PENGENALAN MUDAH PENGATURCARAAN C

Dayang Norhayati Abang Jawawi
Rosbi Mamat

Kandungan

1. Pengenalan Kepada Komputer

1.0 Apa itu Komputer ?

- Mesin/peranti yang digunakan untuk mengendali data input sebagaimana yang diarahkan oleh pengguna.

- Untuk mengendalikan satu komputer, komputer itu mesti diarahkan dengan tepatnya apa tindakan-tindakan yang perlu dilakukan.

- Satu jujukan arahan kepada komputer dipanggil ***aturcara komputer*** *(computer program)*.

- Arahan-arahan yang difahami secara langsung oleh komputer dipanggil ***kod mesin/bahasa mesin***.

- Proses menyediakan aturcara komputer dipanggil ***pengaturcaraan komputer*** *(computer programming).*

- Aturcara komputer ditulis dalam ***bahasa pengaturcaraan*** (programming language).

1.1 Komputer Dulu & Sekarang

- SEKARANG - hampir semua urusan & aktiviti harian kini menggunakan komputer.
  Cth: bayar bil, beli tiket, ketuhar mikro, mesin basuh, telefon bimbit dll.....(terima kasih)

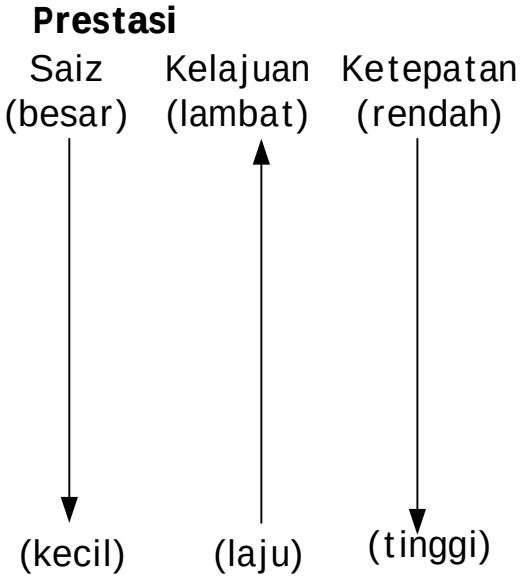
  Bidang kejuruteraan & sains ? Cth: Industri membuat kereta:
  dari proses rekabentuk hingga proses penjualan kereta.

1.1 Komputer Dulu & Sekarang (samb....)

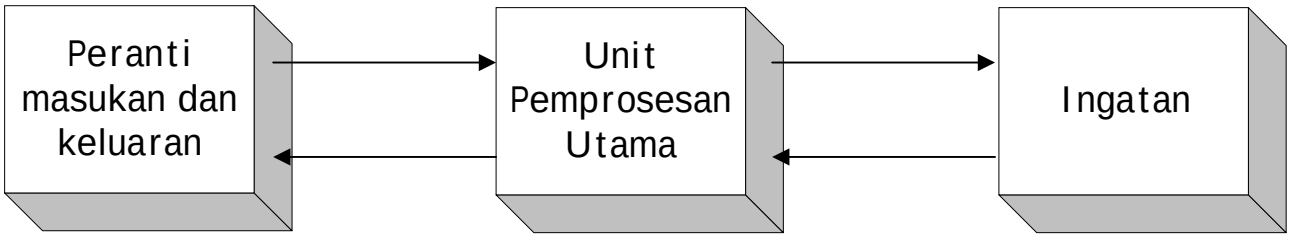
- Dulu hingga sekarang

Komputer yg dicipta dikelaskan mengukut teknologi & masa berikut:

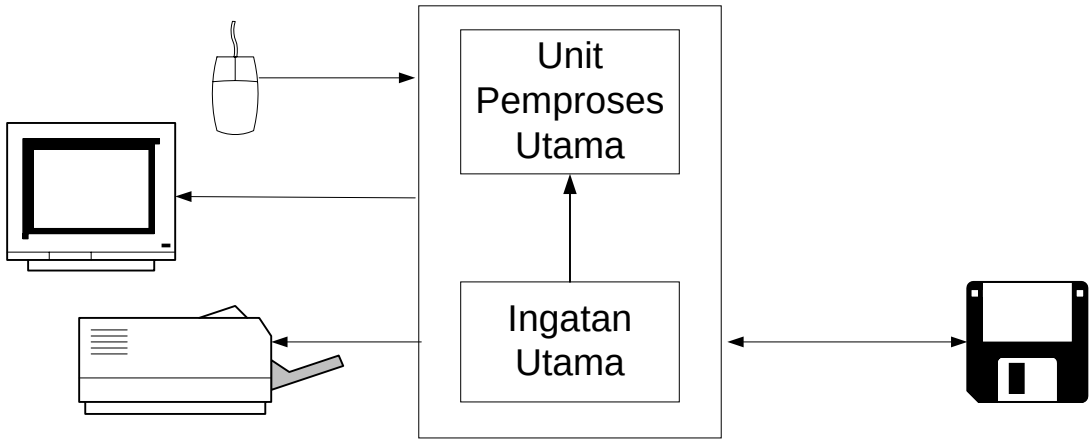
Generasi (tahun)	Teknologi
Pertama (1939-1958)	Teknologi tiub hampa gas
Kedua (1959-1964)	Teknologi transistor
Ketiga (1965-1974)	Teknologi litar bersepadu
Keempat (1975-sekarang)	Teknologi VLSI



1.2 Komponen Satu Komputer



Contoh satu unit komputer:



1.2 Komponen Satu Komputer (samb....)

- Unit Pemprosesan Unit (CPU) - Bertanggungjawab melaksanakan aturcara.

- Ingatan - Menyimpan aturcara dan data.
  - Ingatan utama (EPROM/ROM, RAM)
  - Storan sekunder (Cakera padat)
- Peranti Masukan dan Keluaran

  - Aturcara & data dimasukkan ke ingatan melalui peranti masukan.

  - Hasil pemprosesan dihantar ke peranti keluaran.

- Perkakasan (Hardware) - Peralatan komputer yang diperlukan untuk melaksana aturcara.
- Perisian (Software) - Aturcara yang menjadikan perkakasan berguna.

Apakah fungsi asas Komputer? = Bagaiamana komputer beroperasi ?

1. Input	memasukan data & arahan ke dlm ingatan. menggunakan peranti input.
2. Pemprosesan	melaksanakan arahan aturcara & memproses data. menggunakan CPU
3. Output	mengeluarkan/memaparkan keputusan & maklumat yang dihasilkan. menggunakan peranti output.
4. Storan	menyimpan maklumat utk kegunaan masa hadapan. menggunakan ingatan utama & ingatan sekunder. ingatan utama lwn. ingatan sekunder.

1.3 Kategori Komputer

- Dikelaskan mengikut saiz fizikal, kapasiti ingatan dan kelajuan pemprosesannya, 3 kategori:

1. Komputer mikro (microcomputer)	Dikenali sbg komputer peribadi (PC) atau desktop. Untuk kegunaan peribadi & perseorangan Terbesar-worstation & terkecil notebook. Cth kegunaan kejuruteraan - untuk lukiran & lakaran rekabentuk untuk pembangunan produk baru
2. Komputer mini (minicomputer)	Digunakan oleh lebih dpd seorang pengguna pada satu masa. Biasanya digunakan oleh syarikat kecil & sederhana. Cth kegunaan kejuruteraan - mengawal proses pengeluaran & chemical plants
3. Kerangka Utama (mainframes)	Digunakan oleh syarikat/organisasi yg besar. Terbesar-supercomputer. Cth kegunaan kejuruteraan - pemprosesannumber (number cruncher).

Era baru komputer ?

1.4 Perisian

- Setakat ini kita telah menyebut secara ringkas ?perisian & ?aturcara.

- Perisian
  merupakan sekumpulan arahan yg disusun dlm sintak yg tertentu untuk melaksanakan sesuatu tugas. Dlm kelas ini kita akan belajar menyusun arahan dgn. menggunakan bahasa pengaturcaraan C.

- 2 bahagian perisian komputer:
  1. Perisian sistem
  2. Perisian pengguna@aplikasi

1.4.1 Perisian Sistem

- Aturcara perkakas yang memudahkan komputer untuk digunakan, terutama kepada pembangun & pengaturcara perisian.

- Cth[2] perisian sistem:

Sistem pengendalian (operating system)	Perisian komputer yg berfungsi mengtakbir pengendalian, peruntukan & penjadualan sumber & proses dlm sistem komputer supaya pengguna dpt menggunakan sistem dgn mudah. (cth: Windows, UNIX & OS/2)
Penyunting teks	Perisian yg digunakan utk menulis teks, mengedit teks & menyimpan teks. Ada berbagai penyunting teks ttp utk penulisan aturcara C pemilihan penyunting teks yg betul, boleh meemudahkan proses penulisan aturcara.
Pentejemah bahasa	Tukar bhs pengaturcaraan aras tinggi @ bhs himpunan kpd aturcara yg sesuai dgn bhs mesin.
pemaut (linker)	Aturcara yg berfungi utk memautkan @ menggabungkan aturcara utk menjadi satu aturcara yg boleh dilaksanakan.
Pemuat (loader)	Aturcara kawalan dlm sistem pengendalian yg bertugas utk memuatkan rutin sistem pengendalian cakera, aturcara pengguna @ data dari punca luar spt cakera liut @ pita, kedalam simpanan utama sistem.

Kebanyakan perisian ini telah digunakan & akan digunakan dlm kelas C.

1.4.2 Perisian Aplikasi

- Aturcara yang dibangunkan khusus utk melakukan kerja^2 tentu @ menyelesaikan masalah tertentu dgn menggunakan perisian sistem.

- 2 jenis perisian aplikasi:
  1. Perisian Penggunaan Umum (PPU).
  2. Perisian Penggunaan Khusus (PPK).

- PPU - aplikasi yg mudah dibeli dipasaran @ 'off-the-shelf', juga dipanggil pakej perisian.

  - memudahkan pengguna melaksanakan kerja^2 tertentu. cthnya:

Jenis	Fungsi & cth aplikasi
Pemproses kata	utk membuat dokumen spt Microsoft Word & Word Perfect.
helaian hamparan	pengiraan & pemplotan graf spt Microsoft Excel.
lukisan grafik	utk lukisan gambarajah @ lukisan rekabentuk spt AutoCad, Visio Technical & Microsoft Paint

- PPK - aplikasi yg dibangunkan khas untuk menyelesaikan masalah yang tertentu bagi sebuah organisasi tertentu @ org persendirian.

  - perisian tersebut boleh ditulis oleh pengaturcara dari syarikat tersebut @ oleh syarikat yg pembangun perisian.
  Cthnya ..... aturcara yg dibangunkan oleh pusat komputer utk mencetak keputusan peperiksaan UTM.

Dlm kelas ini kita akan belajar bhs pengaturcaraan C untuk menghasilkan perisian jenis mana ??????

1.5 Bahasa Pengaturcaraan

- Bahasa pengaturcaraan boleh dibahagikan kpd. 3 kumpulan:

a. Bahasa mesin (BM)
b. Bahasa himpunan (BH)
c. Bahasa aras tinggi (BAT)

a. Bahasa Mesin

Bahasa tabii sesuatu komputer/CPU.

Satu-satunya bahasa yang difahami secara langsung oleh komputer.

Suruhan berbentuk gabungan nombor-norbor perduaan.

Setiap jenis CPU mempunyai bahasa mesin yang berlainan.

Contoh bahasa mesin untuk mencampur 2 nombor:

```
0101 1000 0001 0000
0101 1011  0001 0000
0101 0000 0001 0000
```

b. Bahasa Himpunan

Terdiri dari nemonik[2] (singkatan) yang lebih mudah difahami dari bahasa mesin.

Bahasa himpunan **TIDAK** difahami oleh CPU.

⇨ perlukan perisian **penghimpun** untuk menukarkan BH kpd BM.

Contoh :

MOVE #$1,D1
MOVE #$8,D0
ADD D0,D1
	0101 1000 0001 0000
	0101 1011 0001 0000
	0101 0000 0001 0000

c. Bahasa Aras Tinggi

Terdiri dari suruhan[2] yang menyerupai bahasa Inggeris.

Satu ruruhan BAT terdiri dari banyak BM/BH.

Contoh BAT :

FORTRAN, COBOL BASIC, Pascal, Ada, C, C++ dsbnya

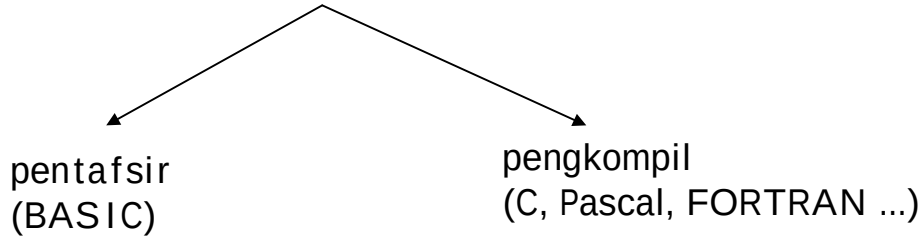
Contoh suruhan BAT:

```
LUAS = PANJANG * LEBAR
IF LUAS > 50
        GOTO BESAR
ELSE
        GOTO KECIL
```

Kebaikan BAT :
- tidak bergantung kpd jenis CPU
- mudah difahami oleh manusia

Keburukan BAT :
- tidak difahami oleh CPU !

⇨ perlukan **penterjemah bahasa** utk menukar BAT kpd BM.



Latihan:

1. Nyata sama ada komponen[2] berikut perkakasan komputer atau perisian komputer:

a) Ingatan utama
b) Sistem pengendalian
c) Peranti masukan & keluaran
d) Prapemproses
e) pemaut (linker)
f) Storan sekunder
g) Unit prmprosesan utama
f) Pemuat (loader)

2. Soalan ulangkaji:

a) Senaraikan 2 fungsi CPU.
b) Senaraikan 2 peranti masukan/keluaran & 2 peranti storan skunder.
c) Komputer hari ini boleh berfikir? Betul @ Salah
d) Senaraikan 3 kumpulan bhs pengaturcaraan.
e) Senaraikan 3 nama BAT.
f) Apakah kebaikan BAT.

2. Penyelesaian Masalah & Pengenalan Kepada Pengaturcaraan

2.0 Penyelesaian masalah dgn komputer

- Dari Bab 1 : Komputer boleh menyelesaikan masalah tertentu jika diberi arahan dgn menulis aturcara.

- Pengaturcaraan **bukan** hanya menulis aturcara.

- Penyelesaian masalah dengan komputer adalah prosedur untuk menentukan penyelesaian kepada sesuatu masalah dan menyatakannya dalam bentuk yang jelas - PALING SUSAH.

- Penyelesaian masalah (PM) memerlukan kemampuan seni & sains. Bidang kejuruteraan & sains menggunakan kaedah kejuruteraan & sains utk menyelesaikan masalah.

- Proses pembangunan perisian yg menggunakan kaedah PM kejuruteraan & sains dinamakan kejuruteraan perisian.

- PM bidang kejuruteraan lwn kejuruteraan perisian (KP).

Langkah^2 PM Kejuruteraan	**Langkah^2 PM KP**
1. Mengenali & memahami masalah	1. Dapatkan spesifikasi keperluan masalah/sistem
2. Mengumpul maklumat	2. Analisa masalah
3. Memilih teori & prinsip yg bersesuaian 4. Membuat andaian^2 yg diperlukan	3. Rekabentuk penyelesaian
5. Selesaikan masalah	4. Pengkodan & pelaksanaan
6. Menguji keputusan	5. Pengujian perisian
	6. Penyenggaraan

2.1 Pembangunan Perisian

- Langkah^2 proses pembangunan perisian:

1. Spesifikasi keperluan
2. Analisa masalah
3. Rekabentuk
4. Pengkodan & pelaksanaan
5. Penentusahan & Pengujian
6. Penyenggaraan

2.1.1 Spesifikasi Keperluan

- Memahami masalah & persekitaran masalah yg hendak diselesaikan.

- Andaian^2 juga perlu dilakukan utk memastikan skop masalah.

Cth masalah mudah : membuat kopi.
1. Memahami masalah : Kopi utk berapa orang, kopi O @ kopi susu @ kedua2nya dll.
2. Persekitaran masalah : Situasi buat kopi di pejabat, rumah & apa kelengkapan yg ada dll.
3. Andaian : Taraf gula yg diambil sama, air yg digunakan belum dimasak dll.

2.1.2 Analisa Masalah

- Mengkaji/menganalisa spesifikasi masalah dgn terperinci. 2 elemen utama yg perlu di kenal pasti:

i. input masalah

Apakah data @ maklumat yg perlu diberi ke dalam sistem aturcara utk penyelesaian masalah tersebut.
Format input juga perlu ditentukan.

cth: kopi, gula, air (format: yg belum dimasak) & krim.

2.1.2 Analisa Masalah (samb ....)

ii. output masalah

Apakah data @ maklumat yg perlu dari aturcara sebagai penyelesaian pada masalah.
Format keluaran perlu ditentukan.

cth : Format 5 cawan kopi susu @ Format 5 cawan kopi 'O' dgn 5 sudu susu dlm bekas berasingan.

ii. proses

Menganalisa apakah proses yg perlu dilalui oleh input utk menghasilkan output. Kekangan^2 & andaian^2 pd PM yg lebih terperinci dilakukan disini.

cth : formula yg digunakan, proses masuk kopi dlm cawan dll.

2.1.3 Rekabentuk

- 2 tugas utama dlm rekabentuk aturcara:
  - memilih struktur yg sesuai utk menyusun & menyimpan data dalam proses penyelesaian masalah.
  - merekabentuk **algoritma** utk memproses input & mengeluarkan output.

- Bagaimana algoritma dibuat dlm proses pembangunan aturcara ? ..... TUNGGU.....

2.1.4 Pengkodan & Pelaksanaan

- Proses melaksanakan algoritma yg direkabentuk ke dlm bahasa pengaturcaraan tertentu (dlm kelas ini bhs C).

- Format input yg telah dikenal pasti akan digunakan bersama algoritma tersebut & keluaran akan dikeluarkan mengikut format yg juga telah dikenal pasti.

- Bagaimana perlaksanaan algoritma dilakukan dlm pengaturcaraan C?.....TUNGGU.....

2.1.5 Penentusahan (verification) & Pengujian

- Utk menentukan aturcara yg dihasilkan betul.

- Pengujian aturcara perlu dilakukan dgn memberi input & memeriksa output sama ada output yg betul dikeluarkan dari aturcara yg ditulis.

- Perlu dilakukan pada semua fasa sebelum ini..... KENAPA?

  Fasa 1 & 2
  Berinteraksi pd orang yg memberi masalah (pakar) yg utk diselesaikan supaya pemahaman masalah tepat & tidak terpesong (pakar masalah kelas ini adalah SAYA).

  Fasa 3
  Pastikan rekabentuk yg terhasil memenuhi spesifikasi keperluan yg telah dihasilkan dr fasa 1 & 2.

  Fasa 4
  Pastikan aturcara selaras dgn algoritma rekabentuk & lengkap.
  Pastikan tiada kesilapan sintaks (ralat yg dikesan semasa mengkompail aturcara) & kesilapan logik (ralat masa larian) pada aturcara. AKAN DIALAMI SEMUA.....

2.1.6 Penyenggaraan

- Proses penyenggaraan perisian amat unik jika dibanding dgn produk kejuruteraan lain.
  Cthnya kereta : jika roda kereta rosak tukar pada roda yg baru.
  jika fungsi utk antara muka perisian word perlu ditukar BAGAIMANA????

- Taraf kesukaran penyenggaran sesuatu perisian bergantung kpd fass 1-5 yg telah ditempuhi oleh perisian tersebut... SAMADA PENULISNYA MENGIKUT PRINSIP KP @ TIDAK.....

2.2 Algoritma & Bagaimana Mengungkap Algoritma

- Sila sambung TUNGGU 1.1.3 disini.

- Untuk menyelesaikan masalah dengan komputer, kita perlu:

  1. Reka algoritma (fasa rekabentuk)
  2. Nyatakan algoritma sebagai aturcara dalam bahasa aturcara yang sesuai (fasa pengkodan & perlaksanaan)
  3. Laksanakan aturcara tersebut di atas komputer (fasa pengkodan & perlaksanaan)

- Algoritma - resipi bagaimana untuk menyelesaikan masalah dengan komputer. Ia menerangkan cara & langkah yang perlu dilakukan untuk menyelesaikan masalah.

- Tanpa algoritma tiada aturcara, tanpa aturcara tiada penyelesaian.

- Algoritma tidak bergantung kepada bahasa pengaturcaraan.

- Apa yg perlu ada dlm algoritma:
  - input
  - output
  - tidak mengelirukan/ringkas/mudah
  - dlm bentuk umum - boleh difahami semua golongan pembaca yg berkeperluan
  - tepat utk menyelesaikan masalah
  - terhingga - ada kesudahan
  - cekap

- Berbagai kaedah mengungkap algoritma :

Kenyataan bahasa	Carta alir	Kod Pseudo
Gambarajah N-S	Jadual aksi	Mesin keadaan terhingga

  ... dsbnya.

2.2.1 Kaedah kod pseudo

- Kod pseudo adalah jujukan langkah yg ditulis menggunakan bhs tabii yg menerangkan pemprosesan yg terlibat dlm penyelesaian masalah.

- Tujuan utama mempersembahkan penyelesaian masalah dalam bentuk yang logik dan sistematik.

- Ada beberapa cara penulisan kod pseudo digunakan, dlm kelas ini 2 jenis akan diperkenalkan:
  - kenyataan bahasa
  - kenyataan bahasa menyerupai bahasa pengaturcaraan

- Setiap langkah dlm penyelesaian masalah disususn satu-persatu secara sistematik. Setiap baris kod pseudo mewakili satu langkah yang dilaksanakan sekali shj.

i. Kaedah kod pseudo - kenyataan bahasa

- Algoritma yg ditulis dlm bahasa tabii & tidak mempunyai peraturan yg khusus.

- Dalam kaedah ini nombor yg diberikan pada jujukan langkah adalah sangat penting.

- Contoh : algoritma untuk menyediakan secawan kopi

1. masak air
2. masukkan kopi dalam cawan
3. jika perlu gula
   3.1. tambahkan gula
4. jika perlu krim
   4.1 tambahkan krim
5. masukkan air panas ke dalam cawan

- Tertib perlaksanaan langkah[2] mengikut nombor yg ditulis.

2.2.1 Kaedah kod pseudo (samb...)

- Sesuai utk masalah yg mudah krn tiada jujukan & struktur yg kompleks.

a) Struktur jujukan:

1. mula
2. langkah_1
3. langkah_2
:
n. langkah_n
n+1. tamat

b) Struktur pemilihan:

n. jika syarat
    n.1. langkah_n.1
    n.2. langkah_n.2
    :
    n.n. langkah_n.n

c) Struktur ulangan:

n. pergi ke langkah sebelum n

b) Struktur lompat:

n. pergi ke langkah selepas n

ii. Kaedah kod pseudo - kenyataan bhs menyerupai bhs pengaturcaraan

- Kod pseudo adalah pengungkapan algoritma dlm bahasa Inggeris mudah yg hampir menyerupai bahasa pengaturcaraan PASCAL - TAPI ia BELUM terikat kpd sebarang bahasa pengaturcaraan !

- Tertib perlaksanaan langkah^2 mengikut aturan langkah^2 itu ditulis.

Contoh algoritma utk memasak air:
    isi air ke dlm cerek
    nyalakan api
    masak air hingga mendidih

Tertib perlaksanaan

2.2.1 Kaedah kod pseudo (samb...)

- Tertib perlaksanaan langkah^2 boleh dikawal dgn menggunakan beberapa struktur kawalan. Kata-kunci^2 digunakan utk menjelaskan struktur^2 kawalan ini.

a) Struktur jujukan:

```
begin
    langkah_1
    langkah_2
    :
    langkah_n
end
```

b) Struktur pemilihan:

```
if  syarat then
      bhg_benar
else
      bhg_takbenar
endif
```

c) Struktur ulangan:

```
repeat
    bhg_yg_perlu diulang
until  syarat
     @
while  syarat   do
    bhg_yg_perlu diulang
endwhile
     @
repeat N kali
    bhg_yg_perlu diulang N x
```

Perhatikan:
i. syarat adalah ungkapan samada benar @ tidak
ii. bhg tertentu teks (blok) diindenkan (diberi jidar) utk menjelaskan kod pseudo
iii. Blok kadangkala dikumpulkan diantara kata kunci ***begin*** & ***end***

- Kod pseudo menyediakan secawan kopi

```
masak air
masukkan kopi dalam cawan
if  perlu_gula then
     tambahkan gula
endif
if  perlu_krim then
     tambahkan krim
endif
masukkan air panas ke dalam cawan
```

2.2.1 Kaedah kod pseudo (samb...)

- Kod pseudo menyediakan secawan kopi

```
masak air
masukkan kopi dalam cawan
if  perlu_gula
    then tambahkan gula
endif
if  perlu_krim
    then tambahkan krim
endif
masukkan air panas ke dalam cawan
```

- Penghalusan berlangkah algoritma

Perhatikan, kod pseudo di atas masih lagi dinyatakan dlm ungkapan umum. Contohnya, *masak air, masukkan kopi* dsbnya. Algoritma ini perlu dihalusi lagi utk menghapuskan kekeliruan dlm melaksanakannya.

Kod pseudo di atas adalah contoh bagaimana fasa pertama algoritma ditulis, bermula dari langkah^2 yg agak umum. Dlm fasa^2 yg seterusnya, langkah^2 umum ini dihalusi sedikit demi sedikit sehingga menghasilkan langkah^2 yg lebih spesifik.

Teknik merekabentuk algoritma sebegini dipanggil **teknik Atas ke Bawah** (Top-down). Proses menghalusi algoritma sedikit demi sedikit dipanggil **Penghalusan berlangkah**.

Contoh: Penghalusan berlangkah ungkapan *masak air*

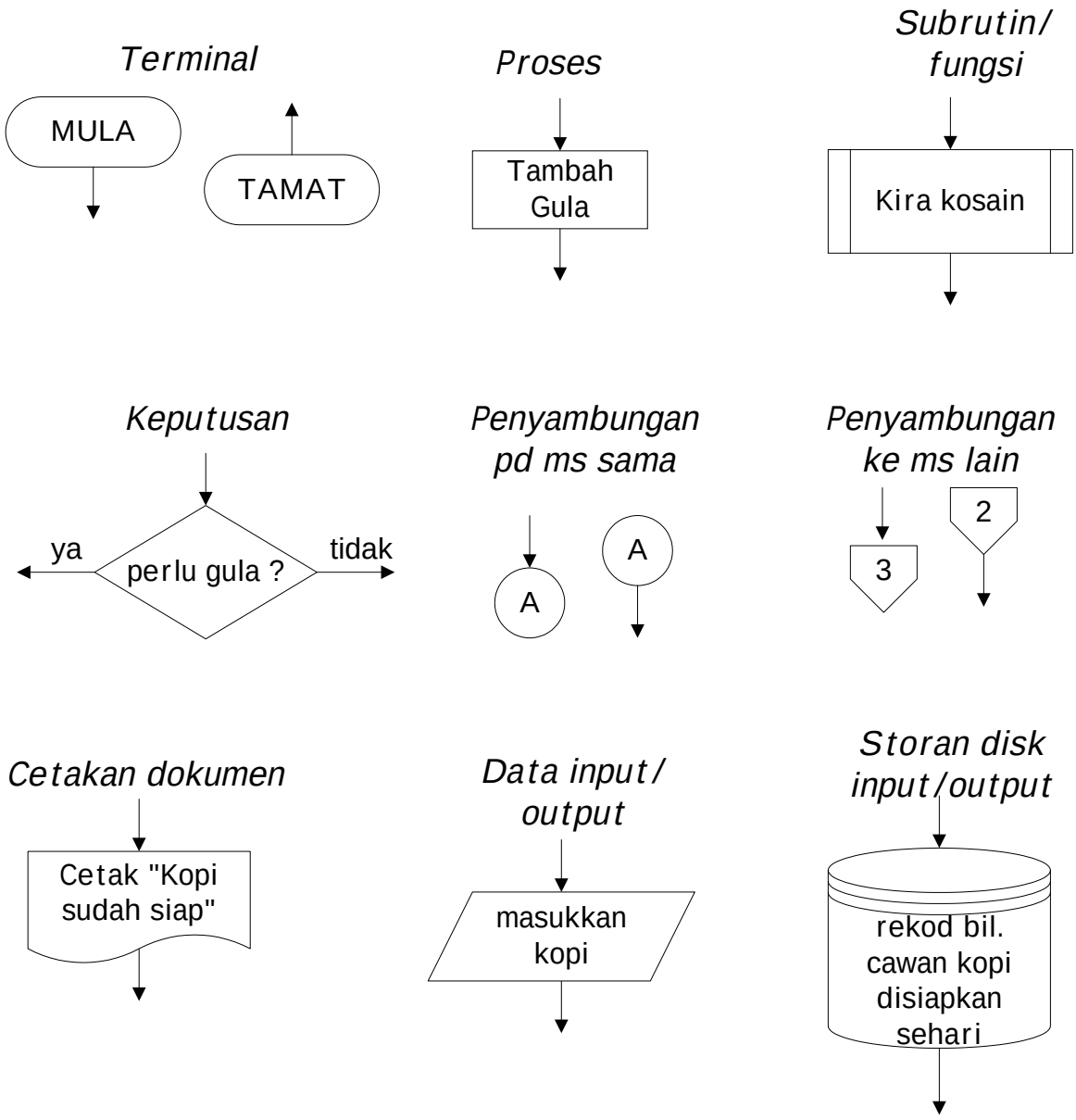
masak air ⟶

```
isi air ke dlm cerek
nyalakan api
repeat
    tunggu
until  air_mendidih
```

2.2.2 Kaedah carta alir

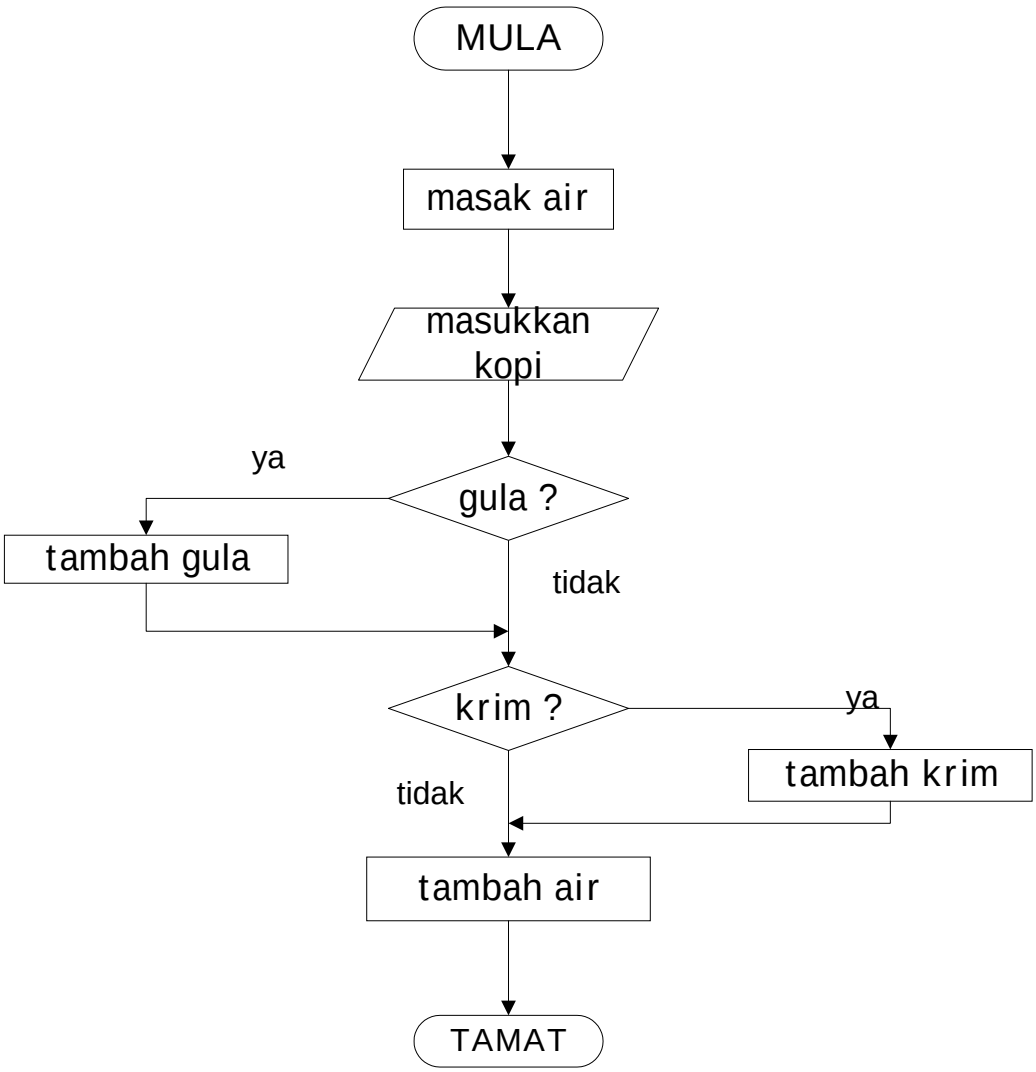
- Carta alir terdiri dari kotak-kotak berbagai bentuk yang menunjukkan operasi yg perlu dilakukan dan anak-panah[2] yg menyatakan jujukan operasi tersebut.

