanjutkan ke proses pengulangan

### Contoh

```
#include <iostream.h>
void main()
{
    int i=0,a=5;
    while (i<=10)
    {
        cout<<"nilai = "<<i<<\n;
        if (i<=5)
            continue;
        cout<<"di bawah continue"<<\n;
        i++;
    }
}
```

### Penjelasan

Program di atas jika dijalankan akan melakukan proses pengulangan terus-menerus. Hal ini disebabkan saat nilai  $i=0$  dilakukan pengecekan apakah  $0 \leq 5$ , maka proses bernilai benar dan akan mengerjakan proses continue. Akibat perintah continue ini berarti proses kembali ke pengulangan dan nilai  $i$  tidak pernah bertambah (nilai tetap  $i=0$ ).

**7.9.Latihan**

1. Jumlah penduduk kota A pada tahun 1970 adalah 1.000 dengan pertambahan penduduk rata-rata 7%. Pada tahun yang sama jumlah penduduk kota B adalah 2.000 dengan pertumbuhan rata-rata 5 %. Jika pertambahan penduduk kedua kota tersebut tetap, buatlah program untuk menentukan pada tahun ke berapa penduduk kedua kota A lebih banyak dari Kota B

Dalam sistem bunga majemuk, besar pinjaman pada akhir suatu periode dinyatakan dengan persamaan

$$\text{Pakhir} = \text{P awal} (1+\text{bunga})^n$$

Dengan

Pakhir = pinjaman pada akhir periode n

Pawal = pinjaman awal

Bunga =besarnya bunga dalam persen

N = periode peminjaman

2. Menghitung bilangan pangkat 3 yang diperoleh dengan penjumlahan suku-sukunya

**Logika**

$$2^3 = 3 + 5 = 8$$

$$3^3 = 7 + 9 + 11 = 27$$

$$4^3 = 13 + 15 + 17 + 19 = 64$$

$$5^3 = 21 + 23 + 25 + 27 + 29 = 125$$

dan seterusnya ....

Untuk mencari suku awal rumusnya :  $n(n-1)+1$

Misal

$6^3$  maka suku awalnya adalah  $6(6-1)+1 = 31$  dan untuk suku selanjutnya selalu ditambah 6 sampai suku ke 6

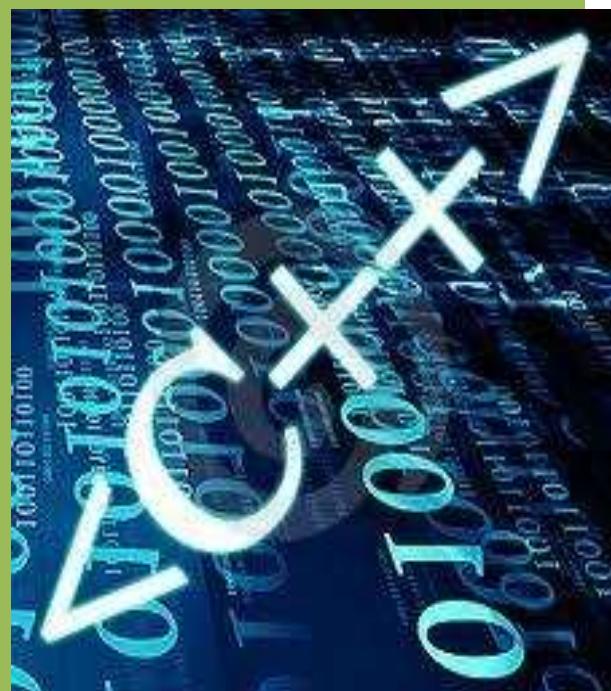
Jadi hasilnya  $31+33+35+37+39+41$

**Algoritma**

1. Masukan data pangkat yang akan dihitung, n
2. Hitung suku awal  $\rightarrow$  awal= $n*(n-1)+1$
3. Cetak awal
4. Ulang i dari 2 sampai n
  - Awal=awal+2
  - Cetak awal

# 8

## FUNGSI



## Tujuan Instruksional

Setelah membaca bab ini, diharapkan pembaca memahami konsep fungsi, manfaat fungsi serta aturan dalam pembuatan fungsi. Dalam materi ini pembaca diharapkan dapat memahami parameter, nilai balik serta lingkup variabel (variabel lokal, global dan statik)

## Materi

-  Memahami Fungsi
-  Fungsi Tanpa Nilai Balik
-  Fungsi dengan Nilai Parameter
-  Fungsi dengan Nilai Balik
-  Lingkup variabel

Dalam pembuatan suatu program yang panjang dan kompleks sering dijumpai suatu bagian/ potongan program sering digunakan di bagian lain dalam program tersebut. Potongan program yang sering digunakan ini tentunya menjadi persoalan diantaranya program jadi panjang karena beberapa hal sama ditulis berulang-ulang dan akan menimbulkan persoalan jika dilakukan proses modifikasi.

ilustrasi

```
=====
MAHESWARI
=====
AQUILA
=====
VERNANDA
=====
```

Buat program dengan tampilan di atas

```
# include <iostream.h>
void main()
{
    cout<<"======"<<endl;
    cout<<"MAHESWARI "<<endl;
    cout<<"======"<<endl;
    cout<<"AQUILA "<<endl;
    cout<<"======"<<endl;
    cout<<"VERNANDA "<<endl;
}
```



#### Permasalahan :

- Dalam program di atas ada perintah `cout<<"======"<<endl;` yang selalu berulang. Bisakah program di atas diselesaikan dengan proses perulangan.

#### Jawabannya : TIDAK

Mengingat `cout<<"======"<<endl;` memang berulang tetapi tidak berturut-turut, sehingga kalau dibuat dengan perintah perulangan tidak bisa

- Bagaimana kalau tanda ===== diganti dengan tanda -----
  - Berapa kali harus dilakukan proses modifikasi

Dari permasalahan tersebut solusi yang dapat digunakan adalah dengan menggunakan fungsi.

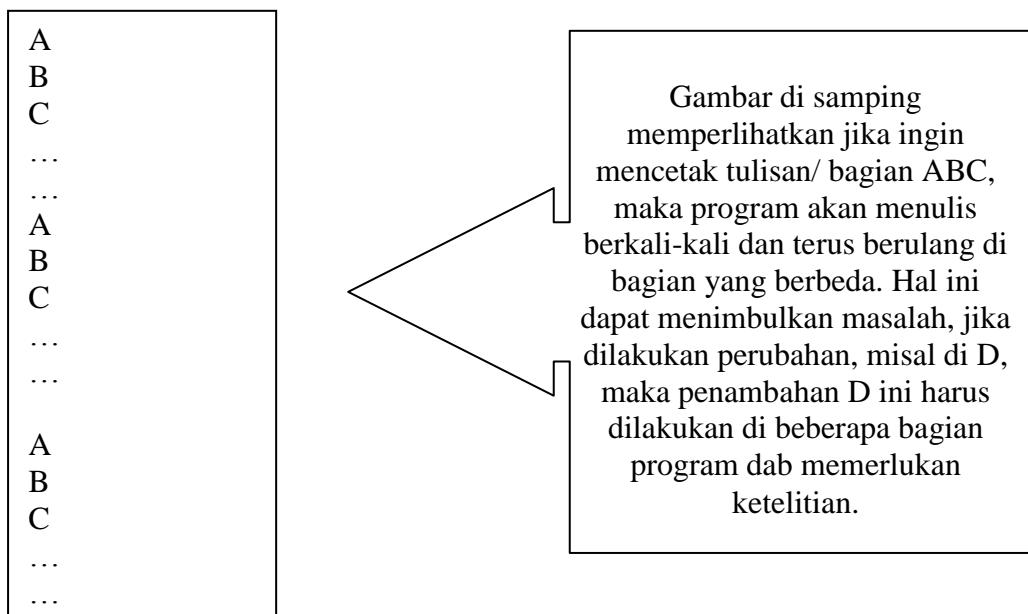
### 8.1. Memahami Fungsi

Fungsi atau lebih dikenal sebagai sub program merupakan cara membuat program dengan membuat bagian-bagian tertentu dari suatu program. Walaupun dibuat dengan bagian-bagian yang lebih kecil semua bagian ini tetap menjadi satu-kesatuan.

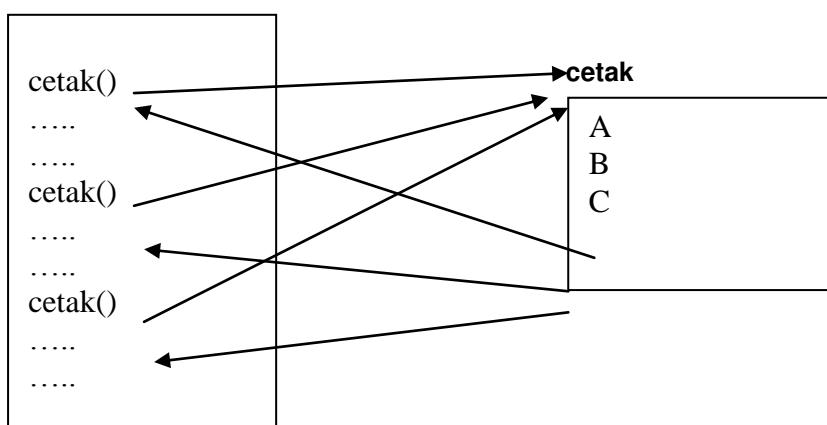
Tujuan pembuatan fungsi adalah

- Memudahkan dalam pengembangan program.
- Menghemat ukuran program

#### Tanpa Fungsi



#### Dengan Fungsi



Dari ilustrasi di atas, jika suatu fungsi sudah dibuat (misal diberi nama cetak), maka fungsi tersebut dapat dipanggil berkali-kali. Artinya jika menginginkan mencetak tulisan ABC, programmer tidak perlu menulis lagi, tetapi cukup memanggil nama fungsi tersebut. Demikian juga jika dilakukan perubahan maka perubahan cukup dilakukan di bagian fungsi saja.

Misal ingin menambah D, maka proses perubahan/ penambahan cukup dilakukan di bagian fungsi saja

Dalam pembuatan suatu program yang panjang dan kompleks sering dijumpai suatu bagian/ potongan program sering digunakan di bagian lain dalam program tersebut. Potongan program yang sering digunakan ini tentunya menjadi persoalan diantaranya program jadi panjang karena beberapa hal sama ditulis berulang-ulang dan akan menimbulkan persoalan jika dilakukan proses modifikasi.

### Ilustrasi

```
=====
MAHESWARI
=====
AQUILA
=====
VERNANDA
=====
```

Buat program dengan tampilan di atas

```
# include <iostream.h>
void main()
{
    cout<<"======"<<endl;
    cout<<"MAHESWARI "<<endl;
    cout<<"======"<<endl;
    cout<<"AQUILA "<<endl;
    cout<<"======"<<endl;
    cout<<"VERNANDA "<<endl;
}
```



### Permasalahan :

- Dalam program di atas ada perintah `cout<<"======"<<endl;` yang selalu berulang. Bisakah program di atas diselesaikan dengan proses perulangan.

### Jawabannya : TIDAK

Mengingat `cout<<"======"<<endl;` memang berulang tetapi tidak berturut-turut, sehingga kalau dibuat dengan perintah perulangan tidak bisa

- Bagaimana kalau tanda ===== diganti dengan tanda -----
  - Berapa kali harus dilakukan proses modifikasi

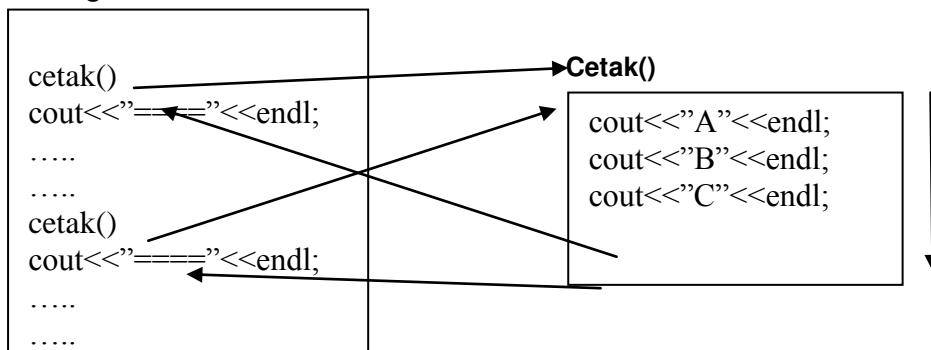
Dari permasalahan tersebut solusi yang dapat digunakan adalah dengan menggunakan fungsi.

## 8.2.Fungsi Tanpa Nilai Balik

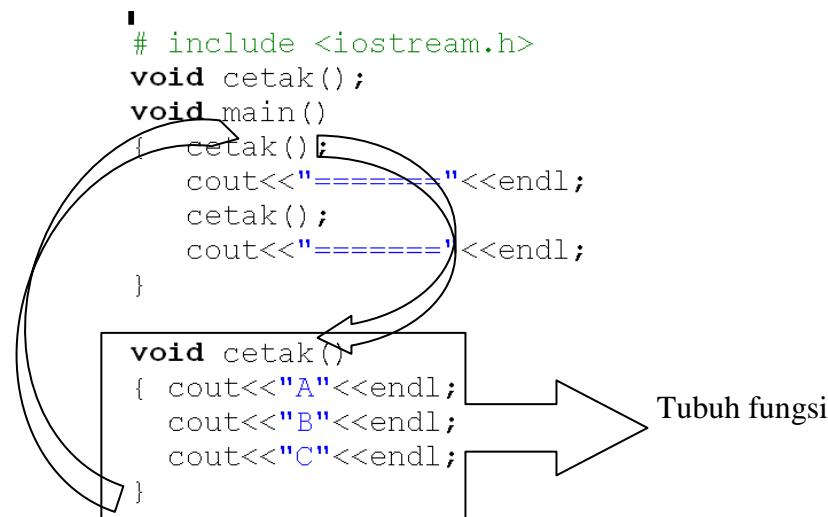
Fungsi yang menggunakan kata kunci void sering disebut juga fungsi tanpa nilai balik. Fungsi ini dalam proses pemanggilan hanya berguna untuk menjalankan program yang ada dalam tubuh fungsi dan tidak membawa suatu nilai balik.

### Logika

#### Program utama



Dalam program utama perintah cetak artinya memanggil fungsi cetak dan fungsi cetak akan menjalankan perintah-perintah yang ada dalam tubuh fungsi dan setelah selesai semua perintah dikerjakan fungsi kembali ke program utama tanpa memberikan suatu nilai balik. Jadi pemanggilan fungsi cetak hanya berguna untuk menjalankan apa yang ada dalam tubuh fungsi tersebut.



