DAYANG NORHAYATI ABANG JAWAWI
ROSBI MAMAT

Bab 6: Pengaturcaraan Bermodul

6.1 Pengenalan

- Pengaturcaraan bermodul adalah kaedah penggunaan modul-modul atur cara yang kecil dan tak bersandar yang diintegrasikan untuk memudahkan pembinaan, pengujian, dan penyenggaraan sesuatu perisian yang bersaiz besar.
- Pengaturcaraan C menggunakan konsep fungsi untuk menyokong pelaksanaan kaedah pengaturcaraan bermodul.
- Fungsi - Kumpulan jujukan suruhan2 yang diberi nama. Contoh fungsi2 pratakrif (ditulis dan disediakan oleh pembekal pengkompil C yang disimpan dalam perpustakaan piawai C) yang telah dipelajari:

```
pow()          putchar()
printf()       gets()
scanf()        puts()
getchar()
```

6.1 Pengenalan

- Fungsi takrifan pengguna pula adalah atur cara yang direkabentuk dan ditulis sendiri oleh pengatur cara.

- Fungsi `main()` adalah salah satu contoh fungsi takrifan pengguna yang mana semua kenyataan-kenyataan di dalam fungsi ini perlu ditulis oleh pengatur cara.

6.2 Reka Bentuk Atur Cara Bermodul

- Sebelum algoritma modul boleh direka bentuk, penguraian masalah perlu dibuat dengan kaedah reka bentuk atas-bawah.

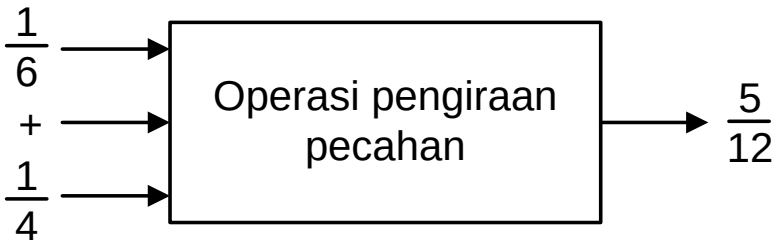
- Penyelesaian masalah dengan kaedah penguraian masalah atas-bawah dilakukan untuk mereka bentuk struktur atur cara

- Struktur atur cara terdiri daripada koleksi modul dan hubungan hierarki antara modul bagi memudahkan pengaturcaraan dan pengujian atur cara bermodul dilakukan.

6.2.1 Kaedah Penguraian Masalah Atas-Bawah

- Penyelesaian masalah dengan kaedah penguraian masalah atas-bawah dilakukan pada masalah yang perlu dipecahkan kepada masalah yang kecil dan skop masalah yang kecil tersebut diselesaikan satu persatu dengan menggunakan fungsi atur cara.

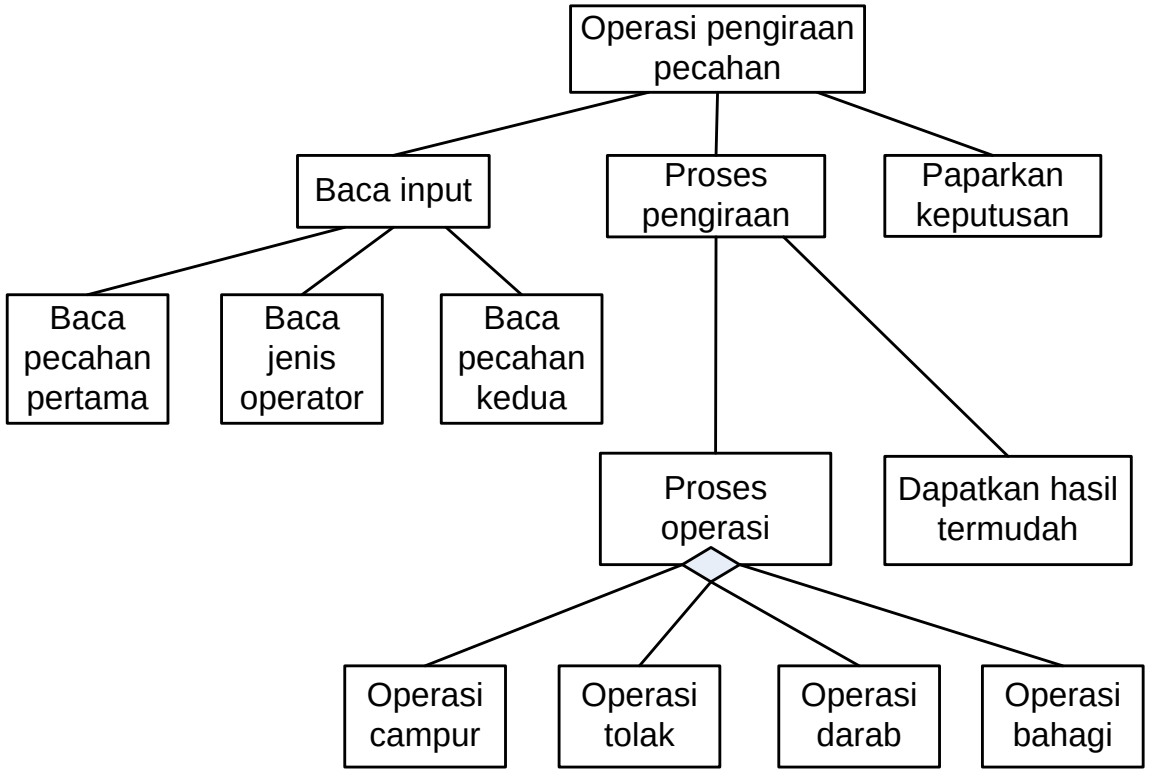
- Contoh pengiraan operasi tambah bagi pecahan input $\frac{1}{6}$ dan $\frac{1}{4}$:



Anda diminta utk menyediakan satu aturcara yg boleh mengira nombor2 pecahan.Operasi pengiraan adalah operasi campur, tolak, darab dan bahagi. hasil dari pengiraan tersebut mestilah dalam bentuk pecahan termudah.

6.2.1 Kaedah Penguraian Masalah Atas-Bawah

- Penguraian atas-bawah masalah pengiraan pecahan:



6.2.2 Carta Struktur

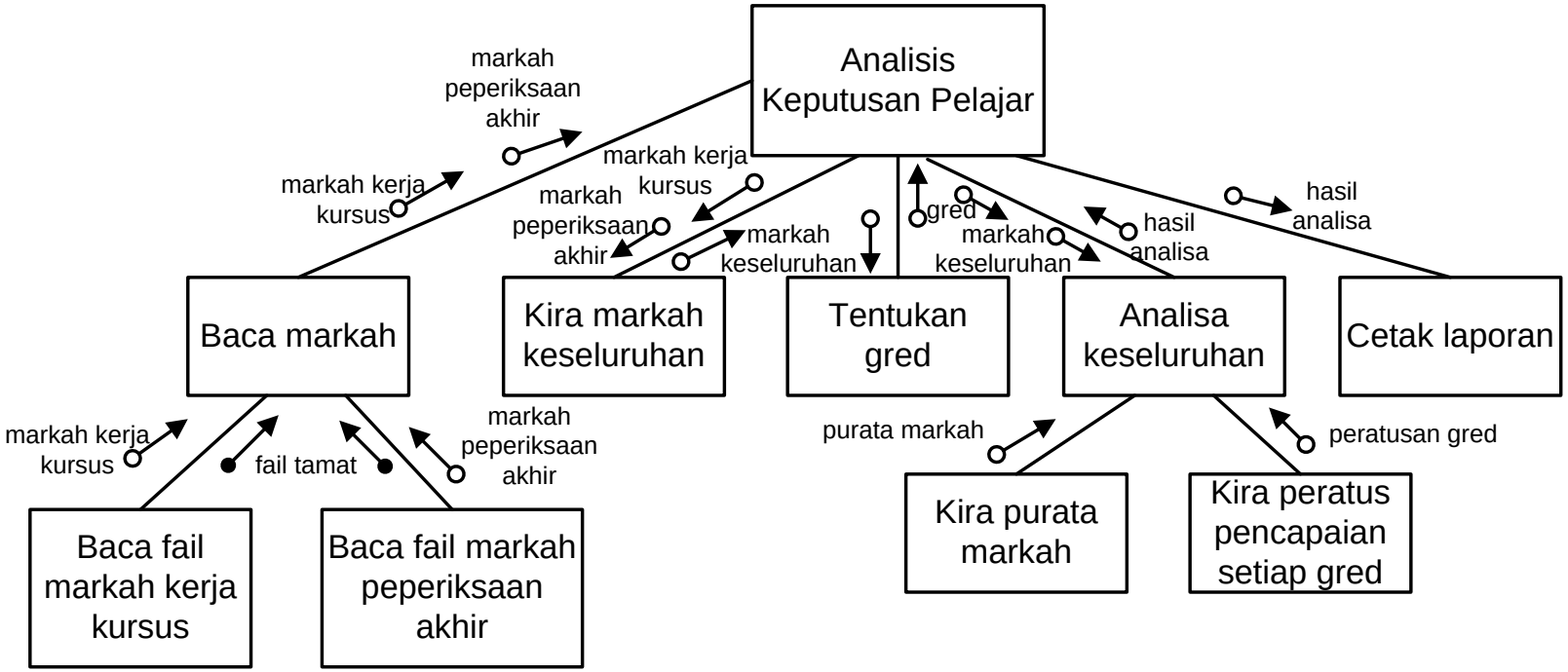
- Carta struktur adalah carta hierarki dalam bentuk gambaran grafik yang digunakan sebagai alat reka bentuk untuk menguraikan penyelesaian masalah secara berstruktur.

- Carta ini menggambarkan pembahagian atau penguraian masalah ke sub-sub masalah dan menunjukkan hubungan hierarki antara bahagian sub-sub masalah tersebut.