- Simbol carta alir:



2.2.2 Kaedah carta alir (samb...)

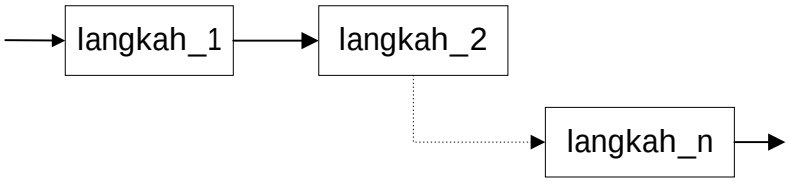
- Carta alir menyediakan secawan kopi



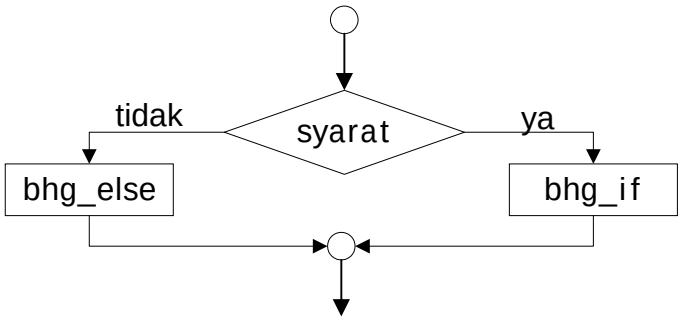
2.3 Carta Alir lwn. Kod Pseudo

Kod Pseudo	Carta Alir

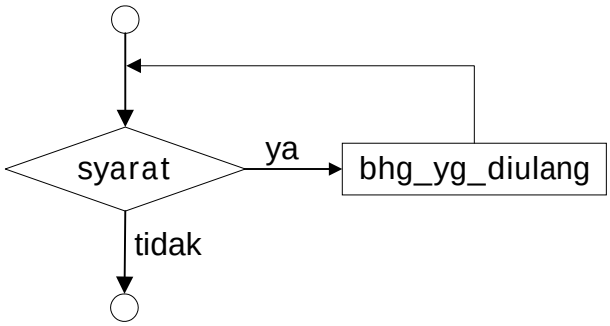
begin
langkah_1
langkah_2
. . . .
. . . .
langkah_n
end



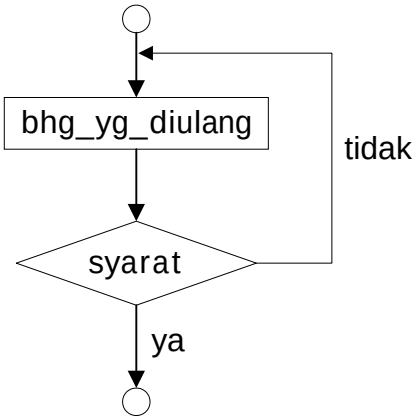
if syarat **then**
bhg_if
else
bhg_else
endif



while syarat **do**
bhg_yg_perlu diulang
endwhile



repeat
bhg_yg_perlu diulang
until syarat



2.4 Contoh PM & Algoritma

Tulis aturcara yg akan menerima input sepasang nombor & mencetak mesej "SAMA" jika dua nombor tersebut mempunyai nilai yg sama, mesej "NOMBOR PERTAMA LEBIH BESAR" jika nombor pertama lebih besar dpd nombor kedua & mesej "NOMBOR KEDUA LEBIH BESAR" jika nombor kedua lebih besar dpd nombor pertama.

Fasa: Spesifikasi keperluan

- Fasa : Analisa Masalah

  Input:
  1. media : papan kekunci
  2. data :
     nombor pertama (nom1) &
     nombor kedua (nom2)

  Output:
  1. media : skrin
  2. data :
     "SAMA" @
     "NOMBOR PERTAMA LEBIH BESAR" @
     "NOMBOR KEDUA LEBIH BESAR"

  Proses:
  1. bandingkan 2 nonbor & tentukan output

- Fasa : Rakabentuk

  i. Kod pseudo I

  1. Mula
  2. Baca nom1, nom2
  3. Jika nom1=nom2
     3.1. cetak mesej "SAMA"
     3.2. pergi ke langkah 6
  4. Jika nom1>nom2
     4.1. cetak mesej "NOMBOR PERTAMA LEBIH BESAR"
     4.2. pergi ke langkah 6
  5. Jika nom1<nom2
     5.1. cetak mesej "NOMBOR KEDUA LEBIH BESAR"
     5.2. pergi ke langkah 6
  6. Tamat

no. adalah penting!!

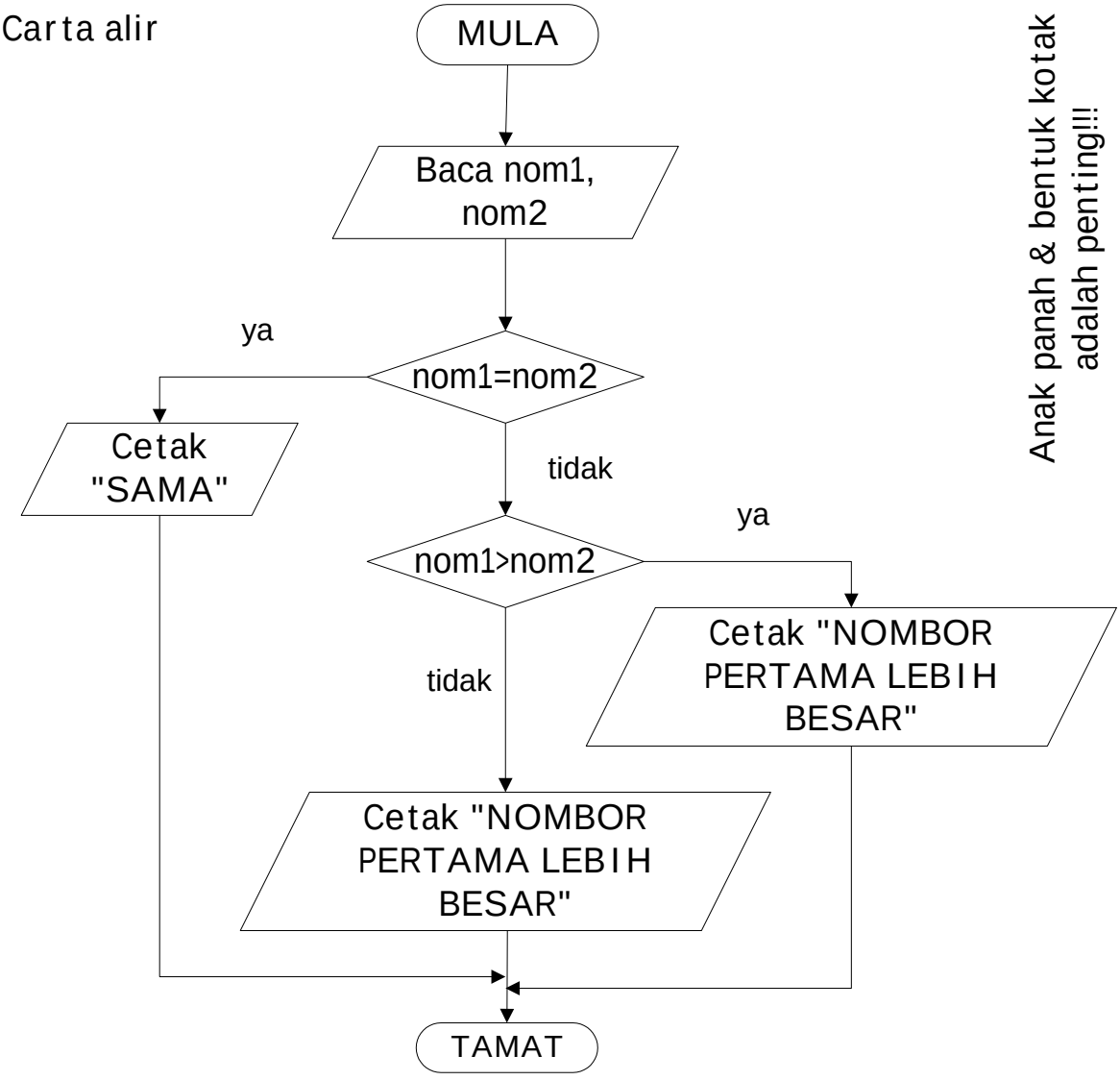
2.4 Contoh PM & Algoritma (samb....)

ii. Kod pseudo I I

```
begin
   Baca nom1, nom2
   if nom1=nom2 then
      cetak mesej "SAMA"
   else
      if nom1>nom2
         cetak mesej "NOMBOR PERTAMA LEBIH BESAR"
      else
         cetak mesej "NOMBOR KEDUA LEBIH BESAR"
      endif
   endif
end
```

Esotan (indent) & penggunaan kata-kunci adalah penting!!!

iii. Carta alir



Anak panah & bentuk kotak adalah penting!!!

2.5 Proses Menyediakan Aturcara

- Langkah[2] penyelesaian masalah menggunakan disiplin KP telah difahami dari bhg 2.1.
  Bagaimana hendak menyerapkan disiplin tersebut dalam proses penyedian aturcara.

- Kajian kes
  Walaupun contoh kajian kes ini mudah, ttp proses ini boleh digunakan utk masalah mudah atau rumit.

Langkah 1 : Definisi Masalah @ Spesifikasi Keperluan Masalah

Daripada jejari yg diberikan kira luas bulatan & ukurlilit bulatan tersebut.

Langkah 2 : Analisa Masalah

Input:
1. media: papan kekunci
2. data:
   radius - dimasukkan oleh pengguna
   π = 3.14159 pemalar

Output
1. media: skrin
2. data:
   luas bulatan
   ukurlilit bulatan

Proses:
formula 1: luas bulatan = πj^2
formula 2: ukurlilit bulatan = $2\pi j$
j-jejari bulatan

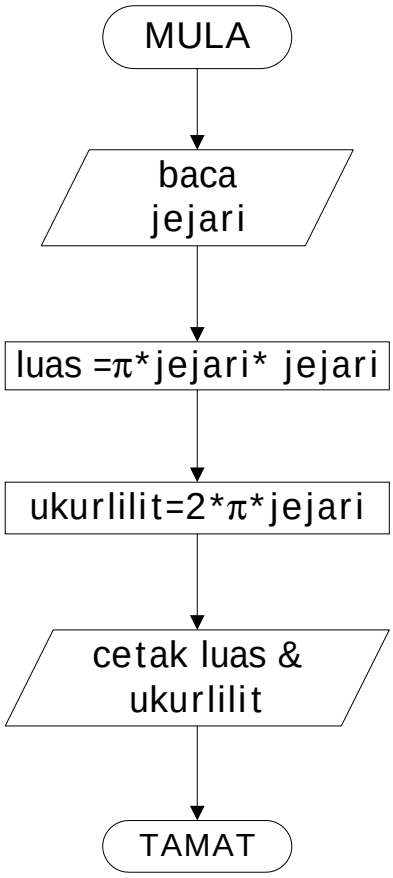
2.5 Proses Menyediakan Aturcara (samb ....)

Langkah 3 : Rekabentuk

Langkah 3.1
Mulakan rekabentuk awal dgn menggunakan kaedah kod pseudo kenyataan bahasa:

1. Baca jejari bulatan
2. Kira luas bulatan
3. Kira ukurlilit bulatan
4. Cetak luas dan ukurlilit bulatan

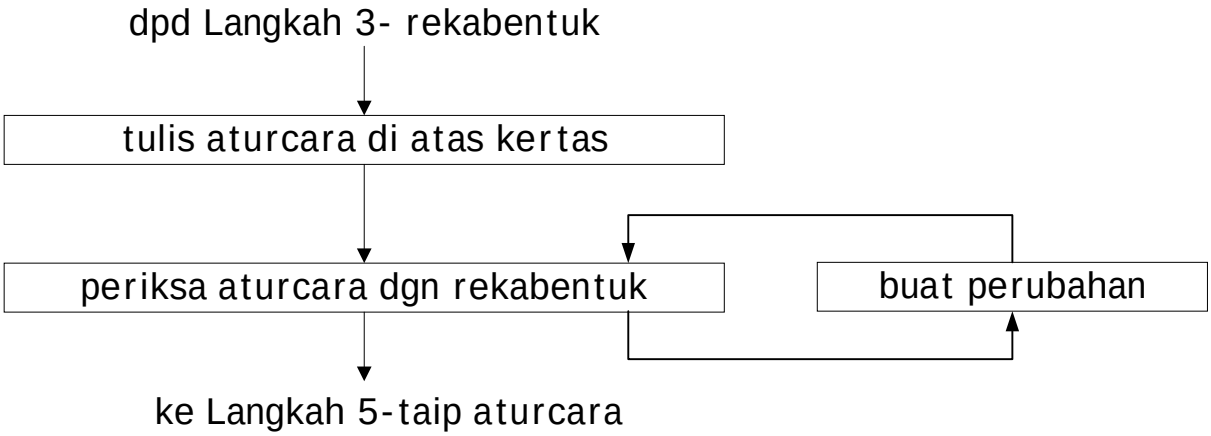
Langkah 3.2
Algoritma ini perlu dihalusi & dilukis menggunakan carta alir.



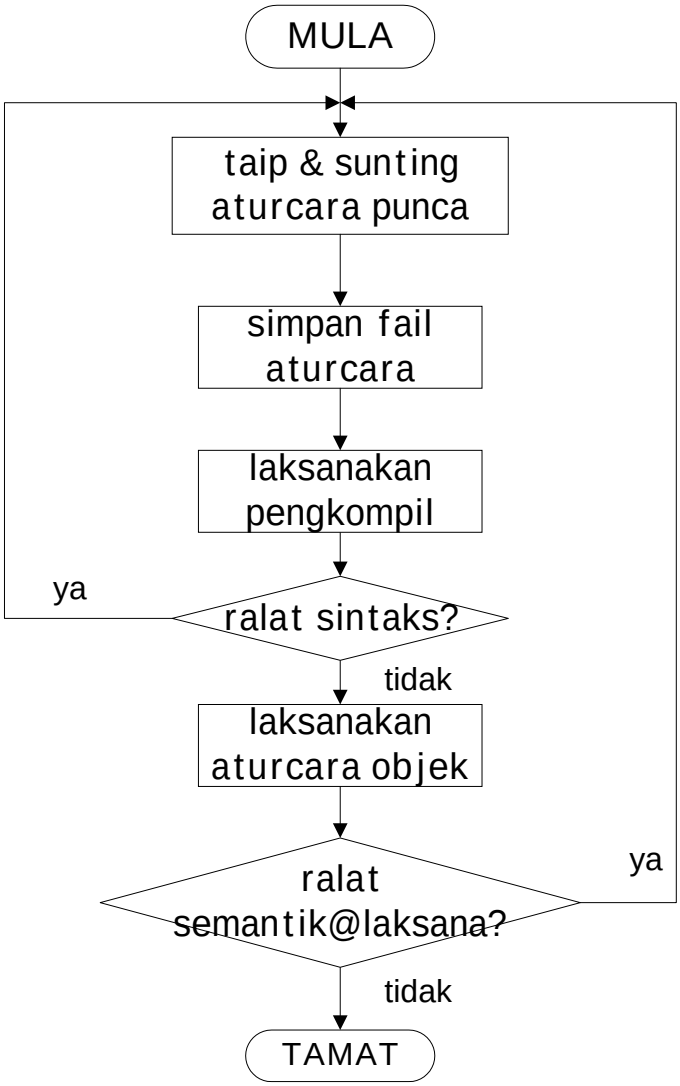
Langkah 3.3
Jika algoritma melibatkan struktur ulangan & pemilihan sila tulis kod pseudo menggunakan kenyataan bhs menyerupai bhs pengaturcaraan.

2.5 Proses Menyediakan Aturcara (samb ....)

Langkah 4 : Tulis Kod
Cadangan strategi penulisan kod :



Langkah 5 & **Langkah 6**: Taip Aturcara & Uji Aturcara
Carta alir taip aturcara & uji aturcara:



guna penyunting fail EDIT, WORD dsbnya

contoh : FAILSAYA.C

contoh : Turbo C, Borland C, Microsoft C dsbnya

C> FAILSAYA.EXE

Latihan:

1. Soalan ulangkaji.

a) Senaraikan fasa/langkah pembangunan perisian.

b) Algoritma bergantung kpd bhs pengaturcaraan. betul @ salah.

c) Fasa perlaksanaan bergantung kpd bhs pengaturcaraan. betul @ salah.

d) Senaraikan 4 kaedah pengungkapan algoritma.

2. Nyatakan pada fasa mana proses^2 berikut dilakukan:

a) Penyataan masalah diperolehi dari pengguna.

b) Kos pseudo ditulis untuk mengira luas segi tiga.

c) Setelah algoritma siap dibina, data contoh digunakan dgn memasukkan data ke dalam algoritma & memastikan output dari algoritma tersebut memberi nilai yg betul.

d) Mengenal pasti input sistem.

e) Menulis arahan^2 bahasa C utk PM.

f) Memastikan ralat masa-larian & ralat sintaks tiada pada kod.

g) Menulis dokumen^2 yg perlu dlm proses PM.

Latihan (samb...)

3. Rekakan algoritma dlm bentuk kod pseudo dan carta alir bagi menyelesaikan masalah[2] berikut:

   a) menentukan samada satu integer itu genap atau ganjil.

   b) mencari nilai rintangan setara bagi 4 perintang yg disambungkan selari.

   c) mencari nilai rintangan setara bagi M perintang yg disambungkan selari.

   d) mencari punca bagi persamaan kuadratic $Ax^2+Bx+C=0$.

4. Tuliskan hasil algoritma berikut dlm bentuk kod pseudo dan carta alir selepas penghalusan berlangkah dilakukan :

   a)
   ```
   if perlu_gula
       then tambahkan gula
   else
       jangan tambah gula
   endif
   ```

   b)
   ```
   isi air ke dlm cerek
   nyalakan api
   repeat
       tunggu
   until air_mendidih
   ```

3 PENGENALAN KEPADA C

3.0 Latar Belakang

- Asal C bermula daripada bahasa BCPL (Basic Combined Programming Language) oleh Martin Richards pada 1967
- BCPL diubah suai oleh Ken Thompson menjadi bahasa B
- B pula diubah suai ke C oleh Dennis Ritchie pada 1972
- Digelar bahasa pertengahan kerana mengabungkan ciri-ciri bahasa aras tinggi dan aras rendah, antara ciri^2 gabungan C
  - ☆ mempunyai kemudahan yang biasa disediakan oleh bahasa peringkat rendah contohnya pengolahan bit dan penggunaan pembolehubah daftar
  - ☆ mempunyai pembinaan kawalan dan jenis struktur data yang mana biasa terdapat dalam bahasa peringkat tinggi

- Kenapa C ?

  Kenapa C menjadi pilihan untuk menyelesaikan masalah saintifik dan masalah kejuruteraan ?

  - Mudah alih : C dipiawaikan (ANSI C), mudah dialihkan dari satu sistem ke sistem yg lain.
  - Kecekapan C tinggi berbanding bahasa lain.
  - Struktur C bermodular
  - Berkuasa utk mengungkapkan pengolahan data dan aritmetik.
  - Sintaks yg mudah
  - Fungsi^2 perpustakaan yg banyak utk menyokong pengaturcaraan.

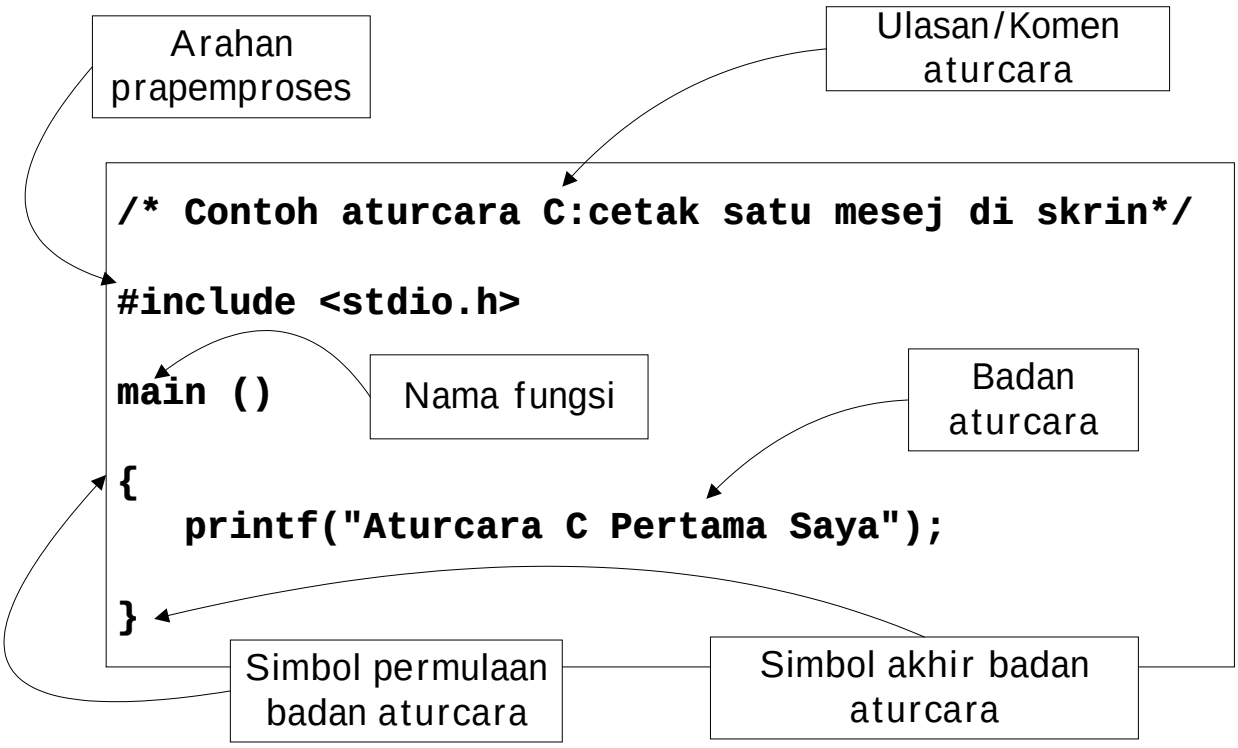
3.1 Contoh Aturcara C yang Mudah -
mencetak satu mesej di atas skrin komputer

```
/* Contoh aturcara C:cetak satu mesej di skrin*/

#include <stdio.h>

main ()
{
   printf("Aturcara C Pertama Saya");
}
```

- Keluaran pada skrin :

```
Aturcara C Pertama Saya
```

- Penjelasan aturcara di atas:



3.1 Contoh Aturcara C yang Mudah (samb.....)

◎ Komen/ulasan aturcara

1. Tujuan untuk menjelaskan maklumat mengenai aturcara.
2. Untuk dibaca oleh pembaca - diabaikan oleh komputer !
3. Bermula dengan /* dan diakhiri dengan */

◎ Arahan prapemproses (akan dibincangkan kemudian !)

1. Memasukkan kandungan fail lain.
2. Setiap arahan prapemproses dimulakan dengan simbol #

◎ Nama fungsi

1. Aturcara C terdiri dari satu atau lebih fungsi.

2. Fungsi utama dipanggil fungsi **main**.

3. Selain fungsi **main** kita bebas untuk memberi nama pada fungsi[2] lain. **printf** ialah contoh fungsi lain yg dipanggil oleh **main** utk mencetak mesej.

4. Tujuan ( ) bagi fungsi adalah untuk pemboleh ubah masukan dan keluaran sesuatu fungsi.

5. Perlaksanaan sesuatu aturcara bermula dari fungsi **main**.

◎ Badan aturcara/fungsi

Semua kenyataan/arahan sesuatu fungsi ditulis selepas simbol **{** dan ditamatkan dgn simbol **}**.

```
{
     kenyataan/arahan[2];
          :
}
```

Setiap kenyataan/arahan dlm badan fungsi **mesti** ditamatkan dgn semikolon (;).

3.2 Contoh Aturcara C dgn Pembolehubah -
mencetak satu mesej di atas skrin komputer

```
1: /* Aturcara C dgn pembolehubah*/
2: #include <stdio.h>
3:
4: main ( )
5: {
6:    int umur;
7:    umur = 25;
8:    printf("Aturcara C Pertama Saya\n");
9:    printf("Selepas umur saya %d tahun", umur);
10: }
```

Perhatikan ! 1:, 2: -- 10: **bukan** sebahagian dari aturcara. Ia dimasukkan utk menunjukan baris[2] dlm aturcara.

- Keluaran pada skrin :

```
Aturcara C Pertama Saya
Selepas umur saya 25 tahun
```

- Penjelasan aturcara

Baris 1-2: bhg ulasan & prapemproses - telah dibincangkan
Baris 3: kosong ! - guna utk menyenangkan pembacaan aturcara
Baris 4-10: mentakrifkan satu fungsi bernama **main** yg terdiri dari 4 kenyataan/arahan di baris 6-9.
Baris 6: mengisytiharkan satu pembolehubah bernama **umur** yg boleh menyimpan nilai **int**eger shj.
Baris 7: memberikan nilai 25 kpd pembolehubah **umur** - iaitu **umur diumpukkan** (assigned) dgn nilai 25.
Baris 8: panggil fungsi **printf** utk cetak mesej dlm "...". Simbol **\n** mengarahkan supaya mesej yg seterusnya dicetak bermula di baris yg baru.
Baris 9: cetak mesej dlm "..." dan nilai pembolehubah **umur**. Simbol **%d** menyatakan jenis dan format utk mencetak nilai pembolehubah **umur**.
Baris 5&10: memulakan & menamatkan takrifan fungsi **main**.

3.3 Aturcara C Yg Jelas di Baca

C adalah bahasa 'Gaya bebas'. Contoh aturcara yg sah:

Format 1:

```
#include <stdio.h>
main ( )
{
    int umur;
    umur = 125;
    printf("umur saya %d tahun", umur);
}
```

Format 2:

```
#include <stdio.h> main ( ) {    int umur;
    umur = 125;  printf("umur saya %d
tahun", umur); }
```

Format 3:

```
#include <stdio.h>
main ( ) { int
         umur;
        umur
      = 125;
     printf("
            umur saya %d tahun", umur
           );
 }
```

Format 4:

```
#include <stdio.h>
main ( ) {

    int umur;
    umur = 125;
    printf("umur saya %d tahun", umur);
}
```

Format 1 & 4 mudah dibaca -- **digalakkan** penggunaannya !

Latihan:

1. Tulis aturcara C untuk menyelesaikan maslah berikut:

   a) cetak mesej - Hurey ! Aturcara C saya.

   b) cetak butir^2 diri anda.
   nama
   umur - *simpan dlm pembolehubah*
   alamat

   c) cetak mesej
   20hb Mac Tahun **2005**
   simpan maklumat yg dihitamkan dlm pembolehubah.

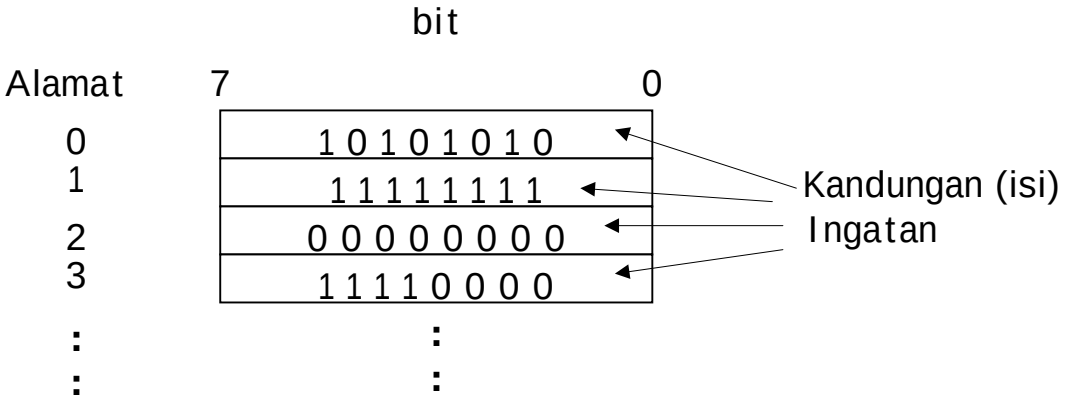
2. Tentukan ralat dlm aturcara berikut:

```
#include <stdio.h>
/* contoh aturcara **/
main ( )
{
    int umur
    umur = 125;
    printf "umur= %d tahun", umur);
```

3.4 Ingatan Dan Simpanan Data Dlm Komputer

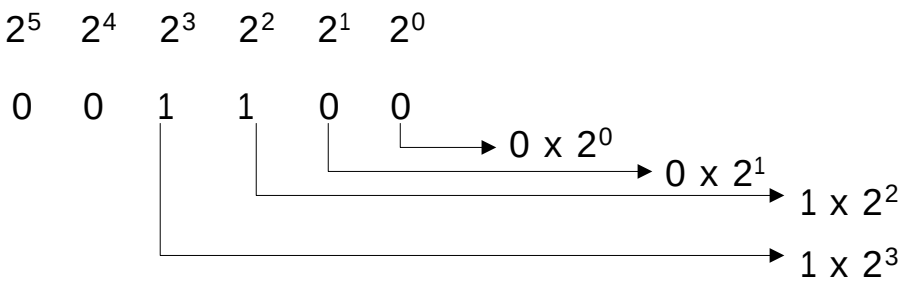
- Oleh kerana komputer adalah sistem digital, unit asas yg boleh disimpan dlm ingatan ialah nombor perduaan '0' @ '1'.
  Satu digit perduaan dipanggil **bit**.
  8 bit = 1 **bait** (byte)
  2 bait = 1 **kata** (word)
  2 kata = 1 **kata panjang** (long word)

- Ingatan dlm komputer terdiri dari senarai bait[2] (bytes).
  Setiap bait mempunyai satu alamat.



- Nilai max yg boleh disimpan di satu alamat ialah 255_{10} (11111111_2).
  Nilai min " " 0_{10} (00000000_2)

- Penukaran perduaan ke desimal



$$001100_2 = 0 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 1 \times 2^3 + 0 \times 2^4 + 0 \times 2^5$$

$$= 12_{10}$$

3.4 Ingatan Dan Simpanan Data Dlm Komputer (samb...)

- Menulis nilai integer besar dlm btk perduaan memerlukan bil. bit yg banyak - mudah membuat kesilapan.

  Nilai integer besar lebih ringkas jika ditulis dlm btk perenambelasan (heksadesimal -Hex).

- Jadual penukaran Hex-Binari-Desimal

Simbol Hex	Binari	Desimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

$4FC_{16} = 4 \times 16^2 + 15 \times 16^1 + 12 \times 16^0 = 1276_{10}$

$= 0100\ 1111\ 1100_2$

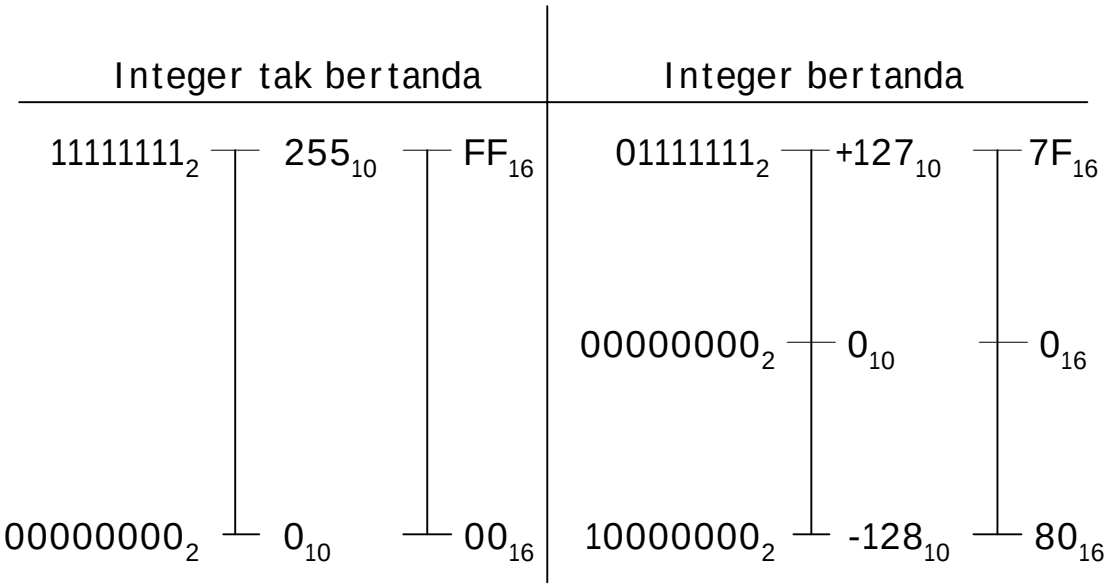
3.4 Ingatan Dan Simpanan Data Dlm Komputer (samb...)

- **Integer tak bertanda** (unsigned integer) - integer dgn nilai positif shj.

  Integer bertanda (signed integer) - integer dgn nilai positif dan negatif. Bit paling kiri (MSB) digunakan sbg tanda.

  MSB = 1 : integer negatif
  MSB = 0 : integer positif

  Contoh: dgn perwakilan 8 bit



- **Aksara** (character)
  Simbol bagi angka (0..9), abjad (A..Z) & simbol istimewa spt @ # % dsbnya diwakili sbg integer 8 bit dlm komputer. Nilai simbol[2] ini disimpan sbg Kod ASCII (American Standard Code for Information Interchange) di dlm ingatan.
  Contoh: Kod ASCII bagi 'A' = 65
  Kod ASCII bagi '$' = 36

- **Integer besar.**
  Utk menyimpan nilai[2] lebih besar dari 8 bit, lebih dari satu ingatan perlu digunakan.