Dari ilustrasi di atas, pemanggilan fungsi cetak hanya menjalankan isi dari fungsi cetak dan tidak ada nilai yang dikembalikan ke program utama. Hal seperti inilah yang dimaksud dengan fungsi tanpa nilai balik (fungsi void)

Ciri-ciri dari jenis fungsi Void adalah sebagai berikut:

- Tidak adanya keyword return
- Tidak adanya tipe data di dalam deklarasi fungsi
- Menggunakan keyword void
- Tidak dapat langsung ditampilkan hasilnya
- Tidak memiliki nilai kembalian fungsi

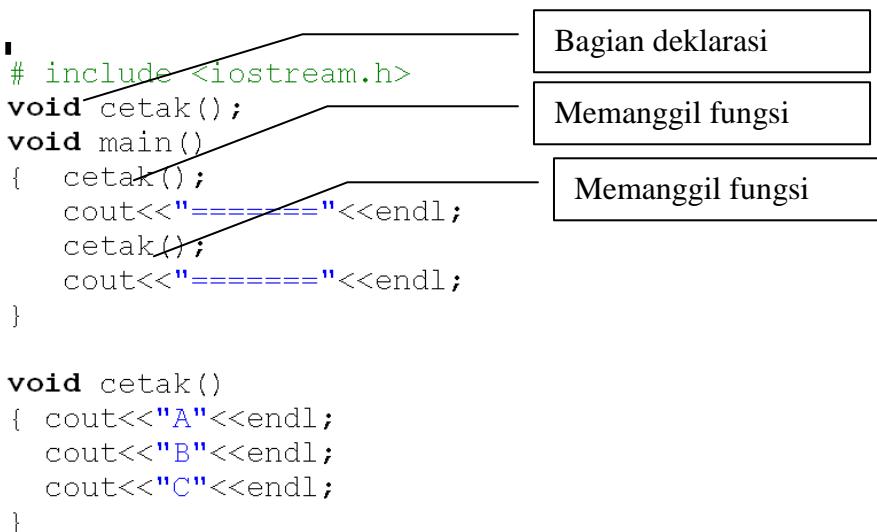
Langkah-langkah membuat fungsi tanpa nilai balik

1. Membuat deklarasi fungsi .Letak deklarasi di atas void main(). Bentuk umum pendeklarasian

```
void nama_fungsi();
```

2. Pendefinisian fungsi. Letak definisi di bagian bawah program utama. Tubuh fungsi diletakkan dalam pendefinisian ini.

```
void nama_fungsi()
{
    Tubuh fungsi
}
```



### Penjelasan

- `cetak();` → program memanggil fungsi
- Fungsi `cetak` dijalankan, setelah selesai program fungsi kembali ke program utama (program yang memanggil)
- Fungsi `cetak` dipanggil dan dijalankan kembali

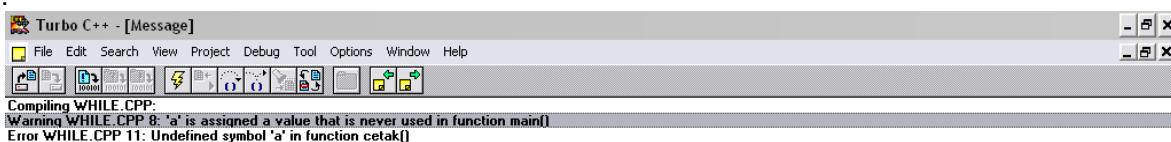
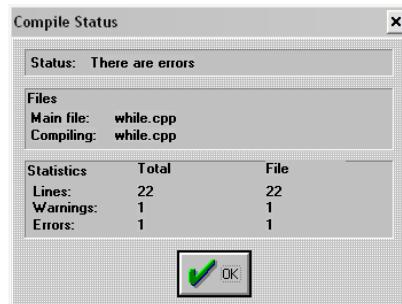
### 8.3.Fungsi dengan Nilai Parameter

Dalam pemrograman akan sering dijumpai melakukan proses ‘pemindahan data’ dari program utama ke bagian suatu fungsi. ‘Pemindahan’ data harus dilakukan pada bagian parameter. Jika data yang ada di program utama ingin dipindahkan ke bagian fungsi tidak menggunakan parameter akan mengalami kesalahan

Contoh :

```
# include <iostream.h>
void cetak();
void main()
{ int a=5;
  cetak();
}

void cetak()
{ cout<<"isi a = "<<a<<endl;
}
```



### Kenapa salah :

Nilai suatu variabel bersifat lokal, artinya jika variabel tersebut didefinisikan di bagian program utama, maka nilai variabel tersebut hanya dikenal pada bagian program utama tersebut. Hal ini akan berakibat salah, jika nilai variabel tersebut ditampilkan di bagian lain, misal ditampilkan di bagian suatu fungsi.

Dalam contoh di atas, variabel a dideklarasikan di program utama dan dicetak di bagian fungsi lain, hal ini tentunya berakibat salah atau variabel a tidak dikenal di fungsi cetak. Agar nilai a dikenal di bagian fungsi salah satu cara yang bisa digunakan adalah melewatkannya nilai a dari suatu program utama ke bagian suatu fungsi. Cara seperti ini sering disebut dengan melewatkannya nilai dengan parameter.

### Logika parameter

#### Program utama

```
int a=5
cetak(a )
cout<<"===="<<endl;
.....
A=a+10
cetak(a)
cout<<"===="<<endl;
.....
.....
```

#### Cetak(int x)

```
cout<<"x"<<x<<endl;
```

- `cetak(a)` artinya memanggil fungsi cetak dan memberikan/ melewatkannya nilai variabel a ke fungsi cetak. Fungsi cetak akan menerima data tersebut dan disimpan di variabel x.
- Jadi yang diberikan ke fungsi hanyalah isi dari suatu variabel dan disalin ke variabel x.
- Perubahan nilai x tidak akan berpengaruh pada variabel a

Langkah membuat fungsi dengan parameter sama dengan membuat fungsi diatas hanya ada penambahan daftar parameter yang akan dilewatkan.

- 1 Membuat deklarasi fungsi .Letak deklarasi di atas void main(). Bentuk umum pendeklarasian  
**void nama\_fungsi(daftar parameter);**
- 2 Pendefinisan fungsi. Letak definisi di bagian bawah program utama. Tubuh fungsi diletakkan dalam pendefinisan ini.  
**void nama\_fungsi(daftar parameter)**  
{  
    Tubuh fungsi  
}

```
#include <iostream.h>
void cetak(int,int);

void main()
{
    int a,b;
    cetak(5,6);
    a=80;
    b=45;
    cetak(a,b);
}

void cetak(int x,int y)
{
    cout<<"isi x = "<<x<<endl;
    cout<<"isi y = "<<y<<endl;
}
```



### Penjelasan

- **void cetak(int,int);**
  - ➔ mendeklarasikan suatu fungsi tanpa nilai balik tetapi ada daftar parameter. Daftar parameter ada sebanyak 2 dan masing-masing bertipe integer
- **void cetak(int x,int y)**

```
{ cout<<"isi x = "<<x<<endl;
    cout<<"isi y = "<<y<<endl;
}
```

  - ➔ Bagian tubuh fungsi int x dan int y digunakan sebagai tempat untuk menampung nilai parameter yang dilewatkan dari pemanggil fungsi.

- Cetak(5,6) → memanggil fungsi cetak dan melewatkannya dengan parameter dan data 5 akan ditampung di variabel x dan data 6 akan ditampung di variabel y
- Cetak(a,b) → memanggil fungsi cetak dan melewatkannya dengan isi variabel a dan isi variabel b sebagai parameter dan data variabel a akan ditampung di variabel x dan data variabel b akan ditampung di variabel y

### Contoh

```
#include <iostream.h>
void hitung(float,float);

void main()
{ int a,b;
    cout<<"masukan nilai alas = ";
    cin>>a;
    cout<<"masukan nilai tinggi = ";
    cin>>b;
    hitung(a,b);
}

void hitung(float alas,float tinggi)
{ float luas;
    luas=1/2.0*alas*tinggi;
    cout<<"isi luas segitiga = "<<luas<<endl;
}
```

### Penjelasan

```
#include <iostream.h>
void hitung(float,float);
```

→ deklarasi fungsi tanpa nilai balik dan ada 2 parameter berupa bilangan pecahan

```
void hitung(float alas,float tinggi)
{ float luas;
    luas=1/2.0*alas*tinggi;
    cout<<"isi luas segitiga = "<<luas<<endl;
}
```

→ definisi fungsi tanpa nilai balik dan ada 2 parameter berupa bilangan pecahan

hitung(a,b);

→ memanggil fungsi hitung dengan parameter isi variabel a dan isi variabel b. Isi variabel a dan b didapat dari proses inputan.

#### 8.4. Fungsi dengan Nilai Balik

Berbeda dengan fungsi tanpa nilai balik, fungsi dengan nilai balik berguna untuk melakukan suatu proses yang dapat mengembalikan sebuah nilai. Dalam fungsi ini harus di definisikan tipe data dari nilai yang akan dikembalikan.

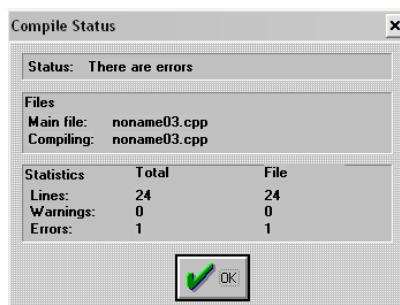
Perhatikan program di bawah ini

```
#include <iostream.h>
void hitung(float,float);

void main()
{ int a,b;
cout<<"masukan nilai alas = ";
cin>>a;
cout<<"masukan nilai tinggi = ";
cin>>b;
hitung(a,b);
cout<<"isi luas segitiga = "<<luas<<endl;
}

void hitung(float alas,float tinggi)
{ float luas;
luas=1/2.0*alas*tinggi;
cout<<"isi luas segitiga = "<<luas<<endl;
}
```

Program di atas merupakan modifikasi dari program sebelumnya. Penambahan dilakukan dengan menambahkan perintah **cout<<"isi luas segitiga = "<<luas<<endl;**; tetapi justru pada baris ini akan dinyatakan salah saat program dijalankan.



#### Kenapa hal ini terjadi kesalahan ?

Kesalahan ini terjadi karena variabel luas dalam perintah di atas tidak dikenal di program utama. Memang variabel luas sudah dideklarasikan di tubuh fungsi hitung. Pendefinisian suatu variabel hanya dikenal di bagian dimana variabel tersebut dideklarasikan, sehingga variabel luas hanya dikenal di fungsi hitung dan tidak dikenal di bagian lain atau di program utama.

Jadi, setelah semua perintah yang ada di fungsi selesai dikerjakan, maka semua variabel yang ada dalam fungsi tersebut akan dihapus, sehingga variabel tersebut tidak dikenal lagi.



Persoalannya bagaimana agar nilai hasil perhitungan (variabel luas) yang dilakukan di bagian suatu fungsi dapat ditampilkan atau digunakan pada bagian program/ fungsi yang berbeda. Agar suatu nilai dapat digunakan di fungsi lain, maka solusinya adalah menggunakan fungsi dengan nilai balik.

Langkah membuat fungsi dengan nilai balik sama dengan membuat fungsi diatas hanya ada penambahan tipe nilai balik dan data yang menjadi nilai balik daftar parameter yang akan dilewatkkan.

- 1 Membuat deklarasi fungsi .Letak deklarasi di atas void main(). Bentuk umum pendeklarasian

```
Tipe_data nama_fungsi(daftar parameter);
```

- 2 Pendefinisan fungsi. Letak definisi di bagian bawah program utama. Tubuh fungsi diletakkan dalam pendefinisan ini.

```
Tipe_data nama_fungsi(daftar parameter)
```

```
{
```

```
.....
```

```
return nilai balik;
```

```
}
```

#### contoh

```
#include <iostream.h>
float hitung(float, float);
```

Fungsi dengan nilai balik dan nilai balik berupa bilangan float (pecahan)

```
void main()
{ int a,b;
cout<<"masukan nilai alas = ";
cin>>a;
cout<<"masukan nilai tinggi = ";
cin>>b;
cout<<"isi luas segitiga = "<<hitung(a,b)<<endl;
}
```

Ada 2 parameter dan bertipe float

```
float hitung(float alas, float tinggi)
{ float luas;
luas=1/2.0*alas*tinggi;
return luas;
}
```

Memanggil fungsi cetak dengan memberikan 2 parameter

```
(Inactive D:\FUNSI.EXE)
masukan nilai alas = 4
masukan nilai tinggi = 5
isi luas segitiga = 10
```

Nilai yang menjadi nilai balik adalah isi variabel luas

### Penjelasan

`cout<<"isi luas segitiga = "<<hitung(a,b)<<endl;`

- ➔ memanggil fungsi hitung dengan melewatan 2 parameter a dan b. Nilai a dan b dapat dari proses pemasukan data

```
float hitung(float alas,float tinggi)
{
    float luas;
    luas=1/2.0*alas*tinggi;
    return luas;
}
```

- ➔ fungsi hitung merupakan fungsi dengan nilai balik dan ada 2 parameter. Nilai balik adalah variabel luas

Hasil dari nilai balik akan diberikan pada instruksi `cout<<"isi luas segitiga = "<<hitung(a,b)<<endl;` artinya nilai balik akan ditampilkan ke layar monitor.

### contoh

```
#include <iostream.h>
float hitung(float, float);

void main()
{   int a,b,c;
    cout<<"masukan nilai alas = ";
    cin>>a;
    cout<<"masukan nilai tinggi = ";
    cin>>b;
    c=hitung(a,b);
    cout<<"isi luas segitiga = "<<c<<endl;
}

float hitung(float alas, float tinggi)
{   float luas;
    luas=1/2.0*alas*tinggi;
    return luas;
}
```

### Penjelasan

- 1 `C=hitung(a,b)` ➔ memanggil fungsi hitung dengan memberikan 2 parameter berupa nilai a dan b, dimana nilai a dan b dimasukan dari keyboard
- 2 Nilai a akan diberikan ke variabel alas dan nilai b akan diberikan variabel tinggi
- 3 Return luas ➔ memberikan nilai balik berupa nilai dari isi variabel luas dan hasil dari nilai balik ini akan disimpan di variabel c

### 8.5.Lingkup variabel

Dalam bahasa C suatu variabel yang didefinisikan mempunyai ruang lingkup yang terbatas. Suatu variabel yang dideklarasikan dalam suatu fungsi hanya dikenal dalam fungsi itu sendiri, kecuali jika variabel tersebut dideklarasikan secara global. Dalam bahasa c++ lingkup variabel dibedakan :

#### Variabel Lokal

Suatu variabel yang dideklarasikan dalam suatu tubuh fungsi hanya dikenal dalam tubuh fungsi itu sendiri. Sifat variabel seperti ini disebut dengan variabel lokal.

Contoh 1

```
#include <iostream.h>
void luas(float, float);
void main()
{
    int a, b;
    a=5;           ← Variabel lokal
    b=9;           ← Variabel lokal
    cout<<"isi a = "<<a<<endl;
    cout<<"isi b = "<<b<<endl;
    luas(a,b);
    cout<<"isi a = "<<a<<endl;
    cout<<"isi b = "<<b<<endl;
}
void luas(float x, float y)
{
    int a, b;      ← Variabel lokal
    a=x+y;
    b=a+6;
    cout<<"isi a = "<<a<<endl;
    cout<<"isi b = "<<b<<endl;
}
```

#### Penjelasan

Dalam program diatas, antara program utama dan fungsi luas mempunyai variabel yang sama ( a dan b) tetapi antara variabel yang ada di program utama dan fungsi luas tidak ada sangkut pautnya. Perubahan nilai pada variabel a dan b baik di program utama maupun di fungsi tidak saling mempengaruhi. Sehingga bila dijalankan hasilnya adalah

```
isi a = 5
isi b = 9
isi a = 14
isi b = 20
isi a = 5
isi b = 9
```

#### Kenapa hasilnya seperti diatas

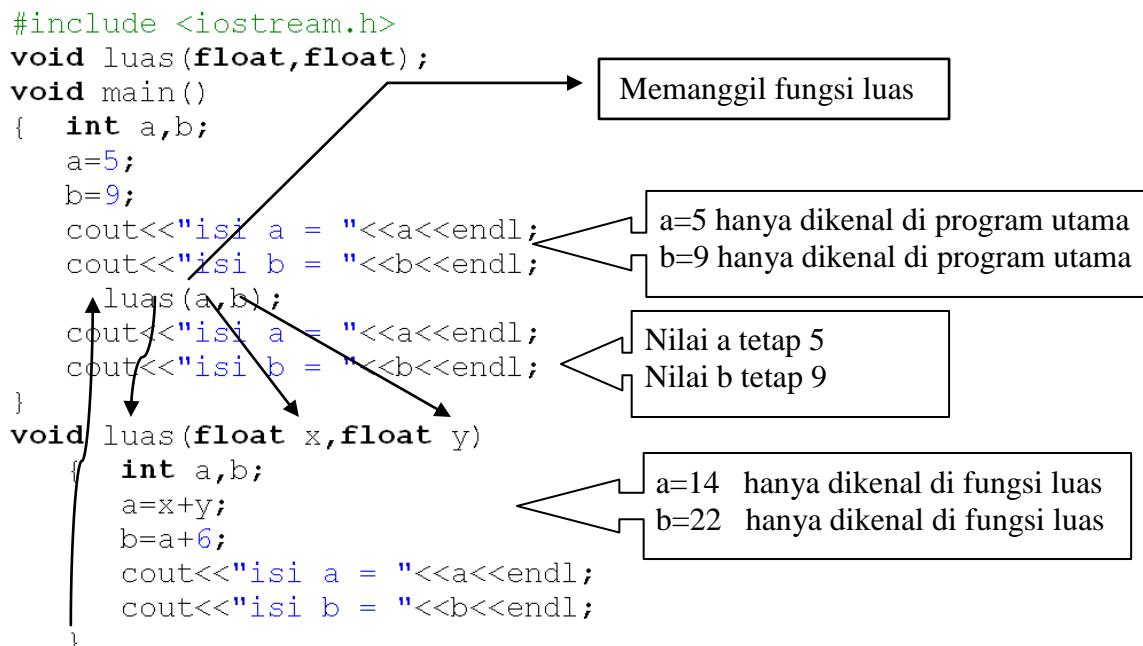
|                            |                                                       |
|----------------------------|-------------------------------------------------------|
| int a,b;                   | → variabel lokal di program utama                     |
| a=5;                       | → menset variabel a dengan 5 (lokal di program utama) |
| b=9;                       | → menset variabel a dengan 5 (lokal di program utama) |
| cout<<"isi a = "<<a<<endl; | → menampilkan isi variabel a                          |
| cout<<"isi b = "<<b<<endl; | → menampilkan isi variabel b                          |
| luas(a,b);                 | → memanggil fungsi luas                               |

```

void luas(float x,float y)
{
    int a,b;           ➔ variabel lokal di fungsi luas
    a=x+y;            ➔ variabel a=14 ➔ lokal di fungsi luas
    b=a+6;            ➔ variabel b=22 ➔ lokal di fungsi luas
    cout<<"isi a = "<<a<<endl;
    cout<<"isi b = "<<b<<endl;
}