Aliran data	Aliran kawalan
	
- laluan yang diambil oleh data menerusi modul-modul sistem sewaktu pelaksanaan atur cara.	- aliran data kawalan pelaksanaan modul yang disasarkan dan ia biasanya digunakan sebagai rujukan bagi mengenal pasti, memilih, atau mengolah rutin, rekod dan fail.

6.2.2 Carta Struktur

- Reka bentuk carta struktur untuk masalah analisis markah pelajar:



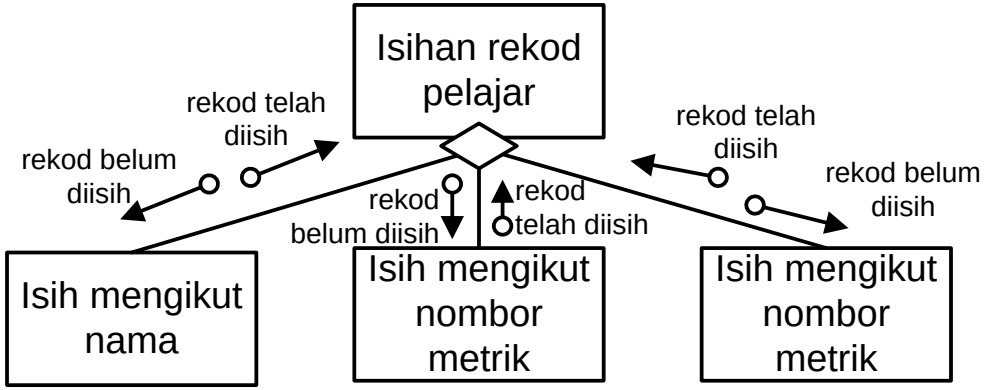
6.2.2 Carta Struktur

- Carta struktur tidak menunjukkan urutan atau jujukan modul-modulnya dilaksanakan, tetapi carta struktur menunjukkan bagaimana sub-sub modul dipanggil oleh modul aras lebih atas dengan menggunakan struktur logik:

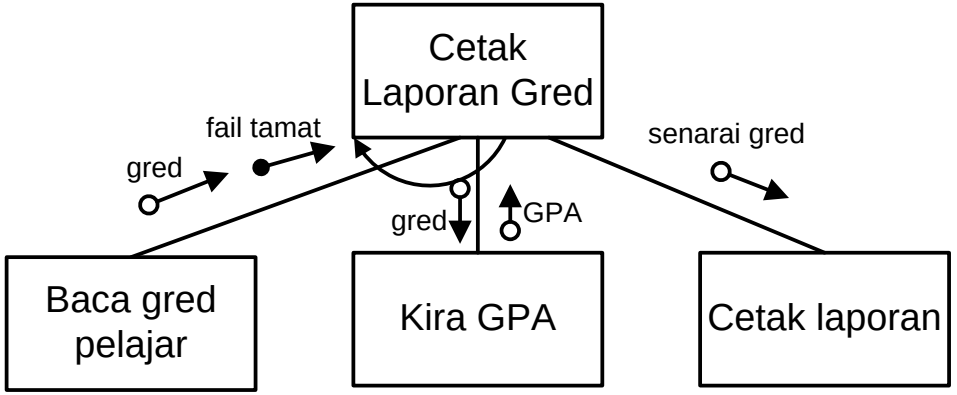
- pemilihan, atau ulangan. Pemilihan dalam carta struktur diwakili oleh simbol berlian  dan syarat akan disemak untuk menentukan modul yang akan dipanggil.

6.2.2 Carta Struktur

- Contoh carta struktur dengan struktur logik pemilihan:



- Contoh carta struktur dengan struktur logik ulangan



6.3 Elemen Pengaturcaraan Bermodul

- Setiap modul dalam pengaturcaraan bermodul akan dilaksanakan sebagai fungsi.

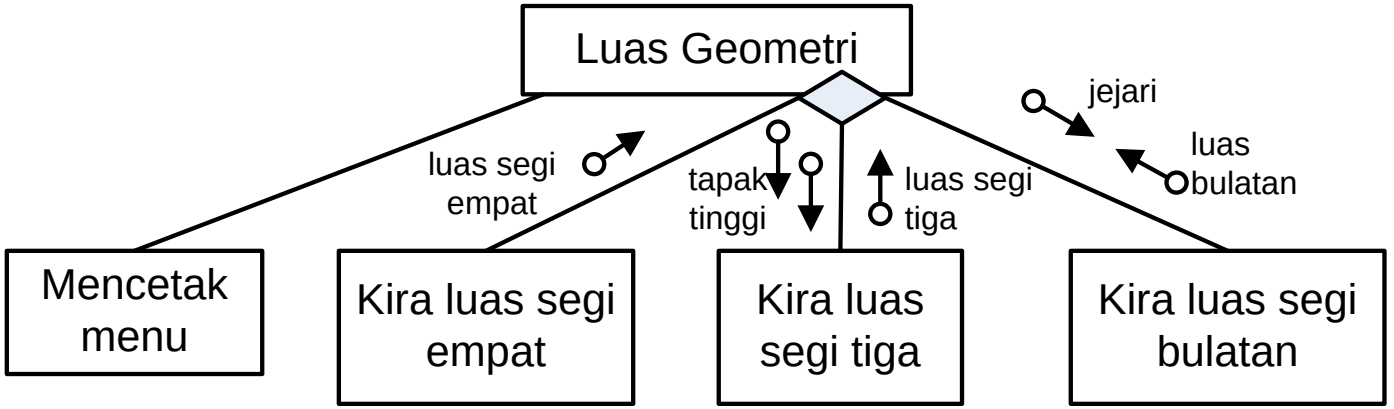
- Semua modul yang direka bentuk pada carta struktur adalah fungsi takrifan pengguna yang perlu dilaksanakan sebagai fungsi.

- Sebelum fungsi-fungsi ini boleh dilaksanakan, setiap fungsi perlu reka bentuk algoritmanya.

6.3 Elemen Pengaturcaraan Bermodul

- Contoh: mengira dan memaparkan luas satu segi empat, satu bulatan atau satu segi tiga atas pilihan pengguna dan saiz sisi diberikan oleh pengguna melalui papan kekunci.

- Carta struktur operasi pengiraan luas tiga bentuk geometri:



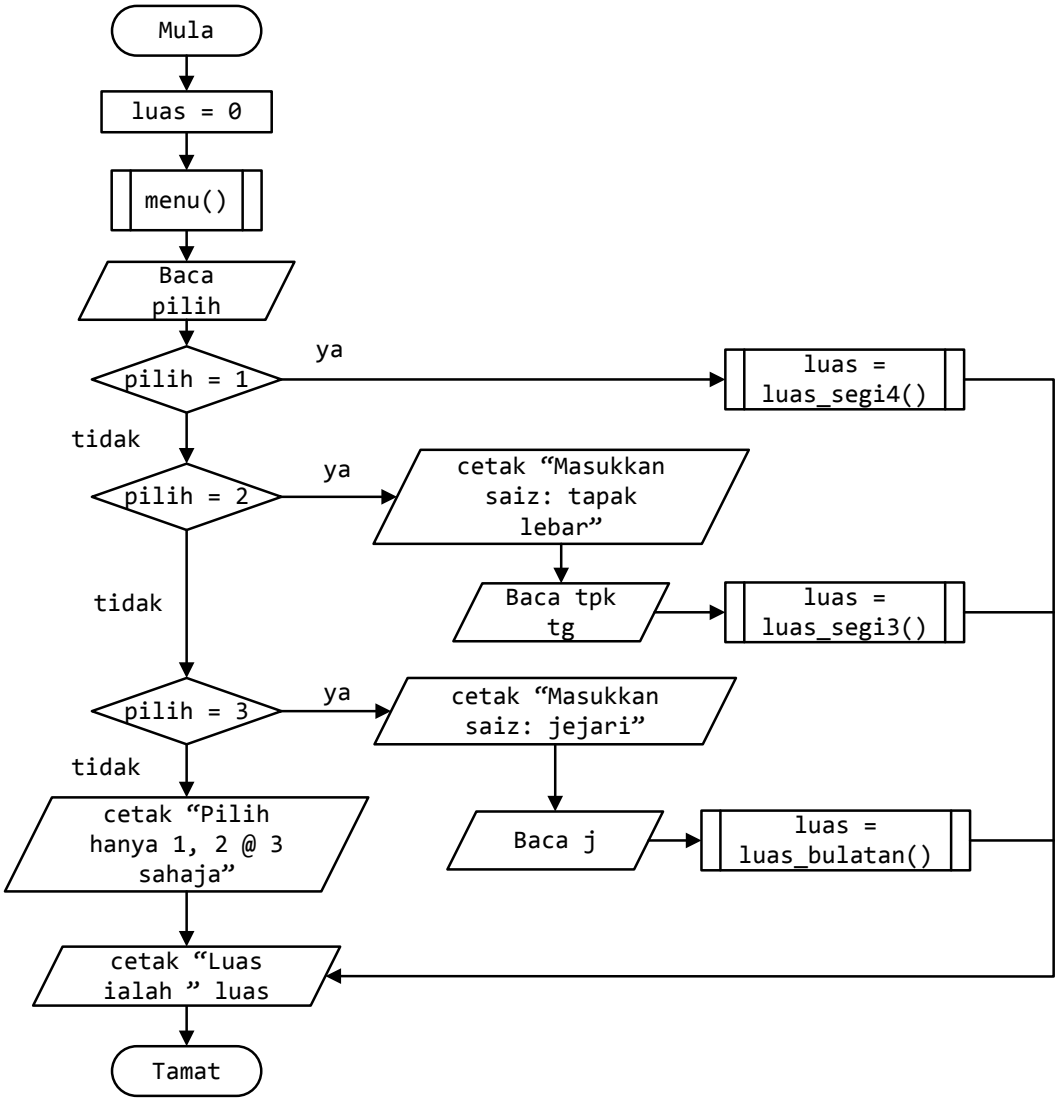
6.3 Elemen Pengaturcaraan Bermodul

- Analisis masalah untuk mengira luas tiga bentuk geometri:

Input
Media: papan kekunci panjang nilai masukan panjang sisi segiempat lebar nilai masukan lebar sisi segi empat j nilai masukan jejari bulatan tpk nilai masukan tapak segi tiga tg nilai masukan tinggi segi tiga
Output
Media: skrin luas hasil proses luas segi empat, luas segi tiga atau luas bulatan
Proses
1. Luas segi empat = panjang x lebar 2. luas segi tiga = (tpk x tg)/2 3. luas bulatan = π x j^2

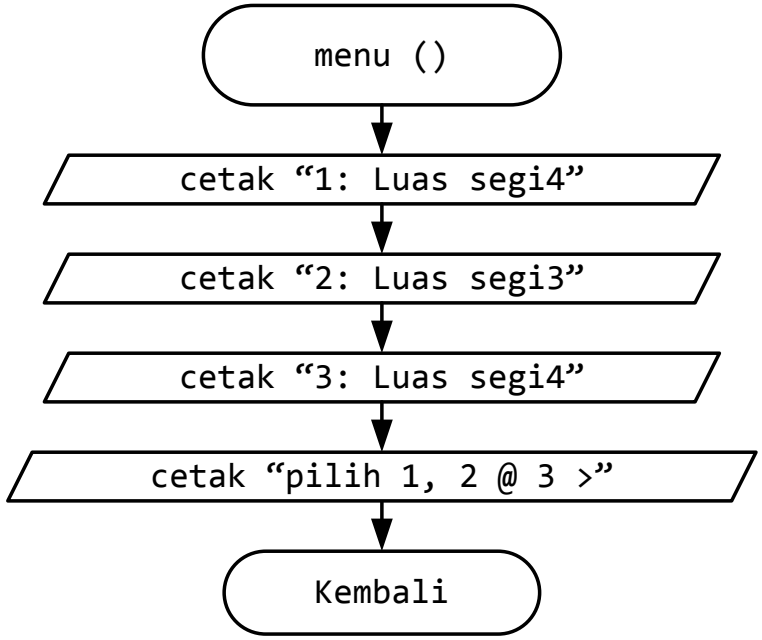
6.3 Elemen Pengaturcaraan Bermodul

- Algoritma modul Luas Geometri yang diungkap dengan carta alir:



6.3 Elemen Pengaturcaraan Bermodul

- Algortima modul `Mencetak menu` yang diungkap dengan carta alir:



6.3 Elemen Pengaturcaraan Bermodul

- Untuk mengkod atur cara yang menggunakan fungsi-fungsi yang telah direka bentuk ini, bahasa pengaturcaraan C menggunakan tiga elemen utama di dalam perlaksanaan fungsi, iaitu:

1. Takrifan fungsi,
2. Panggilan fungsi dan
3. Pengisytiharan fungsi.

6.3 Elemen Pengaturcaraan Bermodul

- Contoh atur cara:

```
 1:  //Mengira dan memaparkan luas satu segiempat, satu bulatan
 2:  //atau satu segitiga atas pilihan pengguna
 3:  #include <stdio.h>
 4:
 5:  // Pengisytiharan fungsi atau prototaip fungsi
 6:  float luas_segi4 ();
 7:  float luas_segi3 (float tapak, float tinggi);
 8:  float luas_bulatan (float jejari);
 9:  void menu ();
10:
11:  // Takrifan fungsi main
12:  int main(){
13:     float luas=0, j, tpk, tg;
14:     int pilih;
15:
16:     menu();  // Panggilan fungsi
17:     scanf ("%d", &pilih);
18:     if (pilih == 1)
```

UTM UNIVERSITI TEKNOLOGI MALAYSIA

6.3 Elemen Pengaturcaraan Bermodul

```
19:       luas = luas_segi4 (); // Panggilan fungsi
20:    else if (pilih == 2) {
21:       printf("Masukkan saiz: tapak lebar\n");
22:       scanf("%f %f", &tpk, &tg);
23:       luas = luas_segi3 (tpk, tg); // Panggilan fungsi
24:    } else if (pilih == 3 ) {
25:    printf("Masukkan saiz: jejari\n");
26:         scanf("%f", &j);
27:         luas = luas_bulatan (j);  // Panggilan fungsi
28:    } else   printf("Pilih 1, 2 @ 3 sahaja\n");
29:    printf ("Luas ialah %0.2f\n", luas);
30:    return 0;
31: }
32:
33: //Takrifan fungsi untuk mengira luas segi empat
34: float luas_segi4 (){
35: float pjg, lbr;
36:
37: printf("masukkan saiz: panjang lebar\n");
38: scanf("%f %f", &pjg, &lbr);
39: return (pjg*lbr);
40: }
```

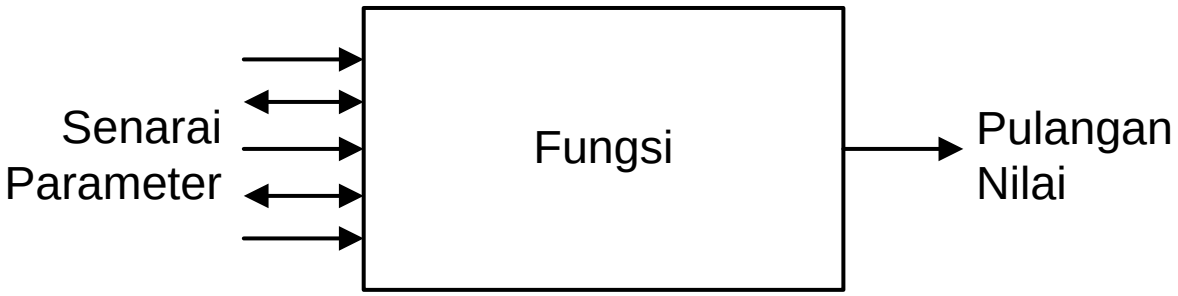
6.3 Elemen Pengaturcaraan Bermodul

```
41:
42:   //Takrifan fungsi untuk mengira luas segi tiga
43:   float luas_segi3 (float tapak, float tinggi){
44:     float luas;
45:
46:     luas =(tapak*tinggi)/2.0;
47:     return ( luas );
48:   }
49:
50:   //Takrifan fungsi untuk mengira luas bulatan
51:   float luas_bulatan (float jejari){
52:     return ( 3.1415*jejari*jejari );
53:   }
54:
55:   //Takrifan fungsi untuk mencetak menu
56:   void menu (){
57:     printf("1: Luas segi4\n");   printf("2: Luas segi3\n");
58:     printf("3: Luas bulatan\n"); printf("pilih 1, 2 @ 3 >");
59:   }
```

6.3.1 Takrifan Fungsi

- Sebelum satu fungsi takrifan pengguna boleh digunakan, ia perlu ditakrifkan. Empat bahagian yang perlu dimasukkan dalam proses takrifan fungsi iaitu: jenis kembali, nama fungsi, senarai parameter dan badan fungsi.

```
jenis_kembali nama_fungsi
(senarai_parameter)
{
badan_fungsi;
return(nilai); /* nilai kembali
     sebahagian daripada badan_fungsi*/
}
```



6.3.1 Takrifan Fungsi

- **`jenis_kembali :`**
  - jenis data seperti char, int, float dan long yang akan dihasilkan oleh fungsi tersebut sebagai nilai kembali. Jika tiada nilai kembali, maka jenis kembali akan dinyatakan sebagai void.

- **`nama_fungsi :`**
  - digunakan untuk memanggil fungsi dan melaksanakan fungsi. Nama fungsi mesti mematuhi syarat yang sama dengan syarat pengenal pasti.

- **`Senarai_parameter :`**
  - Pembolehubah2 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 senarai_parameter mestilah samasemasa fungsi ditakrifkan dan digunakan.