Saiz ingatan	Nilai integer
16 bit	0 ... 65535
	-32768 .. +32767
32 bit	0 ... 4,294,967,295
	-2,147,483,648 ... +2,147,483,647

Latihan:

1. Tukar nombor perduaan berikut ke hex dan desimal:

   a) 10001110_2 b) 10101010_2 c) 00100111_2

2. Tukar nombor hex berikut ke desimal dan binari:

   a) 25 c) 1FF b) E6

3.6 Pencam (identifiers)

- Dlm cth aturcara C yg lepas, fungsi & pembolehubah diberi nama simbolik spt **printf** & **umur**.
  Nama simbolik ini dipanggil **pencam** bagi fungsi & pembolehubah tersebut.
- Dlm C pencam digunakan sbg nama rasmi bagi : fungsi, pembolehubah, pemalar dan label.
- Pencam dlm C mesti memenuhi syarat^2 berikut:
  - bermula dgn abjad @ aksara 'underscore' (_)
  - terdiri dari abjad (A...Z, a...z), digit (0...9) @ aksara (_) shj
  - saiz max 30/1/2 aksara
  - bukan katakunci^2 C. Katakunci^2 C :

auto	**extern**	**sizeof**	**#define**
break	**float**	**static**	**#include**
case	**for**	**struct**	
char	**goto**	**switch**	
const	**if**	**typedef**	
continue	**int**	**union**	
default	**long**	**unsigned**	
do	**register**	**void**	
double	**return**	**volatile**	
else	**short**	**while**	
enum	**signed**		

contoh^2 pencam:

Sah	**Tak sah**
`utm`	`2utm`
`_bek`	`meow?`
`rekod_201`	`rekod-301`
`ifi`	`if`

- C membezakan diantara pencam huruf besar atau pencam huruf kecil @ gabungan huruf besar & huruf kecil.

  contoh 4 pencam yg berbeza:
  ba BA Ba bA

- Apakah perbezaan 2 pencam berikut:

  dayang_norhayati_binti_abang_jawawi
  dayang_norhayati_binti_abang_jawawi2

3.6 Pencam (samb....)

- Pencam terdiri dpd:

  i. Pencam Piawai
  ii. Pencam Pengguna

i. Pencam Piawai

- Nama pencam yg digunakan didlm perpustakaan C.
  Mempunyai maksud tertentu spt : printf & scanf

- Pencam ini boleh ditakrifkan semula ttp fungsi asal pencam tersebut tidak ditukar semula, cth aturcara:

```
int printf;    /*printf menyimpan nilai integer*/
printf("Cuba pasti salah\n");
```

ii. Pencam Takrifan Pengguna

- Nama pencam yg direka oleh pengguna utk 3 tujuan penakrifan:
  1. Pembolehubah
  2. Pemalar
  3. Fungsi

- Pembolehubah -pencam digunakan utk menyimpan nilai.
  -Format pengisytiharan :

```
jenis_data   nama_pembolehubah;
```

- cth:

```
int umur;
double jejari, luas,  ukurlilit;
```

3.6 Pencam (samb....)

- Pemalar -pencam digunakan utk menyimpan nilai yg sama sepanjang masa.

  -2 cara pengisytiharan pemalar :

```
const jenis_data nama_pemalar = nilai_pemalar;
```

@

```
#define nama_pemalar nilai_pemalar
```

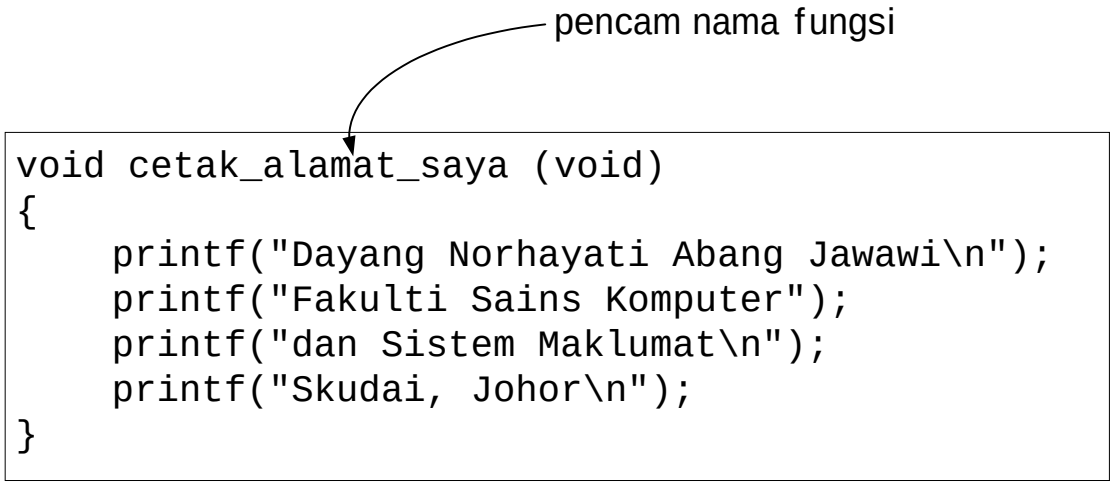
- cth pemalar π:

```
const double PI = 3.1459;
                @
#define PI 3.1459
```

-jika satu nilai pemalar diisytiharkan & pengguna cuba mengumpukkan nilai lain kpd pemalar - RALAT Sintaks!

- Nama Fungsi -satu kumpulan arahan yg ditulis oleh pengguna utk tujuan tertentu & setiap fungsi mempunyai nama yg unik.

  -cth :



-Fungsi akan dipelajari dgn terperinci dlm bab dihadapan ....Tunggu....

3.7 Jenis-jenis Data

- Dlm aturcara mudah yg lepas kenyataan :

```
int umur;
```

katakunci **int** mengisytiharkan pembolehubah **umur** untuk menyimpan data jenis **integer** shj.

- Dlm C katakunci[2] berikut digunakan utk mengisytiharkan jenis data yg. boleh disimpan oleh sesuatu pembolehubah :

katakunci (jenis data)	saiz dlm bait	julat no yg disimpan
char	1	-128 - 127
int	2	-32768 - 32767
short	2	-32768 - 32767
long	4	-2147483648 - 2147483647
float	4	$\pm 1.18 \times 10^{-38}$ - $\pm 3.4 \times 10^{38}$
double	8	$\pm 9.46 \times 10^{-308}$ - $\pm 1.79 \times 10^{308}$

- Perhatikan :
  1. **char, int, short** & **long** adalah utk menyimpan **nombor integer** (.. -2, -1,0,1,2,3....)
  2. **float** & **double** adalah utk menyimpan **nombor nyata** (-2.1, 1.1 dsb)

- Utk membezakan nombor integer **tak bertanda** (positif shj) & nombor **bertanda** (negatif & positif) C menggunakan katakunci **unsigned** bagi nombor[2] integer tak bertanda.

katakunci (jenis data)	julat no yg disimpan
unsigned char	0 - 255
char	-128 - 127
unsigned int	0 - 65535
int	-32768 - 32767
unsigned long	0 - 4294967295
long	-2147483648 - 2147483647

3.7 Jenis-jenis Data (samb....)

- **char** biasanya digunakan utk menyimpan kod[2] ASCII bagi aksara spt 'a', 'M', '#', '!', '%' .....

- Cth mengisytiharkan pembolehubah dlm C :

```
unsigned char kod;
char alfa, c_kod, no;
int hari, tahun;
unsigned int luas, isipadu;
long int bil_pelajar;
long bilBiri_biri;
float gaji_pm;
double luas_planet;
```

- Pembolehubah boleh diberikan nilai awalnya semasa ia diisytiharkan :

```
int hari = 28, tahun = 201;
float gaji_pm = 654321.25;
```

- Nilai[2] pemalar (constant) dlm C boleh ditulis spt berikut :

Jenis Pemalar	Contoh nilai pemalar
Perpuluhan (desimal)	65
Perenambelasan (heksadesimal)	0x41, 0X41
Perlapanan (oktal)	0101
Aksara	'A'
Perpuluhan panjang	65L
Nombor nyata	65.0, 65., 65e0, 650.E-1

- Cthnya aksara 'A' boleh ditulis dlm berlainan format spt berikut :

	Pemalar aksara	Desimal	Hex	Oktal
Format	'A'	65	0x41	0101
Format dlm ingatan (binari)	01000001	01000001	01000001	01000001

3.7 Jenis-jenis Data (samb....)

- Dlm C terdapat beberapa pemalar aksara istimewa yg mempunyai maksud tertentu.

Pemalar Aksara	Maksud
'\n'	newline - pergi ke baris seterusnya pd lajur 1
'\t'	horizotal tab - anjak ke kanan
'\v'	vertical tab - anjak ke bawah pd lajur yg sama
'\r'	carriage return - baris seterusnya
'\x41'	nombor hexa 0x41
'\101'	nombor oktal 101
'\0'	null - aksara nol digunakan utk menandakan penghujung rentetan aksara (string)
'\''	tanda (')
'\"'	tanda (")
'\\'	tanda (\)
'\b'	backspace - undur ruang
'\f'	formfeed - halaman seterusnya (utk pencetak)
'\a'	alert - membunyikan loceng

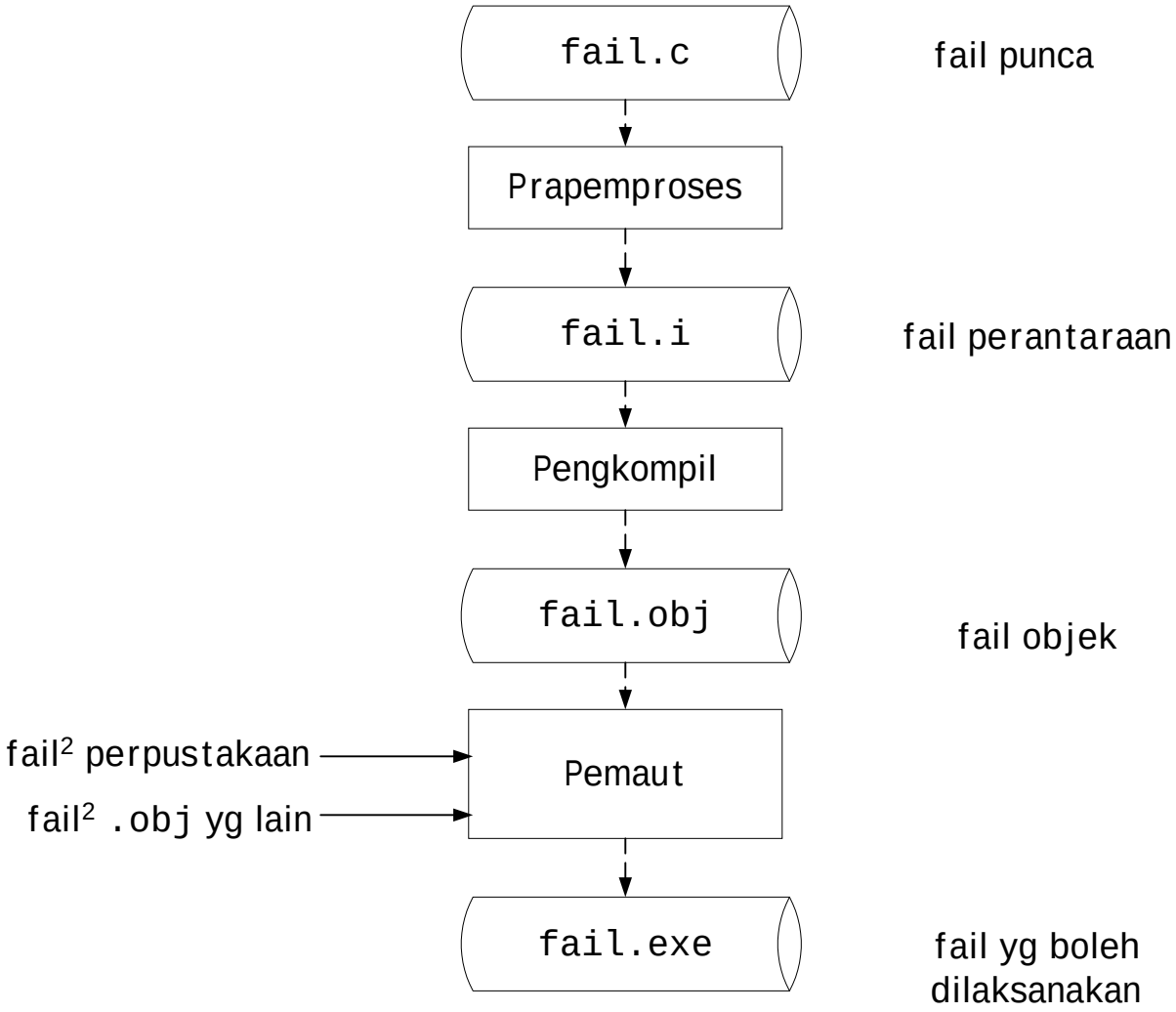
- Pembolehubah jenis **`const` tidak boleh diubah** nilainya oleh aturcara. Ia biasanya digunakan utk melindungi pembolehubah yg dihantar kpd sesuatu fungsi drpd diubah nilainya.

Contoh aturcara menggunakan pembolehubah & pemalar

```
/* Contoh aturcara kira luas segiempat & bulatan */
/* Penulis : Rosbi Mamat Tarikh : 20.02.2020 */
#include <stdio.h>

main ()
{
   const float pi = 3.1415;
   int panjang, lebar, luas_segi4;
   float jejari, luas_bulat;

   panjang = 12; lebar = 3; luas_segi4 = panjang * lebar;
   jejari = 3.5; luas_bulat = pi * jejari * jejari;

   printf("Luas bulatan = %f\t
          Luas segi4 = %d\n", luas_bulat, luas_segi4 );
   printf("\a Luas bulatan = %f\r
          Luas segi4 = %d\n", luas_bulat, luas_segi4 );
}
```

3.8 Arahan Prapemproses

- Arahan C prapemproses akan diproses dahulu sebelum dikompil. Perhatikan proses yg dilalui oleh fail punca/sumber sebelum boleh dilaksanakan.



- Format panggilan kepada prapemproses :

```
#jenis_prapemproses arahan²_berkaitan
```

- 2 prapemproses yg biasa digunakan `#define` & `#include`

3.8 Arahan Prapemproses (samb....)

i. Arahan prapemproses #define

-untuk mengisytiharkan pemalar simbolik

- Format panggilan prapemproses #define rujuk 3.6.

ii. Arahan prapemproses #include

- digunakan untuk menggabungkan fail luaran ke dalam satu aturcara sumber/punca dgn memanggil fail pengepala (namafail.h), 2 jenis fail kepala:
  - piawai - utk mencapai perpustakaan C
  - takrifan pengguna - utk mencapai aturcara punca lain dlm fail yg berlaian yg ditulis oleh pengguna.

- Format panggilan prapemproses #include piawai :

```
#include <nama_pengepala>
```

- Format panggilan prapemproses #include takrifan pengguna :

```
#include "nama_pengepala"
```

- Cth aturcara dgn #include piawai & #define:

```
#include <stdio.h>
#define PI 3.14159
#define NAMAPENULIS "Dayang Norhayati"

int main(void)
{
   double jejari,luas,ukurlilit;

   printf("Penulis aturcara : %s\n", NAMAPENULIS);

   jejari = 1.0;
   luas = PI * jejari * jejari;
   ukurlilit = 2 * PI * jejari;

   printf("Luas bulatan ialah %f\n", luas);
   printf("Ukurlilit bulatan ialah %f", ukurlilit);

   return 0;
}
```

3.8 Arahan Prapemproses (samb....)

- output aturara:

```
Penulis aturcara : Dayang Norhayati
Luas bulatan ialah 3.141590
Ukurlilit bulatan ialah 6.283180
```

- Cth aturcara dgn `#include` takrifan pengguna & `#define`:

```
/* Nama Fail   : bulat.c*/

#include "bulat.h"

int main(void)
{
   printf("Penulis aturcara: %s\n", NAMAPENULIS);

   jejari = 1.0;
   luas = PI * jejari * jejari;
   ukurlilit = 2 * PI * jejari;

   printf("Luas bulatan ialah %f\n", luas);
   printf("Ukurlilit bulatan ialah %f", ukurlilit);

   return 0;
}
```

```
/* Nama Fail   : bulat.h*/

#include <stdio.h>
#define PI 3.14159
#define NAMAPENULIS "Dayang Norhayati"

double jejari,luas,ukurlilit;
```

3.3 Latihan

1. Tentukan jenis data yg akan digunakan utk mewakili data^2 berikut:

   a) jumlah penduduk Johor Bahru
   b) berat sebiji kelapa
   c) bilangan pelajar UTM yang berbasikal
   d) jumlah import Malaysia sepanjang 5 tahun yg lepas
   e) warna kereta

2. Apakah keluaran pd skrin jika suruhan^2 berikut ditambah dlm aturcara dlm m/s 16:

   a) **printf("bulatan\b\bin \'\"");**

   b) **panjang= '\0x64'; lebar = 'A'; luas_segi4=panjang*lebar;**
      printf("%d", luas_segi4);

3. Cari 7 kesilapan dlm aturcara berikut:

```
KiraHutang ( )
{
    int nama_yang_tersangat_panjang_jenis_int;
    float nama_yang_tersangat_panjang_jenis_float;
    const float kadar = 25.23, goto=1.3;
    float pinjambank, pinjamkawan, samanpolis,hutang;
    char kod;
    int bil_guli = 5.0;

    Hutang= pinjambank*kadar+pinjambak+pinjamkawan+samanpolis;
    kadar=20.1; nama_yang_tersangat_panjang_jenis_int =80000;
    kod = 66;
}
```

4 OPERATOR DAN UNGKAPAN

4.1 Operator

- Data yg disimpan di dlm ingatan (lihat bab 3) boleh diolah dgn menggunakan **operator**. C mempunyai 45 operator utk mengolah berbagai jenis data.

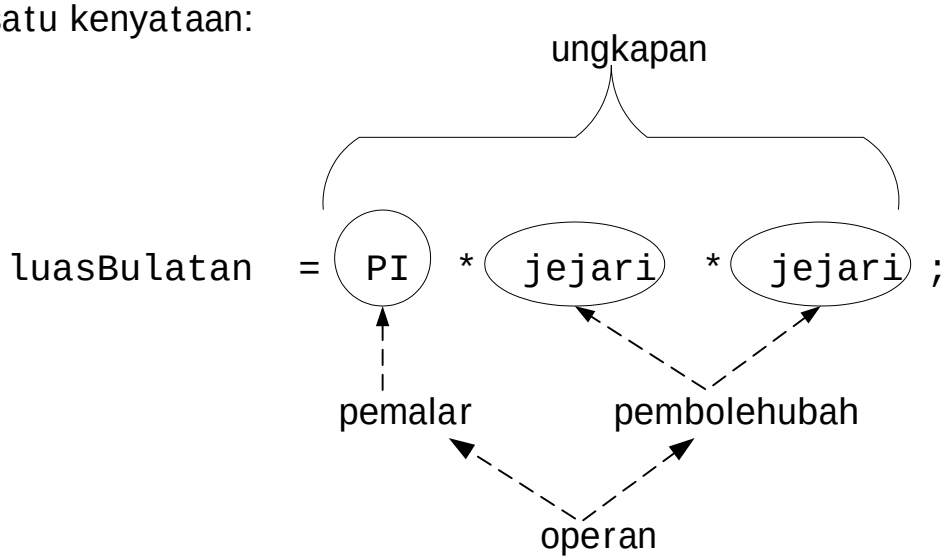
- Beberapa operator telah diperkenalkan sebelum ini:

+	operator tambah
-	operator tolak
=	operator umpukan

- Operator digunakan utk menggabungkan operan (pembolehubah@pemalar) menjadi ungkapan. Ungkapan yg ditamatkan dgn ; menjadi kenyataan.:

ungkapan `gaji + elaun - hutang`
kenyataan `pendapatan = gaji + elaun - hutang;`

- Cth satu kenyataan:



- Operator terbahagi kepada:-
  i) operator aritmetik
  ii) operator hubungan
  iii) operator logik

- Dlm bab ini kita akan belajar fungsi operator² tertentu & syarat² penggunaannya.

4.2 Operator aritmetik

- Operator aritmetik dalam C terbahagi kepada 2:-
  i) unari ii) binari

4.2.1 Operator Aritmetik unari

Operator unari dikenakan kpd **satu** pembolehubah shj.

Operator	Fungsi
+	operator positif
-	operator negatif
++	operator tokok
--	operator susut
!	operator TIDAK/NOT
~	operator salingan/pelengkap satu
&	operator alamat bagi sesuatu pembolehubah
*	operator kandungan sesuatu alamat
sizeof	saiz sesuatu pembolehubah dlm bait

```
main()
{
     int x, c;

     c = -64;
     x = +c;              /* x = -64 */
     x = -c;              /* x =  64 */
     x = ++c;             /* x =  65 */
     x = --c;             /* x =  64 */
     c = sizeof x;        /* c =   2 */
     c = ~c;              /* c = 253 */
}
```

Jika operator tokok dan susut dikenakan **selepas** (posfix)satu pembolehubah, nilai pembolehubah tersebut yg asal akan digunakan **dan selepas** itu baru nilai pembolehubah tersebut ditokok @ disusut.

```
main()
{
     int x,xcc,xtt;
     x = 24;
     xcc = x++;     xtt = ++x;
     printf( "x= %d, xcc= %d , xtt= %d", x, xcc, xtt);
}
```

Keluaran skrin:

x= 26, xcc= 24, xtt= 26

4.2.2 Operator Aritmetik binari

- operator yang terletak di antara dua operan.
- Simbol pewakilan operator binari:

+	operator tambah
-	operator tolak
*	operator darab
/	operator bahagi
%	operator hasil bahagi @ modulus (integer shj.)

Contoh:

```
main()
{
     int x, y, z;

     x =10; y = 17;
     z = x + y;     /* z diumpukkan dgn. nilai 27 */
     y = y - x;     /* y diumpukkan dgn. nilai 7 */
     x = y * z;     /* x diumpukkan dgn. nilai 189 */
     z = x / 20;    /* z diumpukkan dgn. nilai 9 */
     y = z % x;     /* y diumpukkan dgn. nilai 9 */
}
```

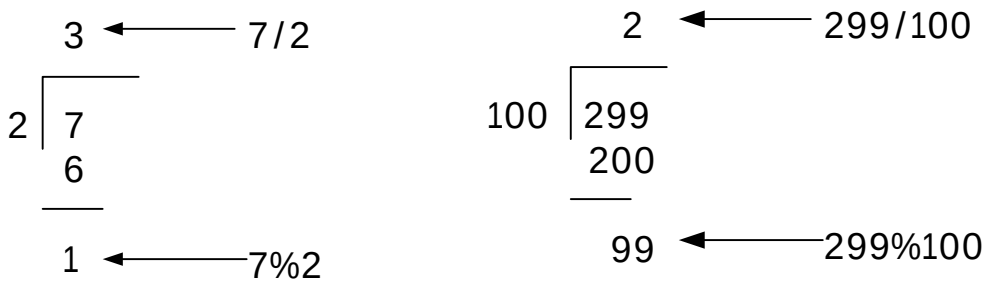
i) Operator / & %

- / - Melakukan operasi bahagi ttp. bila digunakan dgn pembolehubah integer ia akan memberikan hasil bhg integer(no. bulat).
  Contoh:
  7.0/2.0 hasil 3.5
  7.0/2 hasil 3.5
  7/2.0 hasil 3.5
  7/2 hasil 3

- %- Memberi nilai baki bg pembahagian 2 no. HANYA boleh digunakan bersama nilai atau operan berjenis integer
  Contoh:
  7%2 hasil 1

4.2.2 Operator Aritmetik binari (samb....)

Contoh operasi / & % :



ii) Penggunaan Jenis Data Pelbagai dlm Ungkapan Aritmetik binari

- Dlm satu unkapan aritmetik binari, min. 2 operan & 1 operator

Format:

$operan_1$ $operator_1$ $operan_2$ ....$operator_{n-1}$ $operan_n$

- Operan `int` & `double`
  jika kesemua operan adalah `integer` nilai terhasil adalah `integer`, jika tidak, akan menghasilkan `double`.

Contoh:

```
main()
{
     int i1=3, i2=2, hasil1, hasil2;
     double d=2.0, hasil3, hasil4;

     hasil1 = i1/i2;     /* 3/2   */
     hasil2 = i1/d;      /* 3/2.0 */
     hasil3 = i1/i2;     /* 3/2   */
     hasil4 = i1/d;      /* 3/2.0 */
}
```

hasil1	hasil2	hasil3	hasil4
1	1	1.0	1.5

4.2.2 Operator Aritmetik binari (samb....)

iii) Cast

- Dari contoh berlaku pertukaran jenis data dlm satu ungkapan - pertukaran jenis data secara automatik.
  Pertukaran jenis data boleh diarahkan mengunakan operasi casting.

Format cast:

```
(jenis data) pembolehubah@ungkapan
```

Contoh i:

```
double a=3.0, b=2.0, hasil;
hasil = a % b; /*Ralat Sintaks!!!*/
```

Penyelesaian:

```
hasil = (int)a % (int)b; /*Bebas Ralat*/
```

Contoh 2:

```
main()
{
     int jum_markah = 456, bil_pelajar = 5;
     double purata_markah1, purata_markah2;

     purata_markah1=jum_markah/bil_pelajar;
     purata_markah2=(double)jum_markah / bil_pelajar;
}
```

purata_markah1	purata_markah2
91.0	91.2

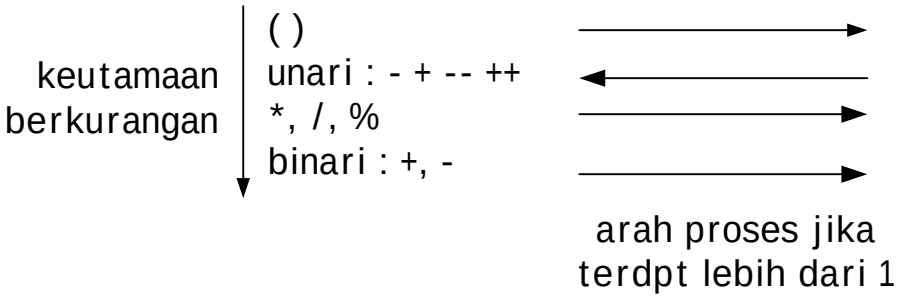
4.3 Kenyataan dgn pelbagai operator

- Setakat ini kita membincangkan ungkapan menggunakan satu operator pd satu kenyataan.
  contoh : `x= -y; x = y + z;`

- Jika pelbagai operator dlm satu ungkapan spt:

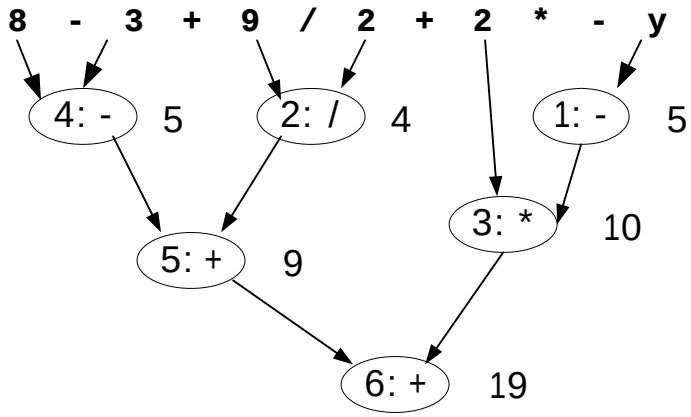
```
x= -y + z * 7 % 2 /++z;
```

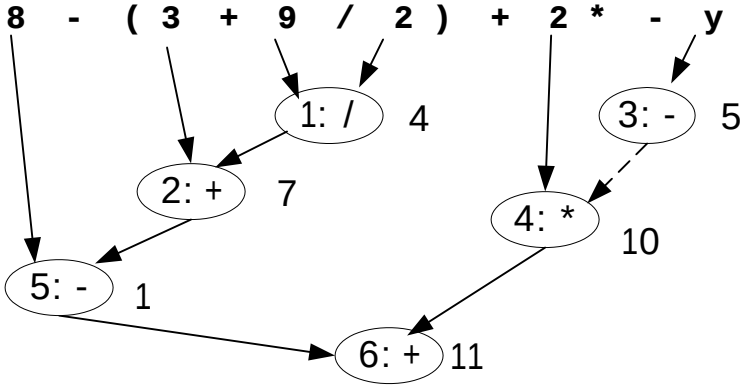
Perlukan syarat keutamaan utk memproses aritmetik tersebut.
Kadar Keutamaan operator unari adalah seperti berikut:



Contoh:

```
int z, y=-5;
z= 8 - 3 + 9 / 2 + 2 * - y;
z= 8 - (3 + 9 / 2) + 2 * - y;
```





4.3 Ungkapan dgn pelbagai operator (samb....)

- Mengetahui syarat keutamaan utk memproses aritmetik membantu kita memahami bgm C memproses kenyataan. TTp utk menghasilkan kod aturcara yang mudah dibaca penggunaan ( ) amat digalakkan.
  contoh `x*y*z+y/b-c`
  lebih mudah difahan jika `x*((y*z)+y)/(b-c)`

Contoh formula math & kenyataan C:

Formula	kenyataan C
$b^2 - 4ac$	`b * b - 4 * a * c`
$\frac{a+b}{c+d}$	`(a + b) / (c + d)`
$\frac{1}{1+x^2}$	`1/ (1 + x * x)`

4.4 Penyataan Umpukan =

- pernyataan umpukan menggunakan simbol =. Operator umpukan ini juga boleh digunakan dalam 2 cara:
  i) pernyataan umpukan berganda
  ii) pernyataan umpukan kompaun

i) pernyataan umpukan berganda

- Terdapat lebih dari satu operator (=) dalam sesuatu pernyataan umpukan berganda.

Contoh 1:

```
int a=b=c=d=e=250;
```

Contoh 2:

```
int b=2, nombor=0, jum=0, purata=3;
nombor= b = 10 + (jum=10-purata);
printf("\n%d %d %d %d", b , nombor, jum, purata);
```

Output:

```
17 17 7 3
```

4.4 Penyataan Umpukan = (samb....)

ii) pernyataan umpukan kompaun

- Kenyataan yg memudahkan kenyataan berikut:

```
pembolehubah = pembolehubah operator ungkapan;
```

kepada:

```
pembolehubah operator= ungkapan;
```

- Gabungan beberapa operator yang dibenarkan adalah seperti berikut:

+=
%=
-=
/=
***=**

Contoh:

```
masa = masa -1;          /*sama*/      masa -= 1;
bil = bil + 1;           /*sama*/      bil += 1;
jum_ti = jum_ti + ti;    /*sama*/      jum_ti += ti;
n = n * (x + 1);         /*sama*/      n *= x + 1;
```

4.5 Fungsi Perpustakaan Matematik

- Setakat ini kita mengenali operator math yg mudah spt +, -, *, / dan %. Ttp ini tidak cukup utk menyelesaikan masalah spt:

$\sqrt{x}$ $\quad$ $|q+z|$ $\quad$ $\left[\frac{h}{12.3}\right]^3$

- Rutin/fungsi utk sesetengah pengiraan penting ini telah disediakan C. Fungsi ini boleh dipanggil bersama dengan arahan prapemproses #include. Contoh 2 panggilan perpustakaan matematik:

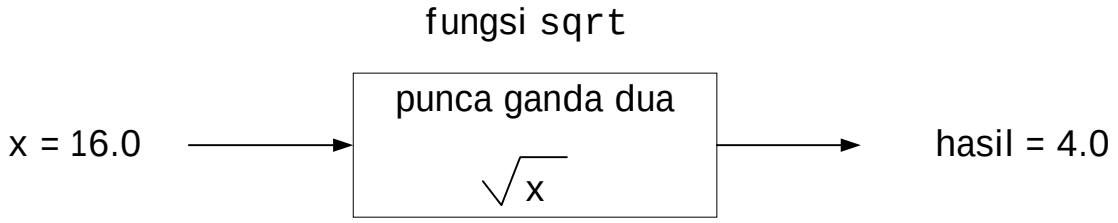
```
#include <math.h>
#include <stdlib.h>
```

4.5 Fungsi Perpustakaan Matematik (samb....)

- Beberapa fungsi dpd perpustakaan matematik:

fungsi	fail perpus.	tujuan@nilai dikembalikan & cth	argumen	hasil
`abs(x)`	`stdlib.h`	mutlak x abs(-5) hasil 5	int	int
`exp(x)`	`math.h`	e^x exp(1.0) hasil 2.71828	double	double
`log(x)`	`math.h`	$\log_e(x)$ log(2.71828) hasil 1.0	double	double
`pow(x,y)`	`math.h`	x^y pow(0.16, 0.5) hasil 0.4	double, double	double
`sqrt(x)`	`math.h`	$\sqrt{x}$ dan x >=0.0 sqrt(2.25) hasil 1.5	double	double

- Contoh operasi fungi sqrt dlm kotak hitam:



4.6 Operator hubungan

Operator hubungan digunakan utk melakukan perbandingan. Ia digunakan oleh suruhan[2] yg membuat keputusan. Ia memerlukan 2 pembolehubah.

<	kurang dari
<=	kurang dari atau sama dgn
>	lebih besar dari
>=	lebih besar dari atau sama dgn
==	sama dgn
!=	tidak sama dgn

Perhatikan:

x = 5 bermaksud berikan nilai 5 kpd pembolehubah x

x == 5 bermaksud bandingkan samada nilai x sama dgn 5

Jangan terkeliru operator umpukan dgn operator sama dgn !