```

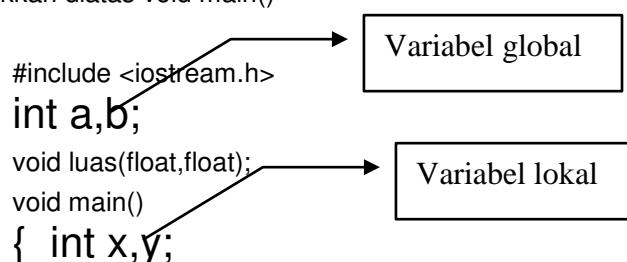
Jadi setelah memanggil fungsi luas, perubahan variabel a dan b di tubuh fungsi luas tidak berpengaruh di program utama sehingga isi a dan b tetap sama



Dengan menggunakan variabel lokal ini, walaupun dalam fungsi yang berbeda mempunyai nama variabel yang sama oleh kompiler ini tetap dianggap berbeda dan masing-masing mempunyai nilai sendiri-sendiri.

### Variabel Global

Lain dengan variabel global, perubahan data pada suatu bagian fungsi akan berpengaruh pada semua bagian. Jika menginginkan variabel bersifat global, proses pendeklarasian variabel diletakkan diatas void main()



**contoh**

```

#include <iostream.h> → Variabel global
int a,b;
void luas(float,float) → Mensest variabel a = 5 dan b=9
void main()
{
    a=5;
    b=9;
    cout<<"isi a = "<<a<<endl;
    cout<<"isi b = "<<b<<endl;
    luas(a,b);
    cout<<"isi a = "<<a<<endl;
    cout<<"isi b = "<<b<<endl;
}
void luas(float x,float y)
{
    a=x+y;
    b=a+6;
    cout<<"isi a = "<<a<<endl;
    cout<<"isi b = "<<b<<endl;
}

```

**Penjelasan**

```

int a,b; → set variabel a dan b bersifat global
void luas(float,float);

```

```

void main()
{
    a=5; → memberi nilai 5 pada variabel a
    b=9; → memberi nilai 5 pada variabel a
    cout<<"isi a = "<<a<<endl; → menampilkan isi a → 5
    cout<<"isi b = "<<b<<endl; → menampilkan isi b → 9
    luas(a,b); → memanggil fungsi luas
    cout<<"isi a = "<<a<<endl; → menampilkan isi a → 14
    cout<<"isi b = "<<b<<endl; → menampilkan isi a → 22
}
void luas(float x,float y)
{
    a=x+y; → nilai a diubah menjadi 14
    b=a+6; → nilai b diubah menjadi 22
    cout<<"isi a = "<<a<<endl; → menampilkan isi a → 14
    cout<<"isi b = "<<b<<endl; → menampilkan isi a → 22
}

```

### Variabel Statis

Variabel static merupakan variabel yang hanya dikenal dalam suatu fungsi dan nilai dari variabel ini tidak akan hilang saat keluar dari suatu fungsi. Jika suatu variabel dideklarasikan sebagai variabel statis maka:

1. Variabel hanya dapat diakses dalam fungsi di mana variabel tersebut dideklarasikan
2. Nilai variabel tidak akan hilang walaupun sudah keluar dari suatu fungsi

### Contoh

```
#include <iostream.h>
void hitung();

void main()
{ hitung();
cout<<"======"<<endl;
hitung();
}

void hitung()
{ static int a=7;
a++;
cout<<"isi a = "<<a<<endl;
}
```

### Penjelasan

1. Hitung() → memanggil fungsi hitung()
  - void hitung()
    - { static int a=7; → mendeklarasikan variabel static a dan diset 7
    - a++; → a=a+1 sehingga a terisi data 8
    - cout<<"isi a = "<<a<<endl; → tampilkan isi a → 8
2. kembali ke program utama
3. Hitung() → memanggil fungsi hitung() dipanggil kembali
  - void hitung()
    - { static int a=7; → karena sudah dideklarasikan pada pemanggilan fungsi pertama maka isi variabel tidak kembali 7 tetapi akan mengingat isi variabel sebelumnya. yaitu 8
    - a++; → a=a+1 sehingga a terisi data 9
    - cout<<"isi a = "<<a<<endl; → tampilkan isi a → 9
4. kembali ke program utama

```
#include <iostream.h>
void hitung();
int x;
void main()
{   x=9;
    cout<<"isi x = "<<x<<endl;
    hitung();
    cout<<"isi x = "<<x<<endl;
}
void hitung()
{   cout<<"isi x = "<<x<<endl;
    x=x+1;
}
```

```
isi x = 9
isi x = 9
isi x = 10
```

### Penjelasan

- void hitung()  
int x; → mendeklarasikan variabel x yang bersifat global
- void main()  
x=9; → mengisi variabel x dengan 9  
cout<<"isi x = "<<x<<endl → mencetak isi x → 9
- hitung() → memanggil fungsi hitung()
- void hitung()  
{ cout<<"isi x = "<<x<<endl; → mencetak isi x → 9  
x=x+1; → isi x ditambah 1, jadi x sekarang berisi 10
- setelah mengerjakan fungsi, maka program kembali ke program utama dan mengerjakan perintah di bawahnya  
cout<<"isi x = "<<x<<endl → mencetak isi x → 10

### Contoh

```
#include <iostream.h>
void hitung();
int x;
void main()
{   x=9;
    cout<<"isi x = "<<x<<endl;
    hitung();
    cout<<"isi x = "<<x<<endl;
}
void hitung()
{   int x=3;
    cout<<"isi x = "<<x<<endl;
    x=x+1;
}
```

(Inactive D:\FUNSI2.EXE)

```
isi x = 9
isi x = 3
isi x = 9
```

x sebagai variabel lokal di fungsi hitung

### Penjelasan

- void hitung()
- int x;                      → mendeklarasikan variabel x yang bersifat global
- void main()
- x=9;                      → mengisi variabel x dengan 9
- cout<<"isi x "<<x<<endl → mencetak isi x → 9
- hitung()                  → memanggil fungsi hitung()
- void hitung()
- { int x=3;              → x sebagai variabel lokal. Bukan global  
        cout<<"isi x = "<<x<<endl; → mencetak isi x → 3  
        x=x+1;                 → isi x ditambah 1, jadi x sekarang berisi 4
- setelah mengerjakan fungsi, maka program kembali ke program utama dan mengerjakan perintah di bawahnya  
cout<<"isi x "<<x<<endl → mencetak isi x → 9  
Kenapa 9, bukan 4 , hal ini karena x=4 bersifat lokal dan perubahan ini tidak berpengaruh pada variabel x yang ada di program utama.

### Contoh

```
#include <iostream>
void hitung();
void main()
{
    int x=9;
    cout<<"isi x = "<<x<<endl;
    hitung();
    cout<<"isi x = "<<x<<endl;
}
void hitung()
{
    int x=3;
    cout<<"isi x = "<<x<<endl;
    x=x+1;
}
```

(Inactive D:\FUNSI2.EXE)

isi x = 9  
isi x = 3  
isi x = 9

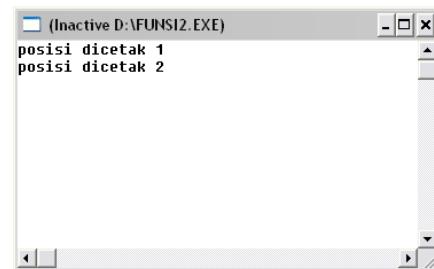
### Penjelasan

- 1 Memanggil fungsi hitung1 → program akan mengerjakan/ memanggil fungsi hitung1
- 2 Setelah selesai akan kembali ke pemanggil fungsi (fungsi main)

int x=9 → mendeklarasikan variabel x dan variabel x bersifat lokal di fungsi main()  
int x=3 → mendeklarasikan variabel x dan variabel x bersifat lokal di fungsi hitung()

### Contoh

```
#include <iostream.h>
void cetak1();
void cetak2();
void main()
{ cetak1();
  cetak2();
}
2 void cetak1()
{ cout<<"posisi dicetak 1 "<<endl;
}
4 void cetak2()
{ cout<<"posisi dicetak 2 "<<endl;
}
```

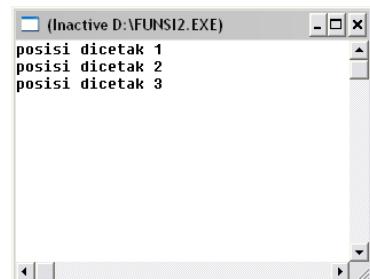


### Penjelasan

- 1 Memanggil fungsi cetak1() → program akan mengerjakan/ memanggil fungsi1
- 2 Setelah selesai akan kembali ke pemanggil fungsi (fungsi main)
- 3 Memanggil fungsi cetak2() → program akan mengerjakan/ memanggil fungsi2
- 4 Setelah semua perintah di fungsi cetak2 dikerjakan, program akan kembali ke pemanggil fungsi pemanggil (fungsi main)

### Contoh

```
#include <iostream.h>
void cetak1();
void cetak2();
void cetak3();
void main()
{ cetak1();
  cetak2();
}
2 void cetak1()
{ cout<<"posisi dicetak 1 "<<endl;
}
3 void cetak2()
{ cout<<"posisi dicetak 2 "<<endl;
}
6 void cetak3()
{ cout<<"posisi dicetak 3 "<<endl;
}
```



### Penjelasan

- 1 Memanggil fungsi cetak1() → program akan mengerjakan/ memanggil fungsi1
- 2 Setelah selesai akan kembali ke pemanggil fungsi (fungsi main)
- 3 Memanggil fungsi cetak2() → program akan mengerjakan/ memanggil fungsi2 dan di dalam fungsi cetak2 ternyata memanggil fungsi cetak3
- 4 Memanggil fungsi cetak3()
- 5 Setelah selesai akan kembali ke pemanggil fungsi (fungsi cetak2())
- 6 Setelah semua perintah di fungsi cetak3 dikerjakan, program akan kembali ke pemanggil fungsi main

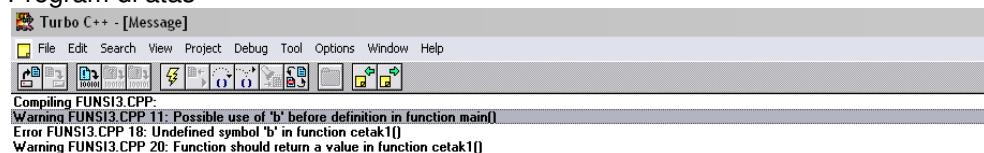
### Contoh

```
#include <iostream.h>
float cetak1();
int a;
void main()
{ a=4;
  int b;
  cout<<"posisi di fungsi main "<<endl;
  cout<<"isi a = " <<a<<endl;
  cout<<"isi b = " <<b<<endl;
}
float cetak1()
{ int c=8;
  a=40;
  cout<<"posisi dicetak 1 "<<endl;
  cout<<"isi a = " <<a<<endl;
  cout<<"isi b = " <<b<<endl;
  cout<<"isi c = " <<c<<endl;
}
```

Error, karena variabel b bersifat lokal di fungsi main tetapi dipanggil di tempat lain (fungsi cetak1)

### Penjelasan

- Program di atas



akan terjadi kesalahan

- Ada fungsi cetak1() tetapi fungsi ini belum digunakan

**Contoh**

```
#include <iostream.h>
void cetak1();
void cetak2();
void cetak3();
int a;
void main()
{ a=10;
    cout<<"isi a di posisi di fungsi main = "<<a<<endl;
    cetak1();
    cout<<"isi a di posisi di fungsi main = "<<a<<endl;
    cetak2();
    cout<<"isi a di posisi di fungsi main = "<<a<<endl;
    cetak2();
    cetak3();
}
void cetak1()
{ int a=20;
    cout<<"isi a di posisi dicetak 1 = "<<a<<endl;
}
void cetak2()
{ static int a=30;
    cout<<"isi a di posisi dicetak 2 = "<<a<<endl;
}
void cetak3()
{ a=80;
    cout<<"isi a di posisi dicetak 3 = "<<a<<endl;
}
```

### 8.6.Latihan

1. Program ini, bila dijalankan apa hasilnya

```
#include <iostream.h>
#include <math.h>
//prototipe dari fungsi
void pers(int x);

void main(void)
{
int b=5;
cout << "Nilai b = " << b << endl;
pers(b);
    cout << "Nilai b = " << b << endl;
}

void pers(int x)
{
int y,b=10;
cout << "Nilai b = " << b << endl;
y=pow(x,2)+6*x+8;
cout << "hasil y = " <<y << endl;
return;
}
```

- 2 Cari simpan baku dengan rumus

$$S = \sqrt{\sum \frac{(x_i - \bar{x})^2}{n}}$$

- S : simpan baku yang akan dihitung
- Xi : data ke i dari n buah data
- X̄ : nilai rata-rata dari keseluruhan data
- N : caca data

Kerjakan dengan rumus, minimal terdapat fungsi :

- Memasukan data-data
- Mencari-rata-rata
- Mencari rumus  $\sum (X_i - \bar{X})^2$
- Mencari S

3. Rumus korelasi pearson sebagai berikut

$$r = \frac{N \sum XY - (\sum X)(\sum Y)}{\sqrt{N \sum X^2 - (\sum X)^2} \sqrt{N \sum Y^2 - (\sum Y)^2}}$$

Tentukan fungsi-fungsi yang akan dibuat



9

# ARRAY



### Tujuan Instruksional

Setelah membaca bab ini, diharapkan pembaca memahami kegunaan array, dapat membedakan variabel biasa dengan variabel array serta dapat mengimplementasikan array dalam pemrograman

### Materi



Array dimensi satu



Array Berdimensi Dua

### 9.1. Array dimensi satu

Penggunaan variabel yang selama ini digunakan terdapat kelemahan yang mendasar. Kelemahan tersebut adalah variabel yang digunakan tidak bisa menyimpan lebih dari satu data. Variabel yang sering dipakai dalam pemrograman diatas sering disebut dengan variabel tunggal. Artinya variabel yang digunakan hanya dapat menyimpan 1 data yaitu data yang terakhir.

int a=10;

 10

a

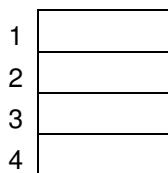
jika diganti a=20;

 20

a

isi a berubah menjadi 20 dan data sebelumnya sudah dihapus. Persoalan bagaimana jika menginginkan data-data sebelumnya juga disimpan. Solusinya gunakan array.

Array dapat digambarkan dalam bentuk larik dengan nama variabel satu tetapi mempunyai tempat yang berbeda-beda.