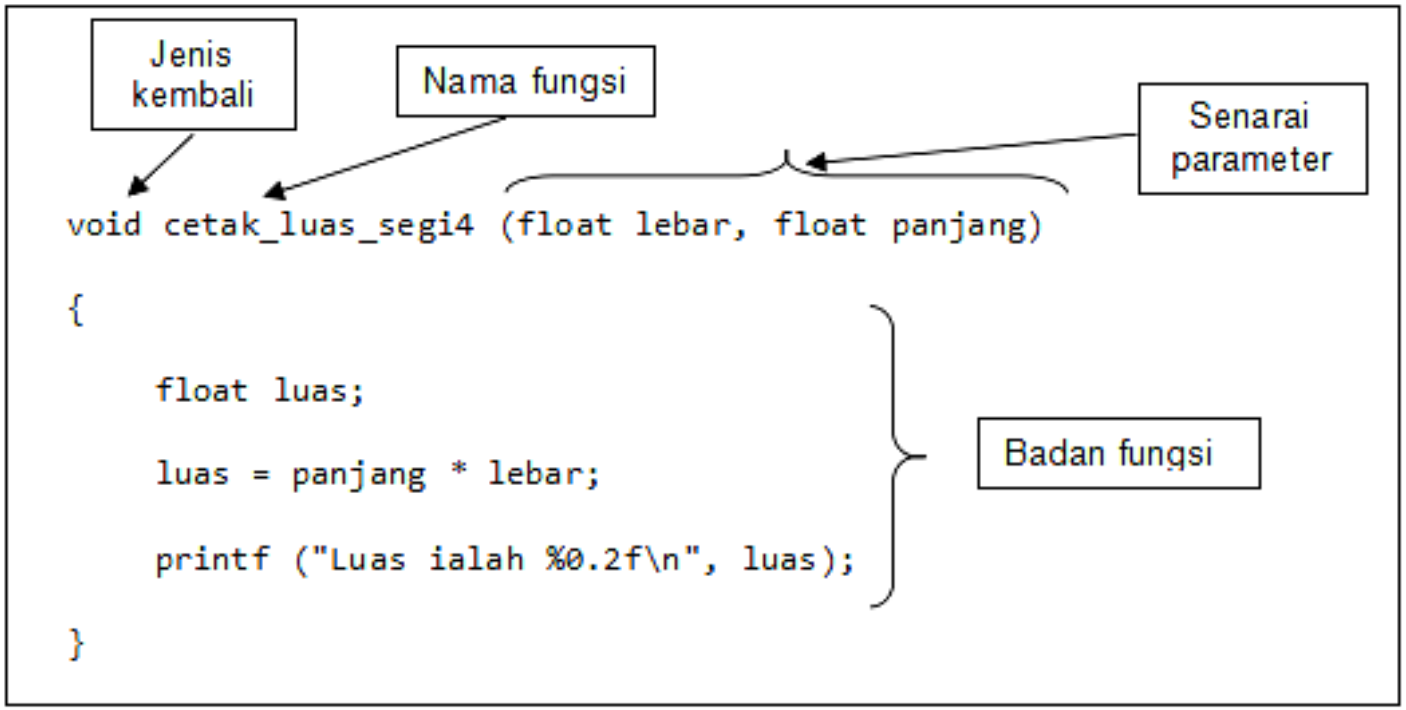
6.3.1 Takrifan Fungsi

- **`badan_fungsi :`**
  - Terdiri dari satu atau lebih kenyataan2 C termasuk pengisytiharan pembolehubah2 yg melakukan sesuatu kerja. Cara menulisnya sama dgn cara menulis badan fungsi `main()`.

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

6.3.1 Takrifan Fungsi

- contoh takrifan fungsi `cetak_luas_segi4()` dengan dua parameter yang disenaraikan.



6.3.2 Panggilan Fungsi

- Fungsi-fungsi hanya boleh dilaksanakan setelah dipanggil dengan format tertentu

- Tiga perkara yang perlu diberi perhatian untuk memanggil fungsi ialah:

1. Nama fungsi mestilah sama dengan nama yang ditakrif di kepala fungsi.

2. Senarai argumen atau data yang dihulurkan ke fungsi mestilah mempunyai jenis dan bilangan yang sama dengan senarai parameter di kepala fungsi dalam turutan yang sama dengan senarai parameter tersebut.

3. Jenis kembali fungsi yang sama dengan yang di takrif di kepala fungsi.

6.3.2 Panggilan Fungsi

- Format panggilan fungsinya :

```
nama_fungsi();
```

- Contoh:

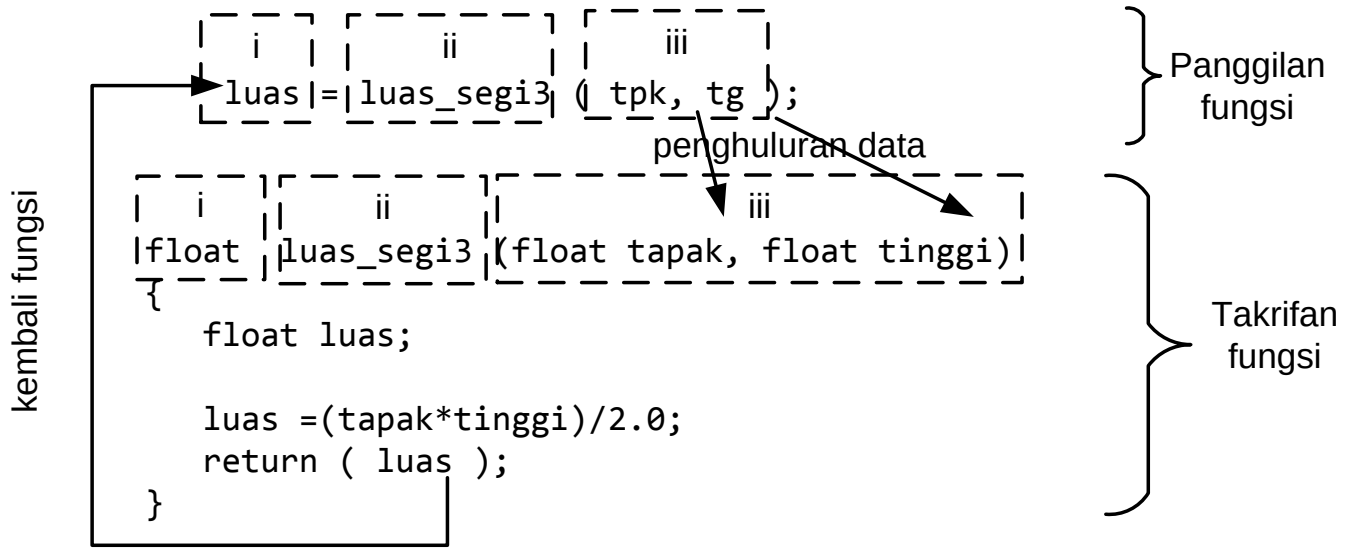
```
float luas2;
//nilai kembali digunakan pada ungkapan aritmetik
luas2 = luas_segi4() * 2;
 // nilai kembali terus dicetak
 printf ("Luas segi empat ialah %0.2f\n", luas_segi4 ());
```

6.3.2 Panggilan Fungsi

- Format panggilan fungsi yang menghantar data ke fungsi dan tidak menerima nilai kembali fungsi, format panggilan fungsi :

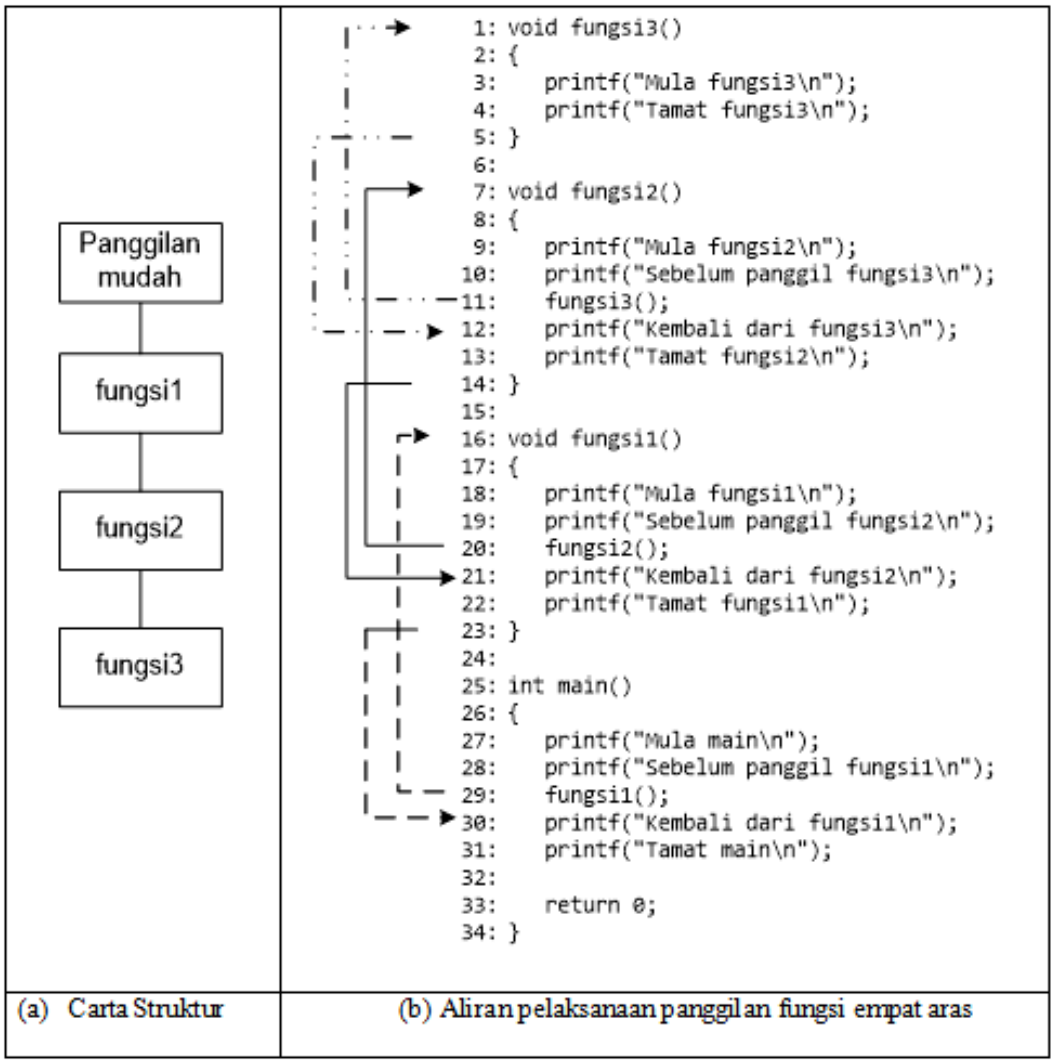
```
nama_fungsi(senarai_argumen);
```

- Contoh penghuluran data ke fungsi dan kembali data dari fungsi



6.3.2 Panggilan Fungsi

- Aliran panggilan fungsi pada carta struktur dan atur cara dan output:



(a) Carta Struktur

```
 1: void fungsi3()
 2: {
 3:    printf("Mula fungsi3\n");
 4:    printf("Tamat fungsi3\n");
 5: }
 6:
 7: void fungsi2()
 8: {
 9:    printf("Mula fungsi2\n");
10:    printf("Sebelum panggil fungsi3\n");
11:    fungsi3();
12:    printf("Kembali dari fungsi3\n");
13:    printf("Tamat fungsi2\n");
14: }
15:
16: void fungsi1()
17: {
18:    printf("Mula fungsi1\n");
19:    printf("Sebelum panggil fungsi2\n");
20:    fungsi2();
21:    printf("Kembali dari fungsi2\n");
22:    printf("Tamat fungsi1\n");
23: }
24:
25: int main()
26: {
27:    printf("Mula main\n");
28:    printf("Sebelum panggil fungsi1\n");
29:    fungsi1();
30:    printf("Kembali dari fungsi1\n");
31:    printf("Tamat main\n");
32:
33:    return 0;
34: }
```

(b) Aliran pelaksanaan panggilan fungsi empat aras

Output:

```
Mula main
Sebelum panggil fungsi1
Mula fungsi1
Sebelum panggil fungsi2
Mula fungsi2
Sebelum panggil fungsi3
Mula fungsi3
Tamat fungsi3
Kembali dari fungsi3
Tamat fungsi2
Kembali dari fungsi2
Tamat fungsi1
Kembali dari fungsi1
Tamat main
```

6.3.3 Prototaip Fungsi

- menurut pada rajah panggilan fungsi sebelum ini, fungsi-fungsi yang dipanggil diletakkan sebelum fungsi yang memanggil, kes ini tidak memerlukan prototaip fungsi.