Hasil dari operator hubungan ialah samada **BENAR (TRUE)** atau **PALSU (FALSE)**. Dlm C, PALSU diwakili oleh nombor/nilai sifar (0) dan BENAR oleh sebarang nombor bukan sifar(biasanya 1).

Contoh:

kenyataan : **y = (4 == 5)** akan memberikan y nilai sifar

kenyataan : **y = (x <= z)**

memberikan y nilai bukan sifar jika x lebih kecil atau sama dgn z

memberikan y nilai sifar jika x lebih besar dari z

4.7 Operator logik

Operator logik digunakan utk menggabungkan beberapa ungkapan yg mengandungi operator hubungan. Berguna utk melakukan perbandingan kompleks bagi membuat keputusan.

&&	DAN (AND)
||	ATAU (OR)
!	TIDAK (SALINGAN)

4.7 Operator logik (samb...)

Jika **ungkapan1** dan **ungkapan2** adalah ungkapan[2] yg mengandungi operator hubungan mudah spt x < 10, z == 0 dsbnya, maka:

ungkapan1 && ungkapan2	adalah BENAR jika & hanya jika kedua[2] ungkapan adalah BENAR
ungkapan1 || ungkapan2	adalah BENAR jika satu atau kedua[2] ungkapan adalah BENAR
!ungkapan1	adalah BENAR jika ungkapan1 adalah PALSU dan sebaliknya

Aturan **ungkapan1** dan **ungkapan2** dinilai adalah dari kiri ke kanan. Oleh itu utk memastikan ungkapan[2] dinilai spt yg dikehendaki, gunakan kurungan '( )' utk menentukan aturan ungkapan[2] dinilai.

Contoh:

a) (6 > 2) && (3==3) adalah BENAR
b) (6 < 2) && (3==3) adalah PALSU
c) ! ((6 < 2) && (3==3)) adalah BENAR
d) j != 0 && (x/10 < 3) adalah BENAR jika j bukan sifar dan x lebih kecil dari 30

e) (12 != 12) || ((2 > 1) && 5 ==2))

f) !(10==10) || (20 <= 22) || 2000

g) -100 && -155

h) x = ((9==0) && (1<1))

Latihan

1. Diberi pengenalan pemalar dan pembolehubah

```
#define PI 3.14159
#define MAX_I 1000
.....
double x, y;
int a, b, i;
```

yang mana menunjukkan kepada pernyataan yang sah (cari nilai yang disimpan oleh setiap pernyataan) dan yang mana tidak sah (kenapa).
anggap pembolehubah a adalah 3, b adalah 4, dan y adalah -1.0.

```
a. i = a % b;
b. i = (989 - MAX_I) / a;
c. i = b % a;
d. x = PI * Y;
e. i = a / -b;
f. x = a / b;
g. x = a % (a / b);
h. i = b / 0;
i. i = a % (990 - MAX_I);
j. i = (MAX_I - 990) / a;
k. x = a / y;
l. i = PI * a;
m. x = PI / y;
n. x = b/a;
o. i = (MAX_I - 990) % a;
p. i = a % 0;
q. i = a % (MAX_I -990);
```

2. Anggapkan anda mempunyai pengenalan pembolehubah berikut:

```
int   color, lime, straw, kuning, merah, oren;
double hitam, putih, hijau, biru, unggu, crayon;
```

Cari niali setiap pernyataan a.-f. yg mana nilai pembolehubah 'color' adalah 2, 'hitam' adalah 2.5, 'crayon' adalah -1.3, 'straw' adalah 1, 'merah' adalah 3, 'unggu' adalah 0.3E + 1.

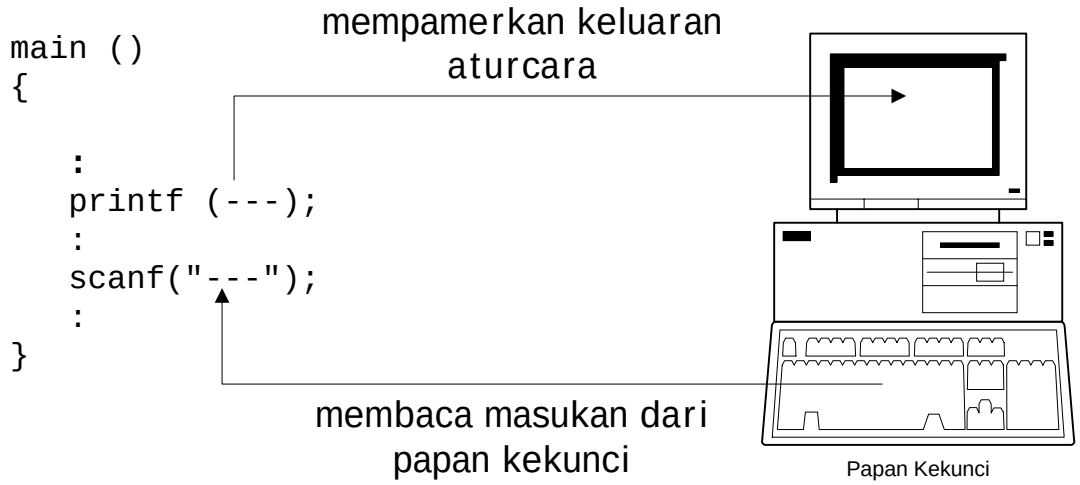
```
a. putih = color * 2.5 / unggu;
b. hijau = color / unggu;
c. oren  = color / red;
d. biru  = (color + straw) / (crayon +0.3);
e. lime  = merah / color + red % color;
f. unggu = straw / merah * color;
```

5 FUNGSI INPUT & OUTPUT

5.1 Fungsi Input & Output

- Dlm aturcara yg lepas, kita telah menggunakan fungsi **`printf (...)`** utk mempamir/mencetak mesej dlm format tertentu.

- Fungsi **`scanf (...)`** digunakan utk membaca masukan dr papan kekunci dlm format tertentu.



- Fungsi **prinf** & **scanf** membolehkan kita berhubung dgn sesuatu aturcara. Fungsi^2 ini dipanggil fungsi masukan & keluaran (I/O). Fungsi^2 ini **bukan** sebahagian dari bahasa C. Mereka dipanggil fungsi^2 perpustakaan yg ditulis/dibekalkan oleh pengeluar pengkompil C.

- Selain fungsi **prinf** & **scanf** C jiga menyediakan fungsi-fungsi lain untuk operasi input & output. Fungsi2 ini boleh dicapai dari perpustakaan C melalui fail kepala piawai <stdio.h> & <conio.h>.

  Contoh:

conio.h	: clrscr	getch	inport
stdio.h	: scanf	getchar	feof

5.2 Fungsi `printf` & cara Menggunakannya

- Format memanggil fungsi **`printf`** ialah :

  `printf(kawalan format, senarai cetakan);`

  Kawalan format ialah satu rentetan aksara diantara tanda " " yg menerangkan bagaimana pembolehubah[2] akan dicetak di skrin.
  Contohnya :

  "luas = %f\n" "jumlahLembu = %d"

  Kawalan format mengandungi 2 bentuk maklumat :

  i) aksara[2] yg akan dicetak secara terus
  ii) pencam data @ tentuan penukaran @ tentuan format

Tentuan Format	Keluaran
%d	integer desimal
%c	satu aksara
%s	rentetan aksara
%f	nombor nyata dlm desimal
%e	nombor nyata dlm eksponen
%E	nombor nyata eksponen E dlm huruf besar
%g	guna %f atau %e mana yg lebih pendek
%u	integer desimal tak bertanda
%x	integer hexa tak bertanda
%X	integer hexa tak bertanda (huruf besar)
%o	integer oktal tak bertanda
%%	cetak tanda (%)

  Contoh:

```
char a;
printf("%c  %d  %x  %0", a, a, a, a);
```

  Keluaran pd skrin:

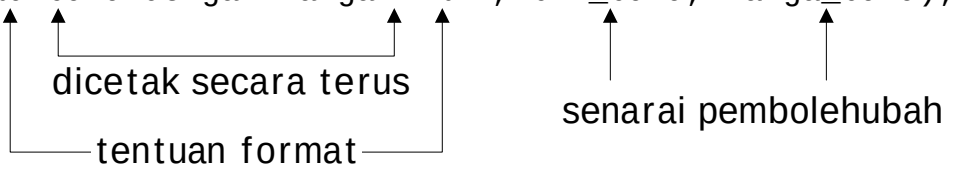
```
A  65  41  101
```

- Senarai cetakan (jika ada) terdiri dari nama[2] pembolehubah, pemalar, ungkapan & nama fungsi yg nilainya hendak dicetak.
  Setiap tentuan format dlm kawalan format mesti mempunyai satu pembolehubah dlm senarai.

5.2 Fungsi `printf` & cara Menggunakannya (samb....)

- Contoh 1:

```
bil_buku = 50; harga_buku = 2599.15;
printf("%d buku dengan harga RM%f", bil_buku, harga_buku);
```



Keluaran pd skrin:

```
50 buku dengan harga RM2599.15
```

- Contoh 2:

```
/* Contoh aturcara mencetak luas segiempat & bulatan */
/* Penulis : Dayang Norhayati Tarikh: 28.06.2020 */
#include <stdio.h>
#include <conio.h>

main ()
{
   const float pi = 3.1415;
   int panjang, lebar;
   float jejari;

   panjang = 12; lebar = 3;
   jejari = 3.5;

   clrscr();
   printf("Luas bulatan = %f  Luas segi4 = %d\n",
           pi * jejari * jejari, panjang * lebar );

}
```

Keluaran pd skrin:

```
Luas bulatan = 38.483375 Luas segi4 = 36
```

5.2 Fungsi `printf` & cara Menggunakannya (samb....)

- Format pembolehubah yg dicetak boleh diubahsuai dgn cara memasukkan pengubahsuai di antara % & aksara tentuan format. Contohnya :

  %-6d, %5d, %6.2f

Pengubahsuai	Makna
digit	Lebar medan minimum bagi mencetak pembolehubah. Medan yg lebih lebar akan digunakan jika nombor yg dicetak tidak muat dlm medan yg diberi.
.digit	Ketepatan bagi nombor nyata. Mewakili bil digit yg dicetak selepas titik perpuluhan.
-	Pembolehubah akan dicetak bermula dari kiri lebar medannya.
l	Pembolehubah yg akan dicetak adalah jeis integer panjang (long).

Contoh :

Suruhan	Keluaran Skrin
`printf("|%d|\n", 987);`	`|987|`
`printf("|%2d|\n", 987);`	`|987|`
`printf("|%8d|\n", 987);`	`|     987|`
`printf("|%-8d|\n", 987);`	`|987     |`
`printf("|%0.2f|\n", 9876.54);`	`|9876.54|`
`printf("|%4.2f|\n", 9876.54);`	`|9876.54|`
`printf("|%3.1f|\n", 9876.54);`	`|9876.5|`
`printf("|%10.3f|\n", 9876.54);`	`|  9876.540|`
`printf("|%10.3e|\n", 9876.54);`	`| 9.876e+03|`

5.3 Fungsi `scanf` & cara Menggunakannya

- Format memanggil fungsi **scanf** ialah :

```
scanf(kawalan format,  senarai alamat pembolehubah);
```

Kawalan format menyatakan format & jenis data yg akan dibaca dari papan kekunci. Bentuk dan maknanya sama spt format printf.

- Senarai alamat pembolehubah terdiri dari nama^2 pembolehubah yg dimulai dgn simbol (&). Nilai yg dibaca dari papan kekunci akan diisikan ke dlm pembolehubah tersebut.
  Setiap tentuan format dlm kawalan format mesti mempunyai satu pembolehubah dlm senarai.

- Contoh :

```
scanf("%d %f", &bil_buku, &harga_buku);
```

tentuan format senarai alamat pembolehubah

- Contoh : Rekakan satu aturcara untuk mencetak luas satu bulatan, jejari bulatan dibaca dari pengguna aturcara.

```
/* Contoh aturcara kira luas bulatan */
#include <stdio.h>

main ()
{
   const float pi = 3.1415;
   float jejari, luas_bulat;

   printf ("Masukkan jejari bagi bulatan : ");
   scanf("%f", &jejari);
   luas_bulat = pi * jejari * jejari;

   printf("Luas bulatan = %6.2f\n", luas_bulat );
}
```

5.4 Fungsi^2 Input & Output Mudah

i) Fungsi `getchar()` & `putchar()`

- dicapai dari perpustakaan piawai <stdio.h>

- Fungsi **`getchar`** digunakan utk membaca satu aksara dari papan kekunci dan menunggu aksara <ENTER> dimasukkan. Aksara yang dimasukkan akan dipamir.

  Contoh memanggil `getchar`:

```
ch = getchar ();
```

  nilai yg dibaca dari papan kunci akan disimpan dalam ch

- Fungsi **`putchar`** digunakan utk mencetak satu aksara ke skrin.

  Contoh memanggil putchar:

```
putchar (ch);
```

  parameter pembolehubah @ pemalar aksara yg hendak dicetak

- Contoh 1: baca satu aksara dari papan kekunci & mencetak aksara yg dibaca & aksara selepasnya :

```
char huruf;

huruf = getchar();
putchar (huruf);    putchar (huruf+1);
```

  Keluaran skrin jika 'E' ditaip :

```
EF
```

5.4 Fungsi[2] Input & Output Mudah (samb....)

- Contoh 2:

```
char hurufA = 0x41;
putchar (hurufA);   putchar ('b');
putchar ('\n');     putchar ('C');
```

Keluaran skrin :

```
Ab
C
```

ii) Fungsi `getch()` & `putch()`

- dicapai dari perpustakaan piawai <conio.h>

- Fungsi **`getch`** digunakan utk membaca satu aksara dari papan kekunci tanpa menunggu <ENTER> dimasukkan. Aksara yang dimasukkan tidak akan dipamir.

Contoh memanggil `getch`:

```
ch = getch ();
```

nilai yg dibaca dari papan kunci akan disimpan dalam ch

- Fungsi **`putch`** digunakan utk mencetak satu aksara ke skrin.

Contoh memanggil putch:

```
putch (ch);
```

parameter pembolehubah @ pemalar aksara yg hendak dicetak

- Jika contoh dari `getchar` & `putchar` ditukarkan kpd getch & putch ia akan memberi keluaran yg sama, hanya cara masukan berbeza.

5.4 Fungsi[2] Input & Output Mudah (samb....)

iii) Fungsi `getc()` & `putc()`

- dicapai dari perpustakaan piawai <stdio.h>

- Fungsi **`getc`** digunakan utk membaca satu aksara dari peranti input piawai. Contoh input piawai fail dan papan kekunci. Operasi bacaan aksara sama spt **`getchar()`**.

  Format memanggil fungsi `getc`:

  `getc(jenis_peranti)`

- Contoh memanggil `getc` & aksara dibaca dari papan kekunci:

  `ch = get (stdin);`

  nilai yg dibaca dari papan kunci akan disimpan dalam ch

- Fungsi **`putc`** digunakan utk mencetak satu aksara ke skrin.
  Format memanggil fungsi `putc`:

  `putc (operan, jenis_peranti);`

- Contoh fungsi **`putc`** digunakan utk mencetak satu aksara ke skrin.

  `putc(ch, stdout);`

  parameter pembolehubah @ pemalar aksara yg hendak dicetak

- Jika contoh dari `getchar` & `putchar` ditukarkan kpd `getc` & `putc` ia akan dilaksanakan dgn cara yg sama.

5.4 Fungsi[2] Input & Output Mudah (samb....)

iv) Fungsi `gets()` & `puts()`

- dicapai dari perpustakaan piawai <stdio.h>

- Fungsi **gets** digunakan utk membaca rentetan(string) dari keyboard mengakhiri bacaan rentetan jika aksara baris baru (\n) @ <ENTER> dimasukan. Rentetan akan disimpan & aksara baris baru akan ditukar ke aksara nul (\0).

- Fungsi **puts** digunakan utk mencetak rentetan di skrin memasukan aksara baris baru (\n).

- Contoh fungsi **gets & puts** digunakan dlm aturcara membaca dr papan kekunci & mencetak rentetan di skrin.

```
#include <stdio.h>

main ()
{
   char nama[30];

   printf("\nEnter Your Name please >>");
   gets(nama);    /*baca rentetan*/

   printf("Good day ");
   puts(nama);    /*cetak rentetan*/
   puts("It's your lucky day !!");

   return 0;
}
```

Keluaran pd skrin jika masukan "Dayang Norhayati" dimasukkan:

```
Good day Dayang Norhayati
It's your lucky day !!
```

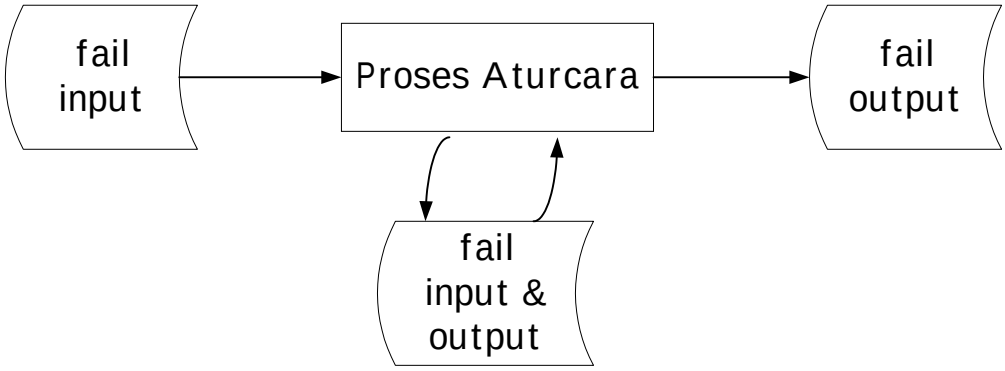
5.5 Pemprosesan Fail

- 2 ragam operasi komputer adalah:
  1. berkelompok (batch)
  2. interaktif (interactive).

  Aturcara C yg kita tulis setakat ini adalah ragam interaktif.
  Untuk menulis aturcara yang beroperasi secara berkelompok fail perlu digunakan.

- 3 operasi yg dilakukan utk memproses fail:
  1. membina fail - wajib
  2. membaca data dr fail
  3. menulis data ke fail

  } salah satu @ kedua2



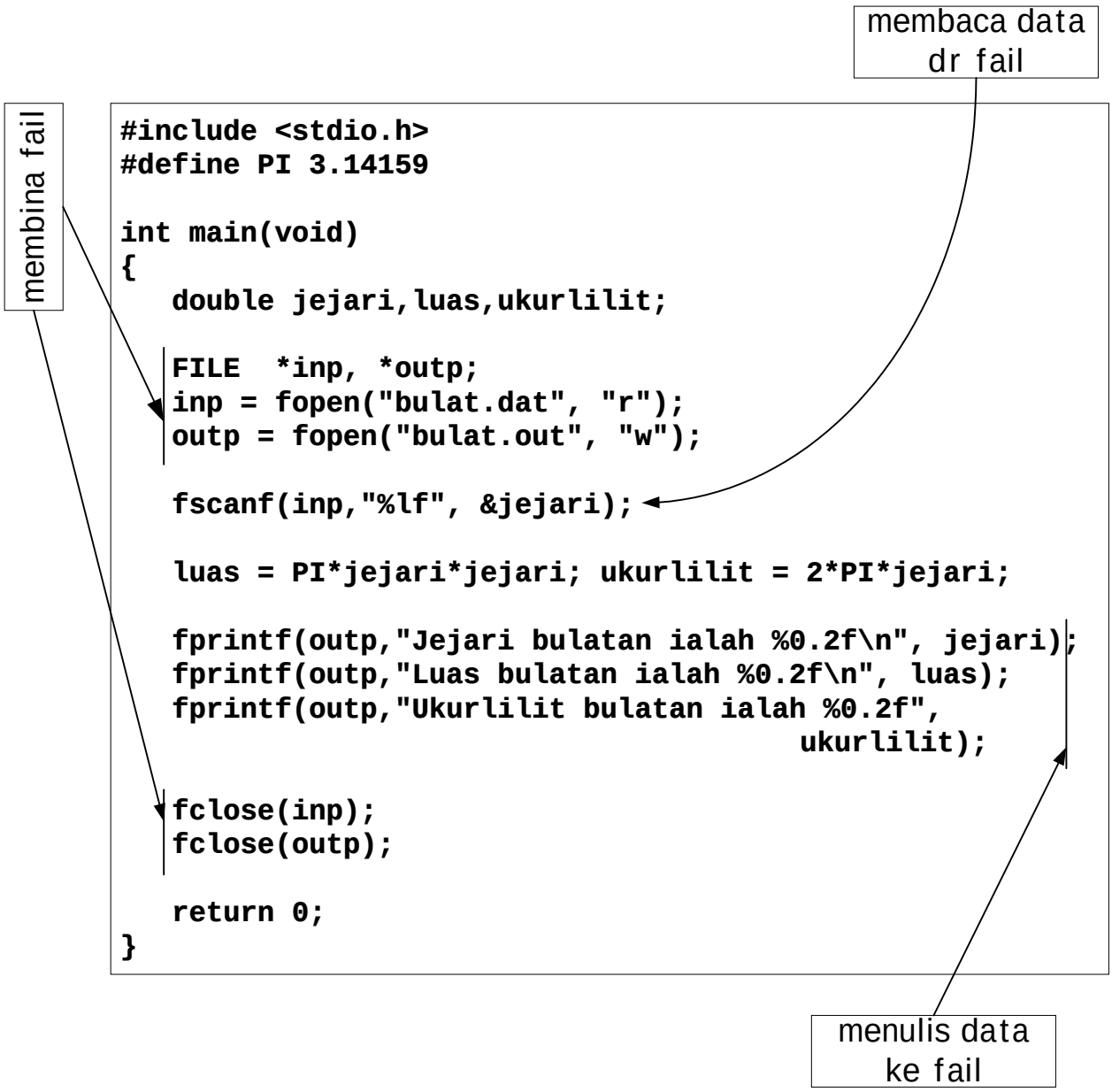
- Merujuk kpd masalah bulat.c, jika kita tukarkan masalah tersebut menggunakan fail:
  - membaca dr papan kekunci kpd membaca fail `bulat.dat`
  - mencetak output ke skrin kpd menulis di fail `bulat.out`

  Katakan isi `bulat.dat` spt berikut:

```
5.0
```

5.5 Pemprosesan Fail (samb....)

- Contoh aturcara menggunakan fail, membaca fail `bulat.dat` & menulis output di fail `bulat.out`.



Output dr aturcara akan dikeluarkan ke fail `bulat.out` & isinya adalah spt berikut:

```
Jejari bulatan ialah 5.00
Luas bulatan ialah 78.54
Ukurlilit bulatan ialah 31.42
```

5.5 Pemprosesan Fail (samb....)

i) Membina fail

- Pengisytiharan fail menggunakan pembolehubah penuding fail, format pengiystiharan:

```
FILE *pembpenudingfail;
```

- Untuk membolehkan fail digunakan, fail tersebut mesti dibuka terlebih dahulu.
  Format membuka fail:

```
pembpenudingfail = fopen("namafail","jenis_capaian");
```

pembolehubah penuding fail yg disytiharkan sebelum

nama fail yg sah

jenis mod capaian menentukan samada fail tersebut adalah fail input, output atau keduanya

- jenis mod capaian fail

Mod	Jika fail wujud	Jika fail tidak wujud
"r"	buka fail utk dibaca	Ralat
"w"	buka fail utk ditulis	Bina fail baru
"a"	buka fail utk ditambah di hujung fail	Bina fail baru
"r+"	buka fail utk dibaca & ditulis	Ralat
"w+"	buka fail utk dibaca & ditulis	Bina fail baru
"a+"	buka fail utk dibaca & ditambah di hujung fail	Bina fail baru

- Jika fail tidak digunakan lagi fail perlu ditutup. Format utk menutup fail:

```
fclose(pembpenudingfail);
```

5.5 Pemprosesan Fail (samb....)

ii) Membaca dan menulis fail

- Banyak cara boleh dilakukan untuk membaca dan menulis data ke @ dari fail, Ini bargantung kpd cara data hendak diproses:
  - sama ada aksara demi aksara pada satu masa
  - satu baris data (satu rentetan aksara) pada satu masa

- Antara fungsi-fungsi yang digunakan untuk membaca & menulis data:
  - `fscanf()& fprinf()` - pelbagai jenis data pada satu masa.
  - `fgetc()& fputc()` - satu aksara demi satu aksara pada satu masa (data berjenis char)
  - `fgets() & fputs()` - satu baris data (rentetan aksara) pada satu masa (data berjenis rentetan)

- Contoh aturcara menggunakan fail, membaca fail `stor12.dat` & menulis output di fail yg mana nama fail dimasukan oleh pengguna melalui papan kekunci.

Isi fail `stor12.dat` adalah spt berikut:

```
MAKMAL KAWALAN
ABC
10 45.20
23 17.39
4 100.14
```

Hasil dari aturcara di ms sebelah, fail output yg terhasil.

```
Laporan Tahunan Stor MAKMAL KAWALAN

Kod Part     Bil      Harga
A            10       45.20
B            23       17.39
C            4        100.14


Nilai stok 1252.53 dan purata harga satu part 54.24
```

5.5 Pemprosesan Fail (samb....)

```
#include <stdio.h>
int main(void)
{
   char   namastor[30], failLaporan[12];
   char   partA, partB, partC;

   int    bilpartA, bilpartB, bilpartC;
   float  harga1partA, harga1partB, harga1partC, nilaistok,
           purataharga;

   FILE  *stor,
         *laporan;

   stor = fopen("stor12.dat", "r");
   printf("Masukkan nama fail laporan > ");
   gets(failLaporan);
   laporan = fopen(failLaporan, "w");

   fgets(namastor,30,stor);
   partA = fgetc(stor); partB = fgetc(stor); partC =
                                              fgetc(stor);
   fscanf(stor,"%d %f", &bilpartA, &harga1partA);
   fscanf(stor,"%d %f", &bilpartB, &harga1partB);
   fscanf(stor,"%d %f", &bilpartC, &harga1partC);

   nilaistok = (bilpartA*harga1partA) + (bilpartB*harga1partB)
                                      + (bilpartC*harga1partC);
   purataharga= (harga1partA + harga1partB +harga1partC)/3;

   fprintf(laporan,"Laporan Tahunan Stor ");
   fputs(namastor, laporan);
   fprintf(laporan,"\nKod Part\tBil\t\tHarga\n");
   fprintf(laporan,"%c\t\t%d\t\t%0.2f\n", partA, bilpartA,
                                             harga1partA);
   fprintf(laporan,"%c\t\t%d\t\t%0.2f\n", partB, bilpartB,
                                             harga1partB);
   fprintf(laporan,"%c\t\t%d\t\t%0.2f\n", partC, bilpartC,
                                              harga1partC);
   fprintf(laporan,"\n\nNilai stok %0.2f dan purata
              harga satu part %0.2f", nilaistok, purataharga);

   fclose(stor);
   fclose(laporan);

   return 0;
}
```

Latihan

1. Apakah keluaran bagi setiap suruhan berikut :

a) **printf("Panjang seekor semut ialah : %2.2f sm", 2.445e-2);**
b) **printf("%c %d %u", 66, 0x50, 'C');**
c) **printf("%-8d\n%6.3f %-6.3f", 4356, 1.52, 1.52);**
d) **putchar('\"'); putchar('\\'); putchar(66);**

2. Cari kesilapan[2] dlm aturcara berikut:

```
#include <stdio.h>
main()
{
     int nom1, nom2;
     unsigned int jum1;
     char warna;
     jum1=-1;
     scanf("%d %d", nom1, num2);
     printf("%u",jum1);
     &warna=getchar();
     printf("%d %c %u", warna,warna,warna);
)
```

3. Tulis aturcara untuk melakukan perkara berikut :

a) Baca 6 nombor dari papan kekunci dan cari purata nombor[2] tersebut.
b) Cari isipadu satu bungkah dimana dimensi[2] diberikan oleh pengguna.
c) Tukar jarak dlm batu kpd km
d) Tukar nombor heksadesimal 3 digit kpd nombor desimal (2 cara)

6 STRUKTUR KAWALAN ATURCARA

Bab 2 kita telah mempelajari bagaimana hendak menyelesaikan masalah dgn menggunakan struktur jujukan, struktur pemilihan, struktur perulangan & gelung.

C mempunyai beberapa struktur kawalan utk mengawal aliran aturcara bagi melakukan pemilihan, perulangan & gelung. Struktur[2] ini berguna utk membuat keputusan.
Katakunci[2] yg digunakan dlm suruhan kawalan:

if	***else***	***goto***
switch	***case***	***while*** ***do***
for		

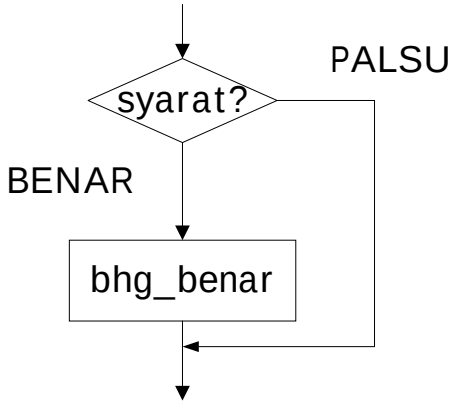
6.1 Struktur Pemilihan

- Dlm struktur pilihan, arahan[2] tertentu akan dipilih utk dilaksanakan, pemilihan ini berganting kpd syarat pemilihan.
- C melaksanakan struktur pemilihan dgn 3 cara:
  I) Pemilihan dgn `if & else`
  II) Pemilihan dgn operator@pengolah bersyarat
  III) Pemilihan dgn `switch`, `case` & `break`

6.1.1 Pemilihan dgn *if & else*

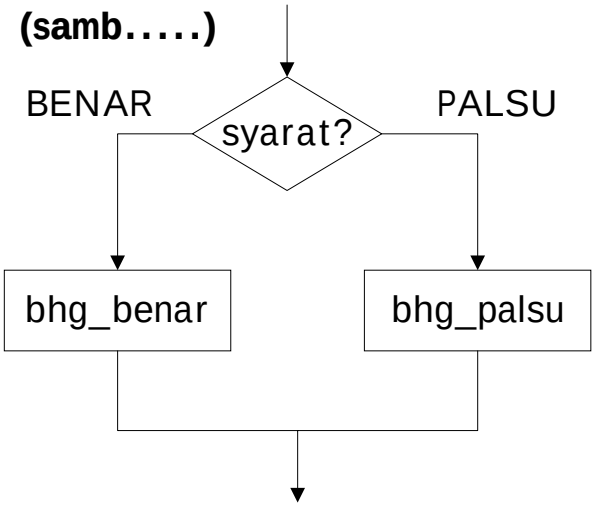
- Katakunci ***if & else*** digunakan utk memilih aliran aturcara @ membuat keputusan. Ada 3 format penggunaan ***if & else*** dlm aturcara:
- Ada 3 format penggunaan ***if & else*** dlm aturcara:

a)
```
if (syarat)
    bhg_benar;
```

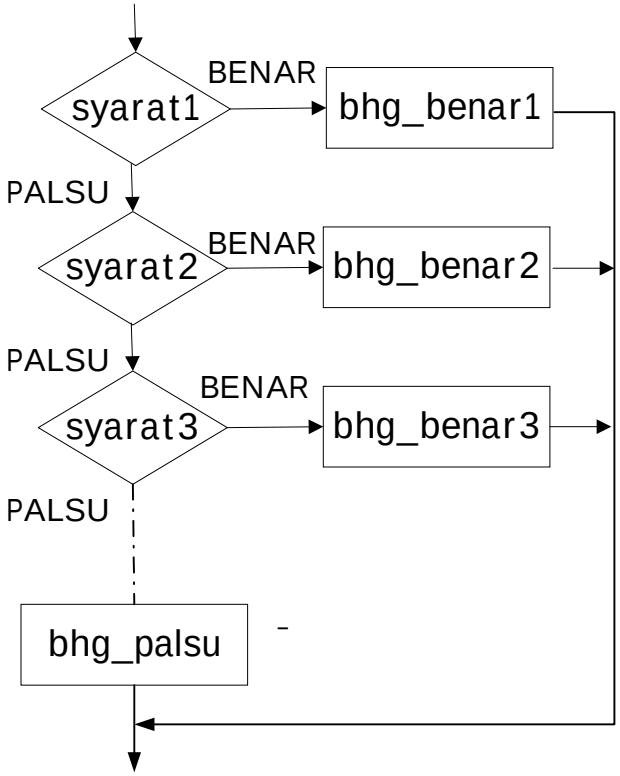


6.1.1 Pemilihan dgn if & else (samb.....)

b)
```
if (syarat)
    bhg_benar;
else
    bhg_palsu;
```



c)
```
if (syarat1)
    bhg_benar1;
else if (syarat2)
    bhg_benar2;
else if (syarat3)
    bhg_benar3;
     :
else
    bhg_palsu;
```



- Di dlm struktur a) - c) di atas:

  syarat, syarat1 ... adalah ungkapan[2] logik dan hubungan mudah/ kompleks.

  bhg_benar ... terdiri dari 1 @ lebih kenyataan[2] C yg akan dilaksanakan jika **syarat**... adalah BENAR. Jika **bhg_benar** mempunyai lebih dari satu kenyataan, ia **mesti** diletakkan di antara tanda { }.

  bhg_palsu ... terdiri dari 1 @ lebih kenyataan[2] C yg akan dilaksanakan jika syarat... adalah PALSU. Jika bhg_palsu mempunyai lebih dari satu kenyataan, ia mesti diletakkan di antara tanda { }.