Variabel array → a

Dari ilustrasi di atas akan terdapat variabel dengan nama a dan dapat menyimpan data sebanyak. Proses penyimpanan dengan menggunakan nama a[1],a[2],a[3] dan a[4].

#### Deklarasi Array

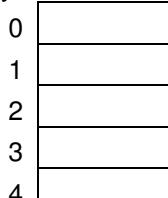
Variable array dideklarasikan dengan mencantumkan tipe dan nama variabel yang diikuti dengan banyaknya lokasi memori yang ingin dibuat.

**Tipe\_data nama\_variabel[banyak array]**

Pemberian nomer array dimulai dari 0

#### Contoh :

int x[5]; → artinya mendeklarasikan variabel a dengan banyak array sebanyak 5



x bertipe integer

Jadi, Array merupakan koleksi data dimana setiap elemen memakai nama dan tipe yang sama serta setiap elemen diakses dengan membedakan indeks array-nya.

### Inisialisasi variabel array

Inisialisasi atau memberikan nilai pada array hampir sama dengan variabel tunggal, hanya dalam memberikan nilai ini diperjelas dengan memberi nomor array.

```
x[0] = -45;
x[1] = 6;
x[2] = 0;
x[3] = 72;
x[4] = 1543;
```

|   |      |
|---|------|
| 0 | -45  |
| 1 | 6    |
| 2 | 0    |
| 3 | 72   |
| 4 | 1543 |

Cara lain untuk memberi nilai atau inisialisasi dapat dilakukan langsung dalam proses pendeklarasian variabel array

```
Int x[5] = {-45, 6, 0, 72, 1543 }
```

### Menampilkan isi variabel array

Cara menampilkan isi array dilakukan dengan menyebutkan nama array dan nomor array yang ditampilkan. Contoh

```
cout<<"isi array 4 "<<x[4];
```

#### contoh

```
#include <iostream.h>
main()
{ int x[5];
  x[0] = -45;
  x[1] = 6;
  x[2] = 0;
  x[3] = 72;
  x[4] = 1543;
  cout<<"isi array 0 "<<x[0]<<endl;
  cout<<"isi array 1 "<<x[1]<<endl;
  cout<<"isi array 2 "<<x[2]<<endl;
  cout<<"isi array 3 "<<x[3]<<endl;
  cout<<"isi array 4 "<<x[4]<<endl;
}
```

Deklarasi variabel array

Mengisi variabel array

Menampilkan variabel array

```
(Inactive D:\WONAME03.EXE)
isi array 0 -45
isi array 1 6
isi array 2 0
isi array 3 72
isi array 4 1543
```

#### Penjelasan :

- Mendeklarasikan variabel array x sebanyak 5 (dihitung mulai 0)
- Mengisi variabel array satu persatu
- Menampilkan isi array satu persatu

**Catatan:**

Untuk mengisi variabel bisa menggunakan perintah masukan/ cin

Untuk menampilkan semua data dapat menggunakan proses perulangan

**Contoh**

```
#include <iostream.h>
main()
{ int x[5] = {-45, 6, 0, 72, 1543 };
  cout<<"isi variabel x[0] = "<<x[0]<<endl;
  cout<<"isi variabel x[1] = "<<x[1]<<endl;
  cout<<"isi variabel x[2] = "<<x[2]<<endl;
  cout<<"isi variabel x[3] = "<<x[3]<<endl;
  cout<<"isi variabel x[4] = "<<x[4]<<endl;
}
```

**Penjelasan :**

- int x[5]={-45,6,0,72,1543}  
→ Mendeklarasikan variabel array x dan sekaligus memberi nilai awal pada array
- Menampilkan isi array satu persatu

**Contoh**

```
#include <iostream.h>
main()
{ int i, x[5] = {-45, 6, 0, 72, 1543 };
  for (i=0;i<=4;i++)
    cout<<"isi variabel x "<<i<<" = "<<x[i]<<endl;
}
```

**Penjelasan :**

- Mendeklarasikan variabel array x dan sekaligus memberi nilai awal pada array
- Menampilkan isi array dengan menggunakan array

### Contoh

```
#include <iostream.h>
main()
{ int i,x[5];
  x[0] = -45;
  x[1] = 6;
  x[2] = 0;
  cout<<"masukan isi array ke 3 ";
  cin>>x[3];
  cout<<"masukan isi array ke 4 ";
  cin>>x[4];
  for (i=0;i<=4;i++)
  cout<<"isi array "<<i<<" = "<<x[i]<<endl;
}
```



```
masukan isi array ke 3 40
masukan isi array ke 4 25
isi array 0 = -45
isi array 1 = 6
isi array 2 = 0
isi array 3 = 40
isi array 4 = 25
```

### Penjelasan :

- Mendeklarasikan variabel array x sebanyak 5 (dihitung mulai 0)
- Mengisi variabel array x(0) dengan -45
- Mengisi variabel array x(1) dengan 6
- Mengisi variabel array x(2) dengan 0
- Mengisi variabel array x(3) dengan meminta data dari masukan keyboard
- Mengisi variabel array x(4) dengan meminta data dari masukan keyboard
- Menampilkan semua isi array dengan menggunakan perulangan (for)

### Studi kasus

Hitung jumlah dan rata-rata dari suatu data yang dimasukan dari keyboard. Banyak data sebanyak N dan juga dimasukan dari keyboard

### Algoritma :

- Masukan banyak data yang akan dihitung, misal N data
- Ulang i dari 0 sampai N-1
  - Masukan data ke i+1, misal di simpan di arraya data[i]
  - Lakukan proses penjumlahan misal jum=jum+1
- Hitung rata-rata
- Tampilkan jumlah
- Tampilkan rata-rata

**Contoh**

```
#include <iostream.h>
main()
{ int n,i,data[10];
  float jum,rata;
  cout<<"Masukan banyak data = ";
  cin>>n;
  jum=0;
  rata=0;
  for(i=0;i<=n-1;i++)
  { cout<<"masukan data ke "<<i+1<<" = ";
    cin>>data[i];
    jum=jum+data[i];
  }
  rata=jum/n;
  cout<<"hasil penjumlahan semua data = "<<jum<<endl;
  cout<<"hasil rata-rata data = "<<rata<<endl;
}
```

Masukan banyak data = 5  
masukan data ke 1 = 20  
masukan data ke 2 = 15  
masukan data ke 3 = 10  
masukan data ke 4 = 12  
masukan data ke 5 = 25  
hasil penjumlahan semua data = 82  
hasil rata-rata data = 16.4

**Penjelasan**

- int n,i,data[10];  
float jum,rata;
  - memesan variabel-variabel yang akan digunakan
- cout<<"Masukan banyak data = ";  
cin>>n;
  - meminta banyak data yang akan dihitung dan dicari rata-ratanya
- jum=0;  
rata=0;
  - memberi nilai awal 0 untuk variabel jum dan rata
- for(i=0;i<=n-1;i++)  
{ cout<<"masukan data ke "<<i+1<<" = ";  
cin>>data[i];  
jum=jum+data[i];  
}
  - melakukan proses pengulangan sebanyak n data
    - Proses pengulangan dari 0 sampai n-1,
      - n-1 , hal ini karena proses penyimpanan mulai dari 0, sehingga data dari 0 sampai n-1 bukan 0 sampai n
      - cout<<"masukan data ke "<<i+1<<" = "; i+1, hal ini agar tampilan di monitor tetap menampilkan angka 1 sampai n.
      - jadi walaupun proses penyimpanan dimulai dari 0, tetapi untuk menampilkan informasi tetap mulai dari 1
      - dalam proses pengulangan juga dilakukan proses perhitungan yang lain, misal menghitung jumlah, jum=jum+data[i]

- `rata=jum/n;`
  - menghitung rata-rata
- `cout<<"hasil penjumlahan semua data = "<<jum<<endl;`  
`cout<<"hasil rata-rata data = "<<rata<<endl;`
  - memapilkkan jumlah dan rata

### contoh

```
#include <iostream.h>
void main ()
{
int bln,bulan[12]={31,28,31,30,31,30,31,31,31,30,31,30,31};
cout<<"Masukkan Bulan : ";
cin>>bln;
switch(bln)
{ case 1:
    cout<<"jumlah hari = "<<bulan[0];
 case 2:
    cout<<"jumlah hari = "<<bulan[1];
 case 3:
    cout<<"jumlah hari = "<<bulan[2];
 case 4:
    cout<<"jumlah hari = "<<bulan[3];
}
}
```



### Penjelasan

- `bulan[12]={31,28,31,30,31,30,31,31,30,31,30,31};`  
Mendeklarasikan variabel array dengan nama bulan sebanyak 12 dan langsung diberi nilai awal
- `switch(bln)`  
`{ case 4:`  
 `cout<<"jumlah hari = "<<bulan[3];`  
melakukan proses seleksi dengan case dan jika dimasukan 4, maka akan seleksi case 4 dan menampilkan `cout<<"jumlah hari = "<<bulan[3]` → menampilkan isi array bulan yang 3 (dari sisi array dihitung mulai 0). Jadi `bulan[3]` menampilkan isi bulan April

## 9.2. Array Berdimensi Dua

Pemanfaatan array tidak hanya dapat digunakan untuk menyimpan data dalam bentuk satu dimensi tetapi juga dapat digunakan untuk menyimpan data dalam bentuk 2 dimensi. Misal ada data dalam bentuk representasi sebagai berikut :

| Nomhs | Tugas 2 | Tugas 3 |
|-------|---------|---------|
| 2001  | 80      | 80      |
| 2002  | 75      | 80      |
| 2003  | 90      | 75      |
| 2004  | 65      | 60      |

### Deklarasi Array

Variable array dimensi dua dideklarasikan dengan mencantumkan tipe dan nama variabel yang diikuti dengan banyaknya lokasi memori yang ingin dibuat.

**Tipe\_data nama\_variabel[subscript\_baris][ subscript\_kolom ]**

Pemberian nomor array dimulai dari baris 0 dan kolom 0, sehingga dalam contoh tabel di atas dapat dideklarasikan :

```
int data[4][3]
pada pendefinisian diatas :
    4 menyatakan jumlah baris
    3 menyatakan jumlah kolom
```

### Inisialisasi variabel array

Inisialisasi atau memberikan nilai pada array hampir sama dengan variabel tunggal, hanya dalam memberikan nilai ini diperjelas dengan memberi nomor array.

```
x[0][0] = -45;
x[0][1] = 6;
x[1][0] = 0;
x[1][1] = 72;
x[2][0] = 4;
x[2][1] = 34;
```

|   | 0   | 1  |
|---|-----|----|
| 0 | -46 | 6  |
| 1 | 0   | 73 |
| 2 | 4   | 34 |

Cara lain untuk memberi nilai atau inisialisasi dapat dilakukan langsung dalam proses pendeklarasian variabel array

```
int x[3][2] = { {-45, 6},
                { 0, 72 },
                { 4, 34 }, };
```

### Menampilkan isi variabel array

Cara menampilkan isi array dilakukan dengan menyebutkan nama array dan nomor array yang ditampilkan. Contoh

```
cout<<"isi array 4 "<<x[1][1];
```

### Contoh

```
#include <iostream.h>
main()
{ int data[4][3];
  data[0][0]= 2001;
  data[0][1]= 80;
  data[0][2]= 80;
  data[1][0]= 2002;
  data[1][1]= 75;
  data[1][2]= 80;
  data[2][0]= 2003;
  data[2][1]= 90;
  data[2][2]= 75;
  data[3][0]= 2004;
  data[3][1]= 65;
  data[3][2]= 60;
}
```

### Penjelasan:

- int data[4][3];
   
⇒ memesan deklarasi variabel array 2 dimensi sebanyak 4 baris dan 3 kolom
- mengisi variabel array 2 dimensi
   
data[2][0]= 2003; → mengisi data baris 2 kolom 0 dengan 2003
   
data[2][1]= 90; → mengisi data baris 2 kolom 1 dengan 90
   
data[2][2]= 75; → mengisi data baris 2 kolom 2 dengan 75

### Contoh

```
#include <iostream.h>
main()
{   int x[2][2]= {
    { 1,2 },
    { 3,4 },
  };
  int huruf[8][8] =
  {
    {1,2,3,4,5,6,7,8},
    {1,1,0,0,0,1,0,0},
    {1,1,0,0,0,1,0,0},
    {1,1,1,1,1,1,1,0},
    {1,1,0,0,0,0,1,0},
    {1,1,0,0,0,0,1,0},
    {1,1,0,0,0,0,1,0},
    {1,1,1,1,1,1,1,0},
  };
}
```

**Penjelasan :**

- int x[2][2] = { { 1,2},  
                    {3,4},  
                    };  
        ⇒ mendeklarasikan variabel x sebanyak 2 baris dan 2 kolom serta memberikan inisialisasi variabel array tersebut
- int huruf[8][8] → mendeklarasikan array 2 dimensi dan memberi nilai awal

**contoh**

```
#include <iostream.h>
main()
{ int data[4][3],i,j;
  data[0][0]= 2001;
  data[0][1]= 80;
  data[0][2]= 80;
  data[1][0]= 2002;
  data[1][1]= 75;
  data[1][2]= 80;
  data[2][0]= 2003;
  data[2][1]= 90;
  data[2][2]= 75;
  data[3][0]= 2004;
  data[3][1]= 65;
  data[3][2]= 60;

  for(i=0;i<=3;i++)
  {
    for(j=0;j<=2;j++)
      cout<<data[i][j]<<" | ";
    cout<<endl;
  }
}
```

**Penjelasan:**

- int data[4][3];  
        ⇒ memesan deklarasi variabel array 2 dimensi sebanyak 4 baris dan 3 kolom
- mengisi variabel array 2 dimensi
  - data[2][0]= 2003; → mengisi data baris 2 kolom 0 dengan 2003
  - data[2][1]= 90;   → mengisi data baris 2 kolom 1 dengan 90
  - data[2][2]= 75;   → mengisi data baris 2 kolom 2 dengan 75
- for(i=0;i<=3;i++)
 { for(j=0;j<=2;j++)
 cout<<data[i][j]<<" | ";   → menampilkan isi array 2 dimensi
 cout<<endl;
 }

**Contoh**

```
#include <iostream.h>
main()
{
    int i,j;
    int x[2][2] = {
        { 1, 2 },
        { 3, 4 },
    };

    for (i=0;i<=1;i++)
    {
        for (j=0;j<=1;j++)
            cout<<x[i][j];
        cout<<endl;
    }
}
```

**Penjelasan:**

```
for (i=0;i<=1;i++)
{
    for (j=0;j<=1;j++)
        cout<<x[i][j];
    cout<<endl;
}
⇒ untuk menampilkan data variabel 2 dimensi harus menggunakan for dalam for
```

**Contoh :****Penjumlahan 2 matrik****Logika**

$$\left\{ \begin{array}{ccc} 2 & 3 & 4 \\ 4 & 2 & 1 \end{array} \right\} + \left\{ \begin{array}{ccc} 3 & 1 & 3 \\ 2 & 2 & 3 \end{array} \right\}$$

Dalam menjumlah suatu matrik harus ada syarat yang harus diperhatikan, 2 matrik dapat dilakukan operasi penjumlahan bila ke dua matrik tersebut mempunyai baris dan kolom yang sama. Untuk mempermudah proses komputasinya diagram alir yang dibuat **harus menggunakan variabel larik**. Penggunaan variabel larik ini adalah variabel laarik yang menggunakan 2 dimensi.