- Format pengisytiharan prototaip fungsi:

```
jenis_kembali nama_fungsi (senarai_parameter);
```

- Prototaip fungsi adalah menyerupai kepala fungsi, kecuali ada koma bertitik di akhir kenyataan. `jenis_kembali`, `nama_fungsi` dan `senarai_parameter` (susunan dan jenis parameter) mestilah sama dengan kepala fungsi. Contoh:

```
float luas_segi3 (float, float);
```

6.4 Kembali Nilai dari Fungsi

- Satu fungsi boleh memulangkan satu nilai ke fungsi yang memanggilnya

- Jenis nilai yang hendak dikembalikan dari fungsi perlu dinyatakan di kepala fungsi, jika tiada nilai kembali,maka kepala fungsi dinyatakan `void`.

- Jenis nilai menggunakan semua jenis data seperti `int`, `float`, `bool`, `double` dan `char`. Kenyataan yang digunakan untuk memulangkan sesuatu nilai ke fungsi yang memanggil ialah `return`.

6.4 Kembali Nilai dari Fungsi

- Format kenyataan untuk memulangkan satu nilai ke fungsi :

```
return ungkapan;
```

- Contoh kenyataan `return` yang sah adalah seperti berikut:

```
return luas;
return ( 3.1415*jejari*jejari );
return 5;
```

6.4 Kembali Nilai dari Fungsi

- Contoh : kenyataan pertama dan kedua pada `fungsi1()` akan dilaksanakan sebelum kembali ke fungsi yang memanggil, manakala `fungsi2()` hanya kenyataan pertama sahaja dilaksanakan kerana kenyataan return diletakkan sebelum kenyataan untuk mencetak perkataan “Kedua”.

```
void fungsi1(void)
{
   printf(“Pertama\n”);
   printf(“Kedua\n”);
}
```

```
void fungsi2(void)
{
  printf(“Pertama\n”);
  return;
  printf(“Kedua\n”);
}
```

6.5 Penghuluran Data ke Fungsi

- Apabila fungsi dipanggil dengan argument maka penghuluran data akan berlaku.

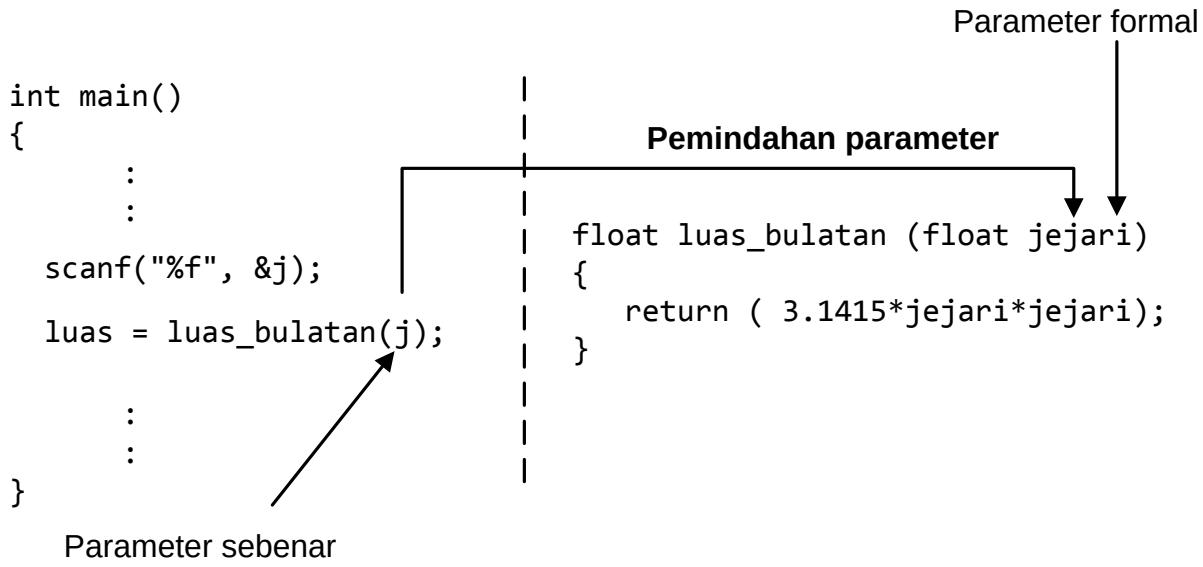
- Dua jenis parameter yang terlibat dalam penghuluran data ke fungsi ialah:
  1. parameter formal
  2. parameter sebenar

- **Paramater formal** adalah pemboleh ubah parameter yang diisytiharkan semasa fungsi ditakrif.

- **Parameter sebenar** adalah nilai sebenar yang diberikan oleh fungsi pemanggil kepada parameter formal semasa fungsi tersebut dipanggil.

6.5 Penghuluran Data ke Fungsi

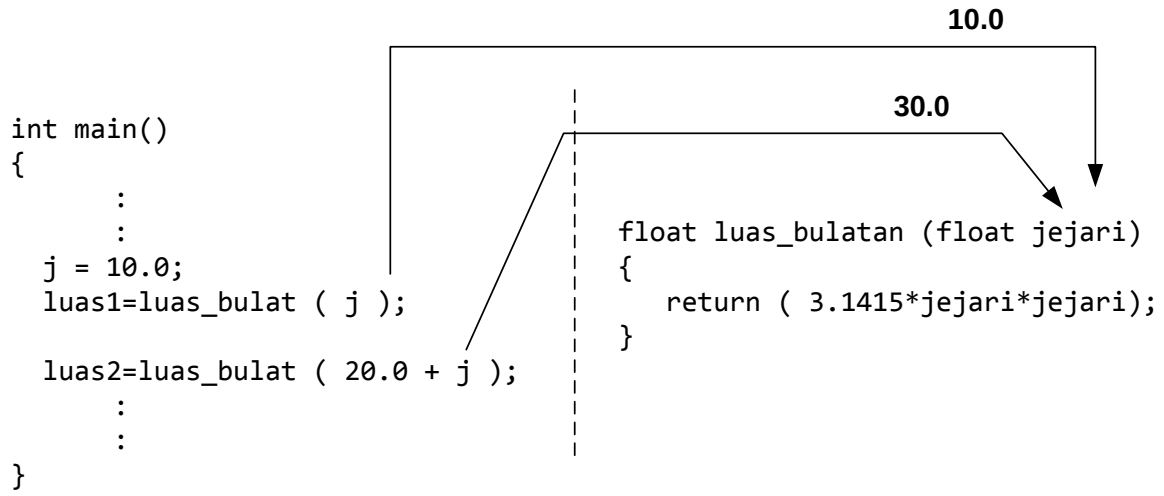
- Contoh pemindahan parameter semasa panggilan fungsi:



- Semasa fungsi dipanggil, penghuluran nilai data dari fungsi pemanggil ke fungsi yang dipanggil boleh dibuat dalam dua kaedah iaitu
  1. penghuluran secara nilai sebenar
  2. penghuluran secara alamat nilai.