  Soalan: Di dlm struktur c) di atas, jika syarat1 & syarat3 adalah BENAR, bahagian benar manakah yg akan dilaksanakan ?

6.1.1 Pemilihan dgn if & else (samb.....)

Contoh penggunaan if & else:

a)
```
if (x == 10)
   printf( "sama dgn 10");
```

b)
```
if (x < 0)
   printf( "negatif");
else
   printf( "positif");
```

c)
```
if (x == 0)
   printf( "sifar");
else if (x < 0)
   printf( "negatif");
else
   printf( "positif");
```

d)
```
if (x > 10)
   {
      printf( "lebih dari 10");
      ++kira;
   }
   else
   {
      printf( "kecil dari 10");
      --kira;
   }
```

e)
```
if ((x>10) || (y>20))
    printf("Memenuhi syarat");
```

- Contoh pengujian fail: Untuk memastikan tiada ralat dlm proses membuka fail, pengujian boleh dilakukan terutama bagi fail yg hendak dibaca. Semasa fail `stor12.dat` hendak dibuka & dibaca pengujian dilakukan.

```
#include <stdio.h>

main()
{
     FILE *failptr;
     failptr = fopen("stor12.dat", "r");

     if (failptr ==NULL)
     {
          printf("Ralat dalam pembukaan fail.\n")
          exit(-1); /*Tamatkan Aturcara*/
     }

     return 0;
}
```

Kod ini boleh ditukar spt berikut - memberi maksud yg sama:

```
if(failptr = fopen("stor12.dat", "r"))== NULL
{
     puts("Ralat dalam pembukaan fail.");
     exit(-1); /*Tamatkan Aturcara*/
}
```

6.1.1 Pemilihan dgn if & else (samb.....)

- Jika dlm satu pemilihan `if` hanya ada 1@2 bahagian pilihan sahaja dinamakan struktur pemilihan mudah.

- Utk lebih dpd 2 pilihan ***if*** **bersarang** perlu digunakan. 2 cara if bersarang digunakan:
  i) Keputusan Berbilang Alternatif
  Telah diguna sebelum ini iaitu `if .. else if ... else` rujuk `if` format c) ms 2.

  ii) Syarat `if` yang diberi dlm beberapa aras. Contoh 3 aras `if` bersarang:

```
if (syarat1)
    if (syarat2)
        if (syarat3)
            bhg_benar;
```

`if` bersarang keputusan berbilang alternatif digunakan bila hanya satu data(1 pembolehubah) shj digunakan utk mementukan bahagian yang benar.

Contoh if bersarang menggunakan kep.berbilang alternatif:
Jika kadar kebisingan bunyi dalam desibel dihubungkan dgn kesannya terhadap tanggapan manusia. Jadual di bawah menghubungkan kadar bunyi dgn tanggapan manusia:

kebisingan dlm desibel (db)	Tanggapan manusia
50 atau kurang	Senyap
51-70	Menggangu
71-90	Menyakitkan hati
91-110	Sangat menyakitkan hati
lebih dpd 110	Tidak selesa

6.1.1 Pemilihan dgn if & else (samb.....)

Penyelesaian:

```
if (bising_db<=50)
   printf("%d db adalah Senyap\n", bising_db);
else if (bising_db<=70)
   printf("%d db adalah Menggangu\n", bising_db);
else if (bising_db<=90)
   printf("%d db adalah Menyakitkan hati\n", bising_db);
else if (bising_db<=110)
   printf("%d db adalah Sangat menyakitkan hati\n",bising_db);
else
   printf("%d db adalah Tidak selesa\n",bising_db);
```

Contoh syarat if yang diberi dlm beberapa aras:
Tulis satu aturcara utk mengira & memaparkan luas satu segiempat. Saiz sisi^2 segiempat ini diberikan oleh pengguna melalui papan kekunci. Aturcara anda mesti memastikan saiz sisi^2 yg diberi bukanlah sifar atau negatif. Jika saiz yg diberikan adalah sifar, cetak mesej amaran kpd pengguna & jadikan luas sifar. Jika saiz yg diberikan adalah negatif, ambil magnitudnya shj.

Penyelesaian:
i) Analisa Masalah

Input:

media : papan kekunci
data : panjang & lebar

Output:

media : skrin
data : luas segiempat

Proses:

1. utk mencari magnitud sesuatu nombor:
   jika nombor adalah negatif, magnitud = -(nombor)
   jika nombor adalah positif, magnitud = nombor
2. luas segiempat = panjang x lebar

6.1.1 Pemilihan dgn if & else (samb.....)

Penyelesaian ....:

ii) Kod pseudo:

```
baca panjang & lebar dari pengguna
if panjang = 0 atau lebar = 0 then
   begin
     cetak mesej amaran
     luas := 0
  end
 else
   begin
      if panjang negatif then
            panjang:= magnitud panjang
      endif
      if lebar negatif then
           lebar:= magnitud lebar
      endif
      luas := panjang * lebar
  end
 endif
 cetak luas
```

iii) Aturcara:

```
#include <stdio.h>
main()
{
     float luas, panjang, lebar;

     printf ("masukkan nilai panjang dan lebar\n");
     scanf ("%f %f", &panjang, &lebar);
     if (panjang== 0 || lebar ==0)
     {
          printf("%*@$%^#***@^\n");
          luas=0;
     }
     else
     {
          if (panjang < 0)
               panjang = -panjang; /*cari magnitud panjang*/
          if (lebar < 0 )
               lebar =  -lebar;    /* cari magnitud lebar*/
          luas = panjang*lebar;
     }

     printf ("luas segiempat ialah %f\n", luas);
}
```

if bersarang 2 aras

6.1.1 Pemilihan dgn if & else (samb.....)

Contoh syarat if yang diberi dlm beberapa aras:
Kementerian pertahanan hendak menyenaraikan nama-nama kakitangan lelaki yg berumur 20 hingga 26 tahun dan belum berkahwin.

Kenyataan if bersarang berikut boleh digunakan:

```
if (jantina == 'L' )
    if (taraf_perkahwinan == 'B')
        if (umur <=26)
            if (umur>=20)
                printf("%s\n",nama );
```

ATAU kenyataan berikut juga boleh digunakan:

```
if (jantina == 'L' )&& (taraf_perkahwinan == 'B')
    && (umur <=26) &&  (umur>=20)
            printf("%s\n",nama );
```

6.1.2 Pemilihan dgn operator@pengolah bersyarat

- C mempunyai singkatan bagi mengungkap kenyataan ***if-else*** menggunakan operator bersyarat **? :**
  Format penggunaan operator bersyarat dlm aturcara:

  syarat ? ungkapan_benar : ungkapan_palsu

  Jika syarat adalah BENAR, maka keseluruhan ungkapan ini akan memberikan nilai ungkapan_benar. Sebaliknya, jika syarat adalah PALSU, maka keseluruhan ungkapan ini akan memberikan nilai ungkapan_palsu.

6.1.2 Pemilihan dgn pengolah bersyarat (Samb..)

- Contoh:

i) **harga_tiket = (umur < 12) ? 10 : 20;**

pembolehubah harga_tiket akan diberikan nilai 10 jika nilai pembolehubah umur kurang dari 12 sebaliknya ia akan diberikan nilai 20 jika nilai pembolehubah umur lebih dari@ sama dgn 12. Kenyataan if-else yg setara:

```
if (umur < 12)
     harga_tiket = 10;
else
     harga_tiket = 20;
```

ii) Aturcara mencari luas segi4 menggunakan operator bersyarat:

```
#include <stdio.h>
main()
{
     float luas, panjang, lebar;

     printf ("masukkan nilai panjang dan lebar\n");
     scanf ("%f %f", &panjang, &lebar);
     if (panjang== 0 || lebar ==0)
     {
          printf("%*@$%^#***@^\n");
          luas=0;
     }
     else
     {
          panjang = (panjang < 0) ? -panjang : panjang;
          lebar = (lebar < 0 ) ?   -lebar : lebar;
          luas = panjang*lebar;
     }
     printf ("luas segiempat ialah %f\n", luas);
}
```

Latihan

1. Bentukkan ungkapan^2 dlm C bagi mengungkap syarat^2 berikut:
   a) **nombor** sama dgn atau lebih besar dari 1 tetapi lebih kecil dari 9
   b) pembolehubah hrf bukan huruf q atau k
   c) **nombor** adalah di antara 1 dan 9 tetapi bukan 5
   d) **nombor** bukan di antara 1 dan 9

2. Tulis satu aturcara untuk memaparkan mesej samada satu huruf yg dibaca dari papan kekunci adalah huruf vokal @ konsonan.

3. Tukar aturcara mencari luas segiempat dlm contoh untuk mengira juga luas satu bulatan yg jejarinya dibaca dari pengguna. Pengguna boleh memilih samada mencari luas bulatan atau luas segiempat.

4. Tulis aturcara yang memberi grade bagi satu mata pelajaran. Pengguna memasukkan peratusan markah pelajar dari papan kekunci dan aturcara akan menukarkan markah tersebut dalam bentuk gred dan dipaparkan ke skrin.

6.1.3 Pemilihan dgn switch, case & break

- Struktur ***if-else-if-...-else*** digunakan utk memilih aliran aturcara yg terdiri dari banyak alternatif. Kadang-kadang ini lebih mudah dilakukan dgn menggunakan struktur ***switch-case-break***.
  Format penggunaan ***switch-case-break*** dlm aturcara:

```
switch (ungkapan_integer)
{
      case pemalar1:  kenyataan2A; break;
      case pemalar2:  kenyataan2B; break;
      case pemalar3:  kenyataan2C; break;
            :
      default:  kenyataan2D;
}
```

Di mana **ungkapan_integer** adalah ungkapan yg akan menghasilkan nilai jenis integer @ char.

Cara operasi:

1) Aturcara akan mencari nilai **ungkapan_integer.**

2) Nilai ini akan dibandingkan dgn nilai[2] `pemalar1,pemalar2` ... dlm kenyataan `case pemalar1, case pemalar2` dsbnya satu demi satu sehingga aturcara jumpa nilai pemalar yg sama dgn nilai `ungkapan_integer`. Jika sama, kenyataan[2] di antara `case .. break` tersebut akan dilaksanakan. Misalnya jika `ungkapan_integer == pemalar2`, maka kenyataan[2B] akan dilaksanakan.

3) Jika tiada `ungkapan_integer` tidak sama dgn kesemua pemalar[2] dlm senarai `case` maka kenyataan[2D] di bhg default akan dilaksanakan. Bhg default adalah pilihan-boleh ditinggalkan jika tak perlu.

4) Katakunci **break** akan menyebabkan perlaksanaan selanjutnya akan bermula luar dari blok kenyataan **switch() {...}**. Jika break ditinggalkan, kenyataan[2] selepasnya akan dilaksanakan.

6.1.3 Pemilihan dgn switch, case & break (Samb..)

i) Aturcara ini akan mencetak nombor 1, 2, @3 yg dimasukkan melalui papan kekunci dlm perkataan- satu, dua @ tiga

```
main()
{
     int x;
     scanf("%d", &x);
     switch (x)
     {
          case 1: puts ("satu"); break;
          case 2: puts ("dua"); break;
          case 3: puts ("tiga"); break;
          default: puts ("bukan satu dua tiga ");
     }
}
```

ii) Perhatikan output jika break dikeluarkan spt berikut:

```
main()
{
     int x;
     scanf("%d", &x);
     switch (x)
     {
          case 1: puts ("satu"); break;
          case 2: puts ("dua");
          case 3: puts ("tiga");
          default: puts ("bukan satu dua tiga ");
     }
}
```

Masukan: 1 **Jawapan:** satu	**Masukan:** 3 **Jawapan:** tiga bukan satu dua tiga
Masukan: 2 **Jawapan:** dua tiga bukan satu dua tiga	**Masukan:** selain dpd 1,2 & 3 **Jawapan:** bukan satu dua tiga

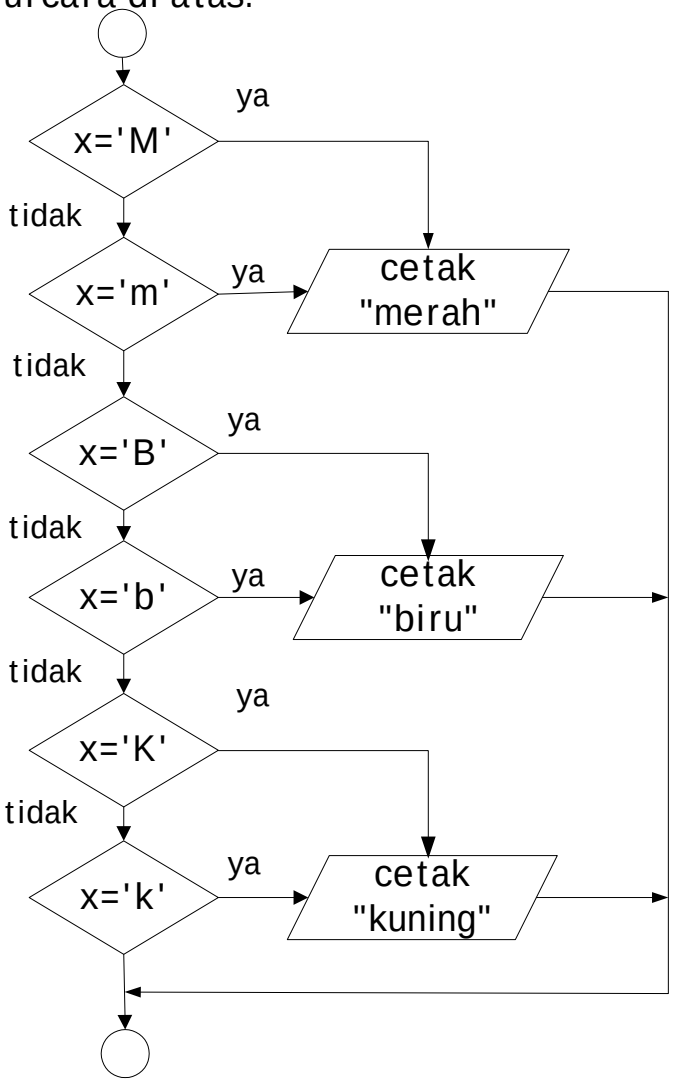
Soalan? Lukiskan carta alir utk bahagian switch, 2 kod di atas .

6.1.3 Pemilihan dgn switch, case & break (Samb..)

iii) Aturcara ini akan mencetak warna pada skrin, akan mencetakan "merah" jika aksara 'M' @ 'm' dimasukan, "biru" jika aksara B @ b dimasukkan & "kuning" jika aksara 'K' @ 'k' dimasukkan melalui papan kekunci.

```
main()
{
     char x;
     scanf("%d", &x);
     switch (x)
     {
          case 'M':
          case 'm': printf ("merah\n"); break;
          case 'B':
          case 'b': printf ("biru\n"); break;
          case 'K':
          case 'k': printf ("kuning\n"); break;
     }
}
```

Carta alir bagi aturcara di atas.



6.2 Struktur Perulangan & Gelung

- Sering kali dlm satu aturcara, satu/blok kenyataan perlu dilaksanakan berulang[2] beberapa kali atau sehingga sesuatu syarat dipenuhi. Perlaksanaan yg berulang ini membentuk **gelung**.

- Dlm C kata kunci 3 digunakan utk melakukan perulangan/gelung:
  I) while
  II) do .. while
  III) for

- Ketiga-tiga gelung tersebut beroperasi dgn cara berlainan. Untuk menentukan gelung mana yg digunakan, beberapa jawapan perlu dikenalpasti sebelum menyelesaikan masalah:
  1. Ada ulangan dlm PM?
  2. Jika soalan 1 Ya, tahu kah, berapa kali perlu diulang?
  3. Jika soalan 2 Tidak, bila nak berhenti @ apakah syarat utk berhenti.

6.2.1 Gelung dgn while & do...while

- Format penggunaan suruhan ***while*** dan ***do .. while*** dlm aturcara:

```
while (syarat)
{
      kenyataan1;
      kenyataan2;
           :
      kenyataann;
}
```

BLOK

```
do
{
      kenyataan1;
      kenyataan2;
           :
      kenyataann;
} while (syarat)
```

BLOK

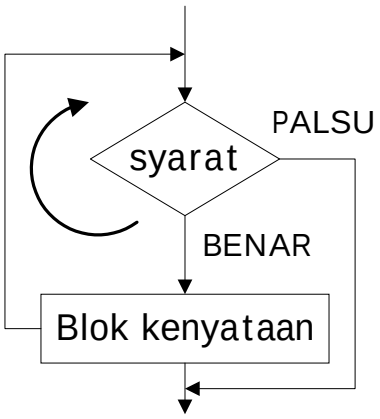
Di mana **syarat** adalah satu ungkapan Boolean yg akan menghasilkan BENAR (1) @ PALSU (0)

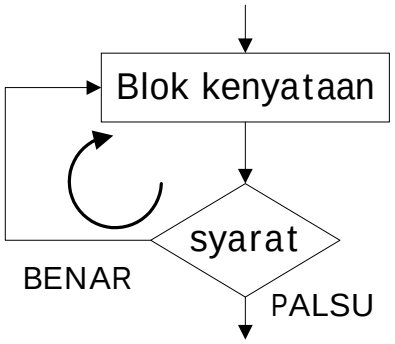
Cara operasi:
Blok kenyataan[2] 1,2...n akan dilaksanakan berulang[2] selagi **syarat** adalah BENAR.

6.2.1 Gelung dgn while & do...while (samb...)

- **Perbezaan *while* dan *do ... while*:**

while	**do ... while**
syarat diuji sebelum masuk ke gelung. Blok kenyataan tidak akan dilaksanakan jika syarat adalah PALSU semasa masuk ke gelung.	syarat diuji setelah masuk ke gelung Blok kenyataan akan dilaksanakan sekurang²nya sekali jika syarat adalah PALSU semasa masuk ke gelung.





- **Contoh 1:** Tulis satu aturcara utk memcetak nama Pn. Dayang di Skrin sehingga hujan. Perhatikan operasi dgn i/p & o/p fail berikut: while [i/p: y, o/p:TIADA] do...while [i/p:yy, o/p:1)Pn. Dayang]

```
#include <stdio.h> #include <conio.h>
void main()
{
     FILE  *keluar; int pengira=1; char hujan='n';
     keluar = fopen("dayang.out", "w");

     printf("Dah hujan? Tekan aksara y jika hujan>");
     hujan = getch();
     while ( hujan != 'y' )
     {
      fprintf (keluar,"%d) Pn. Dayang \n", pengira);
      printf("\nDah hujan? Tekan aksara y jika hujan>");
      hujan = getch();
       ++pengira;
      }
}
```

```
do {
     fprintf (keluar,"%d) Pn. Dayang \n", pengira);
     printf("\nDah hujan? Tekan aksara y jika hujan>");
     hujan = getch();
     ++pengira;
} while (hujan != 'y' );
```

6.2.1 Gelung dgn while & do...while (samb...)

- **Contoh 1:** Tulis satu aturcara utk mempamirkan sifir darab bagi 11. Corak output yg dikehendaki adalah spt berikut:

```
11 X 1 = 11
11 X 2 = 22
     :
11 X 12 = 121
```

Penyelesaian:

```
#include <stdio.h>
main()
{
     int nomb_sifir,
         pengira, kiraan_max;

     nomb_sifir = 11;
     pengira = 1;  kiraan_max =12;

     while ( pengira <= kiraan_max )
     {
        printf ("%d X %d  = %d \n", nomb_sifir, pengira,
                                    pengira*nomb_sifir);
        ++pengira;
     }
}
```

- **Contoh 2:** Tulis satu aturcara permainan meneka nombor. Komputer akan memilih satu nombor & pengguna diminta meneka nombor tersebut. Selagi nombor yg diteka tidak sama dgn nombor yg dipilih oleh komputer, pengguna mesti terus meneka. Paparkan mesej samada tekaan adalah tepat atau sebaliknya.

Penyelesaian:

```
#include <stdio.h>
main()
{
     int nomb_komputer, nomb_teka;

     nomb_komputer = 17;
     do {
        puts ("sila teka nombor yg saya pilih (1-20)\n" );
        scanf ("%d", nomb_teka);
        if (nomb_teka != nomb_komputer)
          puts ("Anda salah, cuba lagi\n");
     }  while ( nomb_teka != nomb_komputer );

     puts ("Tahniah, anda betul\n");
}
```

6.2.2 Gelung dgn for

- Format penggunaan suruhan **if** dlm aturcara:

```
for (bhg_memulakan ; syarat ; bhg_kemaskini)
{
      kenyataan1;
      kenyataan2;
            :
      kenyataann;
}
```

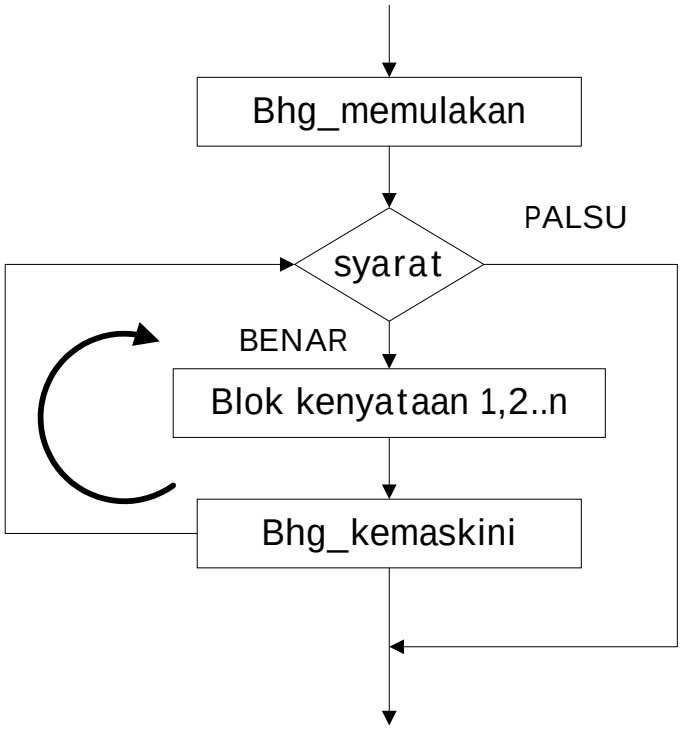
Di mana syarat adalah satu ungkapan Boolean yg akan menghasilkan BENAR (1) @ PALSU (0).

Cara operasi:

1) ungkapan di bhg_memulakan dilaksanakan. Bhg ini biasanya digunakan utk mengumpuk nilai awalan kpd sesuatu pembolehubah. Dilaksanakan sekali sahaja.

2) syarat akan diuji. Jika syarat adalah BENAR, Blok kenyataan[2] 1,2...n akan dilaksanakan. Jika PALSU, gelung akan ditamatkan, dan perlaksanaan bermula selepas blok suruhan for.

3) ungkapan di bhg_kemaskini dilaksanakan dan perlaksanaan akan ulang langkah 2).



- Ungkapan[2] bhg_memulakan @ bhg_kemaskini boleh diabaikan ttp tanda semkolon mesti di kekalkan. Contohnya

```
for ( ; syarat ; )
```

6.2.2 Gelung dgn for (samb....)

- **Contoh 1:** Ulang contoh 1 m/s 15 menggunakan for.

Penyelesaian:

```
#include <stdio.h>
main()
{
     int nomb_sifir,
         pengira, kiraan_max;

     nomb_sifir = 11;
     kiraan_max =12;

     for ( pengira = 1;  pengira <= kiraan_max; pengira++ )
     {
        printf ("%d X %d  = %d \n", nomb_sifir, pengira,
                                    pengira*nomb_sifir);
      }
}
```

- **Contoh 2:** Apakah output bagi aturcara berikut ?

```
#include <stdio.h>
main()
{
     int i,j;

     for ( i = 6;  i ; i-- )
     {
          for ( j = 0;  j < i ; j ++ )
          {
               putchar ('#');
          }
          putchar ('\n');
     }
}
```

Penyelesaian:

```
######
#####
####
###
##
#
```

6.2.2 Gelung dgn for (samb....)

- **Contoh 3:** Aturcara sifir dgn mengabaikan ungkapan bhg_memulakan & bhg_kemaskini .

Penyelesaian:

```
#include <stdio.h>
main()
{
     int nomb_sifir,
          pengira, kiraan_max;

     nomb_sifir = 11;
     kiraan_max =12;
     pengira = 1;

     for ( ;  pengira <= kiraan_max;  )
     {
        printf ("%d X %d  = %d \n", nomb_sifir, pengira,
                                    pengira*nomb_sifir);
        pengira++;
    }
}
```

6.3 Contoh Penggunaan Struktur Kawalan Pemilihan & Gelung

Tulis satu aturcara utk mengira & memaparkan jumlahlah markah pelajar di skrin. Markah[2] dibaca dari fail markah.dat.

Isi fail markah.dat

```
12
56
89
66
78
```

1. Ada ulangan dlm PM?
YA
2. Jika soalan 1 Ya, tahu ke, berapa kali perlu diulang?
Tidak
3. Jika soalan 2 Tidak, bila nak berhenti @ apakah syarat utk berhenti.
Sehingga habis semua markah dibaca di dlm fail

2 cara untuk memeriksa penghujung fail:
1. EOF - pemalar penghujung fail
2. feof - macro yg memulangkan nilai 0 selagi pemalar EOF belum dijumpai

```
#include<stdio.h>
#include<process.h>

void main (void)
{
   FILE *masuk;
   int jumlah = 0, markah, status;

   if ((masuk = fopen("markah.dat", "r"))== NULL)
   {
      puts("Ralat dalam pembukaan fail.");
      exit(-1); /*Tamatkan Aturcara*/
   }

   printf("Markah-markah Saya\n");
   while ((fscanf(masuk,"%d\n", &markah))!=EOF)
   {
      printf("%5d\n",markah);
      jumlah+=markah;
   }
   printf("Jumlah markah %d", jumlah);
}
```

6.3 Contoh Penggunaan Struktur Kawalan Pemilihan & Gelung(samb..)

Gelung utk membaca markah, mencetak markah & mengira jumlah markah boleh ditukar menggunakan macro `feof` spt berikut:

```
while (!feof(masuk))
{
     fscanf(masuk,"%d\n", &markah);
     printf("%5d\n",markah);
     jumlah+=markah;
}
```

Gelung `while` tersebut juga boleh digantikan dgn gelung `for` spt berikut:

```
for (status=fscanf(masuk,"%d\n", &markah);
     status != EOF; status=fscanf(masuk,"%d\n", &markah))
{
      printf("%5d\n",markah);
      jumlah+=markah;
}
```

Output di skrin:

```
Markah-markah Saya
   12
   56
   89
   66
   78
Jumlah markah 301
```

6.4 Gelung Bersarang

- Gelung bersarang - penyataan gelung yg mengandungi gelung lain.

Tulis satu aturcara utk mempamirkan sifir darab bagi 1 hingga 5 & hanya hasil darab yg kurang & sama dgn 5 sahaja akan dicetak.
Penyelesaian:

```
#include <stdio.h>
main()
{
     int nomb_sifir, hasil_darab,
          pengira, sifir_max;

     sifir_max=5;  hasil_darab=0;  pengira = 1;

     for ( nomb_sifir=1; nomb_sifir<=sifir_max ;nomb_sifir++)
     {
          printf("\nSifir %d\n", nomb_sifir);
          hasil_darab = pengira*nomb_sifir;

          while (hasil_darab<=5)
          {
               printf ("%d X %d  = %d \n", nomb_sifir,
                                    pengira, hasil_darab);
              pengira++;
              hasil_darab = pengira*nomb_sifir;
          }
          pengira=1;
     }
     return 0;
}
```

Gelung aras 1

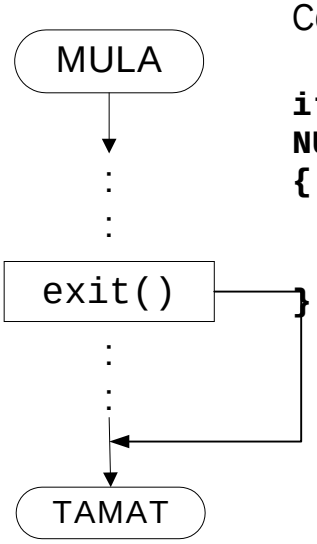
Gelung aras 2

Output pada skrin dari aturcara di atas

```
Sifir 1
1 X 1  = 1
1 X 2  = 2
1 X 3  = 3
1 X 4  = 4
1 X 5  = 5

Sifir 2
2 X 1  = 2
2 X 2  = 4

Sifir 3
3 X 1  = 3

Sifir 4
4 X 1  = 4

Sifir 5
5 X 1  = 5
```

6.5 Pintas Struktur

- 3 struktur PM struktur jujukan, struktur pemilihan, struktur perulangan & gelung menentukan aliran perlaksanaan aturcara. Kadang-kadang wujud situasi keluar dari struktur tertentu. C menyediakan arahan utk tujuan ini, antaranya:
  1. fungsi `exit()`
  2. pernyataan `break`
  3. pernyataan `continue`



Contoh:

```
if(failptr = fopen("stor12.dat", "r"))==
NULL
{
     puts("Ralat dalam pembukaan fail.");
     exit(-1); /*Tamatkan Aturcara*/
}
```

```
while (syarat)
{
     kenyataan1;
     kenyataan2;
          :
     break;
          :
     kenyataann;
}
```

```
do
{
     kenyataan1;
     kenyataan2;
          :
     break;
          :
     kenyataann;
} while (syarat)
```

```
for (bhg_memulakan ; syarat ; bhg_kemaskini)
{
     kenyataan1;
     kenyataan2;
          :
     break;
          :
     kenyataann;
}
```

6.5 Pintas Struktur (samb....)

```
while (syarat)
{
     kenyataan1;
     kenyataan2;
          :
     continue;
          :
     kenyataann;
}
```

```
do
{
     kenyataan1;
     kenyataan2;
          :
     continue;
          :
     kenyataann;
} while (syarat)
```

```
for (bhg_memulakan ; syarat ; bhg_kemaskini)
{
     kenyataan1;
     kenyataan2;
          :
     continue;
          :
     kenyataann;
}
```

Contoh 1:

```
for (i=1; i<=3; i++)
{
   printf("%d\n", i);
   if (i==2)
      break;
   printf("Bahagian bawah gelung\n");
}
printf("Keluar dari gelung");
```

Output di skrin:

```
1
Bahagian bawah gelung
2
Keluar dari gelung
```

6.5 Pintas Struktur (samb....)

Contoh 2:

```
for (i=1; i<=3; i++)
{
   printf("%d\n", i);
   if (i==2)
      continue;
   printf("Bahagian bawah gelung\n");
}
printf("Keluar dari gelung");
```

Output di skrin:

```
1
Bahagian bawah gelung
2
3
Bahagian bawah gelung
Keluar dari gelung
```

Contoh 3:

```
i=0;
while (i<5)
{
   if (i<3)
   {
      i+=2;
      printf("%d\n",i);
      continue;
   }
   else{
      printf("%d\n",++i);
      break;
   }
   printf("Bahagian bawah gelung\n");
}
printf("Keluar dari gelung");
```

Output di skrin:

```
2
4
5
Keluar dari gelung
```

7 FUNGSI PENGATURCARAAN BERMODULAR

FUNGSI -
Kumpulan jujukan suruhan[2] yg diberikan nama. Contoh fungsi[2] yg telah dipelajari:

main()
printf()
scanf()
getchar()
putchar()
puts()

} Fungsi[2] piawai perpustakaan yg ditulis oleh pembekal pengkompil spt Borland & Microsoft

2 kategori fungsi :
1. Fungsi piawai perpustakaan C, spt di atas.
2. Fungsi takrifan pengguna, yg direkabentuk & ditulis sendiri oleh pengaturcara.

7.1 Mengapa Perlu Fungsi ?

- ***Struktur mudah difahami***
  Aturcara menjadi lebih mudah dibaca dan difahami kerana kumpulan suruhan[2] yg melakukan kerja tertentu dikumpulkan dlm beberapa bahagian yg dipanggil **fungsi**. Ia juga memudahkan proses penyahpepijat.

- ***Jimat masa/ruang & Guna semula***
  Jujukan suruhan yg berulang[2] dlm sesuatu aturcara boleh dijadikan fungsi. Fungsi ini hanya perlu ditulis sekali shj & boleh digunakan berulang[2] dgn memanggil nama fungsi tersebut. Contohnya fungsi printf yg panjangnya beratus baris hanya ditulis sekali oleh pembekal pengkompil, tetapi boleh digunakan berulang[2] dlm aturcara kita.

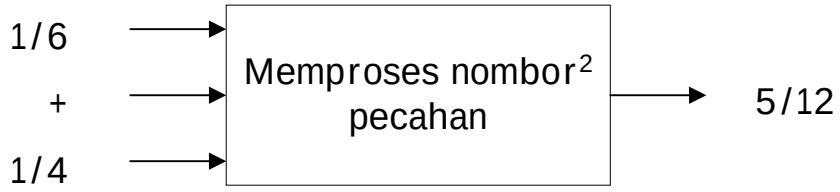
- ***Menggalakkan aturcara bermodular***
  Masalah yg besar dipecahkan kpd masalah yg kecil (pendekatan atas-bawah) & skop masalah yg kecil tersebut diselesaikan satu-persatu.