**A**      A(1,1) = 2    A(1,2) = 3    A(1,3) = 4  
           A(2,1) = 4    A(2,2) = 2    A(2,3) = 1

**B**      B(1,1) = 3    B(1,2) = 1    B(1,3) = 3  
           B(2,1) = 2    B(2,2) = 2    B(2,3) = 3

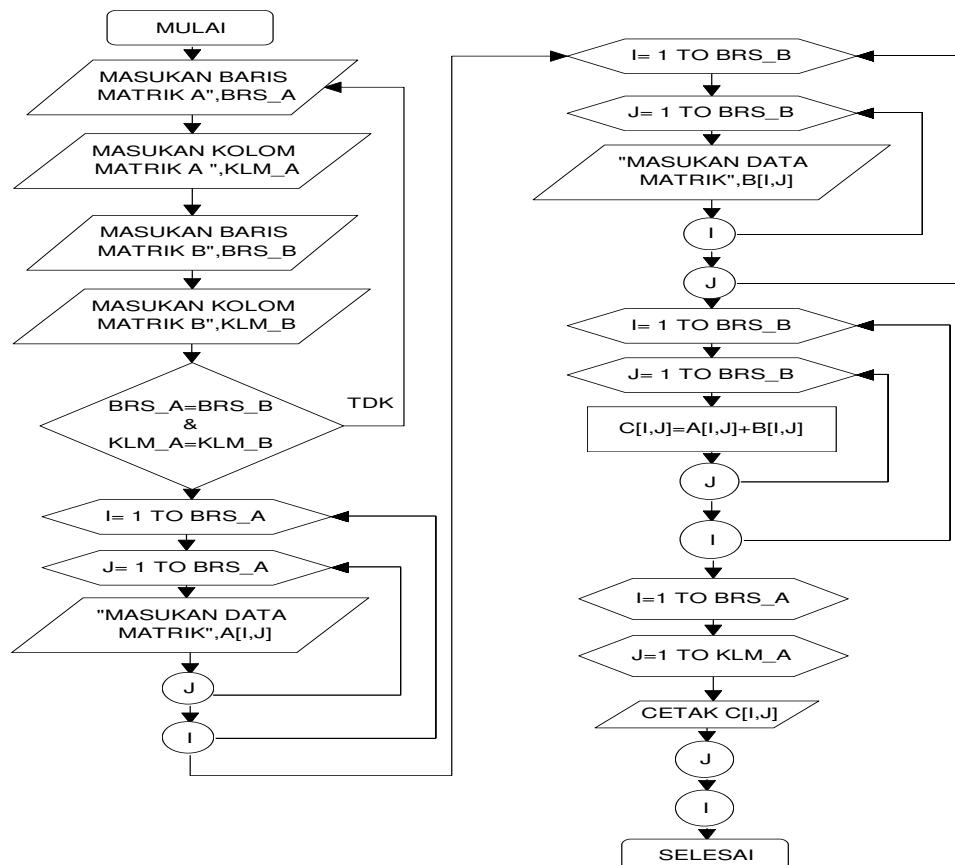
### Proses penjumlahannya

$$\begin{array}{llll}
 A + B & \quad \quad \quad & \\
 \begin{array}{l} A(1,1) + B(1,1) \\ A(2,1) + B(2,1) \end{array} & \begin{array}{l} A(1,2)+B(1,2) \\ A(2,2)+B(2,2) \end{array} & \begin{array}{l} A(1,3)+B(1,3) \\ A(2,3)+B(2,3) \end{array}
 \end{array}$$

### Algoritma :

1. Masukkan ukuran matriks A  
Ukuran baris matrik A → BRS\_A  
Ukuran kolom matrik A → KLM\_A
2. Masukkan ukuran matriks B  
Ukuran baris matrik B → BRS\_B  
Ukuran kolom matrik B → KLM\_B
3. Lakukan pengecekan apakah kedua ukuram matriknya sama  
Apakah BRS\_A = BRS\_B dan apakah KLM\_A = KLM\_B  
Jika tidak sama ulangan langkah 1
3. Masukkan isi data dari matrik A
4. Masukkan isi data dari matrik B
5. Ulang I dari 1 to BRS\_A  
Ulang J dari 1 to KLM\_A  
 $C(I,J)=A(I,J)+B(I,J)$
6. Cetak matriks C
7. Selesai

### Flowchart



Berikut adalah program sederhana untuk menghitung 2 matriks, kembangkan program di bawah ini sesuai flowchart di atas.

```
#include <iostream.h>
main()
{
    int a[3][3]={{ 2,3,5}, { 4,2,4},{ 5,3,5} };
    int b[3][3]={{ 1,2,7}, { 2,2,6},{ 4,3,7} };
    int c[3][3],i,j;

    for(i=0;i<=2;i++)
        for(j=0;j<=2;j++)
            c[i][j] = a[i][j] + b[i][j];

    for(i=0;i<=2;i++)
    {   cout<<endl;
        for(j=0;j<=2;j++)
            cout<<c[i][j]<<" ";
    }
}
```



### Penjelasan

```
for(i=0;i<=2;i++)
    for(j=0;j<=2;j++)
        c[i][j] = a[i][j] + b[i][j];
    ➔ Digunakan untuk proses penjumlahan matriks c=a+b
for(i=0;i<=2;i++)           ➔ untuk berganti baris
{   cout<<endl;
    for(j=0;j<=2;j++)
        cout<<c[i][j]<<" ";  ➔ mencetak isi matriks, tanda << " " untuk memberi jarak
                                dalam mencetak isi matriks
}
```

Jika `cout<<endl;` dihilangkan, maka hasilnya akan berderet untuk semua data matriks



Jika `cout<<c[i][j]<<" ";` perintah << " " dihilangkan, maka hasilnya akan berderet untuk semua data matriks



### 9.3. Melewatkkan Array sebagai parameter dalam suatu Fungsi

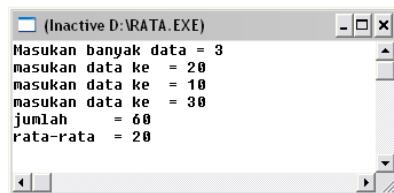
Adakalanya array diberikan kedalam fungsi sebagai parameter. Sebagai misal ada data dalam bentuk array di fungsi main (program utama) dan ingin melewatkkan data-data array tersebut dalam suatu fungsi lain.

```
#include <iostream.h>
void cetak();
main()
{
    int i,j;
    int x[2][2] = { { 1,2 },
                    { 3,4 },
                    };
    for (i=0;i<=1;i++)
    {
        for (j=0;j<=1;j++)
            cout<<x[i][j];
        cout<<endl;
    }
}
void cetak()
```

Dari program di atas, nilai data yang disimpan pada variabel array x dapat dilewatkan di fungsi cetak. Agar dapat melewatkkan nilai array di suatu fungsi, nilai array juga bisa dilewatkan dalam suatu parameter. Untuk menggunakan array sebagai parameter maka yang harus dilakukan saat pendeklarasian fungsi adalah spesifikasi tipe array pada argumen, Contoh :

```
#include <iostream.h>
void rata1(int x[],int n1);
main()
{
    int n,i,data[10];
    cout<<"Masukan banyak data = ";
    cin>>n;
    for(i=0;i<=n-1;i++)
    {
        cout<<"masukan data ke = ";
        cin>>data[i];
    }
    rata1(data,n);
}
void rata1(int x[],int n1)
{
    int i,jum=0,rata=0;
    for(i=0;i<=n1-1;i++)
        jum=jum+x[i];

    rata=jum/n1;
    cout<<"jumlah = "<<jum<<endl;
    cout<<"rata-rata = "<<rata<<endl;
}
```



### Penjelasan

- void rata1(int x[],int n1);  
instruksi void rata1(int x[],int n1); menjelaskan bahwa semua array bertipe int, (x[]) berapa pun panjangnya. Akan menjadi nilai parameter yang akan dilewatkan ke suatu fungsi.
- rata1(data,n); → memanggil fungsi dengan melewatkannya parameter yang disimpan di variabel array data , dalam melewatkannya cukup ditulis nama variabelnya tidak perlu menggunakan [].

### Contoh

Pengurutan data menjadi sesuatu yang sangat penting dalam proses pengolahan data. Dalam pengurutan beberapa perintah yang digunakan diantaranya :

- Array : digunakan untuk menyimpan data-data yang akan diurutkan
- If : digunakan untuk melakukan proses pembanding dalam menentukan data terbesar
- For : digunakan untuk melakukan proses pengulangan dalam mencari data terkecil-terbesar

### Algoritma Pemilihan (*Selection*)

Pada dasarnya algoritma Pemilihan untuk pengurutan *ascending* adalah memilih data terkecil dari daftar, kemudian tukarkan tempat data tersebut dengan data pada posisi pertama dari daftar, kemudian pemilihan diulang, tetapi kali ini dimulai dari posisi kedua, dan bila ditemukan data ditukarkan ke posisi kedua, demikian seterusnya dilakukan pemilihan untuk posisi ketiga, keempat, ke-n hingga seluruh daftar diselesaikan.

### Algoritma Metode Seleksi

0. Baca vector (array)
1. Kerjakan langkah 2 sampai 4  $I \rightarrow$  dari 1 sampai  $N-1$
2. Tentukan LOK  $\rightarrow I$   
Kerjakan langkah 3 untuk  $J \rightarrow I+1$  sampai  $N$
3. Mencari data terkecil  
Test apakah  $A[LOK] > A[J]$   
Jika YA, tentukan  $LOK=J$
4. Tukarkan nilai  $A[LOK]$  dengan  $A[I]$
5. Selesai

### Logika Metode Seleksi

Untuk lebih jelasnya akan diterangkan logika dari algoritma ini

DATA AWAL adalah

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 23 | 45 | 12 | 24 | 56 | 34 |
|----|----|----|----|----|----|

|                 |                           |                                                                                                                                                                                                                                          |
|-----------------|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pengulangan I=1 |                           | Posisi data = 23,45,12,24,56,34                                                                                                                                                                                                          |
|                 | Pengulangan J=I +1<br>J=2 | Cek apakah data[i]>data[j] → apakah 23>45<br>⇒ Tidak , lanjutkan ke pengulangan J selanjutnya → J =3                                                                                                                                     |
|                 | Pengulangan J=3           | Cek apakah data[i]>data[j] → apakah 23>12<br>Ya : tukarkan dua data tersebut<br>temp=data[i] temp=23<br>data[i]=data[j] data[1]=12<br>data[j]=temp data[3]=23<br>jadi array menjadi<br><del>23,45,12,24,56,34</del><br>selanjutnya → j=4 |
|                 | Pengulangan J=4           | Cek apakah data[i]>data[j] → apakah 12>24<br>Tidak , lanjutkan ke pengulangan J selanjutnya → j=5                                                                                                                                        |
|                 | Pengulangan J=5           | Cek apakah data[i]>data[j] → apakah 12>56<br>Tidak , lanjutkan ke pengulangan J selanjutnya → J=6                                                                                                                                        |
|                 | Pengulangan J=6           | Cek apakah data[i]>data[j] → apakah 12>34<br>Tidak , lanjutkan ke pengulangan J selanjutnya → J=7<br>Untuk J sudah habis, lanjutkan ke I, I=2                                                                                            |
| Pengulangan I=2 |                           | Posisi data = 12,45,23,24,56,34                                                                                                                                                                                                          |
|                 | Pengulangan J=I+1<br>J=3  | Cek apakah data[i]>data[j] → apakah 45>23<br>Ya : tukarkan dua data tersebut<br>temp=data[i] temp=45<br>data[i]=data[j] data[2]=23<br>data[j]=temp data[3]=45<br>jadi array menjadi<br><del>23,45,12,24,56,34</del><br>selanjutnya → J=4 |
|                 | Pengulangan J=4           | Cek apakah data[i]>data[j] → apakah 23>24<br>Tidak , lanjutkan ke pengulangan J selanjutnya → J=5                                                                                                                                        |
|                 | Pengulangan J=5           | Cek apakah data[i]>data[j] → apakah 23>56<br>Tidak , lanjutkan ke pengulangan J selanjutnya → J=6                                                                                                                                        |
|                 | Pengulangan J=6           | Cek apakah data[i]>data[j] → apakah 23>34<br>Tidak , lanjutkan ke pengulangan J selanjutnya → J=7                                                                                                                                        |

|                                           |                                    |                                                                                                                                                                                                                                                                                        |
|-------------------------------------------|------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                           |                                    | Untuk J sudah habis, lanjutkan ke I, I=3                                                                                                                                                                                                                                               |
| Pengulangan I=3                           |                                    | Posisi data =12,23,45,24,56,34                                                                                                                                                                                                                                                         |
|                                           | Pengulangan J=I+1<br>J=4           | Cek apakah data[i]>data[j] → apakah 45>24<br>Ya : tukarkan dua data tersebut<br>temp=data[i] temp=45<br>data[i]=data[j] data[3]=24<br>data[j]=temp data[4]=45<br>jadi array menjadi<br>12,23, <del>24,45</del> ,56,34<br>selanjutnya → J=5                                             |
|                                           | Pengulangan J=5                    | Cek apakah data[i]>data[j] → apakah 24>56<br>Tidak , lanjutkan ke pengulangan J<br>selanjutnya → J=6                                                                                                                                                                                   |
|                                           | Pengulangan J=6                    | Cek apakah data[i]>data[j] → apakah 24>34<br>Tidak , lanjutkan ke pengulangan J<br>selanjutnya → J=7<br>Untuk J sudah habis, lanjutkan ke I, I=4                                                                                                                                       |
| Pengulangan I=4                           |                                    | Posisi data =12,23,24,45,56,34                                                                                                                                                                                                                                                         |
|                                           | Pengulangan J=I+1<br>J=5           | Cek apakah data[i]>data[j] → apakah 45>56<br>Tidak , lanjutkan ke pengulangan J<br>selanjutnya → J=6                                                                                                                                                                                   |
|                                           | Pengulangan J=6                    | Cek apakah data[i]>data[j] → apakah 45>34<br>Ya : tukarkan dua data tersebut<br>temp=data[i] temp=45<br>data[i]=data[j] data[4]=34<br>data[j]=temp data[6]=45<br>jadi array menjadi<br>12,23,24, <del>34,56</del> ,45<br>selanjutnya → J=7<br>Untuk J sudah habis, lanjutkan ke I, I=5 |
| Pengulangan I=5                           |                                    | Posisi data =12,23,24,34,56,45                                                                                                                                                                                                                                                         |
|                                           | Pengulangan J=I+1<br>J=6           | Cek apakah data[i]>data[j] → apakah 56>45<br>Ya : tukarkan dua data tersebut<br>temp=data[i] temp=56<br>data[i]=data[j] data[4]=45<br>data[j]=temp data[6]=56<br>jadi array menjadi<br>12,23,24,34, <del>45,56</del><br>selanjutnya → J=7<br>Untuk J sudah habis, lanjutkan ke I, I=6  |
| Pengulangan I=6<br>Untuk I sudah<br>habis | Sehingga posisi terakhir<br>adalah | <b>12,23,24,34,45,56</b>                                                                                                                                                                                                                                                               |

```
#include <iostream.h>
main()
{ int i,j,temp;
  int data[7]={23,45,12,24,56,34};
  cout<<"Data awal "<<endl;
  for(i=0;i<=5;i++)
    cout<<data[i]<<" ";
  for(i=0;i<=5;i++)
    for(j=i+1;j<=5;j++)
      if (data[i]>data[j])
      { temp=data[i];
        data[i]=data[j];
        data[j]=temp;
      }
  cout<<"\nData setelah proses pengurutan "<<endl;
  for(i=0;i<=5;i++)
    cout<<data[i]<<" ";
}
```

### Penjelasan

```
for(i=0;i<=5;i++)
  for(j=i+1;j<=5;j++)
    ⇒ Proses pengulangan berkalang
```

```
    if (data[i]>data[j])
    { temp=data[i];
      data[i]=data[j];
      data[j]=temp;
    }
```

⇒ Digunakan untuk melakukan proses pengecekan, jika nilai `data[i]>data[j]` maka akan dilakukan proses penukaran



#### 9.4.Latihan

Program pendek ini, apa hasilnya bila dijalankan

1. void main(void){  
    int a[5];  
    a[0]=56;  
    a[1]=34;  
    a[2]=16;  
    a[3]=24;  
    a[4]=36;  
    cout << "isi a[0] " <<a[0]<<endl;  
    cout << "isi a[1] " <<a[1]<<endl;  
    cout << "isi a[2] " <<a[2]<<endl;  
    cout << "isi a[3] " <<a[3]<<endl;  
    cout << "isi a[4] " <<a[4]<<endl;  
}
2. #include <iostream.h>  
void main(void){  
{  
    int a[2][3];  
    a[0][0]=1;  
    a[0][1]=2;  
    a[0][2]=3;  
    a[1][0]=4;  
    a[1][1]=5;  
    a[1][2]=6;  
    cout << "a[0,0] = " <<a[0][0]<<endl;  
    cout << "a[0,1] = " <<a[0][1]<<endl;  
    cout << "a[0,2] = " <<a[0][2]<<endl;  
    cout << "a[1,0] = " <<a[1][0]<<endl;  
    cout << "a[1,1] = " <<a[1][1]<<endl;  
    cout << "a[1,2] = " <<a[1][2]<<endl;  
}
4. Ganti program di pengurutan data atas dengan menggunakan data input
5. Ganti program di atas dengan fungsi terdapat fungsi pemasukan\_data, proses sorting serta proses menampilkan hasil akhir