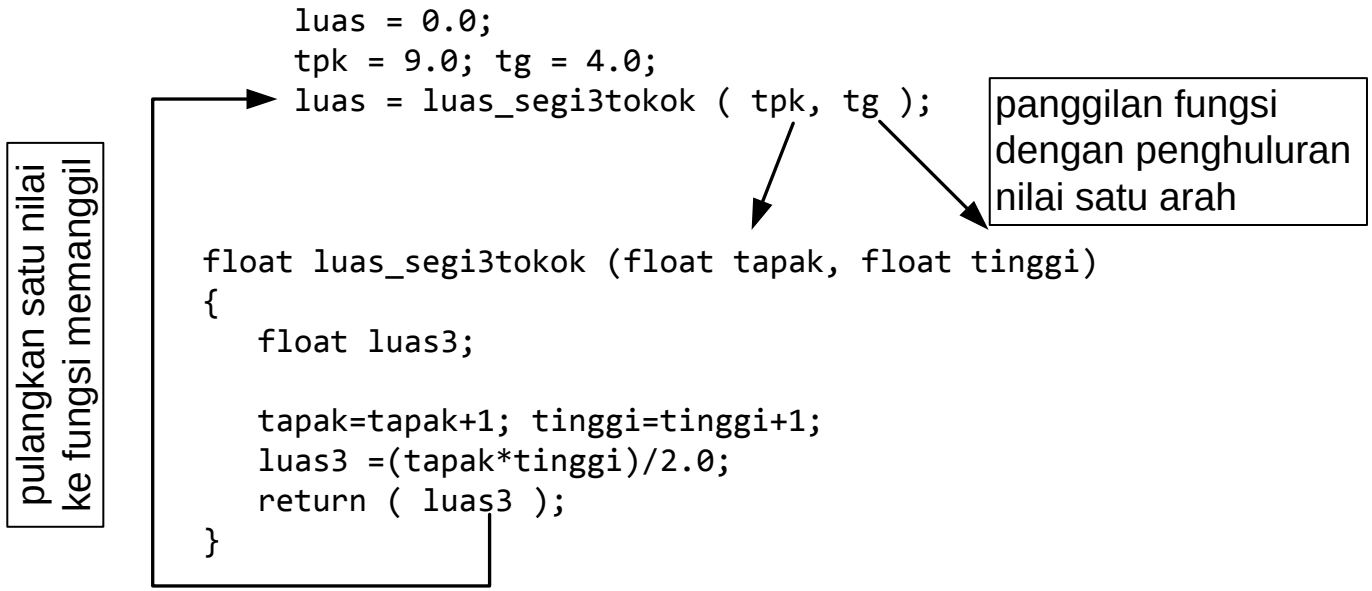
6.5.1 Penghuluran Nilai Sebenar

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



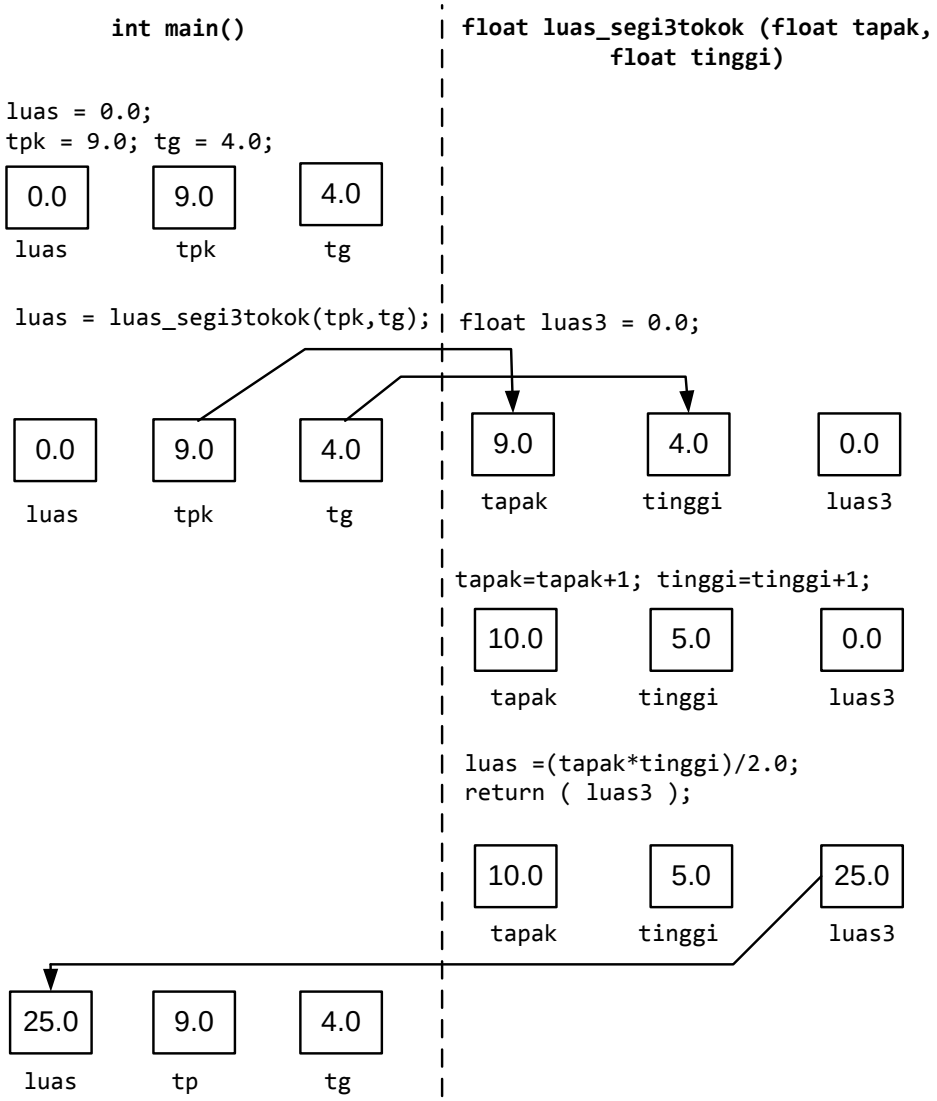
6.5.1 Penghuluran Nilai Sebenar

- Penghuluran nilai sebenar, membolehkan satu atau 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:



6.5.1 Penghuluran Nilai Sebenar

- Perubahan parameter penghuluran nilai sebenar



6.5.2 Penghuluran Alamat Nilai

- Penghuluran alamat nilai menggunakan pemboleh ubah yang merujuk kepada alamat dengan penghuluran alamat parameter sebenar ke parameter formal pada fungsi yang memanggil. Alamat di mana nilai disimpan di RAM akan dihantar kepada parameter formal pada fungsi yang memanggil.

- Alamat sesuatu pemboleh ubah boleh didapati dengan sintaks berikut:

```
&nama_pemboleh_ubah
```

6.5.2 Penghuluran Alamat Nilai

- Fungsi yang menerima alamat boleh menggunakan alamat tersebut untuk mendapatkan nilai yang terkandung dalam alamat yang dihantar.

- Nilai pemboleh ubah yang disimpan pada alamat boleh dirujuk dengan sintaks berikut:

```
*nama_pemboleh_ubah
```

6.5.2 Penghuluran Alamat Nilai

- Pemboleh ubah penuding adalah sel ingatan yang menyimpan alamat sel ingatan lain. Pemboleh ubah penuding pada parameter formal menyimpan alamat parameter sebenar.

- Contoh fungsi `luas_segi3tokok()` dengan panggilan fungsi dari `main()` parameter formal dengan pemboleh ubah penuding atau rujukan.

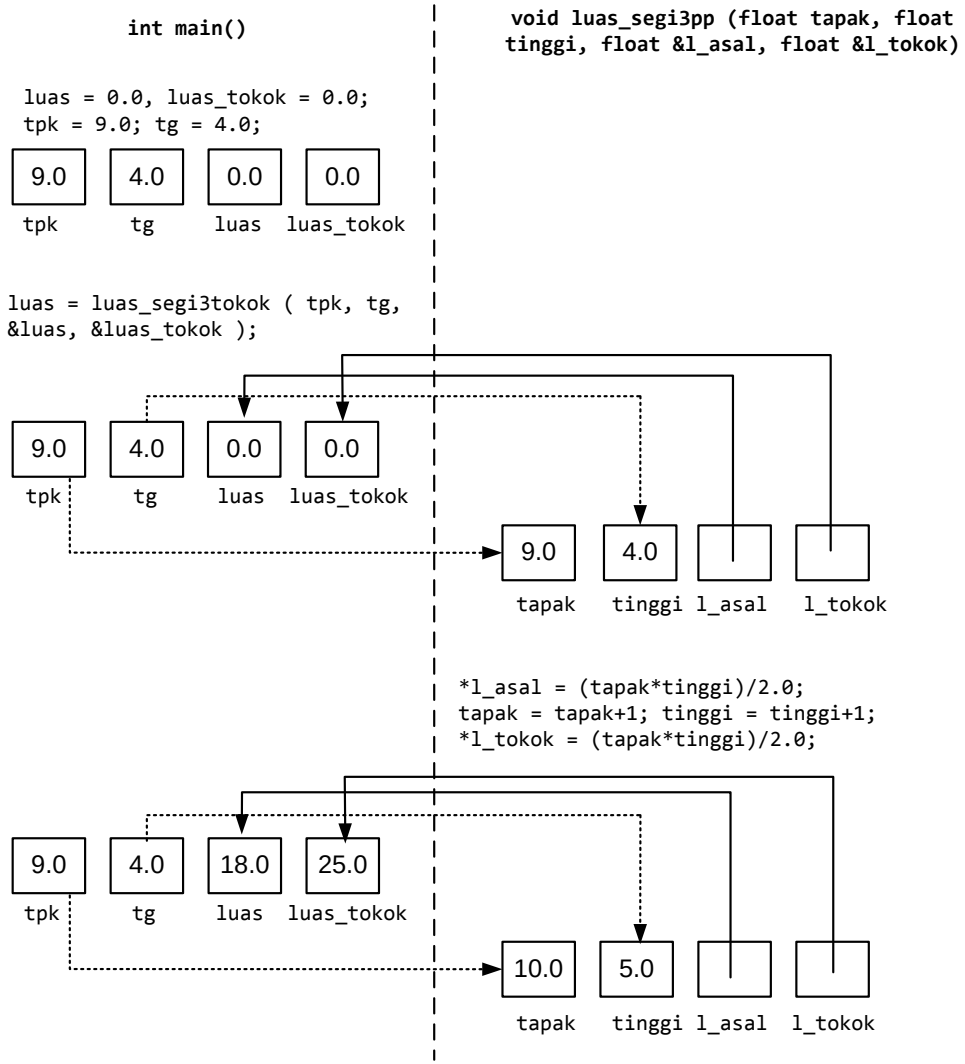
```
luas = 0.0, luas_tokok = 0.0;
tpk = 9.0; tg = 4.0;
luas = luas_segi3tokok ( tpk, tg, &luas, &luas_tokok );
```

panggilan fungsi dengan penghuluran nilai satu dan dua arah

```
void luas_segi3tokok (float tapak, float tinggi, float *l_asal, float *l_tokok )
{
   *l_asal = (tapak*tinggi)/2.0;
   tapak = tapak+1; tinggi = tinggi+1;
   *l_tokok = (tapak*tinggi)/2.0;
}
```

6.5.2 Penghuluran Alamat Nilai

- Perubahan parameter penghuluran nilai sebenar dan alamat nilai:



6.6 Skop Struktur Atur Cara

- Atur cara yag mengandungi berbilang fungsi di dalam lebih daripada satu fail memerlukan cara tertentu untuk menentukan skop pemboleh ubah, fungsi dan cara simpanan data yang akan digunakan oleh fungsi dan fail tertentu.