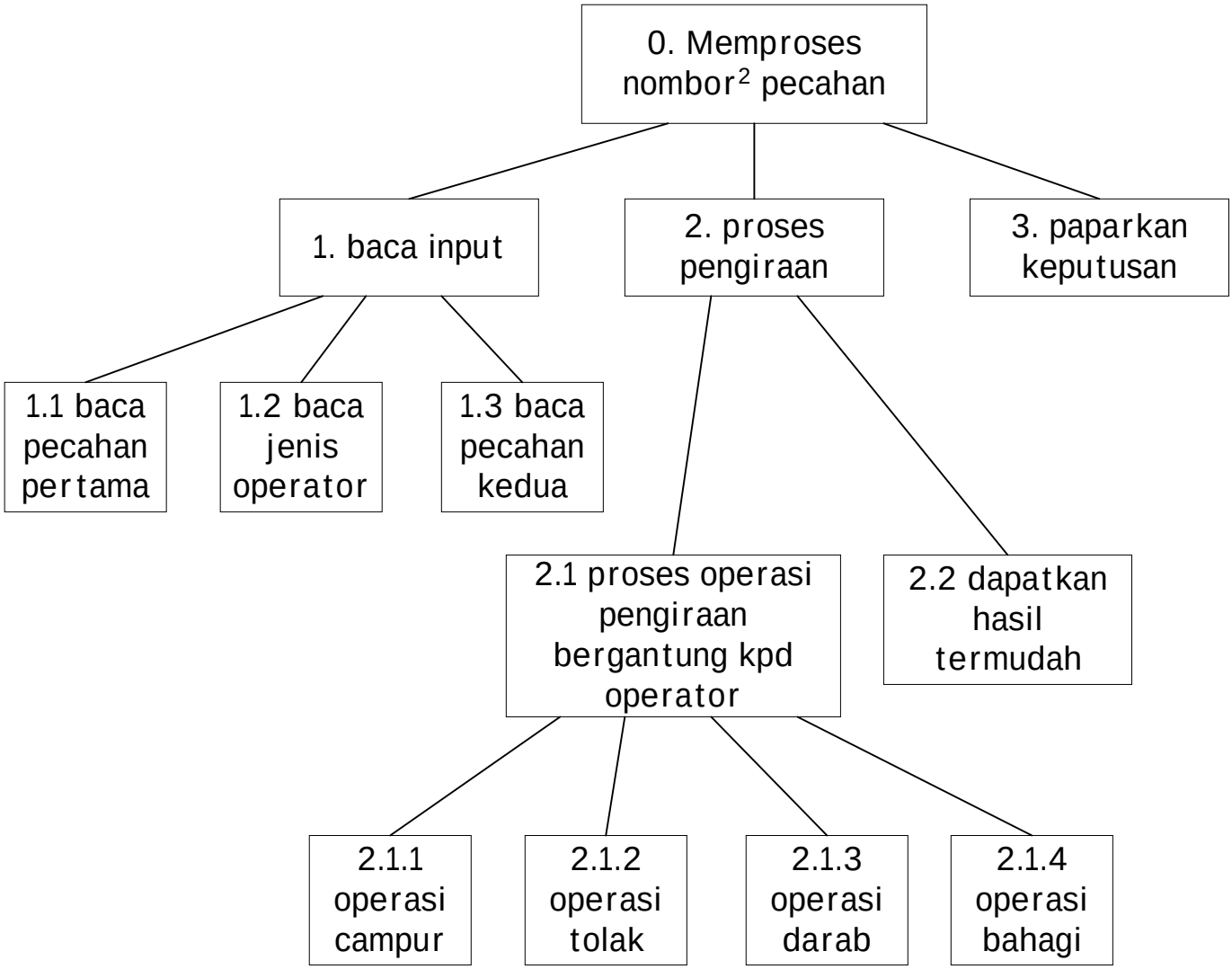
7.1 Mengapa Perlu Fungsi ? (samb....)

- Contoh PM menggunakan pendekatan atas-bawah

Anda diminta untuk menyediakan satu aturcara yang boleh mengira nombor^2 pecahan. Operasi pengiraan adalah operasi campur, tolak, darab dan bahagi. Hasil dari pengiraan tersebut mestilah dalam bentuk pecahan termudah.

Perhatikan contoh satu input & output kpd proses tersebut.





7.2 Cara Menulis Fungsi

Bentuk asas sesuatu fungsi:

```
jenis_pulangan  nama_fungsi (parameter_masukan)
{
        badan_fungsi;
        return(nilai);
}
```

- ***jenis_pulangan*** :
  Jenis (unsigned/signed char, int, float, long) output/pulangan yg dihasilkan oleh fungsi tersebut. Jika fungsi tiada output/ pulangan jenisnya mestilah **void**. Jika *jenis_pulangan* ditinggalkan, jenis **int** adalah dianggap.

- ***nama_fungsi*** :
  Nama fungsi. Nama ini akan digunakan utk memanggil/ melaksanakan fungsi tersebut. Nama fungsi mestilah mematuhi syarat[2] yg sama dgn pencam.

- ***parameter_masukan*** :
  Pembolehubah[2] yg menjadi masukan/input kpd fungsi tersebut bagi melakukan sesuatu kerja. Jika ada lebih dari satu parameter, mereka mesti dipisahkan oleh koma (,). **Jenis, bilangan** dan **aturan** bagi *parameter_masukan* mestilah sama semasa fungsi ditakrifkan dan digunakan.

- ***badan_fungsi*** :
  Terdiri dari satu atau lebih kenyataan[2] C termasuk pengisytiharan pembolehubah[2] yg melakukan sesuatu kerja. Cara menulisnya sama dgn cara menulis badan fungsi main().

- ***return (nilai)*** :
  Suruhan ini akan menghasilkan output/pulangan dari fungsi tersebut. *nilai* ialah sebarang nilai dari pembolehubah/pemalar tetapi jenisnya mestilah sama dgn *jenis_pulangan* yg diberikan di atas.
  Bagi fungsi jenis **void**, suruhan return() tidak ditulis.

7.2 Cara Menulis Fungsi (samb....)

Contoh 1:

a)
```
int campur (int x, int y)
{
    return(x+y);
}
```

b)
```
void cetak_mesej (void)
{
    printf("Hi\n");
}
```

c)
```
float kira_luas (float l, float p)
{
   float A;
   A = p*l;
    return(A);
}
```

Contoh 2:

Tulis satu aturcara utk mengira & memaparkan luas satu segiempat, satu bulatan atau satu segitiga atas pilihan pengguna. Saiz sisi^2 diberikan oleh pengguna melalui papan kekunci. Anggap saiz positif shj yg dimasukkan.
Tulis tiga fungsi utk mencari luas segi4, bulatan & segi3.
Pembolehubah2 jenis float.

Penyelesaian:

i) Analisis:

Input:

media : papan kekunci

data : pilih operasi, panjang & lebar @ jejari @ tapak & tinggi

Output:

media : skrin

data : luas segiempat @ luas bulatan @ luas segitiga

Proses:

luas segiempat = panjang x lebar

luas bulatan = π x jejari2

luas segitiga = (tapak x tinggi)/2

7.2 Cara Menulis Fungsi (samb....)

Penyelesaian ....:

ii) Kod pseudo I:

1. cetak menu pilihan operasi
2. baca pilihan
3. jika pilihan cari luas segiempat
   3.1 baca nilai panjang & lebar
   3.2 luas segiempat = panjang x lebar
4. jika pilihan cari luas bulatan
   4.1 baca nilai jejari
   4.2 luas bulatan = π x jejari^2
5. jika pilihan cari luas segitiga
   5.1 baca nilai tapak & tinggi
   5.2 luas segitiga = (tapak x tinggi)/2
6. cetak luas

ii) Kod pseudo II:

```
cetak menu pilihan operasi
baca pilihan
if pilihan = 1 then
        begin
          baca nilai panjang & lebar
          luas segiempat = panjang x lebar
        end
  else if pilihan = 2 then
        begin
          baca nilai tapak & tinggi
          luas segitiga = (tapak x tinggi)/2
        end
   else if pilihan = 3 then
        begin
          baca nilai jejari
          luas bulatan = π x jejari^2
        end
  else if pilihan luas segitiga then
        begin
          baca nilai tapak & tinggi
          luas segitiga = (tapak x tinggi)/2
        end
  else cetak amaran
  endif
  cetak luas
```

7.2 Cara Menulis Fungsi (samb....)

Penyelesaian ....:

iii) Aturcara:

fungsi luas_segi4

```
#include <stdio.h>

float luas_segi4 (void)
{
   float pjg, lbr;

   printf("masukkan saiz: panjang lebar\n");
   scanf("%f %f", &pjg, &lbr);
   return (pjg*lbr);
}
```

fungsi luas_segi3

```
float luas_segi3 (float tapak, float tinggi)
{
   float luas;

   luas =(tapak*tinggi)/2.0;
   return ( luas );
}
```

fungsi luas_bulat

```
float luas_bulat (float jejari)
{
   return ( 3.1415*jejari*jejari );
}
```

fungsi menu

```
void menu (void)
{
   printf("1: Luas segi4\n");   printf("2: Luas segi3\n");
   printf("3: Luas bulatan\n"); printf("pilih 1, 2 @ 3 >");
}
```

fungsi main

```
void main()
{
     float luas, j, tpk, tg;    int pilih;

     menu();   scanf ("%d", &pilih);
     if (pilih == 1)
        luas = luas_segi4 ();
     else if (pilih == 2) {
        printf("masukkan saiz: tapak lebar\n");
        scanf("%f %f", &tpk, &tg);
        luas = luas_segi3 (tpk, tg);
     } else if (pilih == 3 ) {
        printf("Saiz jejari ?: ");
        scanf("%f", &j);  luas = luas_bulat (j);
     } else   printf("pilih 1, 2 @ 3 shj\n");

     printf ("luas ialah %f\n", luas);
}
```

panggilan^2 fungsi

7.3 Prototaip Fungsi

- Jika rujuk semula penyelesaian mencari luas di ms 6, fungsi^2 yg dipanggil diletakkan sebelum fungsi yg memanggil, kes ini tidak memerlukan prototaip fungsi. Tapi jika sebaliknya kita memerlukan prototaip fungsi.

- Perhatikan 2 kes di bawah:

prototaip fungsi

tidak memerlukan prototaip fungsi

```
void fungsi1 (...)
{
     :
     :
}

void fungsi2 (...)
{
     :
   fungsi1 (...);
     :
}

void fungsi3 (...)
{
     :
     :
}


void main()
{
     fungsi1 (...);
     fungsi2 (...);
     fungsi3 (...);
}
```

memerlukan prototaip fungsi

```
void fungsi1 (...);
void fungsi2 (...);
void fungsi3 (...);

void main()
{
     fungsi1 (...);
     fungsi2 (...);
     fungsi3 (...);
}


void fungsi2 (...)
{
     :
   fungsi1 (...);
     :
}

void fungsi1 (...)
{
     :
     :
}

void fungsi3 (...)
{
     :
     :
}
```

- Syarat menulis prototaip fungsi
  * menyerupai baris pertama takrifan fungsi & ditambah semikolon.
  *nama fungsi, bilangan, susunan & jenis parameter mesti sama
  *nama parameter boleh ditinggalkan, jika tidak ditinggalkan boleh berbeza @ sama

7.3 Prototaip Fungsi (samb....)

Aturcara penyelesaian mencari luas di ms 6 dgn prototaip fungsi.

```
#include <stdio.h>

float luas_segi4 (void);
float luas_segi3 (float, float);
float luas_bulat (float jejari);
void menu (void);

void main()
{
     float luas, j, tpk, tg;    int pilih;

     menu();   scanf ("%d", &pilih);
     if (pilih == 1)
        luas = luas_segi4 ();
     else if (pilih == 2) {
        printf("masukkan saiz: tapak lebar\n");
        scanf("%f %f", &tpk, &tg);
        luas = luas_segi3 (tpk, tg);
     } else if (pilih == 3 ) {
        printf("Saiz jejari ?: ");
        scanf("%f", &j);  luas = luas_bulat (j);
     } else   printf("pilih 1, 2 @ 3 shj\n");
     printf ("luas ialah %f\n", luas);
}

float luas_segi4 (void)
{
   float pjg, lbr;
   printf("masukkan saiz: panjang lebar\n");
   scanf("%f %f", &pjg, &lbr);
   return (pjg*lbr);
}

float luas_segi3 (float tapak, float tinggi)
{
   float luas;
   luas =(tapak*tinggi)/2.0;
   return ( luas );
}

float luas_bulat (float jejari)
{
   return ( 3.1415*jejari*jejari );
}

void menu (void)
{
   printf("1: Luas segi4\n");   printf("2: Luas segi3\n");
   printf("3: Luas bulatan\n"); printf("pilih 1, 2 @ 3 >");
}
```

7.4 Parameter[2] dlm. Fungsi

- **Paramater formal** - nama & jenis parameter yang ditakrif semasa fungsi diisytiharkan.

Parameter formal

```
float luas_bulat (float jejari)
{
   return ( 3.1415*jejari*jejari );
}
```

- **Parameter sebenar** - nilai sebenar yang diberikan oleh fungsi pemanggil kpd parameter formal semasa fungsi itu dipanggil/ digunakan.

```
void main()
{
     :
     :
  scanf("%f", j);
  luas=luas_bulat ( j );
     :
     :
}
```

Pemindahan parameter

```
float luas_bulat (float jejari)
{
   return ( 3.1415*jejari*jejari);
}
```

Parameter sebenar

7.5 Panggilan & Pulangan Fungsi

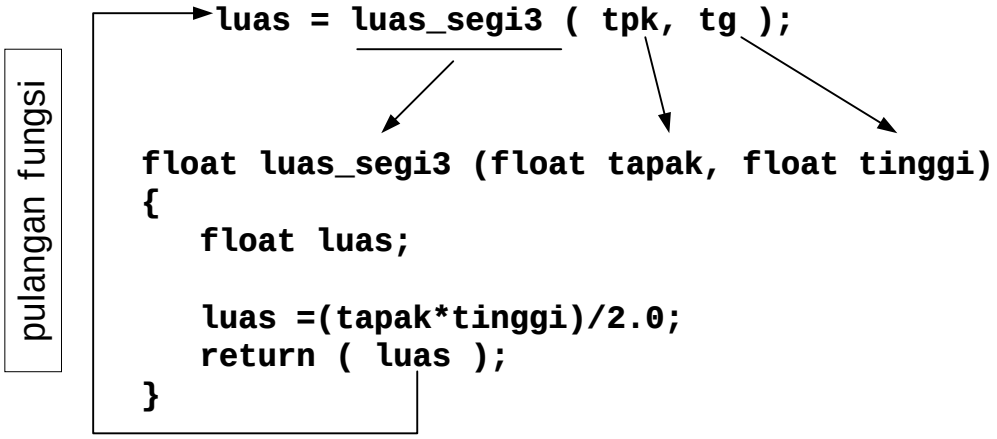
- Untuk melaksankan fungsi, ia perlu dipanggil menggunakan format tertentu.
  Fungsi boleh dipanggil oleh mana[2] fungsi luar atau fungsi yg sama (dinamakan rekursif).

- 3 perkara yg perlu diperhatikan utk memanggil fungsi:

Perkara	Syarat
1. nama fungsi	*nama mesti sama dgn nama fungsi
2. Data yg dihantar : parameter formal lwn. parameter sebenar	*jenis & bilangan mesti sama *mesti dalam turutan yg sama
3. pulangan fungsi	*jenis mesti sama

7.4 Panggilan & Pulangan Fungsi (samb....)

- Contoh panggilan fungsi **luas_segi3**.



- Satu fungsi tidak semestinya memulangkan nilai atau menghantar data. Contoh panggilan fungsi **menu**. yg tidak memulangkan nilai & tidak menghantar apa-apa data spt berikut:

```
menu ( );
```

```
void menu (void)
{
   printf("1: Luas segi4\n");   printf("2: Luas segi3\n");
   printf("3: Luas bulatan\n"); printf("pilih 1, 2 @ 3 >");
}
```

contoh[2] panggilan fungsi yg sah:

```
printf( "Luas segi3 = %f", luas_segi3(tpk,tg));
luassegi3 = luas_segi3(14.1,10.5);
luasbulat = luas_bulat(2.2 + j);
luassegi4_10 = luas_segi4() + 10.0;
```

contoh[2] kenyataan **return** yg sah:

```
return 0;      return luas;        return (luas);
return ( 3.1415*jejari*jejari );
```

- Bila satu fungsi dipanggil kenyataan di dlm badan fungsi akan dilaksanakan satu persatu sehingga tamat badan fungsi atau kenyataan return dilaksanakan. Contoh di bawah **fungsi1** kenyataan1 & 2 akan dilaksanakan sebelum kembali ke fungsi yg memanggil,ttp **fungsi2** hanya kenyataan1 shj dilaksanakan.

```
void fungsi1(void)
{
     kenyataan1;
     kenyataan2;
}
```

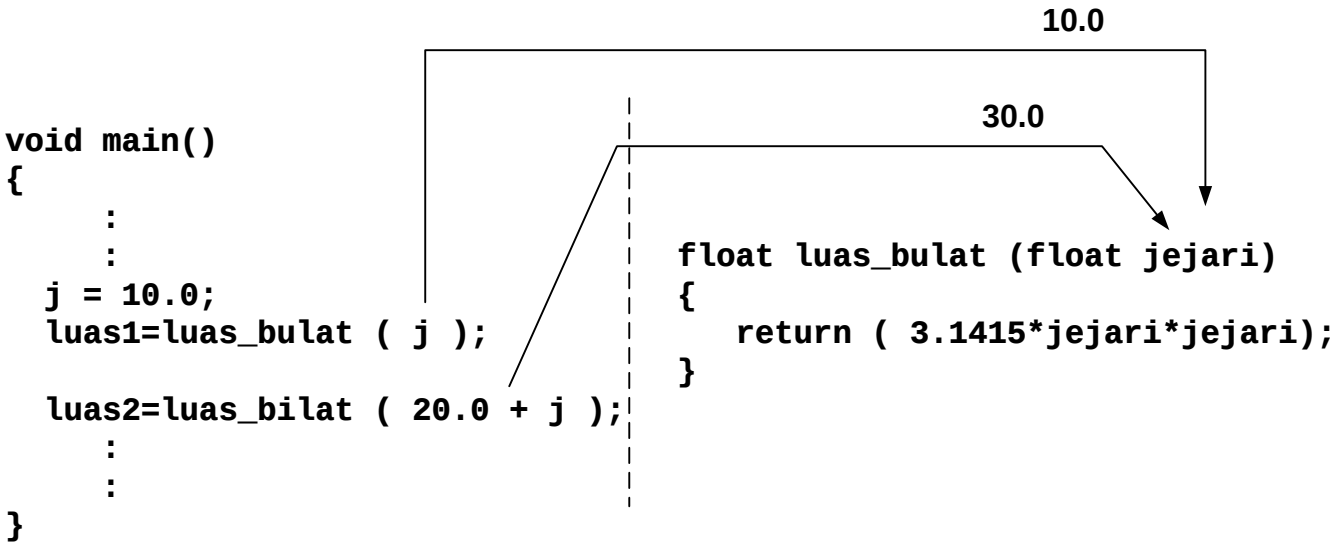
```
int fungsi2(void)
{
     kenyataan1;
     return 0;
     kenyataan2;
}
```

7.5 Penghantaran Nilai kpd Fungsi

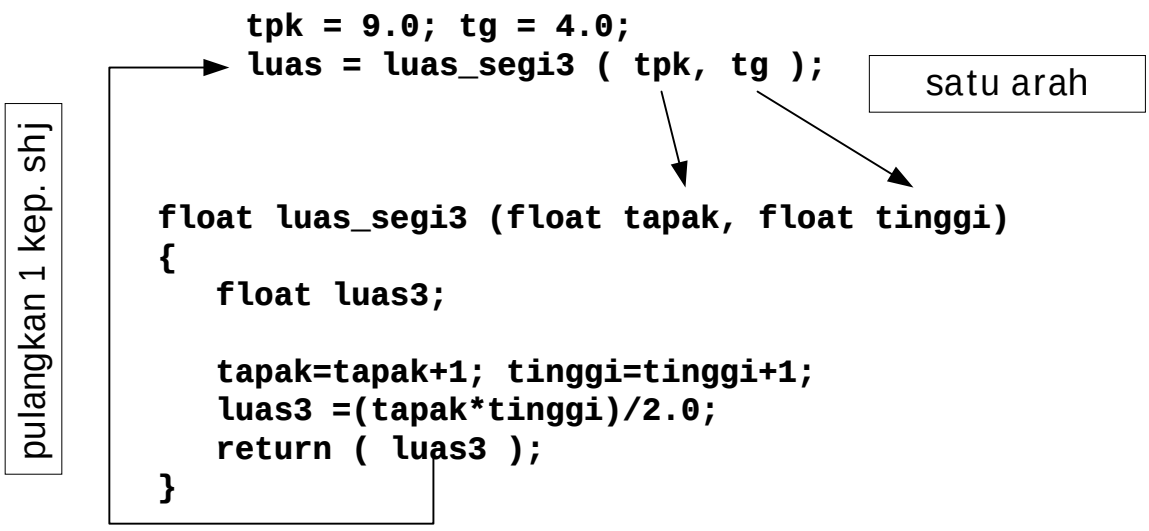
- Semasa fungsi dipanggil, penghantaran nilai oleh pemanggil boleh dibuat dlm 2 kaedah:
  - penghantaran nilai sebenar
  - pnghantaran alamat nilai

7.5.1 Penghantaran Nilai Sebenar

- Nilai sebenar akan diumpukan kpd parameter formal pada fungsi yang memanggil. Contohnya:



  - Penghantaran nilai sebenar, membolehkan satu @ beberapa nilai dihantar kpd sesuatu fungsi dlm bentuk satu arah. Jika kita perlukan fungsi yg dipanggil memberi nilai kpd fungsi yg memanggil, penyataan return boleh digunakan. Masalah pernyataan return hanya boleh pulangkan satu keputusan shj.
  Perhatikan contoh fungsi **luas_segi3** berikut:

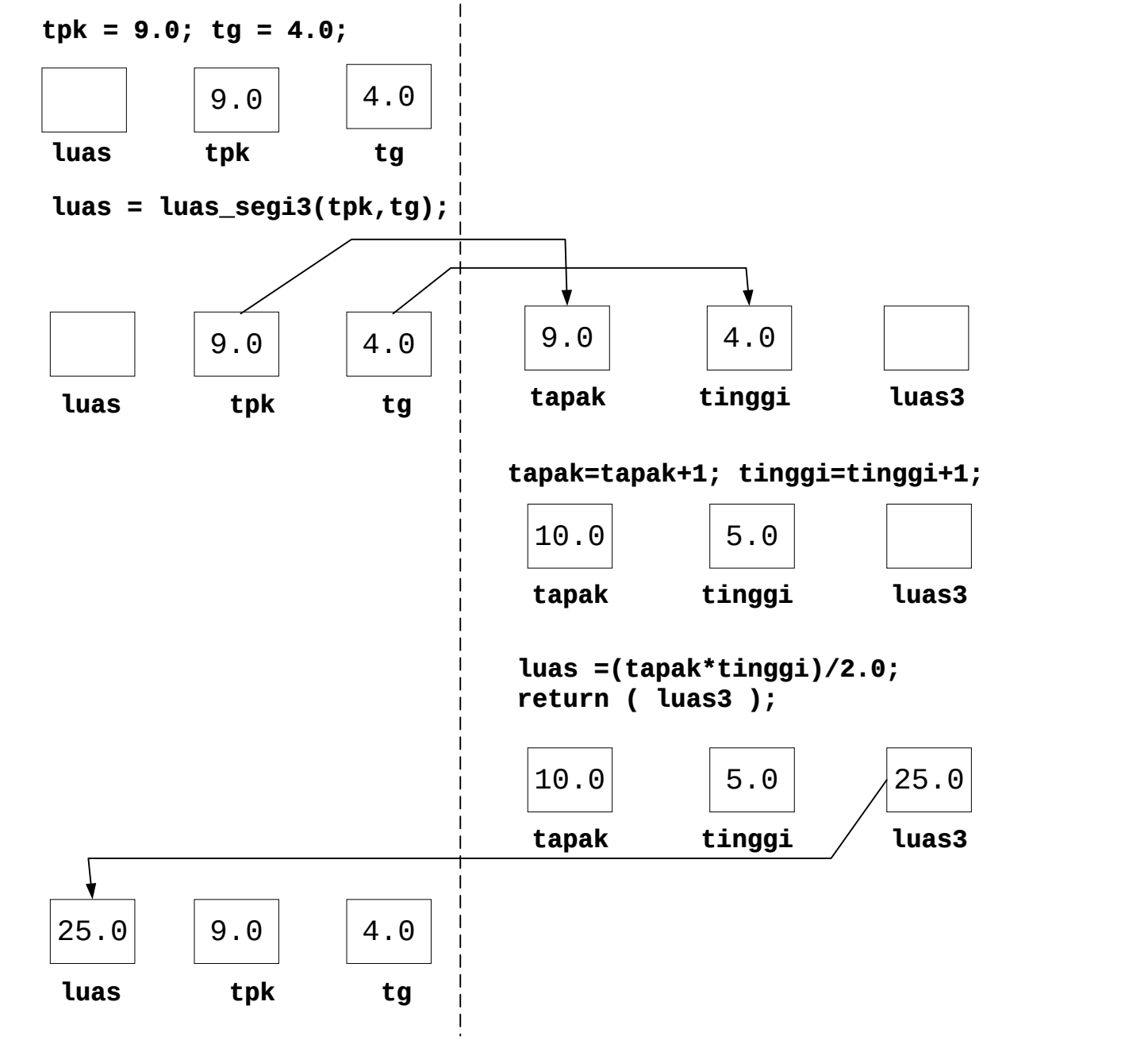


7.5.1 Penghantaran Nilai Sebenar (samb....)

- Apa yg berlaku dlm contoh panggilan fungsi **luas_segi3** ialah:

```
void main()
{
     :
     :
tpk = 9.0; tg = 4.0;

luas = luas_segi3(tpk,tg);

     :
     :
}
```

```
float luas_segi3(float tapak,float tinggi)
{
   float luas3;

   tapak=tapak+1; tinggi=tinggi+1;
   luas3 =(tapak*tinggi)/2.0;
   return ( luas3 );
}
```



7.5.2 Penghantaran Alamat Nilai

- Alamat di mana nilai disimpan di RAM akan dihantar kpd parameter formal pada fungsi yang memanggil. Alamat sesuatu pembolehubah boleh didapati dgn:

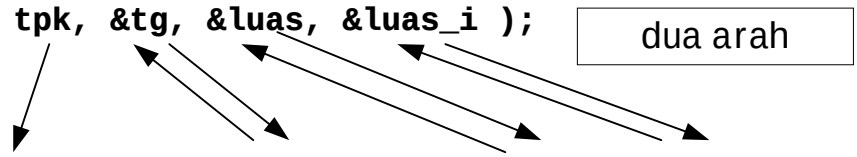
  &nama_pembolehubah

- Fungsi yg menerima alamat tersebut, boleh menggunakan alamat tersebut utk mendapatkan nilai (dengan menuding) yg terkandung dlm alamat yg dihantar. Nilai pembolehubah yg distorkan di dlm alamat boleh dirujuk dgn:

  ***nama_pembolehubah**

- Penghantaran alamat nilai, membolehkan satu @ beberapa nilai dihantar kpd sesuatu fungsi dlm bentuk dua arah. Jika kita perlukan fungsi yg dipanggil memberi nilai kpd fungsi yg memanggil, penyataan return tidak perlu digunakan dan lebih dari satu nilai boleh diberikan.
  Perhatikan contoh fungsi **luas_segi3** berikut:

```
luas = 0.0; luas_i = 0;
tpk = 9.0; tg = 4.0;
luas_segi3 ( tpk, &tg, &luas, &luas_i );
```




```
void luas_segi3 (float tapak, float *tinggi, float *l_f, int *l_i)
{
   tapak=tapak+1; *tinggi=*tinggi+1;

   *l_f =(tapak*(*tinggi))/2.0;
   *l_i = (int)(*l_f/2); /* bhg 2 & tukar kpd no. bulat */
}
```

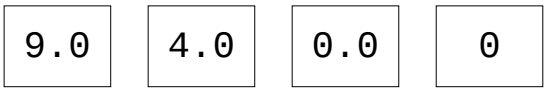
- perhatihan apa yg berlaku dlm fungsi ini di sebelah.

7.5.2 Penghantaran Alamat Nilai (samb....)

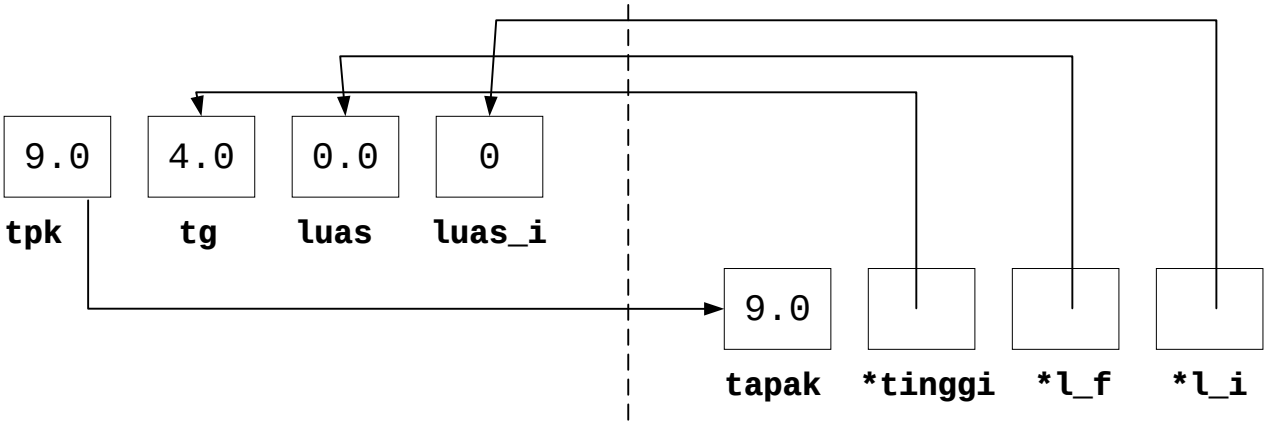
```
void main()
{
     :
luas = 0.0; luas_i = 0;
tpk = 9.0; tg = 4.0;

luas_segi3 ( tpk, &tg,
        &luas, &luas_i );

          :
}
```

```
void luas_segi3 (float tapak, float
       *tinggi, float *l_f, int *l_i)
{
   tapak=tapak+1; *tinggi=*tinggi+1;

   *l_f =(tapak*(*tinggi))/2.0;
   *l_i = (int)(*l_f/2);
    /* bhg 2 & tukar kpd no. bulat */
}
```

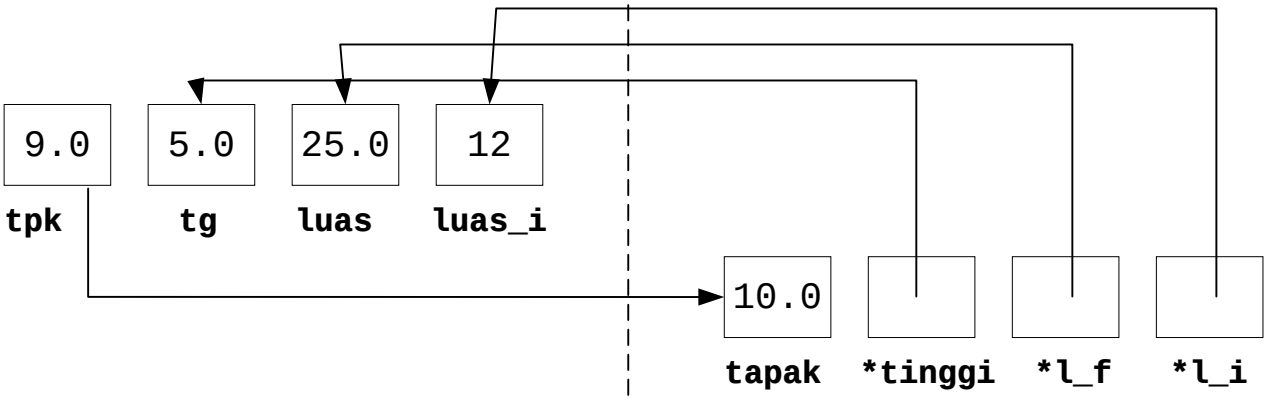
```
luas = 0.0; luas_i = 0;
tpk = 9.0; tg = 4.0;
```




```
luas_segi3 ( tpk, &tg,
        &luas, &luas_i );
```




```
tapak=tapak+1; *tinggi=*tinggi+1;
*l_f =(tapak*(*tinggi))/2.0;
*l_i = (int)(*l_f/2);
```



7.6 Skop Pembolehubah

- Pembolehubah yang diisytiharkan di dlm sesuatu fungsi dinamakan **pembolehubah setempat/lokal**.

- Pembolehubah lokal :
  1. boleh digunakan/dicapai oleh fungsi tersebut shj. Fungi^2 lain termasuk *main* tidak boleh mencapai pembolehubah^2 lokal tersebut
  2. hanya wujud semasa fungsi itu aktif/dipanggil

- Pembolehubah yang diisytiharkan di luar fungsi dinamakan **pembolehubah sejagat/global**.

- Pembolehubah global boleh digunakan/dicapai oleh semua fungsi kecuali jika pembolehubah global tersebut mempunyai nama yang sama dgn pembolehubah lokal.