10

## STRUCTURE



## Tujuan Instruksional

Setelah membaca bab ini, diharapkan pembaca kegunaan structure serta megimplementasikan dalam pemrograman

## Materi



Structure



Structure dengan Array

### 10.1. Structure

Struktur data merupakan kumpulan berbagai tipe data yang memiliki ukuran yang berbeda di kelompokan dalam satu deklarasi unik. Struktur data sangat cocok digunakan untuk merepresentasikan data dalam bentuk tabel, misal

| Nama      | Nilai1 | Nilai2 | rata |
|-----------|--------|--------|------|
| Maheswari | 95     | 90     | 92.5 |
| Aquila    | 75     | 80     | 77.5 |

Representasi tabel diatas, jika diselesaikan dengan array dimensi 2 tentunya tidak cocok. Hal ini karena tipe data dari masing-masing kolom berbeda sedangkan array digunakan jika semua kolom mempunyai tipe data yang sama. Tipe data dari masing-masing kolom adalah

```

Nama :      :: String
Nilai1 :    : integer
Nilai2 :    : integer
rata   :    float

```

Bentuk pendeklarasian struktur adalah :

```

struct model_name {
    type1 element1;
    type2 element2;
    type3 element3;
    .
}
} object_name;

```

dimana *model\_name* adalah nama untuk model tipe strukturnya dan parameter optional *object\_name* merupakan identifier yang valid untuk objek struktur. Diantara kurung kurawal {} berupa tipe dan sub-identifier yang mengacu ke elemen pembentuk struktur. Jika pendefinisan struktur menyertakan parameter *model\_name* (optional), maka parameter tersebut akan menjadi nama tipe yang valid ekuivalen dengan struktur. Contoh :

```

#include <iostream.h>
main()
{ struct data_nilai {
    char nama[30];
    int nilai1;
    int nilai2;
    float rata;
};
    data_nilai matematika;
}

```

Mendefinisikan struktur

Mendeklarasikan variabel  
matematika yang bertipe struktur  
data\_nilai

Sangat penting untuk membedakan antara structure **model**, dan structure **object**. *model* adalah *type*, dan *object* adalah *variable*. Kita dapat membuat banyak *objects* (variables) dari satu *model* (type).

```
#include <iostream.h>
#include <string.h>
main()
{ struct data_nilai {
            char nama [30];
            int    nilai1;
            int    nilai2;
            float rata;
        };
        float rata1;
        data_nilai matematika;
        strcpy (matematika.nama, "Maheswari");
        matematika.nilai1=95;
        matematika.nilai2=90;
        rata1=( matematika.nilai1+matematika.nilai2)/2.0;
        matematika.rata=rata1;
        cout<<" Nama      = "<<matematika.nama<<endl;
        cout<<" Nilai 1   = "<<matematika.nilai1<<endl;
        cout<<" nama  2   = "<<matematika.nilai2<<endl;
        cout<<" rata-rata = "<<matematika.rata<<endl;
    }
}
```

### Penjelasan

- strcpy (matematika.nama, "Maheswari");
  - Mengisi elemen nama milik objek matematika dengan data Maheswari
- matematika.nilai1=95;
- matematika.nilai2=90;
  - Mengisi elemen nilai 1 dan nilai2 milik objek matematika dengan data
- rata1=( matematika.nilai1+matematika.nilai2)/2.0;
  - Menghitung nilai rata-rata
- matematika.rata=rata1;
  - Mengisi elemen nilai 1 dan nilai2 milik objek matematika dengan data
- cout<<" Nama = "<<matematika.nama<<endl;
cout<<" Nilai 1 = "<<matematika.nilai1<<endl;
cout<<" nama 2 = "<<matematika.nilai2<<endl;
cout<<" rata-rata = "<<matematika.rata<<endl;
  - menampilkan semua elemen milik objek matematika

### catatan :

Suatu struktur yang sudah didefinisikan dapat digunakan untuk menciptakan objek-objek yang lain, Jadi untuk membuat tabel yang digunakan untuk menyimpan nilai FISIKA tidak perlu membuat struktur baru tetapi cukup membuat objek baru yang berisi struktur data\_nilai

### Contoh

```
#include <iostream.h>
#include <string.h>
main()
{ struct data_nilai {
    char nama [30];
    int nilai1;
    int nilai2;
    float rata;
};

float rata1;
data_nilai matematika;
data_nilai fisika;
strcpy (fisika.nama, "Anasya");
strcpy (matematika.nama, "Maheswari");

}
}

Mendefinisikan 2 objek dengan struktur yang sama
Mengisi data untuk masing-masing objek
```

### Penjelasan

- data\_nilai matematika;  
data\_nilai fisika;  
 ⇒ memesan 2 objek dengan menggunakan struktur yang sama, 2 objek tersebut adalah matematika dan fisika
- strcpy (fisika.nama, "Anasya");  
strcpy (matematika.nama, "Maheswari");  
 ⇒ mengisi masing-masing objek dengan data masing-masing

### 10.2. Structure dengan Array

Penggunaan struktur di atas masing ada kelemahan, yaitu objek-objek di atas hanya dapat menyimpan 1 data saja. Agar objek dapat menyimpan lebih dari satu data, perlu digabungkan dengan array satu dimensi.

```
#include <iostream.h>
#include <string.h>
main()
{ struct data_nilai {
    char nama [30];
    int nilai1;
    int nilai2;
    float rata;
};

float rata1;
data_nilai matematika;
strcpy (matematika.nama, "Maheswari");
strcpy (matematika.nama, "Aquila");
cout<<" Nama = "<<matematika.nama<<endl;
cout<<" Nama = "<<matematika.nama<<endl;
}
```



### Penjelasan

- `strcpy (matematika.nama, "Maheswari");`  
`strcpy (matematika.nama, "Aquila");`
  - ⇒ mengisi data nama milik objek matematika dengan data maheswara dan aquila. Tetapi karena bersifat variabel tunggal maka yang tersimpan dalam elemen nama adalah data yang terakhir
  - `cout<<" Nama = "<<matematika.nama<<endl;`
- `cout<<" Nama = "<<matematika.nama<<endl;`
  - ⇒ menampilkan data elemen nama, tetapi data yang ditampilkan adalah data yang terakhir, sehingga data maheswara tidak ditampilkan

### contoh

```
#include <iostream.h>
#include <string.h>
main()
{ struct data_nilai {
    char nama [30];
    int nilai1;
    int nilai2;
    float rata;
};

float rata1;
data_nilai matematika[5];
strcpy (matematika[0].nama, "Maheswari");
strcpy (matematika[1].nama, "Aquila");
cout<<" Nama = "<<matematika[0].nama<<endl;
cout<<" Nama = "<<matematika[1].nama<<endl;
}
```



### Penjelasan

- `data_nilai matematika[5];`
  - ⇒ mendeklarasikan objek matematika dengan array sebanyak 5. Objek matematika menggunakan struktur data `data_nilai`.
- `strcpy (matematika[0].nama, "Maheswari");`  
`strcpy (matematika[1].nama, "Aquila");`
  - ⇒ mengisi objek matematika[0] dengan data maheswara
  - ⇒ mengisi objek matematika[1] dengan data aquila
- `cout<<" Nama = "<<matematika[0].nama<<endl;`  
`cout<<" Nama = "<<matematika[1].nama<<endl;`
  - ⇒ menampilkan semua isi objek matematika

### 10.3. Latihan

1. Apa hasilnya bila dijalankan

```
#include <iostream.h>
#include <string.h>
void main(void)
{ struct nama_teman
{
    char nomhs[30];
    char nama[30];
    int umur;
    float ipk;
};
struct nama_teman temanku;
strcpy(temanku.nomhs,"005");
strcpy(temanku.nama,"ista");
temanku.umur=20;
temanku.ipk=2.54;
strcpy(temanku.nomhs,"006");
strcpy(temanku.nama,"ista2");
temanku.umur=19;
temanku.ipk=3.14;

cout <<temanku.nomhs << " ";
cout <<temanku.nama<< " ";
cout <<temanku.umur << " ";
cout <<temanku.ipk <<endl;
}
```

2. Jika ada data struktur sebagai berikut :

- NIK
- Nama
- Bagian
- Golongan

Buat structure nya dan masukan data-data berikut ini

| NIK | Nama | Bagian     | Golongan |
|-----|------|------------|----------|
| 001 | Adi  | Personalia | 3A       |
| 002 | Ade  | Keuangan   | 3B       |
| 003 | Adu  | Personalia | 4A       |



11

## POINTER



## Tujuan Instruksional

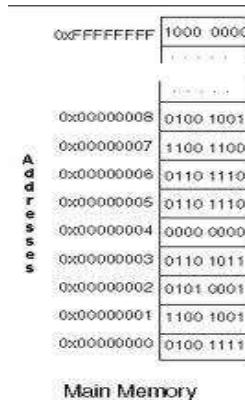
Setelah membaca bab ini, diharapkan memahami bagaimana suatu variabel disimpan di memori, alamat memori serta pointer

## Materi

-  **Pengertian Pointer**
-  **Mendefinisikan dan Mengisi variabel pointer**
-  **Pointer void**
-  **Mengubah isi variabel lewat pointer**

### 11.1. Pengertian Pointer

Pointer adalah variabel yang berisi alamat memori sebagai nilainya dan berbeda dengan variabel biasa yang berisi nilai tertentu. Dengan kata lain, pointer berisi alamat dari variabel yang mempunyai nilai tertentu.



Gambar 4.1 Pengalamatan pada memori  
[http://programmedlessons.org/AssemblyTutorial/Chapter-10/ass10\\_2.html](http://programmedlessons.org/AssemblyTutorial/Chapter-10/ass10_2.html)



Jadi, setiap variabel mempunyai data (isi data) dan alamat memori yang menunjukkan variabel tersebut disimpan dimana, variabel pointer adalah variable yang menunjuk ke objek lain.

### 11.2. Mendefiniskan dan Mengisi variabel pointer

Pendefinisian variable pointer adalah :

**Tipe\_data \* nama\_variabel**

Contoh :

**int \*a;**

variabel a bukan berisi nilai data (bilangan integer), tetapi akan berisi alamat memori. Bila ada pengisian **a=10** akan terjadi kesalahan, hal ini dikarenakan variabel a bukan berisi data tetapi alamat memori.

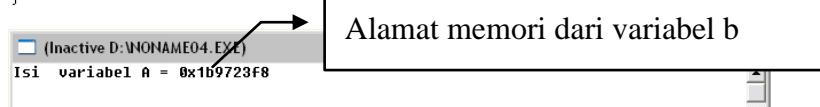
**float \*b;**

variabel b berisi alamat memori dan menunjuk pada variabel yang bertipe float

### Contoh

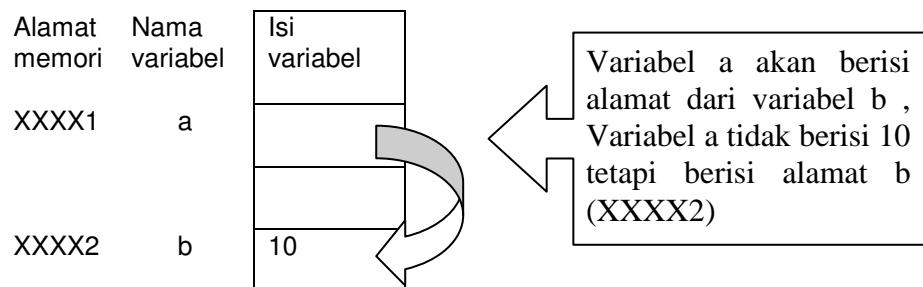
```
#include <iomanip.h>

void main(void)
{ int *a;
  int b;
  b=10;
  a=&b;
  cout <<"Isi variabel A = " << a << endl;
}
```



### Penjelasan

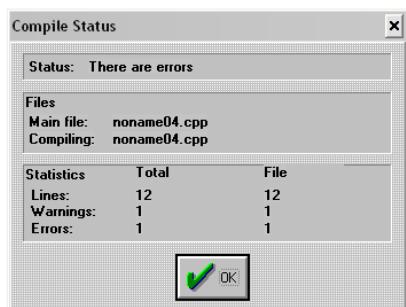
- int \* a; memesan satu variabel a, tetapi variabel a ini berisi alamat memori dari suatu data dan bukan berisi data integer (bilangan bulat)
- a=&b; melakukan pengisian data pada a, untuk mengisi alamat memori digunakan perintah &. Jadi variabel a akan berisi alamat memori dari variabel b.
- ⇒ a=&b dibaca variabel a menunjuk ke variabel b



a=&b → variabel a menunjuk ke variabel b

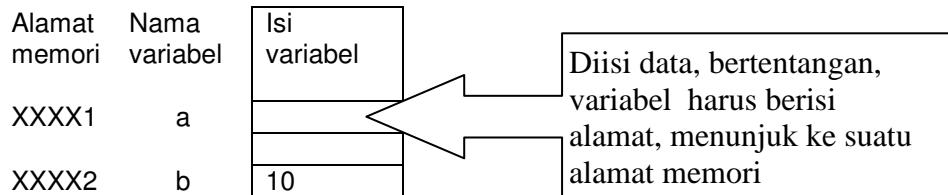
```
#include <iomanip.h>

void main(void)
{ int *a;
  int b;
  b=10;
  a=b;
  cout <<"Isi variabel A = " << a << endl;
}
```



### Penjelasan

- Program di atas terjadi kesalahan karena ada perintah `a=b;` Hal ini terjadi kesalahan karena variabel a berisi alamat memori sedangkan b adalah variabel (berisi data)



### Contoh

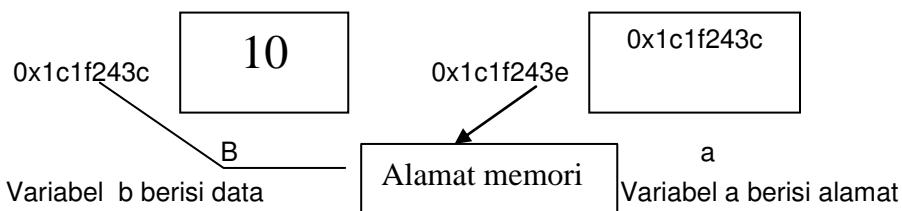
```
#include <iomanip.h>

void main(void)
{ int *a;
  int b;
  b=10;
  a=&b;
  cout <<"Isi variabel A = " << a << endl;
  cout <<"alamat variabel A = " << &a << endl;
  cout <<"isi variabel B = " << b << endl;
  cout <<"alamat variabel B = " << &b << endl;
}
```



### Penjelasan

- `Int * a;` memesan satu variabel a, tetapi variabel a ini berisi alamat memori dari suatu data dan bukan berisi data integer (bilangan bulat)
- `a=&b;` melakukan pengisian data pada a, untuk mengisi alamat memori digunakan perintah `&`. Jadi variabel a akan berisi alamat memori dari variabel b.
- `cout <<"Isi variabel A = " << a << endl;`
  - menampilkan isi variabel a, karena variabel a berisi alamat maka yang akan ditampilkan adalah alamat memori dari suatu variabel. Program diatas yang ditampilkan adalah alamat memori dari variabel b (karena ada perintah `a=&b`)
- `cout <<"alamat variabel A = " << &a << endl;`
  - menampilkan alamat memori dari variabel a sendiri
- `cout <<"isi variabel B = " << b << endl;`
  - menampilkan isi dari variabel b
- `cout <<"alamat variabel B = " << &b << endl;`
  - menampilkan alamat memori dari variabel b