- **Skop pemboleh ubah** menentukan bahagian atur cara yang boleh mencapai dan merujuk pemboleh ubah tertentu yang telah diisytiharkan.

6.6.1 Skop Pemboleh Ubah

- Skop satu pemboleh ubah bermula pada tempat di mana ia diisytiharkan dan akan tamat bergantung kepada cara ini diisytiharkan.

- Terdapat tiga jenis pemboleh ubah dalam pengaturcaraan C iaitu
  1. pemboleh ubah sejagat
  2. setempat
  3. parameter formal fungsi.

- Pemboleh ubah sejagat adalah pemboleh ubah yang diisytiharkan di luar fungsi. Skop pemboleh ubah sejagat bermula pada titik ia diisytiharkan sehingga ke penghujung fail atur cara.

- Pemboleh ubah yang diisytiharkan di dalam sesuatu blok atur cara dinamakan pemboleh ubah setempat.

6.6.1 Skop Pemboleh Ubah

- Contoh:

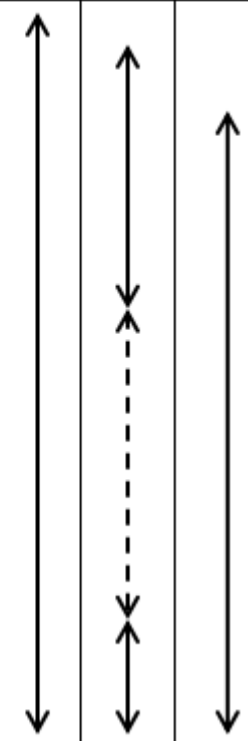
```
float luas_segi4()
{
   float pjg, lbr;

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

- Pengisytiharan dua pemboleh ubah setempat `pjg` dan `lbr` dalam contoh fungsi `luas_segi4()` berikut hanya boleh dicapai oleh fungsi `luas_segi4()` sahaja, fungsi lain termasuk fungsi `main()` tidak boleh mencapai dua pemboleh ubah ini.

6.6.1 Skop Pemboleh Ubah

- Skop pemboleh ubah setempat di dalam blok gelung:

Kod Segmen	baris	sum	j	i
`int sum = 0;`	1			
`int j = 99;`	2			
`int i = 99;`	3			
`for (i=0; i<5; i++){`	4			
`printf("Titik 1: i = %d j = %d\n" ,i, j);`	5			
`int j=10;`	6			
`printf("Titik 2: i = %d j = %d\n" ,i, j);`	7			
`sum +=j;`	8			
`printf("Sum = %d\n", sum);`	9			
`}`	10			
`printf("Gelung tamat\n");`	11			
`printf("i)\n", i);`	12			



Pemboleh ubah sejagat boleh dicapai oleh semua blok atur cara kecuali jika pemboleh ubah tersebut mempunyai nama yang sama dengan pemboleh ubah setempat.

6.6.1 Skop Pemboleh Ubah

- Output menunjukkan skop pemboleh ubah dengan blok gelung:

```
i = 99 j = 99
Akan mula gelung
Titik 1: i = 0 j = 99
Titik 2: i = 0 j = 10
Sum = 10
Titik 1: i = 1 j = 99
Titik 2: i = 1 j = 10
Sum = 20
Titik 1: i = 2 j = 99
Titik 2: i = 2 j = 10
Sum = 30
Titik 1: i = 3 j = 99
Titik 2: i = 3 j = 10
Sum = 40
Titik 1: i = 4 j = 99
Titik 2: i = 4 j = 10
Sum = 50
Gelung tamat
i = 5 j = 99
```

6.6.1 Skop Pemboleh Ubah

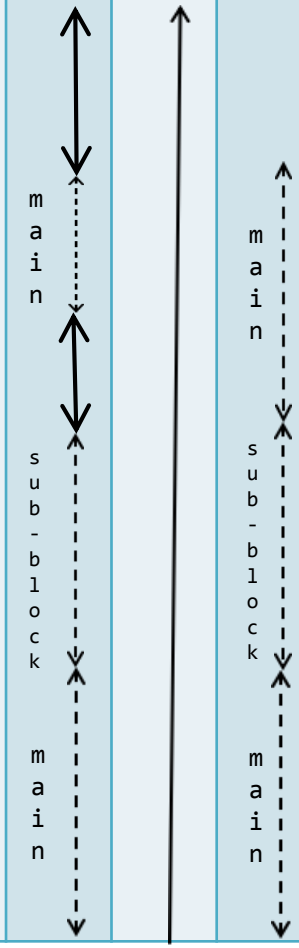
- Perbezaan pemboleh ubah parameter formal dan pemboleh ubah setempat ialah pengisytiharan parameter formal dibuat pada kurungan di kepala fungsi dan pengisytiharan pemboleh ubah setempat pula dibuat di dalam blok badan fungsi.

- Terdapat sedikit perbezaan skop antara dua pemboleh ubah ini jika pemboleh ubah setempat diisytihar bukan dibaris awal badan fungsi kerana skopnya hanya boleh dicapai setelah pemboleh ubah setempat diisytihar, manakala pemboleh ubah parameter formal fungsi boleh dicapai diseluruh lokasi badan fungsi.

6.6.1 Skop Pemboleh Ubah

- Skop tiga jenis pemboleh ubah setempat di dalam empat atur cara :

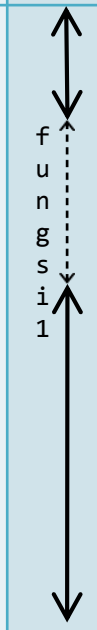
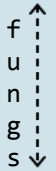
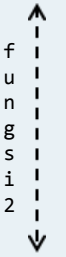
Kod Segmen	baris	x1	x2	x3	X4
`#include <stdio.h>`	1				
`void fungsi1();`	2				
`void fungsi2(int*);`	3				
`int x1 = 1, x2 = 2; //pemboleh ubah sejagat`	4				
	5				
`int main(){  // blok main()`	6				
`  //pemboleh ubah setempat`	7				
`  int x1 = 10, x3 = 30;`	8				
`  printf("x1 di blok main(): %d\n", x1);`	9				
`  printf("x2 di blok main(): %d\n", x2);`	10				
`  printf("x3 di blok main(): %d\n\n", x3);`	11				
`  { // sub-blok main()`	12				
`    //pemboleh ubah setempat`	13				
`    int x1 = 101, x3 = 301;`	14				
`    printf("x1 di sub-blok main(): %d\n", x1);`	15				
`    printf("x2 di sub-blok main(): %d\n", x2);`	16				
`    printf("x3 di sub-blok main(): %d\n\n",`	17				
`             x3);`	18				
`    }`	19				
`    fungsi1();`	20				
`    fungsi2(&x3);`	21				
`    return 0;`	22				
`}`	23				
	24				



6.6.1 Skop Pemboleh Ubah

- Skop tiga jenis pemboleh ubah setempat di dalam empat atur cara :

Kod Segmen	baris	x1	x2	x3	X4

void fungsi1(){ // blok fungsi1()
  // pemboleh ubah setempat
  int x1 = 11, x3 = 31;
  printf("x1 di blok fungsi1(): %d\n", x1);
  printf("x2 di blok fungsi1(): %d\n", x2);
  printf("x3 di blok fungsi1(): %d\n\n", x3);
}
 	
void fungsi2(int* x4){ // blok fungsi2()
  *x4 = 42; //pemboleh ubah parameter formal
  printf("x1 di blok fungsi2(): %d\n", x1);
  printf("x2 di blok fungsi2(): %d\n", x2);
  printf("x4 di blok fungsi2(): %d\n\n", *x4);
}
	25
	26
	27
	28
	29
	30
	31
	32
	33
	34
	35
	36
	37
	38
					

6.6.1 Skop Pemboleh Ubah

- Output contoh atur cara dengan empat blok atur cara :

```
x1 di blok main(): 10
x2 di blok main(): 2
x3 di blok main(): 30

x1 di sub-blok main(): 101
x2 di sub-blok main(): 2
x3 di sub-blok main(): 301

x1 di blok fungsi1(): 11
x2 di blok fungsi1(): 2
x3 di blok fungsi1(): 31

x1 di blok fungsi2(): 1
x2 di blok fungsi2(): 2
x4 di blok fungsi2(): 42
```

6.6.2 Skop Prototap Fungsi

- Seperti skop pemboleh ubah, skop prototaip fungsi juga boleh diisytihar secara sejagat dan setempat.

- Prinsipnya juga sama dengan skop pemboleh ubah. Prototaip sejagat boleh diisytihar di luar fungsi dan prototaip setempat diisytiharkan di dalam fungsi.

- Skop panggilan prototaip sejagat, boleh dipanggil di mana-mana fungsi pada atur cara.

- Skop panggilan prototaip setempat, hanya boleh dipanggil dalam fungsi yang mengisytiharkannya.

6.6.2 Skop Prototap Fungsi

- Contoh aturcara pengiraan luas dengan fungsi prototaip sejagat dan setempat:

```
1:  //Mengira dan memaparkan luas dengan pengisytiharan
2:  // prototaip setempat dan sejagat
3:  #include <stdio.h>
4:
5:  // 3 prototaip fungsi disiytihar sejagat
6:  void luas_segi ();
7:  float luas_bulatan (float jejari);
8:  void menu ();
9:
10: int main(){
11:    float luas=0, j;
12:    int pilih;
13:
14:    menu();
15:    scanf("%d", &pilih);
16:         if (pilih == 1)
17:            luas_segi ();
18:         else if (pilih == 2 ) {
19:         printf("Masukkan saiz: jejari\n");
20:                   scanf("%f", &j);          luas = luas_bulatan (j);
21:          printf("luas bulatan ialah %0.2f\n", luas);
```

6.6.2 Skop Prototap Fungsi

- Contoh aturcara pengiraan luas dengan fungsi prototaip sejagat dan setempat:

```
22:   } else   printf("Pilih 1, 2 @ 3 sahaja\n");
23:
24:     return 0;
25:   }
26:
27:   void menu (){
28:     printf("1: Luas bersegi\n");
29:     printf("2: Luas bulatan\n");
30:     printf("pilih 1 @ 2 >");
31:   }
32:
33:   void luas_segi (){
34:     // 2 prototaip fungsi diisytihar setempat
35:     float luas_segi4 ();
36:     float luas_segi3 (float tapak, float tinggi);
37:     float tpk, tg;
38:     float luas4, luas3;
39:
40:     luas4 = luas_segi4 ();
41:     printf("luas segiempat ialah %0.2f\n", luas4);
42:     printf("masukkan saiz: tapak lebar\n");
```

6.6.2 Skop Prototap Fungsi

- Contoh aturcara pengiraan luas dengan fungsi prototaip sejagat dan setempat:

```
43:      scanf("%f %f", &tpk, &tg);
44:     luas3 = luas_segi3 (tpk, tg);
45:     printf ("Luas segitiga ialah %0.2f\n", luas3);
46:  }
47:
48:   float luas_bulatan (float jejari){
49:    return (3.1415*jejari*jejari);
50:  }
51:
52:  float luas_segi4 (){
53:    float pjg, lbr;
54:
55:    printf("masukkan saiz: panjang lebar\n");
56:    scanf("%f %f", &pjg, &lbr);
57:    return (pjg*lbr);
58:  }
59:
60:  float luas_segi3 (float tapak, float tinggi){
61:    float luas;
62:    luas =(tapak*tinggi)/2.0;
63:    return ( luas ); }
```

6.6.3 Kelas Storan

- Storan kelas adalah ketetapan pemboleh ubah yang akan menentukan skop pemboleh ubah tersebut dalam satu atur cara.

- Terdapat empat jenis kelas storan di dalam C iaitu
  1. luaran
  2. automatik
  3. statik
  4. daftar

- Kelas storan luaran digunakan untuk menyatakan bahawa sesuatu pemboleh ubah sejagat dirujuk pada sesuatu blok atur cara dengan mengguna kata kunci `extern`. Format pengisytiharan:

```
extern jenis_data nama_pemboleh_ubah;
```

6.6.3 Kelas Storan

- Contoh: Dengan menambahkan kenyataan pemboleh ubah luaran pada fungsi `fungsi1()` seperti berikut, ia tidak mengubah hasil perlaksanaan fungsi1() tetapi ia hanya menyatakan dengan lebih jelas bahawa pemboleh ubah x2 merujuk kepada pemboleh ubah sejagat yang telah di isytihar sebelum ini.

```
void fungsi1(){ // blok fungsi1()
    int x1 = 11, x3 = 31; // pemboleh ubah setempat
    extern int x2;    // merujuk kepada pemboleh ubah sejagat
    printf("x1 di blok fungsi1(): %d\n", x1);
    printf("x2 di blok fungsi1(): %d\n", x2);
    printf("x3 di blok fungsi1(): %d\n\n", x3);
}
```

6.6.3 Kelas Storan

- **Kelas storan automatik** adalah merujuk kepada pemboleh ubah setempat, dan untuk menjadikan pengisytiharan pemboleh ubah automatik lebih jelas, maka kata kunci auto boleh ditambah semasa pengisytiharan pemboleh ubah setempat.

- Format pengisytiharan pemboleh ubah automatik atau setempat dengan kata kunci auto adalah seperti berikut:

```
auto jenis_data nama_pemboleh_ubah;
```

6.6.3 Kelas Storan

- **Kelas storan setempat statik** digunakan untuk mengekalkan nilai setempat dan ia membolehkan ingatan kekal diperuntukkan untuk pemboleh ubah ini.

- Kata kunci static digunakan pada pengisytiharan pemboleh ubah setempat dan format pengisytiharan pemboleh ubah setempat statik adalah seperti berikut:

```
static jenis_data nama_pemboleh_ubah;
```

6.6.3 Kelas Storan

- Contoh atur cara penggunaan pemboleh ubah setempat statik dan setempat:

```
 1:  // Mencetak pemboleh ubah setempat statik x dan setempat y
 2:  #include <stdio.h>
 3:
 4:  void fungsi1();
 5:
 6:  int main(){
 7:     int i;
 8:     for (i=0; i<5; i++){
 9:             printf("i = %d\n", i);
10:            fungsi1();
11:          }
12:
13:      return 0;
14:  }
15:
```

6.6.3 Kelas Storan

- Contoh atur cara penggunaan pemboleh ubah setempat statik dan setempat:

```
16: void fungsi1(){
17:     static int x=10; //pemboleh ubah setempat statik
18:     int y=10; //pemboleh ubah setempat
19:     printf("x di fungsi1(): %d\n", x);
20:     printf("y di fungsi1(): %d\n", y);
21:     x++;
22:     y++;
23: }
```

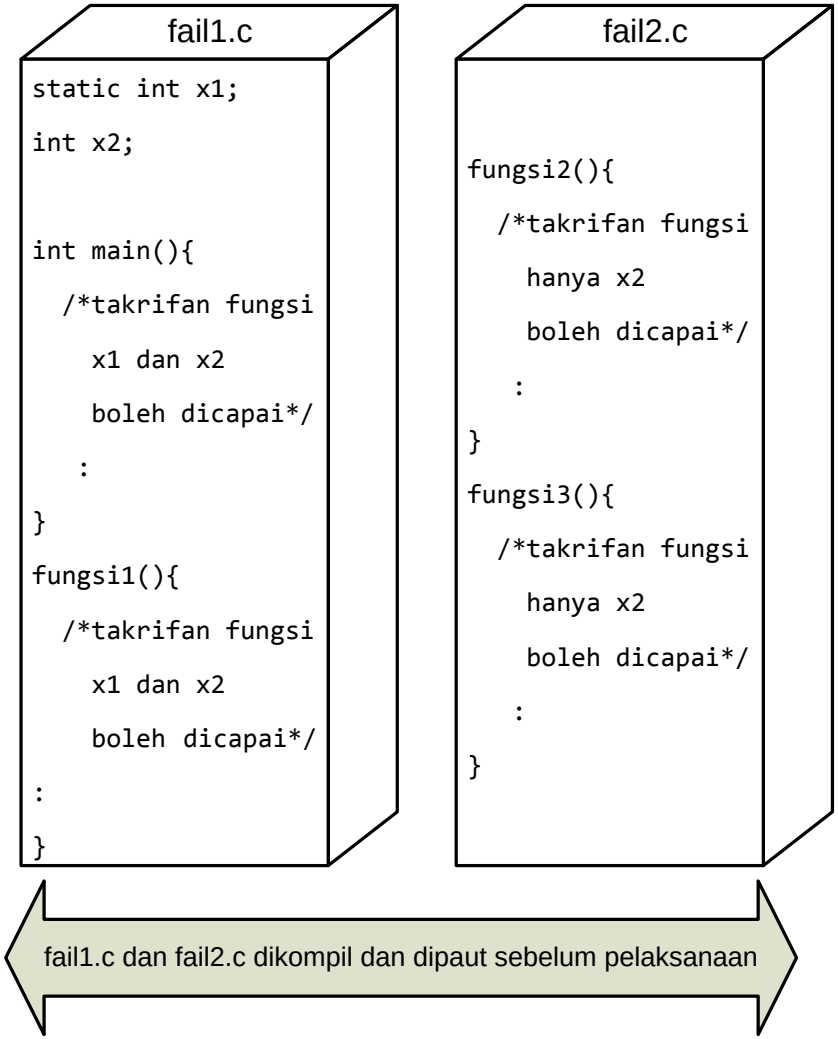
6.6.3 Kelas Storan

- Contoh atur cara penggunaan pemboleh ubah setempat statik dan setempat output:

```
i = 0
x di fungsi1(): 10
y di fungsi1(): 10
i = 1
x di fungsi1(): 11
y di fungsi1(): 10
i = 2
x di fungsi1(): 12
y di fungsi1(): 10
i = 3
x di fungsi1(): 13
y di fungsi1(): 10
i = 4
x di fungsi1(): 14
y di fungsi1(): 10
```

6.6.3 Kelas Storan

- Contoh penggunaan pemboleh ubah statik luaran x1 dan luaran x2:



6.6.3 Kelas Storan

- **Pemboleh ubah daftar** adalah pemboleh ubah automatik yang membolehkan nilai disimpan di dalam daftar CPU.

- bahasa C menyediakan kelas storan daftar supaya pengatur cara boleh mencadangkan kepada pengkompil pemboleh ubah automatik tertentu untuk diperuntukkan kepada daftar CPU.

- pemboleh ubah daftar menyediakan kawalan tertentu ke atas kecekapan pelaksanaan atur cara terutama pada sistem terbenam. Format pengisytiharan pemboleh ubah daftar dengan kata kunci register adalah seperti berikut:

```
register jenis_data nama_pemboleh_ubah;
```

6.6.3 Kelas Storan

- Contoh pemboleh ubah y pada blok `fungsi1()` di Atur cara sebelum ini adalah pemboleh ubah setempat, kenyataan pengisytiharan pemboleh ubah y boleh diubah menjadi pemboleh ubah daftar dengan kenyataan berikut:

```
register int y=10;
```