Contoh 1: Pertimbangkan aturcara berikut.

```
#include <stdio.h>

float luas= 0;

float luas_segi4 (void)
{
   float pjg, lbr, luas;

   luas = pjg*lbr;
   return (luas);
}

float luas_bulat (float jejari)
{
   float luas;

   luas = 3.1415*jejari*jejari;
   return(luas);
}

void main()
{
     float j;

     :
     luas_bulat (j);
     :
}
```

7.6 Skop Pembolehubah (samb....)

- Pembolehubah **pjg, lbr** & **luas** adalah lokal kpd **luas_segi4().** Hanya boleh dicapai oleh fungsi **luas_segi4()** shj.
  Pembolehubah **luas** adalah lokal kpd **luas_bulat()** hanya boleh dicapai oleh fungsi **luas_bulat()** shj.
  Pembolehubah **j** adalah lokal kpd **main()** hanya boleh dicapai oleh fungsi **main()** shj.

- Pembolehubah **luas** di luar fungsi adalah pembolehubah global. Ia boleh dicapai oleh semua fungsi^2 kecuali fungsi^2 **luas_segi4** & **luas_bulat** kerana pembolehubah lokal mempunyai nama yang sama.

7.7 Skop Prototaip Fungsi

- Di ms 7, pada kes yg ke 2 prototaip fungsi perlu digunakan kerana fungsi memanggil diketak sebelum fungsi dipaggil. Seperti skop pembolehubah, skop prototaip fungsi juga boleh diisytihar secara sejagat dan setempat.

- Prinsipnya juga sama dgn skop pembolehubah. Protataip sejagat diisytihar di luar fungsi dan protataip setempat diisytiharkan di dlm fungsi. Skop panggilan protataip sejagat, boleh dipanggil dimana^2 fungsi di dlm aturcara. Skop panggilan protataip setempat, hanya boleh dipanggil dlm fungsi yg mengisytiharkannya.

- Perhatikan rangka aturcara mencari luas di ms sebelah:
  Dlm fungsi **main** prototaip fungsi **luas_segi3** diisytiharkan setempat & dlm fungsi **luas_segi4** prototaip fungsi **luas_bulat** juga diisytiharkan setempat, fungsi^2 ini hanya boleh dipanggil didlm fungi ini shj.
  Protataip fungsi **menu & luas_segi4** diisytihar sejagat, fungsi^2 boleh dipanggil oleh semua fungsi dlm aturcara tersebut.

7.7 Skop Prototaip Fungsi (samb...)

Contoh rangka aturcara mencari luas dgn prototaip fungsi yg pelbagai skop.

```
#include <stdio.h>

void menu (void);
float luas_segi4 (void);


void main()
{
     float luas_segi3 (float, float);
          :
     menu();
     luas = luas_segi4 ();
     luas = luas_segi3 (tpk, tg);
          :
}

float luas_segi4 (void)
{
     float luas_bulat (float jejari);
     :
     luas = luas_bulat (j);
     :

}

float luas_segi3 (float tapak, float tinggi)
{
     :
}

float luas_bulat (float jejari)
{
     menu();
     :
}

void menu (void)
{
     :
}
```

prototaip fungsi diisytiharkan sejagat

prototaip fungsi diisytiharkan setempat

Latihan:

1. Berdasarkan aturcara dibawah, jejakkan aturcara tersebut & berikan hasil output yg akan dicetak di skrin.

```
#include <stdio.h>

void fungsi_1(void);
int  fungsi_2(void);
void fungsi_3(int *a, int b);
int  fungsi_4(int a);
void fungsi_5(int *a, int *b,
              int *c);
int ikan = 10, ayam = 5;

void main ()
{
   int itek = 11,
   kambing = 4, kucing = 2;

   fungsi_1();
   itek++;
   ikan = ++kambing + 1;
   ikan++; ayam++;
   itek = itek + fungsi_2();
   fungsi_3(&kambing, itek);
   fungsi_5(&itek, &kambing,
            &kucing);
   ayam = ayam++ + ikan;
   ikan = ikan + itek++ -
          ++kucing;

   printf("\n Dalam fungsi
    main nilai itek=%d
    kambing=%d kucing=%d",
    itek, kambing, kucing);
   printf("\n Dalam fungsi
    main nilai global ikan=%d
    ayam = %d", ikan, ayam);
}

void fungsi_1(void)
{  int ayam = 0;
   ikan +=2;  ayam +=2;
   printf("\n Dalam fungsi_1
   nilai ikan=%d ayam=%d",
   ikan, ayam);
}
```

```
int  fungsi_2(void)
{  int kucing;
   kucing = ayam * 2;
   printf("\n Dalam fungsi_2
    nilai kucing = %d",
    kucing);
    return kucing;
}

void fungsi_3(int *badak, int
                gajah)
{  *badak = 2 * gajah;
   gajah  = 2 + (*badak);
   printf("\n Dalam fungsi_3
    nilai badak=%d gajah
    = %d", *badak, gajah);
}

int  fungsi_4(int keldai)
{  int kancil;
   kancil = ++keldai +
            ikan + ayam;
  printf("\n Dalam fungsi_4
   nilai keldai=%d kancil=
   %d",keldai, kancil);
   return (kancil);
}

void fungsi_5(int *rama2, int
          *lebah, int *kupu2)
{
   int kuda=6;
   *lebah = *lebah -
            fungsi_4(kuda);
   *rama2 = *lebah - *rama2 +
            *kupu2;
   *kupu2 = kuda +
            (*kupu2)++;
   printf("\n Dalam fungsi_5
   nilai rama2=%d lebah=%d
   kupu2 = %d", *rama2,
   *lebah, *kupu2);
}
```

8. TATASUSUNAN

8.0 Pengenalan

- Sebelum ini utk menyimpan 4 nilai perintang, 4 pembolehubah perlu diisytiharkan, contoh:

```
float perintang1, perintang2,perintang3,perintang4;
```

- Jika bil. perintang bertambah cthnya 20 pengisytiharan di atas menjadi rumit. Utk memudahkan pengisytiharan spt di atas **tatasusunan** boleh digunakan.

- Tatasusunan adalah satu struktur data mudah yang digunakan untuk:
  - menyimpan satu kumpulan nilai data
  - data^2 yg mempunyai jenis data yg sama
  - boleh dirujuk menggunakan satu pembolehubah shj

8.1 Mengisytihar & Merujuk tatasusunan

- Tatasusunan (tts) digunakan untuk menyimpan koleksi item@elemen data yg terdiri dari jenis data yg sama.

  Format pengisytiharan:

```
jenis_data nama_tts[saiz_elemen];
```

  Contoh pengisytiharan tatasusunan yg terdiri dari 5 elemen:

```
int x[5];
```

- Umpukan nilai semasa pengisytiharan boleh dilakukan menggunakan format berikut:

```
jenis_data nama_tts[saiz_elemen]={senarai_data};
```

  saiz_elemen boleh diabaikan jika umpukan awal diberikan kpd tts semasa pengisytiharan.

8.1 Mengisytihar & Merujuk tatasusunan (samb....)

- Contoh penisytiharan & umpukan awal tts x.

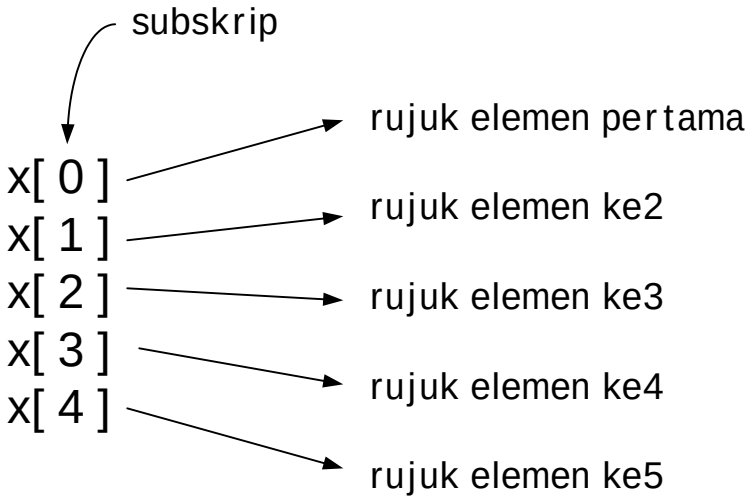
```
int x[] = { 10,  20, 30, 40, 50 };
          atau
int x[5] = { 10,  20, 30, 40, 50 };
```

- Struktur yg terhasil

x	10	20	30	40	50

- kumpulan data @ elemen yg disimpam dirujuk menggunakan pembolehubah & setiap item dlm kumpulan dirujuk melalui subskrip@index.

Contoh:



- Struktur tts x & subsrip setiap elemen

x	10	20	30	40	50

subsrip juga boleh digunakan spt berikut:

```
int x[5] = { 10,  20, 30, 40, 50 }
int a = 1, b = 3;

printf("%d %d %d", x[b], x[a], x[a+b]);
```

Output:

```
40  20  50
```

8.1 Mengisytihar & Merujuk tatasusunan (samb....)

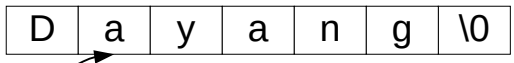
- Perhatikan struktur tts di bawah:

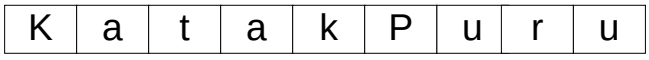
	y[0]	y[1]	y[2]	y[3]	y[4]	y[5]	y[6]	y[7]
y	7.7	5.6	4.1	2.5	1.0	7.7	18.7	9.9

```
y[3]=22.3;
tambah = y[5] + y[3];
tambah += y[2];
y[6] += 1.0;
y[7] = y[0] + y[3];
```

Selepas kenyataan[2] di atas bagaimanakah gambaran baru tatasusunan **y** & apakah nilai akhir **tambah ?**

- Contoh:






```
#include <stdio.h>
#include <string.h>

void main ()
{
 char nama[] = "Dayang";
 char katalaluan[] = {'k','a','t','a','k','P','u','r','u'};

 puts(nama);    puts(katalaluan);
 printf("%c %c\n",nama[3], katalaluan[3*2]);

 strcpy(nama, "Ali");
 katalaluan[0]='C'; katalaluan[1]='o'; katalaluan[2]='w';

 puts(nama);    puts(katalaluan);

 katalaluan[3]='\0';
 puts(katalaluan);
}
```

mencetak sehingga null (\0)

Output:

```
Dayang
katakPuru Dayang
a u
Ali
CowakPuru Ali
Cow
```

8.2 Menggunakan gelung & indek utk operasi tts

- Jika kita membuat operasi yg sama pd setiap elemen di dlm tts, gelung boleh digunakan. Perhatikan contoh berikut:

```
#define SAIZ 11

void main ()
{
   int i;
   int kuasa2[SAIZ];

   for (i=0; i< SAIZ ; i++) {
      kuasa2[i] = i * i;
   }
}
```

sama

```
kuasa2[0] = 0 * 0;
kuasa2[1] = 1 * 1;
kuasa2[2] = 2 * 2;
kuasa2[3] = 3 * 3;
kuasa2[4] = 4 * 4;
kuasa2[5] = 5 * 5;
kuasa2[6] = 6 * 6;
kuasa2[7] = 7 * 7;
kuasa2[8] = 8 * 8;
kuasa2[9] = 9 * 9;
kuasa2[10] = 10 * 10;
```

Tts yg terhasil:

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
kuasa2	0	1	4	9	16	25	36	49	64	81	100

Utk mencetak tts di atas:

```
for (i=0; i< SAIZ ; i++) {
   printf("%d ", kuasa2[i]);
}
```

- Utk menyalin satu struktur tts kpd pembolehubah lain (cth pembolehubah kuasa2 ke pembolehubah baru kuasa2ke2), gelung mesti digunakan. Jika kenyataan berikut digunakan akan memberi ralat sintaks:

```
kuasa2ke2=kuasa2;
```

Gelung digunakan:

```
for (i=0; i< SAIZ ; i++) {
   kuasa2ke2[i] = kuasa2[i];
}
```

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
kuasa2ke2	0	1	4	9	16	25	36	49	64	81	100

8.2 Menggunakan gelung & indek utk operasi tts (samb....)

Contoh 2:
Diberi vektor A dan B seperti berikut:

$$\mathbf{A} \begin{bmatrix} 4 \\ 5 \\ 12 \\ 7 \\ 10 \\ 6 \\ 4 \end{bmatrix} \qquad \mathbf{B} \begin{bmatrix} 3 \\ 7 \\ 2 \\ 2 \\ 11 \\ 10 \\ 1 \end{bmatrix}$$

Masalah:
1. mencari hasil tambah vector **A** & **B**, disimpan dlm vektor **C**
2. mencari vector D yang diberi oleh :

$$\vec{\mathbf{D}} = 2\vec{\mathbf{A}} + \vec{\mathbf{B}}$$

Penyelesaian:

```
#include<stdio.h>

void main (void)
{
   int A[] = {4,5,12,7,10,6,4},
       B[] = {3,7,2,2,11,10,1},
       C[7], D[7];
   int i;

   for (i=0; i<7; i++){
      C[i] = A[i] + B[i];
      D[i] = 2 * A[i] + B[i];
   }

   printf("VEKTOR C");
   for (i=0; i<7; i++){
      printf("\n|%4d|", C[i]);
   }

   printf("\n\nVEKTOR D");
   for (i=0; i<7; i++){
      printf("\n|%4d|", D[i]);
   }
}
```

Output:

```
VEKTOR C
|   7|
|  12|
|  14|
|   9|
|  21|
|  16|
|   5|

VEKTOR D
|  11|
|  17|
|  26|
|  16|
|  31|
|  22|
|   9|
```

8.2 Menggunakan gelung & indek utk operasi tts (samb....)

Contoh 3:
Membaca 8 nombor dari papan kekunci & memilih nombor terbesar & terkecil dari nombor tersebut.

Penyelesaian:

```
#include<stdio.h>
#define BILMAX 8

void main (void)
{
   int nom[BILMAX], besar, kecil, i;

   printf("Masukkan 8 nombor & pisahkan dengan jarak\n>");
   for (i=0; i<8; i++){
      scanf("%d", &nom[i]);
   }

   besar = nom[0]; kecil = nom[0];
   for (i=1; i<8; i++){
      if (nom[i]>besar){
         besar = nom[i];
      }
      if (nom[i]<kecil){
         kecil = nom[i];
      }
   }

   printf("\nNombor terbesar %d", besar);
   printf("\nNombor terkecil %d", kecil);
}
```

Intput & output :

```
Masukkan 8 nombor & pisahkan dengan jarak
>23 45 1 5 -8 11 66 7

Nombor terbesar 66
Nombor terkecil -8
```

8.3 Menghantar Tatasusunan Kepada Fungsi

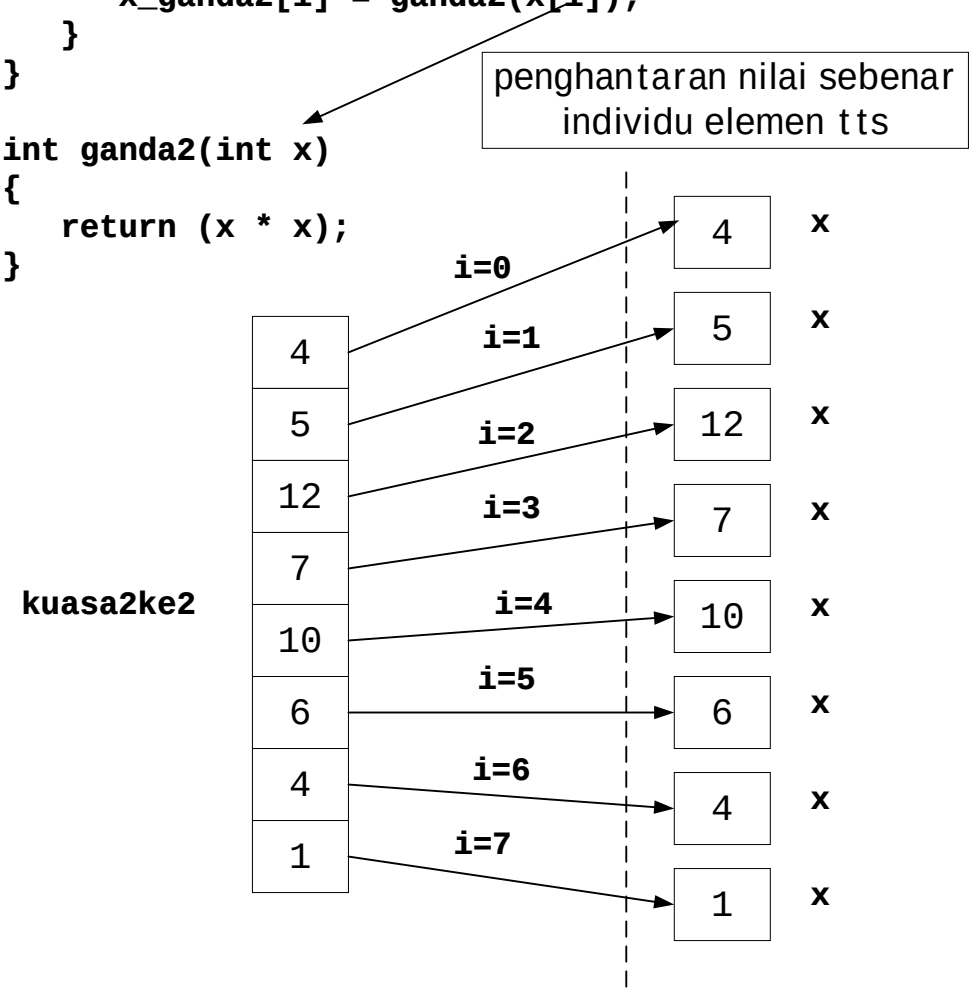
- Tatasusunan boleh dihantar ke fungsi samada:-
  - o individu elemen
  - o kesemua elemen tatasusunan

- Penghantaran individu elemen tatasusunan boleh dilakukan dalam :
  - o penghantaran nilai sebenar - satu arah.
  - o penghantaran alamat nilai - 2 arah.

8.3.1 Penghantaran individu elemen tts melalui nilai sebenar

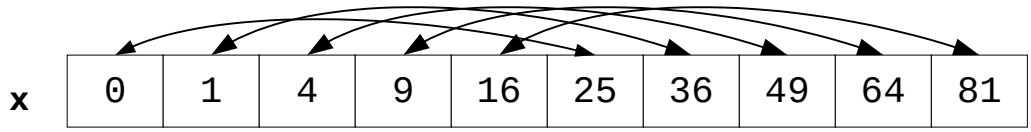
Contoh 1: Ganda 2 nombor dlm tts x

```
#include<stdio.h>
#define BILMAX 8

int ganda2(int x);

void main (void)
{
   int x[BILMAX] = {4,5,12,7,10,6,4,1}, x_ganda2[BILMAX];
   int i;

   for (i=0; i<BILMAX; i++) {
      x_ganda2[i] = ganda2(x[i]);
   }
}

int ganda2(int x)
{
   return (x * x);
}
```



8.3.1 Penghantaran individu elemen tts melalui alamat

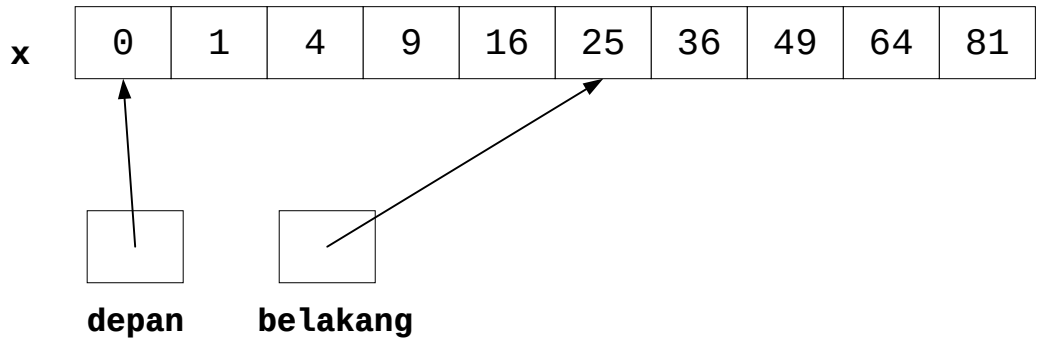
Contoh 2 : 10 nombor dlm tts, menukar 5 elemen hadapan dgn 5 elemen belakang. Operasi spt berikut:




```
#include<stdio.h>

void tukartempat(int *depan, int *belakang);

void main (void)
{
   int x[] = {0,1,4,9,16,25,36,49,64,81};
   int i;

   for (i=0; i<5; i++) {
      tukartempat(&x[i], &x[5+i]);
   }
}

void tukartempat(int *depan, int *belakang)
{
   int salinan;

   salinan = *depan;
   *depan = *belakang;
   *belakang = salinan;
}
```

penghantaran alamat nilai individu elemen tts

Contohnya bila **i=0** penuding **depan & belakang** tunjuk pd alamat yg dihantar.



8.3.3 Menghantar semua elemen Tts Kepada Fungsi

- Ia dihantar dgn menggunakan nama tatasusunan. Nama tatasusunan merupakan alamat lokasi pertama tatasusunan tersebut. Apabila alamat lokasi pertama dihantar bermakna keseluruhan unsur tatasusunan dapat dicapai.

- Oleh itu jika tts dihantar ke fungsi, secara autamatik ia adalah penghantaran melalui alamat (alamat pertama dlm tts). Penghantaran 2 arah sbg input & output.

Contoh 3: Tukarkan penyelesaian masalah contoh 2 kpd penghantaran tts (bukan individu elemen).

```
#include<stdio.h>

void tukartempat(int x[]);

void main (void) {
   int i, x[] = {0,1,4,9,16,25,36,49,64,81};

   tukartempat(x);
   for (i=0; i<10; i++){
      printf("%d ", x[i]);
   }

}

void tukartempat(int x[]) {

   int i, salinan;

   for ( i=0; i<5; i++ ) {
      salinan = x[i];
      x[i] = x[5+i];
      x[5+i] = salinan;
   }
}
```

Output:

```
25 36 49 64 81 0 1 4 9 16
```

8.3.3 Menghantar semua elemen Tts Kepada Fungsi (samb....)

- Untuk menghantar satu tts ke fungsi sbg input shj @ satu arah shj & memastikan elemen^2 tts tidak boleh diubah di dlm fungsi dipanggil, paramater formal fungsi disifatkan **const**.

Contoh 4: Perhatikan aturcara di bawah:

```
#include<stdio.h>

int terbesar(const int nom[8]);

void main (void) {
   int max,
       x[] = {0,1,4,9,16,25,36,49,64,81};

   max = terbesar(x);

   printf("Nombor terbesar %d", max);
}

int terbesar(const int nom[8]){
   int i, besar, salinan;

   besar = nom[0];
   for (i=1; i<8; i++){
      if (nom[i]>besar){
         besar = nom[i];
      }
   }
   return besar;
}
```

- Di contoh atas parameter diisytiharkan dengan const krn nilai^2 dlm tts tidak berubah semasa panggilan fungsi **terbesar** dibuat. Jika ada cubaan utk menukarkan nilai^2 tts di dlm fungsi dipanggil tersebut ralat SINTAKS akan terhasil.

- Saiz tts pd prototaip fungsi & kepala fungsi boleh diabaikan @ ditulis.

```
int terbesar(const int nom[8]);
                  @
int terbesar(const int nom[]);
```

8.3.3 Menghantar semua elemen Tts Kepada Fungsi (samb....)

- Spt telah dibincangkan, penghantaran semua tts ke fungsi dilakukan dgn menghantar alamat lokasi pertama tts tersebut. Apabila alamat lokasi digunakan fungsi memanggil boleh menggunakan penuding utk merujuk elemen tts di dlm fungsi (spt penghantaran melalui alamat dlm bab 7).

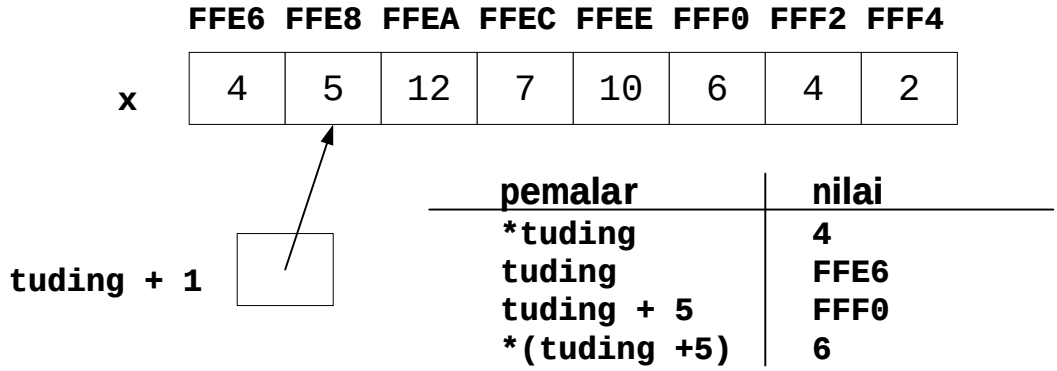
Contoh 5: Aturcara tolak 2 dari setiap elemen tts

```
#include<stdio.h>
#define BILMAX 8

void tolak2(int *tuding);

void main (void)
{
   int x[BILMAX] = {4,5,12,7,10,6,4,2};
   int i;

   tolak2(x);

   for (i=0; i<BILMAX; i++){
      printf("%d ", x[i]);
   }
}

void tolak2(int *tuding)
{
   int i;
   for (i=0; i<BILMAX; i++){
      *(tuding+i)-=2;
   }
}
```

Output:

```
2 3 10 5 8 4 2 0
```

Katakan alamat lokasi pertama tts x dlm RAM FFE6



pemalar	nilai
*tuding	4
tuding	FFE6
tuding + 5	FFF0
*(tuding +5)	6

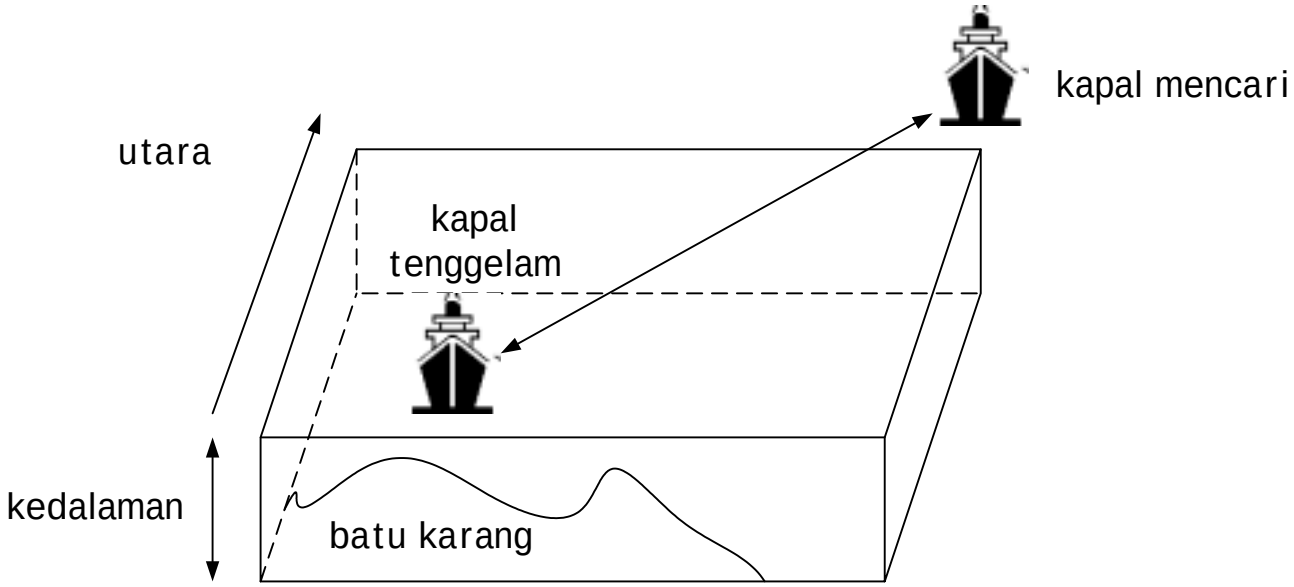
8.4 Tts Multi-Dimensi

- Tts 1 dimensi yg digunakan diseksyen lepas banyak memberi kemudahan kpd pengaturcara, ttp dlm masalah2 tertentu tts lebih dari 1 dimensi diperlukan. Contoh masalah^2 sains @ kejuruteraan yg memerlukan multi-dimensi tts adalah, matrik, ultrasound, radar dll.

Contoh 1: Utk mewakili matrik A tts multi-dimensi diperlukan:

$$A \begin{bmatrix} 1 & 2 & -1 & 4 \\ 2 & 4 & 3 & 5 \\ -1 & -2 & 6 & -7 \end{bmatrix}$$

Contoh 2: Radar utk mencari kapal tenggelam di dasar lautan tts multi-dimensi



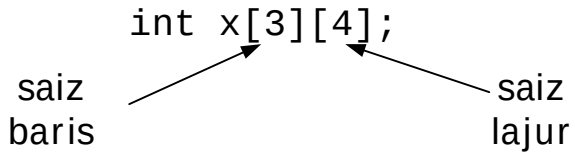
- Dlm contoh 1 tts 2 dimensi diperlukan & dlm contoh 2 tts 3 dimensi diperlukan.

8.5 Mengisytihar & Merujuk Tts Multi-Dimensi

- Format pengisytiharan tts multi-dimensi:

```
jenis_data nama_tts[saiz1][saiz2]....saiz[saizn];
```

Contoh pengisytiharan 2 dimensi:



- Gambarajah tts 2-dimensi yg terhasil & 2 indek@subskrip siperlukan utk merujuk kpd elemen[2] dlm tts.

x	[0]	[1]	[2]	[3]
[0]	x[0][0]	x[0][1]	x[0][2]	x[0][3]
[1]	x[1][0]	x[1][1]	x[1][2]	x[1][3]
[2]	x[2][0]	x[2][1]	x[2][2]	x[2][3]

- Umpukan nilai semasa pengisytiharan boleh dilakukan menggunakan format berikut:

```
jenis_data nama_tts[saiz][saiz]={senarai_data};
```

Contoh pengisytiharan 2 bg matrik A dr ms sebelum:

```
int A[3][4]= {1, 2, -1, 4, 2, 4, 3, 5,
              1, -2, 6, -7};
```

@ utk mudah dibaca

```
int A[3][4]= {{1, 2, -1, 4},
              {2, 4, 3, 5},
              {1, -2, 6, -7}};
```

@ saiz baris boleh diabailan

```
int A[ ][4]= {{1, 2, -1, 4},
              {2, 4, 3, 5},
              {1, -2, 6, -7}};
```

8.5 Mengisytihar & Merujuk Tts Multi-Dimensi (samb....)

- Gambarajah tts 2-dimensi yg terhasil dr pengisytiharan matrik A.