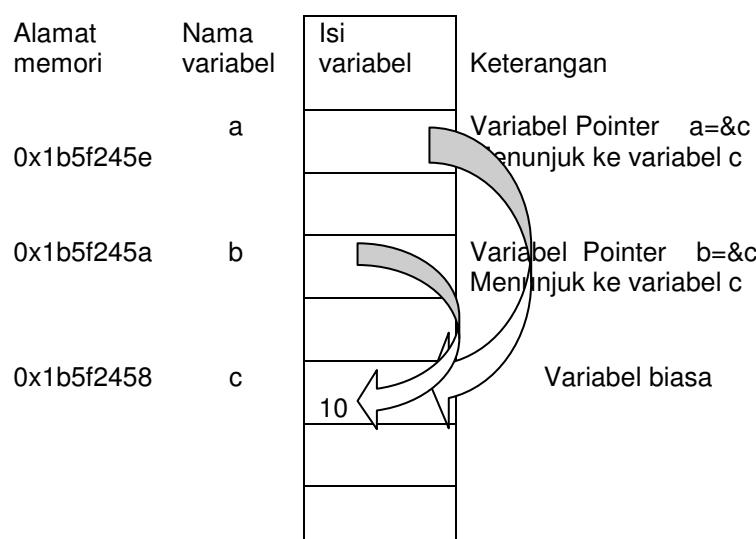
Jadi perintah **&nama\_variabel** adalah menampilkan alamat memori dari variabel tersebut

### Contoh

```
#include <iostream.h>
#include <iomanip.h>
void main(void)
{ int *a,*b;
  int c;
  c=10;
  b=&c;
  a=&c;
  cout <<"Alamat variabel A = " << &a << endl;
  cout <<"Alamat variabel B = " << &b << endl;
  cout <<"Alamat variabel C = " << &c << endl;
  cout <<"Isi variabel A = " << *a << endl;
  cout <<"Isi variabel B = " << *b << endl;
  cout <<"Isi variabel C = " << c << endl;
}
```

```
Alamat variabel A = 0x1c1f245e
Alamat variabel B = 0x1c1f245a
Alamat variabel C = 0x1c1f2458
Isi variabel A = 10
Isi variabel B = 10
Isi variabel C = 10
```

### Penjelasan



- cout <<"Isi variabel A = " << \*a << endl;  
⇒ Menampilkan nilai yang ditunjuk variabel a, nilai yang ditunjuk adalah variabel c , jadi isi yang ditunjuk adalah 10
- cout <<"Isi variabel B = " << \*b << endl;  
⇒ Menampilkan nilai yang ditunjuk variabel a, nilai yang ditunjuk adalah variabel c , jadi isi yang ditunjuk adalah 10

```

#include <iostream.h>
#include <conio.h>
void main()
{
    int *a;
    float *b;
    int d;
    float c;
    a=&d;
    cout<<"isi variabel a (alamat memori) "<<a<<endl;
    a=&c;
    cout<<"isi variabel a (alamat memori) "<<a<<endl;
}

```

Statement ini salah

The screenshot shows a 'Compile Status' window with the following details:

- Status: There are errors
- Files:
  - Main file: noname04.cpp
  - Compiling: noname04.cpp
- Statistics:
 

|           | Total | File |
|-----------|-------|------|
| Lines:    | 330   | 330  |
| Warnings: | 2     | 2    |
| Errors:   | 1     | 1    |

**OK**

Make and run the current program

### Penjelasan

- Perintah `a=&c` merupakan perintah yang salah, hal ini karena variabel pointer `a` yang bertipe integer dipaksa untuk menunjuk ke variabel yang bertipe float (pecahan)
- Perintah `a=&d` merupakan perintah yang benar, hal ini karena variabel pointer `a` yang bertipe integer menunjuk ke variabel yang bertipe integer.

### 11.3. Pointer void

Seperti penjelasan di atas, variabel pointer harus menunjuk ke variabel yang bertipe sama. Agar variabel pointer dapat bersifat fleksibel (dapat menunjuk ke berbagai tipe variabel) dapat menggunakan pointer void.

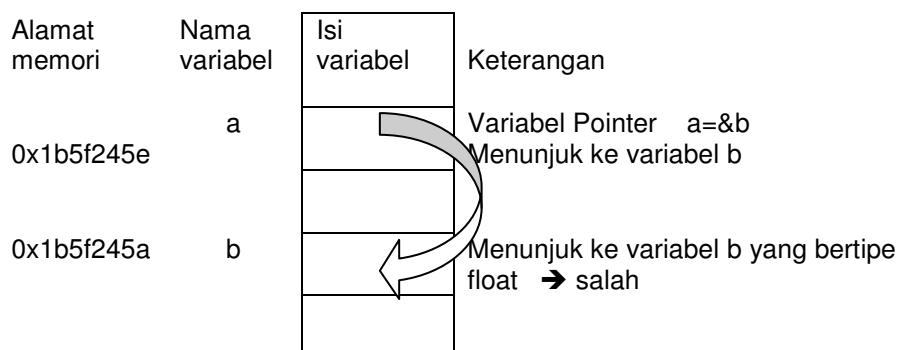
**void \*a;**

Pointer yang tidak menggunakan void

```

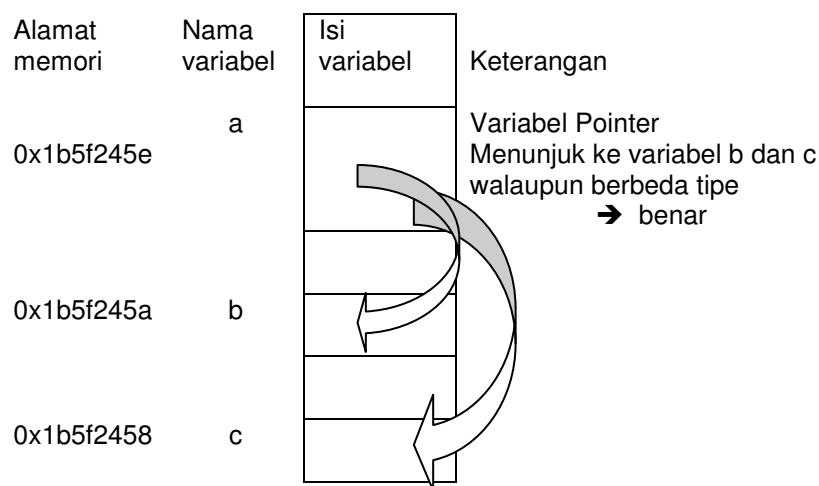
int *a;
float b;
a=&b;

```



Pointer yang menggunakan void

```
void *a;
float b;
int c;
a=&b;
.....
a=&c;
```



### Contoh

```
#include <iostream.h>
#include <conio.h>
void main()
{
    void *a;
    float *b;
    int d;
    float c;
    a=&d;
    cout<<"isi variabel a (alamat memori) "<<a<<endl;
    a=&c;
    cout<<"isi variabel a (alamat memori) "<<a<<endl;
}
```



### Penjelasan

- Variabel pointer \*a bertipe void, artinya variabel pointer tersebut bisa menunjuk ke variabel lain dengan tipe bebas
- a=&d; perintah benar, menunjuk ke tipe data int
- a=&c; perintah benar, menunjuk ke tipe data float

#### 11.4. Mengubah isi variabel lewat pointer

Untuk mengubah Isi variabel dapat dilakukan melalui pointer, misal :

```
int *a;
int b;
b=30;
```

Jika menginginkan menganti nilai b=30 menjadi 40 dapat dilakukan secara langsung (misal b=40) atau melalui variabel pointer.

```
a=&b;
*a=40;
```

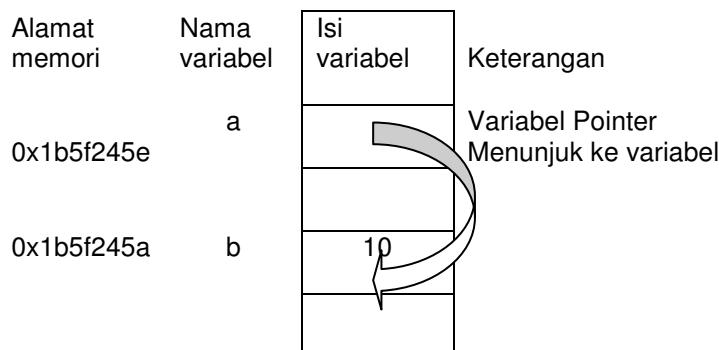
```
#include <iostream.h>
#include <conio.h>

void main(void)
{ int *a;
  int b;
  b=10;
  a=&b;
  *a=30;
  cout <<"Isi variabel A = " << *a << endl;
  cout <<"Isi variabel B = " << b << endl;
}
```

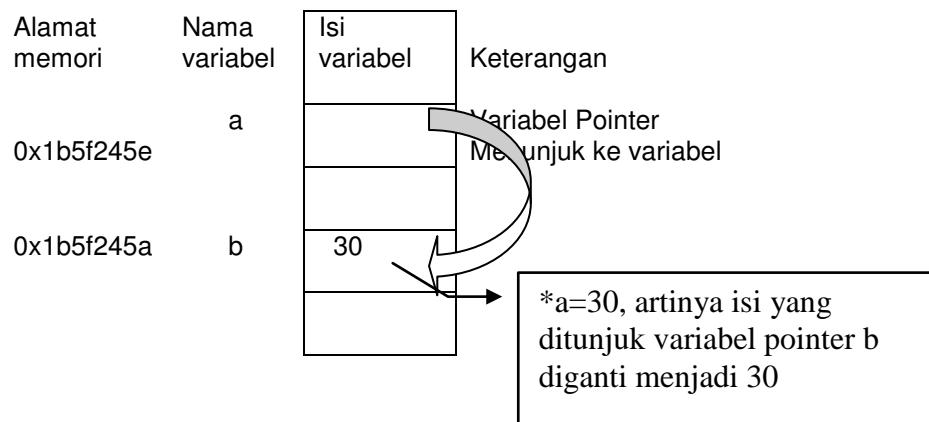


#### Penjelasan

- Int \* a → mendeklarasikan variabel pointer
- a=&b; → variabel pointer a menunjuk ke alamat memori variabel b



- `*a=30;` ➔ Isi yang ditunjuk variabel pointer a diganti menjadi 30, artinya isi yang ditunjuk a adalah variabel b, jadi isi variabel b diganti menjadi 30



### Contoh

```
#include <iostream.h>
#include <iomanip.h>

void main(void)
{ int *a,*d;
  int b,c;
  b=10;
  c=40;
  a=&b;
  d=&c;
  *a=*a * *d;
  *d=*a;
  cout <<"Isi variabel A = " << *a << endl;
  cout <<"Isi variabel B = " << b << endl;
  cout <<"Isi variabel C = " << c << endl;
}
```



### Penjelasan

- `int *a,*d;` ➔ mendeklarasikan variabel pointer
- `a=&b;` ➔ variabel pointer a menunjuk ke variabel b
- `d=&c;` ➔ variabel pointer d menunjuk ke variabel c
- `*a=*a * *d;` ➔ Nilai yang ditunjuk a isinya diganti hasil dari perkalian nilai yang ditunjuk variabel a dikalikan dengan nilai yang ditunjuk variabel d  
➔  $*a=10*40$   
➔ artinya yang diganti adalah variabel b, jadi isi b sekarang 400
- `*d=*a;` ➔ Nilai yang ditunjuk d (isi variabel c) isinya diganti nilai yang ditunjuk a  
➔  $*d=a$  artinya nilai yang ditunjuk d (isi variabel c) diganti dengan nilai yang ditunjuk a, jadi nilai variabel c=400

### 11.5. Pointer dan fungsi

Salah satu cara yang digunakan dalam fungsi adalah melewatkkan nilai pemanggil fungsi kesuatu fungsi, untuk melewatkkan nilai dapat menggunakan pointer.

Contoh :

```
a=88;  
b=77;
```

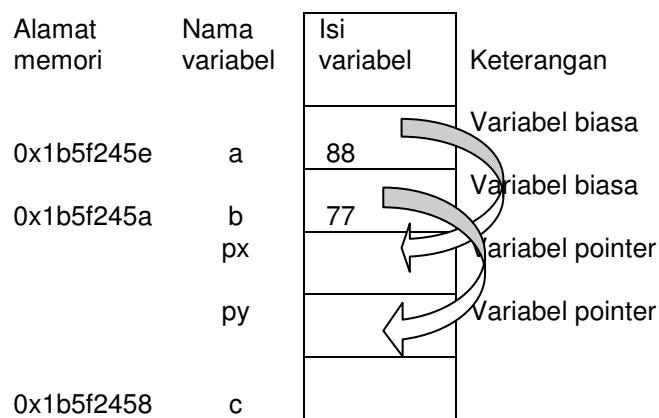
Bagaimana jika diinginkan untuk melakukan proses pertukaran data di atas, sehingga hasilnya menjadi

```
a=77;  
b=88;
```

```
#include <iostream.h>  
#include <conio.h>  
void tukar(int *px, int *py);  
void main(void)  
{  
    int a,b;  
    clrscr();  
    a=88;  
    b=77;  
    cout << "Nilai A sebelum memanggil fungsi = " << a << endl;  
    cout << "Nilai B sebelum memanggil fungsi = " << b << endl;  
    tukar(&a,&b);  
    cout << "Nilai A sesudah memanggil fungsi = " << a << endl;  
    cout << "Nilai B sesudah memanggil fungsi = " << b << endl;  
}  
  
void tukar(int *px, int *py)  
{  
    int z;  
    z=*px;  
    *px=*py;  
    *py=z;  
    cout << "Nilai didalam fungsi " << *px << endl;  
    cout << "Nilai didalam fungsi " << *py << endl;  
}
```

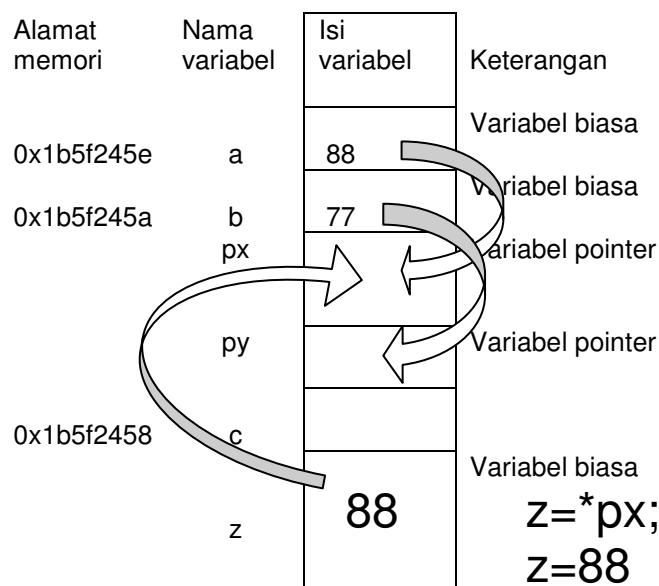
**Penjelasan :**

- Tukar(&a,&b) → artinya yang dilewatkan ke fungsi bukan isi variabel a dan b tetapi alamat memori dari variabel a dan alamat memori dari variabel b
- void tukar(int \*px, int \*py) → artinya alamat memori dari &a akan disimpan di \*px dan alamat memori dari &b akan disimpan di \*py

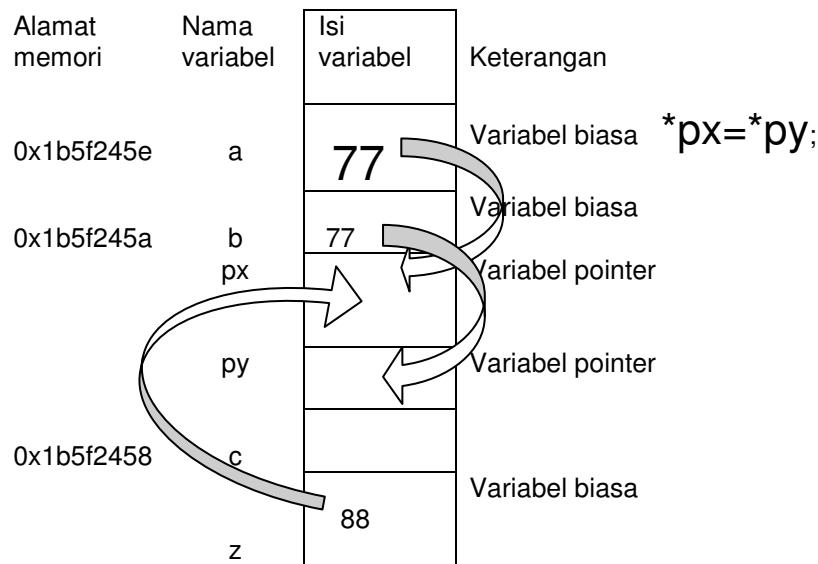


- ```
int z;
z=*&x;
```

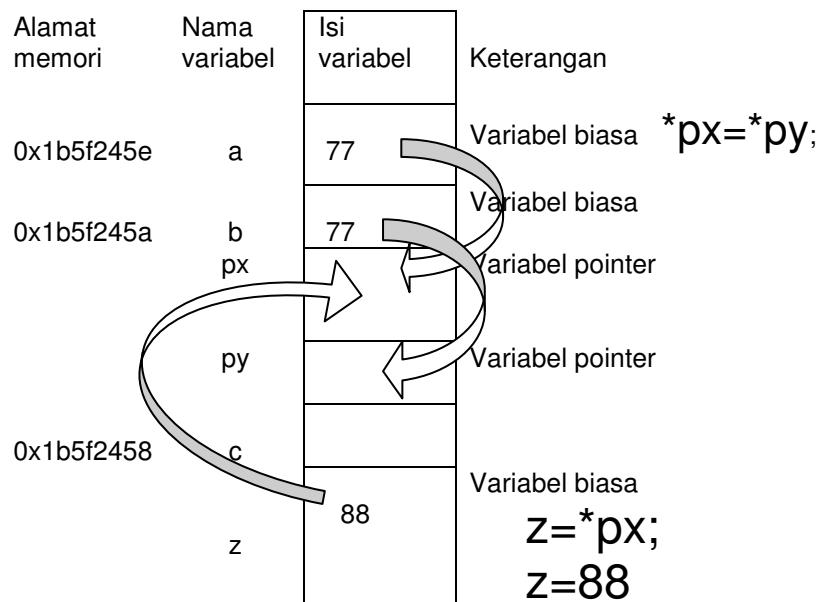
 → variabel z diisi dengan nilai yang ditunjuk oleh \*px  
 Artinya z berisi nilai 88



- `*px = *py;`  
 ➔ Nilai yang ditunjuk `* px(88)` diisi dengan nilai yang ditunjuk oleh `*py (77)`  
 Artinya nilai yang ditunjuk `*px` diganti 77



- `*py = z;`  
 ➔ Nilai yang ditunjuk `* py (77)` diisi dengan nilai dari variabel z (88)



### 11.6. Latihan

Program di bawah ini bila dijalankan hasilnya apa

1.

```
#include <iostream.h>
int main(void)
{
    int x;
    int *y;
    cout<<"masukkan nilai untuk x : ";
    cin>>x;
    cout<<"alamat x : "<<&x<<endl;
    y=&x;
    cout<<"y berisi alamat : "<<y<<endl;
    cout<<"isi y      : "<<*y;
}
```

2

```
#include<iostream.h>
void main()
{
    int x=8;
    int *px;
    px = &x;
    cout<<"Nilai x    = "<<x<<endl;
    cout<<"alamat x   = "<<&x<<endl;
    cout<<"Nilai *px  = "<<*px<<endl;
    cout<<"alamat *px = "<<&px<<endl;
    cout<<"isi    *px = "<<px<<endl;
}
```

3.

Buatlah program terdiri 5 variabel yang di inputkan dari fungsi main kemudian dilewatkan ke fungsi yang tanpa memerlukan nilai balik.

Outputnya adalah kelima variabel tersebut yang nilainya dua kali nilai sebelumnya

## INDEKS

### Simbol

--, ..... 63, 166, 168  
% ..... 61, 62, 117, 118, 129, 133, 163  
++ ..... 63  
==, ..... 59, 166, 168  
\n, ..... 26

### A

abs(), ..... 86  
acos, ..... 86  
algoritma, ..... 9, 10  
and, ..... 59, 118, 119, 120, 122  
aritmatika, ..... 59  
array dimensi dua, ..... 194  
array dimensi satu, ..... 188  
asin, ..... 86  
atan, ..... 86

### B

back slash, ..... 29  
bahasa c++, ..... 11  
binary, ..... 59  
break, ..... 35, 135, 158, 159, 160

### C

case, ..... 35, 104, 135, 193  
casting, ..... 47  
char, 35, ..... 36, 47, 89, 90, 93  
cin, ..... 25, 77, 78, 90, 91, 189, 192  
conio.h, ..... , 25  
continue, ..... 35, 160, 161, 162  
cos, ..... 85, 86  
cout, .... 17, 18, 23, 24, 25, 26, 27, 28, 39, 40,  
    73, 74, 81, 83, 84, 86, 87, 88, 106, 107,  
    108, 109, 110, 112, 113, 141, 153, 156,  
    161, 162, 166, 168, 172, 173, 174, 176,  
    177, 178, 179, 180, 181, 182, 189, 192,  
    193, 195, 196, 197, 199, 209, 211

### D

daftar parameter, ..... 172, 175  
dec, ..... 79, 82  
default, ..... , 29, 35, 135  
define, ..... 41  
do, ..... 35, 156  
double, ..... 35, 36, 86

### E

editor, ..... 15  
else, ..... 110  
endl, 25, 27, 74, 79, 81, 83, 86, 87, 88,  
    106, 107, 108, 109, 110, 112, 113,  
    141, 153, 156, 161, 162, 166, 168,  
    172, 173, 174, 176, 177, 178, 179,  
    180, 181, 182, 193, 196, 197, 199,  
    209, 211

ends, ..... 79  
escape sequences, ..... 28, 29

### F

false, ..... 35, 59, 122  
float, ..... 35, 36, 37, 38, 46, 47, 48, 49, 50,  
    85, 173, 176, 178, 179, 192, 208  
flush, ..... 79  
for, .... 7, 35, 87, 88, 140, 141, 150, 153, 161,  
    191, 192, 196, 197, 199, 204  
fungsi, ..... 79, 84, 85, 86, 92, 167, 169, 170,  
    171, 174, 200

### G

getline, ..... 91  
global, ..... 178

### H

header, ..... 25  
hex, 79, ..... 82

### I

if, ..... 105, 110, 124, 162  
include, ..... 21, 24  
instalasi, ..... 14  
int, .. 35, 36, 37, 38, 39, 42, 43, 47, 48, 60, 77,  
    79, 86, 89, 171, 172, 177, 178, 179, 180,  
    181, 182, 188, 190, 192, 194, 195, 196,  
    201  
interpreter, ..... 6  
iomanipl.h, ..... 25  
iostream.h, ..... 18, 21, 23, 25, 173, 178

**K**

kata kunci, ..... 35  
 keluaran, ..... 73, 101  
 kompilasi, ..... 6  
 kompiler, ..... 6  
 konstanta, ..... 41, 88  
 konversi, ..... 44

**L**

logika, .... 9, 59, 98, 100, 101, 117, 125, 129,  
 131, 144, 146, 163, 169, 171, 197, 202  
 lokal, ..... 177  
 long double, ..... 36  
 long int, ..... 36

**M**

main (), ..... 21  
 majemuk, ..... 66  
 manipulator, ..... 79, 82  
 masukan, ..68, 77, 95, 96, 97, 101, 163, 191,  
 192  
 math.h, ..... 25  
 matrik, ..... 197, 198, 199  
 memori, ..... 11, 24, 25, 33, 36, 89, 191, 197,  
 219, 220, 221, 222, 223, 224, 225

**N**

nilai balik, ..... 169, 174  
 nilai parameter, ..... 170  
 not, ..... 59, 118, 122

**O**

*object\_name*, ..... 208  
 operator, ...35, 47, 48, 58, 59, 62, 65, 73, 74,  
 77, 118  
 operator, ..... 59, 60, 61, 63, 65, 66, 73, 74  
 or, ..... 59, 118, 120, 121, 122  
 overflow, ..... 42

**P**

pemrograman, ..... 4  
 pemrograman, ..... 10  
 pow(), ..... 86  
 program, ..... 4

**R**

rand, ..... 7, 35, 86, 122, 124  
 random, ..... 25, 86, 87, 88  
 relasi, ..... 65  
 relational, ..... 59  
 return, ..... 22, 35, 169, 175, 176

**S**

*selection*, ..... 201  
 setbase, ..... 79  
 setfill, ..... 79, 81, 82  
 setiosflags, ..... 79, 83  
 setprecision, ..... 79, 84  
 setw, ..... 25, 79, 80, 81, 82, 83  
 sin, ..... 85, 86  
 sintaks, ..... 16  
 sqrt, ..... 25, 86  
 statis, ..... 180  
 stdlib.h, ..... 25, 86, 87  
 strcat, ..... 92  
 strcpy, ..... 25, 92, 93, 209, 210, 211  
 string, ..23, 25, 36, 73, 74, 88, 89, 90, 91, 92,  
 93,208  
 strlen(), ..... 25, 92  
 strlwr, ..... 92  
 strrev, ..... 92  
 struct, ..... 35, 208  
 structure, ..... 208, 210  
 struktur bahasa c++, ..... 21  
 strupr, ..... 92, 94  
 switch, ..... 35, 135, 158, 193

**T**

tan, ..... 86  
 time.h, ..... 25  
 tipe data, ..... 36, 44  
 true, ..... 35, 59, 122, 141, 153, 156

**U**

unary, ..... 59  
 variabel, ....33, 34, 36, 37, 40, 45, 46, 47, 49,  
 60, 89, 107, 177, 178, 180, 188

**W**

while, ..... 7, 153, 156, 157, 158, 161

## GLOSARIUM

Algoritma	: Sekumpulan langkah-langkah atau instruksi-instruksi yang terbatas untuk menyelesaikan suatu masalah
Array	: Variabel yang dapat menyimpan lebih dari 1 data
Bahasa Pemrograman	: Bahasa untuk menuliskan langkah-langkah dalam bentuk perintah dan pernyataan ( <i>statement</i> )
cin	: Perintah di C++ yang digunakan untuk meminta data, dimana data tersebut dimasukkan melalui keyboard
cout	: Perintah ini digunakan untuk menampilkan kalimat atau string ke layar monitor
Deklarasi	: Suatu cara untuk mendefinisikan suatu variabel atau fungsi
Editor	: Tempat yang digunakan untuk menulis perintah
Escape Sequences	: Karakter khusus yang menggunakan notasi “\” (back slash)
Flowchart	: Simbol yang digunakan untuk menggambarkan langkah-langkah bagaimana suatu program berjalan
Fungsi Manipulator	: Perintah di C++ yang digunakan untuk mengatur tampilan layar
Logika,	: Proses berfikir manusia untuk menghubungkan fakta atau pernyataan sehingga sampai pada suatu kesimpulan
if –else	: Perintah di C++ yang digunakan untuk melakukan proses seleksi dari suatu kondisi
Inisialisasi	: Suatu cara untuk memberikan nilai awal dari suatu variabel
for	: Perintah di C++ yang digunakan untuk melakukan proses perulangan
Fungsi	: Cara membuat program dengan membuat bagian-bagian tertentu dari suatu program
include	: Merupakan suatu header yang . Mempunyai kegunaan untuk ‘menerjemahkan’ kegunaan dari perintah-perintah yang akan digunakan
Konstanta	: Suatu variabel yang mempunyai nilai bersifat tetap dan tidak bisa diubah/ diganti.
Konversi Tipe Data	: Proses untuk melakukan perubahan tipe data dari suatu variabel
Nilai Balik	: Nilai yang dibawa dari suatu fungsi ke pemanggil fungsi
Operator	: Merupakan simbol yang bisa digunakan untuk melakukan suatu operasi
Overflow Data	: Kondisi dimana suatu isi variabel diisi dengan suatu nilai yang melebihi dari nilai jangkauannya

Parameter	: Nilai yang dibawa dari suatu pemanggil fungsi ke suatu fungsi
Program Komputer :	: Urutan perintah yang diberikan pada komputer untuk membuat fungsi atau tugas tertentu
Programmer	: Seorang praktisi yang memiliki keahlian untuk melakukan penulisan kode dalam bahasa pemrograman
Pointer	: variabel yang berisi alamat memori sebagai nilainya
Proses Kompilasi	: Proses menggabungkan serta menterjemahkan sesuatu (source program) menjadi bentuk lain (bahasa mesin).
Structure	: Kumpulan berbagai tipe data yang memiliki ukuran yang berbeda di kelompokan dalam satu deklarasi unik
Tipe Data	: Tipe data akan mencerminkan isi dari suatu variabel apakah termasuk bilangan atau string serta jangkauan atau maksimal isi data dari variabel
Tipe Casting	: Merubah sementara tipe suatu data yang sudah didefinisikan
Variabel	: Menyimpan data yang akan diolah disimpan oleh komputer
Variabel Lokal	: Suatu nilai variabel yang hanya dikenal di suatu bagian fungsi tertentu saja
Variabel Global	: Suatu nilai variabel yang dikenal di semua bagian fungsi
Variabel Statik	: Merupakan variabel yang hanya dikenal dalam suatu fungsi dan nilai dari variabel ini tidak akan hilang saat keluar dari suatu fungsi
while	: Salah satu perintah di C++ yang digunakan untuk melakukan proses perulangan

## DAFTAR PUSTAKA

- Bäckman, K (2010). Structured Programming with C++. <http://bookboon.com/en/structured-programming-with-c-plus-plus-ebook>.
- Davis, S. R. (2012). Beginning Programming with C++ For Dummies. New Jersey: John Wiley & Sons Inc.
- Davis, S. R. (2004). C++ For Dummies. Indianapolis, Indiana: Wiley Publishing, Inc.
- Fatta, H. A. (2008). Dasar Pemrograman C++ Disertai Dengan Pengenalan Pemrograman Berorientasi Objek. Yogyakarta: Penerbit Andi.
- Gregoire, M. (2011). Professional C++. Indianapolis, Indiana: John Wiley & Sons, Inc.
- Kadir, A. (2009). Mudah Menjadi Programmer C++. Yogyakarta: Penerbit Andi.
- Kadir, A. (2009). Pemrograman C++ Membahas Pemrograman Berorientasi Obyek Menggunakan Turbo C++ dan Borland C++. Yogyakarta: Penerbit Andi.
- Library, W. e. (2004). C++ Programming Open Source Language . [www.WorldLibrary.net](http://www.WorldLibrary.net) .
- Sholeh, M. (2009). Algoritma dan Pemrograman. Yogyakarta: Akprind Press.
- Stroustrup, /. B. (1997). The C Programming Language . Massachusetts: Wesley Publishing.
- Swart, B. (2003). Borland C++ Builder 6 Developoer's Guide. Indianapolis, Indiana: sams Publishing.
- Syahrial, M. (2007). Mahir dan Profesional Pemrograman C++. Medan: Gratech Media Perkasa.

### Internet

- [http://1.bp.blogspot.com/\\_lymwnivw9i8/tup\\_n-v7ebi/aaaaaaaaaaea/pcgcnqhyeuc/s320/1111.jpg](http://1.bp.blogspot.com/_lymwnivw9i8/tup_n-v7ebi/aaaaaaaaaaea/pcgcnqhyeuc/s320/1111.jpg)  
[http://1.bp.blogspot.com/-kHKIrTgdyjo/UIOgB1quII/AAAAAAAABg/-JLoh\\_UuBeA/s320/cm1.JPG](http://1.bp.blogspot.com/-kHKIrTgdyjo/UIOgB1quII/AAAAAAAABg/-JLoh_UuBeA/s320/cm1.JPG)  
<http://imanmauludin.wordpress.com/2012/10/01/bahasa-pemrograman-delphi/>  
<http://desylvia.wordpress.com/2010/09/06/pemrograman-bahasa-c-pendahuluan/>  
[www.facebook.com](http://www.facebook.com)  
[www.wordpress.com](http://www.wordpress.com)  
[elearning.akprind.ac.id](http://elearning.akprind.ac.id)  
<http://blog.banditbatak.com/featured/mencari-tahu-hari-kelahiran-anda/>