A

	[0]	[1]	[2]	[3]
[0]	1	2	-1	4
[1]	2	4	3	5
[2]	-1	-2	6	-7

```
A[1][2] = A[1][1] + 5;
tambah = A[1][2] + A[2][3];
tambah += A[1][3];
```

Selepas kenyataan[2] di atas bagaimanakah gambaran baru tatasusunan **y** & apakah nilai akhir **tambah ?**

- Utk mencapai @ memproses elemen[2] satu-persatu drp tts-2-dimensi gelung bersarang perlu digunakan. Elemen[2] boleh dicapai baris demi baris @ lajur demi lajur. Perhatikan cth berikut utk mencetak tts A:

```
#include <stdio.h>

void main ()
{
   int A[3][4]= {{1, 2, -1, 4},{2, 4, 3, 5},
                 {1, -2, 6, -7}};
   int baris, lajur;

   printf("Baris demi baris\n");
   for (baris=0; baris<3; baris++){
      for (lajur=0; lajur<4; lajur++)
         printf("%d\t",A[baris][lajur]);
      printf("\n");
   }

   printf("Lajur demi lajur\n");
   for (lajur=0; lajur<4; lajur++){
      for (baris=0; baris<3; baris++)
         printf("%d\t",A[baris][lajur]);
      printf("\n");
   }
}
```

Output:

```
Baris demi baris
1    2    -1   4
2    4    3    5
1    -2   6    -7
Lajur demi lajur
1    2    1
2    4    -2
-1   3    6
4    5    -7
```

8.5 Mengisytihar & Merujuk Tts Multi-Dimensi (samb....)

Contoh:
Diberi matrik A dan B seperti berikut:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 3 & 0 \\ -1 & 2 & 5 \end{bmatrix}$$

Masalah:
1. mencari hasil tambah matriks **A** & **B**, disimpan dlm matriks **C**
2. mencari hasil tambah matriks A & B, disimpan dlm matriks D

Penyelesaian:

```
#include<stdio.h>

void main (void)
{
   int A[2][3] = {1,2,3,0,1,4},
       B[2][3] = {2,3,0,-1,2,5},
       C[2][3], D[2][3];
   int baris, lajur;

   for (baris=0; baris<2; baris++){
      for (lajur=0; lajur<3; lajur++) {
         C[baris][lajur] = A[baris][lajur] +
                           B[baris][lajur];
         D[baris][lajur] = A[baris][lajur] -
                           B[baris][lajur];
      }
   }

   printf("MATRIKS C\n");
   for (baris=0; baris<2; baris++){
      for (lajur=0; lajur<3; lajur++)
         printf("%d\t",C[baris][lajur]);
      printf("\n");
   }
   printf("\nMATRIKS D\n");
   for (baris=0; baris<2; baris++){
      for (lajur=0; lajur<3; lajur++)
         printf("%d\t",D[baris][lajur]);
      printf("\n");
   }
}
```

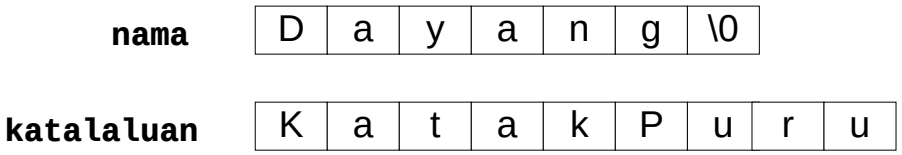
Output:

```
MATRIKS C
3    5    3
-1   3    9

MATRIKS D
-1   -1   3
1    -1   -1
```

8.6 Tts Rentetan

- Jenis data rentetan tidak disediakan oleh bhs C, oleh itu rentetan diwakili oleh tts aksara. Contoh ms 3 kota telah lihat perbezaan umpukan nilai awal rentetan & aksara[2] kpd tts.

```
char nama[] = "Dayang";
char katalaluan[] = {'k','a','t','a','k','P','u','r','u'};
```



- Rentetan banyak digunakan dlm penyelesaian masalah sains & kej. spt kimia $C_{12}H_{22}O_{11}$. Sebelum ini kita hanya melihat beberapa operasi mudah dgn rentetan spt:

```
printf("Cetak Saya\n");
printf("Nama Saya %s\n", nama);
puts(nama);
gets(namakawan);
```

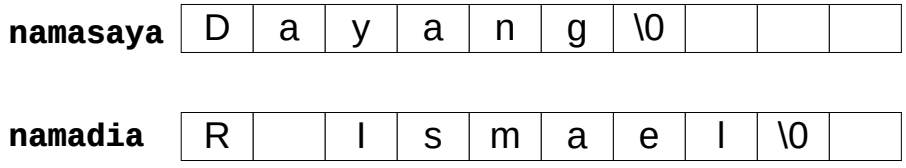
Utk memudahkan operasi[2] ke atas rentetan, beberapa fungsi disediakan dlm perpustakaan C **`string.h`**. Kita akan lihat fungsi:

1. **`strcpy`**
2. **`strcmp`**
3. **`strlen`**

1. **`strcpy`**

Menyalin elemen2 rentetan ke dlm tss aksara.

```
char namasaya[10], namadia[10], nama[] = "Dayang";
strcpy(namasaya, nama);
strcpy(namadia, "R Ismael");
```

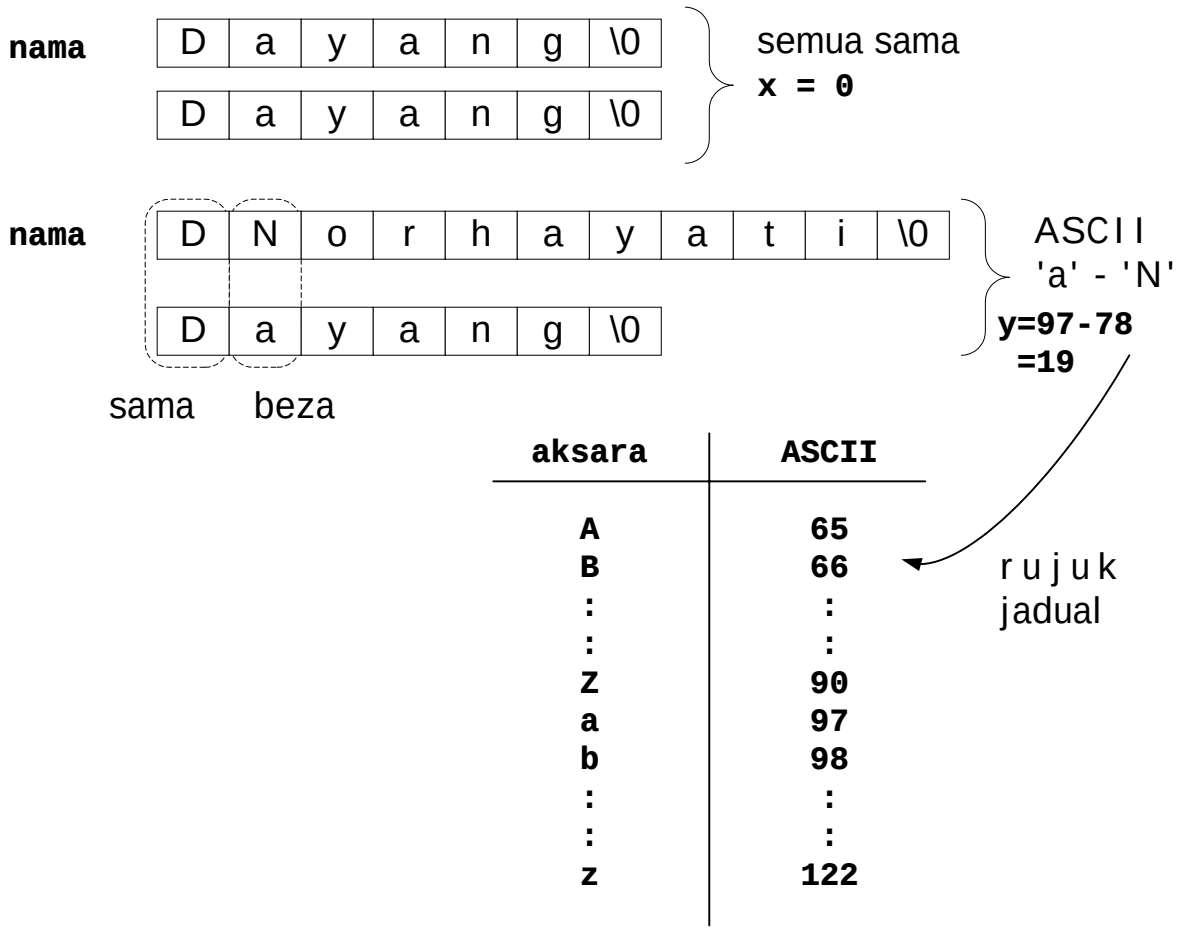


8.6 Tts Rentetan (samb....)

2. **strcmp**

Membandingkan nilai 2 rentetan berdasarkan kpd nilai ASCII, memulangkan integer 0 jika sama & memulangkan nilai perbezaan jika berbeza.

```
char nama[] = "Dayang"; int x, y;
x = strcmp(nama, "Dayang");
y = strcmp(nama, "DNorhayati");
```



3. **strlen**

Mengira bilangan aksara pada satu rentetan.

```
char nama[] = "Dayang";
printf("%d %d", strlen(nama), strlen("Dayang Norhayati"));
```

Output:

```
6 16
```

8.6 Tts Rentetan (samb....)

Contoh: Rentetan dgn 2 dimensi tts.

```
#include<stdio.h>
#include<string.h>

void main ()
{
   char nama[][8] = { "Ismael", "Munirah",
                      "Adilah", "Adibah"};
   int i, beza01, beza23;

   printf("ANALISA NAMA\n");
   for (i=0; i<4; i++) {
      puts(nama[i]);
      printf("Panjang nama %d\n",strlen(nama[i]));
   }

   beza01 = strcmp(nama[0], nama[1]);
   beza23 = strcmp(nama[2], nama[3]);

   printf("\nBeza nama %s & %s ialah %d\n",
                              nama[0], nama[1], beza01 );
   printf("Beza nama %s & %s ialah %d\n",
                              nama[2], nama[3], beza23 );

   strcpy(nama[2], "cahaya");

   for (i=0; i<4; i++) {
      puts(nama[i]);
   }
}
```

Output:

```
ANALISA NAMA
Ismael
Panjang nama 6
Munirah
Panjang nama 7
Adilah
Panjang nama 6
Adibah
Panjang nama 6

Beza nama Ismael & Munirah ialah -
4
Beza nama Adilah & Adibah ialah 10
Ismael
Munirah
cahaya
Adibah
```

tts terhasil dr perlaksanaan aturcara di atas

nama

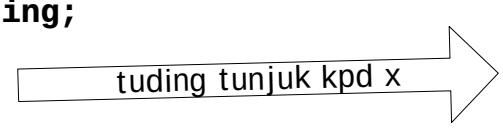
I	s	m	a	e	l	\0	
M	u	n	i	r	a	h	\0
c	a	h	a	y	a	\0	
A	d	i	b	a	h	\0	

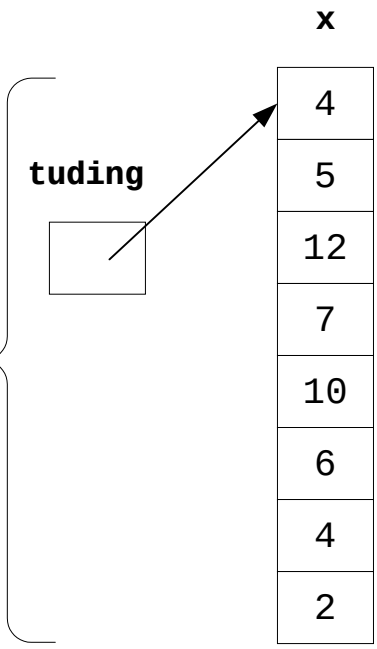
8.6 Tts & Penuding

- Prinsip tts & penuding telah dipelajari secara tidak lansung di seksyen penghantaran tts kpd fungsi ms 11. Prinsip yg sama digunakan utk merujuk elemen[2] tts menggunakan penuding.

  Perhatihan contoh2 berikut - mengubah cth 4 & 5 dr ms 10 & 11 kpd tts ditunjuk dgn penuding:

```
#include<stdio.h>
#define BILMAX 8

void main (void)
{
   int x[BILMAX] = {4,5,12,7,10,6,4,2};
   int i, *tuding;

   tuding = x;

   for (i=0; i<BILMAX; i++){
      *(tuding+i)-=2;
   }
   for (i=0; i<BILMAX; i++){
      printf("%d ", x[i]);
   }
}
```

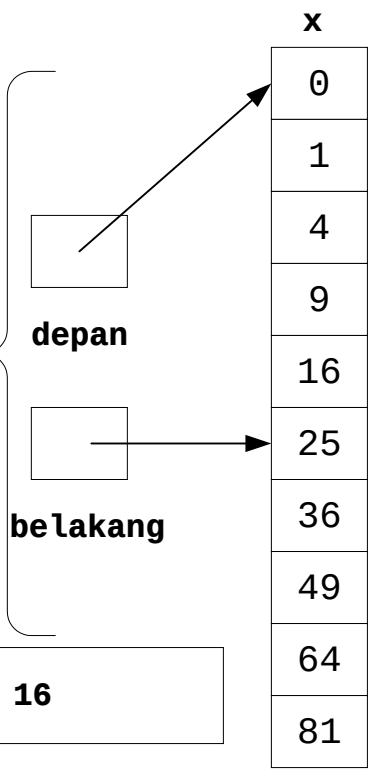




Output:

```
2 3 10 5 8 4 2 0
```

```
#include<stdio.h>
void main (void) {
   int i, salinan,
       x[] = {0,1,4,9,16,25,36,49,64,81};
   int *depan, *belakang;

   depan=x; belakang=x+5;

   for ( i=0; i<5; i++ ) {
      salinan = *depan;
      *depan = *belakang;
      *belakang = salinan;
      depan++; belakang++;
   }
   for (i=0; i<10; i++){
      printf("%d ", x[i]);
   }
}
```



Output:

```
25 36 49 64 81 0 1 4 9 16
```

9. STRUKTUR, UNION & PENOMBORAN

9.0 Pengenalan

- **Struktur –**
  - satu kumpulan pembolehubah yg dikumpulkan & dirujuk di bawah satu nama @ pencam.
  - jenis data yang ditakrif oleh pengguna
  - data terdiri dari jenis yang berlainan yang membentuk maklumat mengenai sesuatu data
  - berbeza dengan tatasusunan yang terdiri dari jenis data yang sama
- **Pengisytiharan:**

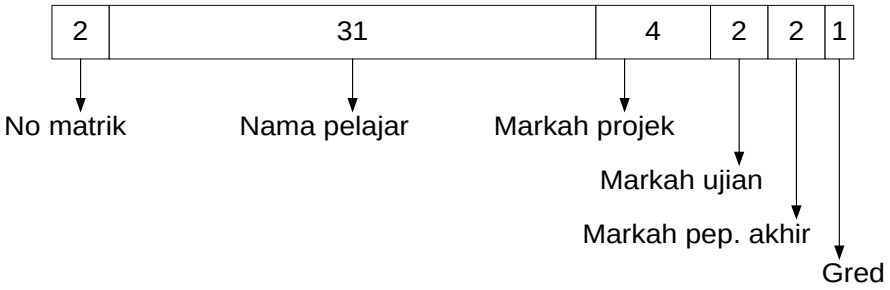
```
struct <nama_struktur>
{
    <jenis_data> <nama_unsur_1>;
    <jenis_data> <nama_unsur_2>;
    :
    <jenis_data> <nama_unsur_n>;
} <senarai_p'ubah_struktur>;
```

- Cth:

Seorang pensyarah hendak menentukan gred kepada setiap pelajarnya pada setiap semester. Pelajar-pelajar telah menjalani projek, ujian dan peperiksaan akhir.

Nama Data	***Jenis Data***	***C***
No matrik	Nilai integer	int
Nama pelajar	Tatasusunan aksara 31	char [31]
Markah projek	Nilai titik terapung	float
Markah ujian	Nilai integer	int
Markah pep. akhir	Nilai integer	int
Gred	Aksara	char

- Saiz ingatan yg diperlukan utk struktur ini dlm bait:



Cth pengisytiharan i

```
struct
{
    int no_metrik;
    char    nama[31];
    float   markah_projek;
    int markah_ujian;
    int markah_final;
    char gred;
} rekod_pelajar;
```

Cth pengisytiharan ii

```
struct pelajar
{
    int no_metrik;
    char    nama[31];
    float   markah_projek;
    int markah_ujian;
    int markah_final;
    char gred;
} rekod_pelajar;
```

Cth pengisytiharan iii

```
struct pelajar
{
    int no_metrik;
    char    nama[31];
    float   markah_projek;
    int markah_ujian;
    int markah_final;
    char gred;
};
struct pelajar rekod_pelajar;
```

- Satu lagi pilihan pengisytiharan struktur menggunakan typedef, cth:

```
typedef struct
{
    int no_metrik;
    char    nama[31];
    int markah_projek1;
    int markah_projek2;
    int markah_projek3;
    int markah_ujian1;
    int markah_ujian2
    int markah_final;
    chargred;
} JENISPELAJAR;

JENISPELAJAR rekod_pelajar;
```

9.1 Umpukan nilai awal kpd struktur

```
struct pelajar
{
    int     no_metrik;
    char    nama[31];
    float   markah_projek;
    int     markah_ujian;
    int     markah_final;
    chargred;
} rekod_pelajar = {722, "Maimunah Muhamad", 30.5,
10, 50, 'B'};
```

Atau

```
struct pelajar
{
    int no_metrik;
    char    nama[31];
    float markah_projek;
    int markah_ujian;
    int markah_final;
    char gred;
};
struct pelajar rekod_pelajar = {722, "Maimunah
Muhamad", 30.5, 10, 50, 'B'};
```

9.2 Operasi ke atas unsur struktur

- Setelah diisytiharkan, unsur struktur boleh dirujuk spt berikut:

```
<p'ubah_struktur>.<nama_unsur>
```

- Contoh cetakan nama pelajar dibuat:

```
printf("%s", rekod_pelajar.nama);
```

- Contoh mengumpukan nilai ke no_metrik:

```
rekod_pelajar.no_metrik = 5443;
```

- Untuk merujuk elemen tertentu didlm tatasusuan *subskrip @ indek* perlu didunakan. Cth menyalin nama pelajar:

```
struct pelajar p1, p11;

for (i=0; i<31; i++)

        p11.nama[i] = p1.nama[i];
```

Latihan

Tulis satu aturcara yang menggunakan struktur utk bagi tujuan penempahan tiket kapal terbang. Minimum data yang perlu ada adalah : Nama penumpang, tempat berlepas, destinasi,no. penerbangan, jumlah yg. perlu dibayar, status bayaran [telah dibayar|belum dibayar] dan umur penumpang.

9.2 Struktur Bersarang

- ahli struktur boleh terdiri dari struct yg lain.
- Perhatikan pengisytiharan struktur berikut:

```
struct struktur_pelajar
{
    char    nama[31];
    int no_metrik;
    char kod_fakulti[5];
    int umur;
    char jantina;
    int markah_projek1;
    int markah_projek2;
    int markah_projek3;
    int markah_ujian1;
    int markah_ujian2
    int markah_final;
    char gred;
} rekod_pelajar;
```

- menstrukturkan semula rekod pelajar menggunakan pengistiharan supaya data boleh diklasifikasi mengikut kategorinya:
    - peribadi pelajar
    - markah pelajar

Cara 1:

```
struct peribadi_pelajar
{
    char    nama[31];
    int no_metrik;
    char kod_fakulti[5];
    int umur;
    char jantina;
}

struct markah_pelajar
{
    int markah_projek1;
    int markah_projek2;
    int markah_projek3;
    int markah_ujian1;
    int markah_ujian2
    int markah_final;
}

struct pelajar
{
    struct peribadi_pelajar  peribadi;
    struct markah_pelajar markah;
    char gred;
};
```

Cara 2:

```
struct pelajar
{
     struct
     {
          char     nama[31];
          int no_metrik;
          charkod_fakulti[5];
          int umur;
          charjantina;
     } peribadi;

     struct
     {
          int markah_projek1;
          int markah_projek2;
          int markah_projek3;
          int markah_ujian1;
          int markah_ujian2
          int markah_final;
     } markah;

     chargred;
};
```

- Jika 2 p'ubah diisytiharkan:

```
struct pelajar p1, p2;
```

Kenyatan[2] berikut ada adalah sah:

```
p1.peribadi.umur = p2.peribadi.umur;

p1=p2;

p1.peribadi.kod_fakulti[4] = p2.peribadi.nama[9];

strcpy(p1.peribadi.nama,"Amirah Iman");

p1.gred = 'A';
```

9.2.1 Umpukan nilai awal kpd struktur

- Cth:

```
struct pelajar p1 = {{“Maimunah Muhamad”, 722,
     FS, 20, P},{10, 10, 20,0,12,60}, ‘C’};
```

Latihan

Dari latihan 1.3, tambahkan fungsi berikut. Setiap penempah boleh menempah lebih dari 1 tiket utk penerbangan yg sama, ttp utk orang yg berbeza. Dengan menggunakan struktur bersarang tambahkan data: no. telepon penempah, nama penempah, tarikh & masa tempahan dibuat.

9.3 P'ubah struktur jenis tatasusunan

- Penisytiharan:

```
struct <n_struk> <n_tts>[dms];
```

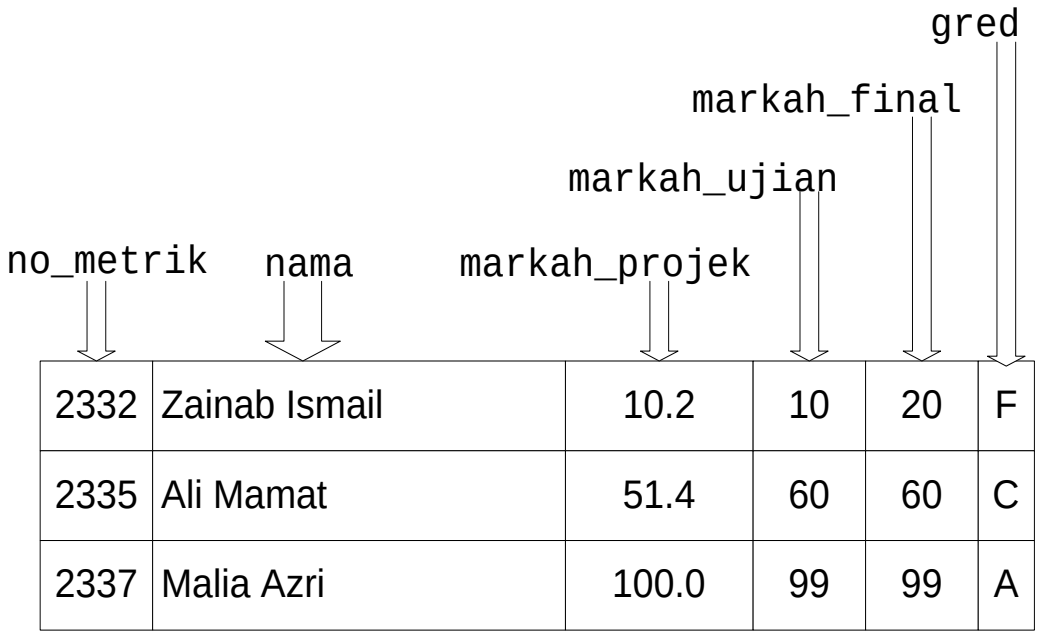
- cth pengisytiharan:

```
struct pelajar
{
    int no_metrik;
    char    nama[31];
    float markah_projek;
    int markah_ujian;
    int markah_final;
    chargred;
};
struct pelajar rekod_pelajar[3];
```

- Saiz tts boleh diabaikan jika nilai awal diber, cth:

```
struct pelajar rekod_pelajar[] = {
        {2332, "Zainab Ismail", 10.2, 10, 20,
        'F'},
        {2335, "Ali Mamat", 51.4, 60, 60, 'C'},
        {2337, "Malia Azri", 100.0, 99, 99, 'A'}
};
```

- Tts yg terbentuk:



2332	Zainab Ismail	10.2	10	20	F
2335	Ali Mamat	51.4	60	60	C
2337	Malia Azri	100.0	99	99	A

- Cth memasukkan maklumat kedlm tts struktur:

```
struct pensyarah_fsksm
    {   int no_pekerja;
        struct
        {   char nama_awal[15];
            char nama_akhir[15];
        } nama;
        int umur;
        float gaji;
    } pensy[100];
```

```
for (j=0; j<100; j++)
{
  printf("\nmasukkan no pek, nama awal, nama
      akhir umur dan gaji %d\n",j+1);
  scanf("%d", pensy[j].no_pekerja);
  gets(pensy[j].nama.nama_awal);
  gets(pensy[j].nama.nama_akhir);
  scanf("%d", pensy[j].umur);
  scanf("%f",pensy[j].gaji);
}
```

Latihan

Dari latihan 2.2, tambahkan fungsi berikut. Boleh menyimpan 20 data dlm tts untuk tujuan:

- kiraan bil penumpang dlm pesawat tertentu
- jumlah tambang yg telah dibayar & belum dibayar.

9.4 P'ubah struktur jenis Penuding

- Pengisytiharan:

```
struct <n_struk> *<n_penuding>;
```

- Perhatikan cth berikut:

```
struct pensyarah_fsksm
    {   int no_pekerja;
        struct
        {   char nama_awal[15];
            char nama_akhir[15];
        } nama;
        int umur;
        float gaji;
}dayang = {777,{"Dayang", "Jawawi"},30, 1500};

struct pensyarah_fsksm *saya,  *kami;
```

- Cth umpukan:

```
saya = &dayang;
```

- Cth merujuk ahli struktur menggunakan penuding:

```
(*saya).no_pekerja

@

saya->no_pekerja
```

- Sambungan cth di atas:

```
struct pensyarah_fsksm pblk162[]= {
   {777,{“Dayang”, “Jawawi”},20, 1500},
   {666,{“Normal”, “Jusoh”},30, 2500}
}
printf(“Alamat #1: %u #2: %u\n”, &pblk162[0],
      &pblk162[1]);
kami = &pblk162[0];
printf(“Penuding #1: %u #2: %u\n”, kami, kami+1);
printf(“1. Kami adalah %s:”, kami-
      >nama.nama_awal);
kami++;
printf(“2. Kami adalah %s:”, kami-
      >nama.nama_awal);
```

- Lengkapkan rajah berikut & tulis output dari cth aturcara di atas.

kami

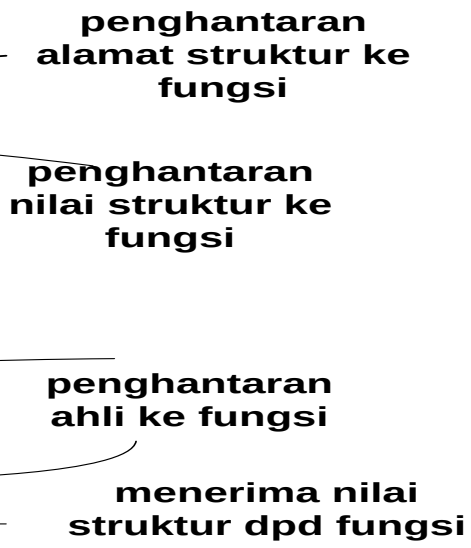
pblk162[0]	777	Dayang 1500
pblk162[0]	666	Normal. 2500

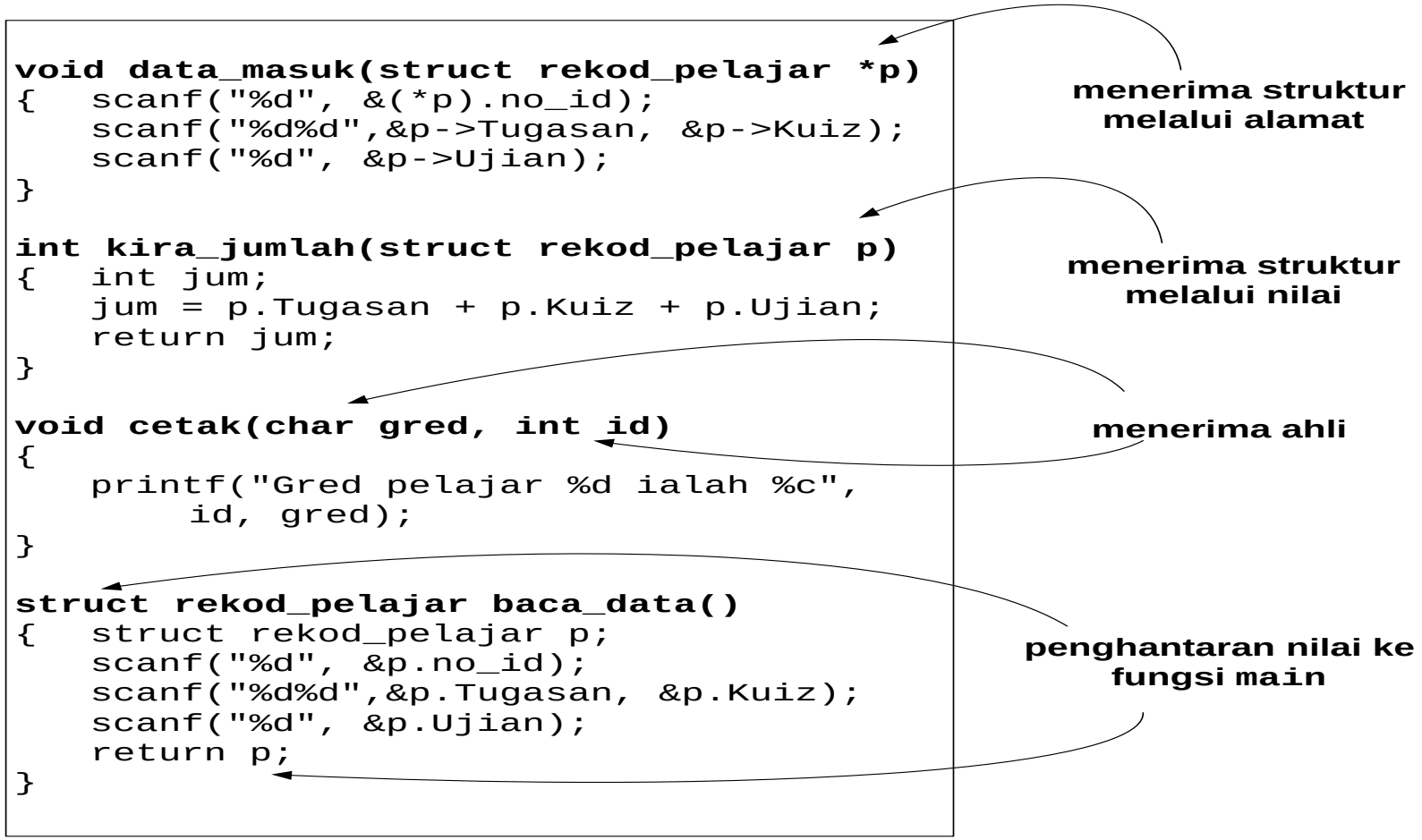
Latihan

Dari latihan 3.1, tukar tts kepada penuding.

9.5 Operasi^2 struktur dgn fungsi

```
struct rekod_pelajar
       {   int no_id;
           int Tugasan;
           int Kuiz;
           int Ujian;
           char gred;
   };

main()
{
   struct rekod_pelajar amir,naim;
   int jum_mar;
:
:
data_masuk(&amir);
jum_mar=kira_jumlah(amir);

if (jum_mar >= 60)
   amir.gred = 'L';
else
   amir.gred = 'G';

cetak(amir.gred, amir.no_id);

naim = baca_data();
```





Latihan

Dari latihan 3.1@ 4.1, anda dikehendaki membentukan sekurang-kurangnya 3 fungsi berikut:

- baca_masukan
- pengiraan
- cetak_laporan

9.6 Struktur: Apa seterusnya?

- Jelas struktur amat berguna utk membentuk data jenis baru yg terdiri dpd jenis^2 data asas.
- Bentuk data yg berbagai jenis ini mewakili data sebenar dlm. penyelesaian masalah dgn komputer.
- Bentuk data ini akan distukturkan mengikut struktur-struktur tertentu spt tindanan, baris gilir & pehohon sebelum diproses.

9.7 Union

- berfungsi lebih kurang sama seperti struktur
- diguna untuk menyimpan jenis data yang berlainan didalam lokasi memori yang sama
- boleh menjimatkan ruang bila pembolehubah tidak digunakan pada satu masa
- hanya satu ahli dari satu jenis data diguna pada satu masa
- pengguna menentukan bila setiap ahli sesuai digunakan

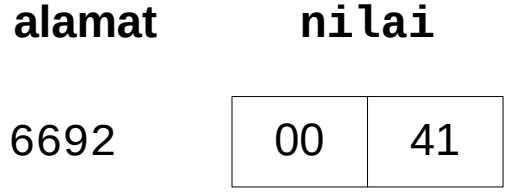
- Cth mudah:

```
#include <stdio.h>
union nombor {
                      char x;
                     int y;};
main( )
{
      union nombor nilai;
      printf("saiz union ialah %d\n",
             sizeof(nilai));
      nilai.x = 0x41;/*ASCII 'A'*/
      printf("char:  %X \t int:   %X\n", nilai.x,
             nilai.y);
      printf("alamat bermula pada  %d\n",
             &nilai.x);
      nilai.y = 0x1234;
      printf("char:  %X \t int:   %X\n", nilai.x,
             nilai.y);
      printf("alamat bermula pada  %d\n",
             &nilai.y);
      return 0;
}
```

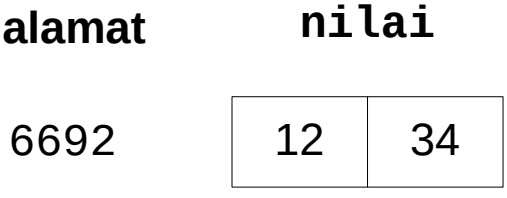
- Output:

```
saiz union ialah 2
char: 41          int:   41
alamat bermula pada  6692
char: 34          int:   1234
alamat bermula pada  6692
```

- Bila nilai.x diumpukkan:



- Bila nilai.y diumpukkan:



9.8 Pernomboran

- set nilai-nilai pemalar integer yagn diwakili oleh pencam.

- Pengisytiharan:

```
enum <nama_jenis_enum>
{
    senarai_enum
};
```

- Cth pengisytiharan jenis data BulanIslam & p'ubah:

```
enum BulanIslam
{
    muharram, safar, rabiulawal,
    rabiulakhir, jamadilawal, jamadilakhir,
    rejab, syaaban, ramadhan, syawal,
    zulkaedah, zulhijjah
};
enum BulanIslam Bulan;
```

- Apabila jenis data pernomboran diisytiharkan, pencam-pencamnya akan diumpukkan nilai integer secara automatik.
- Pencam yang pertama akan diumpukkan nilai '0', pecam kedua bernilai '1' dan begitulah seterusnya.
- Pengaturcara boleh membuat pengubahsuaian berdasarkan peraturan-peraturan berikut:

  - boleh mengumpukkan nilai pemalar (*constant*) kepada senarai pernomboran
  - boleh mengumpukkan nilai kepada senarai pernomboran dengan menggunakan pencam yang sebelumnya dalam ungkapan aritmetik
  - boleh mengumpukkan nilai yang sama kepada lebih dari satu pencam dalam senarai pernomboran

- Cth: Rekabentuk mesin jual air automatik.

```
enum siling
{
    satusen = 1, limasen =5,sepuluhsen= 10,
    samah = limasen*sepuluhsen,serial =
    2*samah
};
enum siling duitmasuk;
```

Untuk kira duit yg dimasukkan oleh pembeli cukup @ tidak, diberi harga air sekotak RM1.20:

```
while (jum_duit<120) {
    switch(duitmasuk) {
        case satusen:
                jum_duit++;
                break;
        case limasen:
                jum_duit=jum_duit+
                limasen;
                break;
        Case sepuluhsen:
                jum_duit=jum_duit+ sepuluhsen;
                break;
        case samah:
                jum_duit=jum_duit+ samah;
                break;
        case serial:
                jum_duit=jum_duit+ serial;
                break;

    }
}
```