DAYANG NORHAYATI ABANG JAWAWI
ROSBI MAMAT

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Bab 7: Tatasusunan dan Penunding

7.1 Pengenalan

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

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

- Jika bil. perintang bertambah cthnya 10 pengisytiharan di atas menjadi rumit. Utk memudahkan pengisytiharan seperti di atas tatasusunan boleh digunakan.

- Tatasusunan adalah satu struktur data mudah yang boleh digunakan untuk memudahkan pengisytiharan satu pemboleh ubah bagi penyimpanan banyak data daripada jenis yang sama.

7.1 Pengenalan

- Kebanyakan operasi tatasusunan boleh dilaksanakan dengan menggunakan penuding.

- Konsep dan pelaksanaan tatasusunan dan penuding membolehkan penyimpanan dan penggunaan data berganda daripada jenis data yang sama.

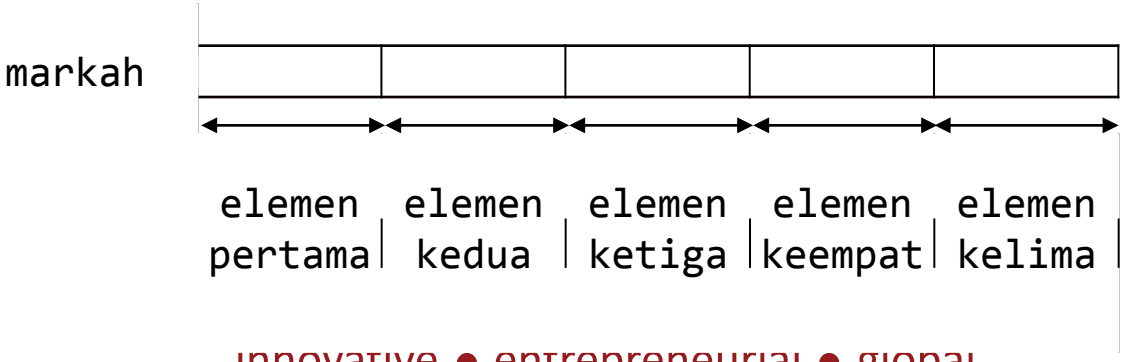
- Jenis data takrifan pengguna dengan tatasusunan boleh ditakrifkan ke dalam beberapa bentuk seperti satu-dimensi dan berbilang dimensi.

7.2 Senarai dengan Tatasusunan

- Senarai adalah himpunan data yang mempunyai jenis yang sama.

- Tatasusunan adalah satu jujukan elemen yang digunakan untuk menyimpan satu kumpulan nilai data yang sama jenisnya dan dirujuk dengan menggunakan satu pemboleh ubah sahaja.

- Elemen tatasusunan disimpan secara berjujukan dalam ingatan utama dengan satu elemen bagi setiap sel ingatan. contoh jujukan elemen tatasusunan markah yang boleh menyimpan lima elemen berjenis integer:



7.2.1 Pengisytiharan dan Pengawalan Eleman Tatasusunan

- Tatasusunan adalah jenis data takrifan pengguna yang memerlukan pengguna mengisytiharkan data adalam struktur tertentu menggunakan jenis data mudah.

- Pengisytiharan tatasusunan menyatakan saiz elemen sesuatu tatasusunan, format sintaks pengisytiharan tatasusunan satu dimensi ialah:

```
jenis_data nama_tatasusunan[saiz_elemen];
```

- Saiz elemen tatasusunan merujuk kepada bilangan elemen tatasusunan dan ia mestilah pemalar integer yang bernilai lebih daripada 0. Contoh:

```
int markah[5];
```

7.2.1 Pengisytiharan dan Pengawalan Eleman Tatasusunan

- Contoh pengisytiharan tatasusunan markah yang menggunakan pemalar untuk bilangan elemen adalah seperti berikut:

```
const int BIL_MARKAH =5;
int markah[BIL_MARKAH];
```

- Tatasusunan boleh diberi nilai awalan dengan menyertakan satu senarai nilai dalam kurungan {} semasa pengisytiharan pemboleh ubah dibuat. Nilai-nilai ini akan diumpuk ke dalam tatasusunan mengikut urutan nilai di dalam kurungan.

- Pengawalan nilai semasa pengisytiharan boleh dilakukan menggunakan format berikut:

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

7.2.1 Pengisytiharan dan Pengawalan Eleman Tatasusunan

- Jika dua pemboleh ubah tatasusunan sejagat dengan contoh pengisytiharan dan umpukan nilai awalan seperti berikut.

```
int hari_hijrah[12]={29, 29, 30, 29, 30, 29, 30, 30,
29, 30, 29, 30};
double markah_tugasan[4]={70.3, 83.5};
```

- Contoh struktur yang terhasil daripada pengisytiharan pemboleh ubah `hari_hijrah` dan `markah_tugasan` yang diberi nilai awalan :

`hari_hijrah`

29	29	30	29	30	29	30	30	29	30	29	30

`markah_tugasan`

70.3	83.5	0.0	0.0

7.2.1 Pengisytiharan dan Pengawalan Eleman Tatasusunan

- Memberi saiz tatasusunan secara tersirat dilakukan apabila satu tatasusunan diisytihar tanpa saiz tetapi mesti diberi nilai awalan dan bilangan nilai awalan ini akan digunakan sebagai saiz tatasusunan tersebut.

- Contoh struktur tatasusunan yang terhasil dari pengisytiharan tatasusunan markah_kuiz dengan pengawalan lalai tatasusunan :

```
float markah_kuiz[]={12.5, 11.0, 22.5, 20.8, 50.0};
```

markah_kuiz	12.5	11.0	22.5	20.8	50.0

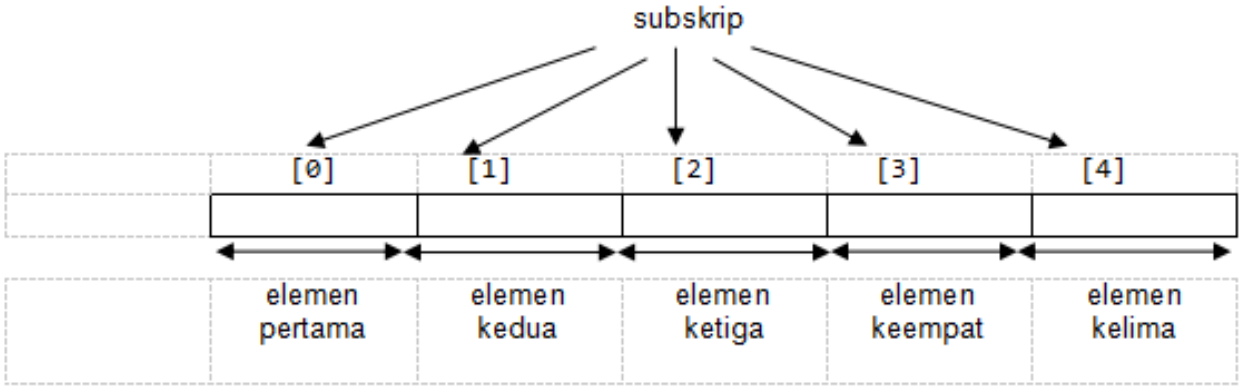
7.2.2 Mencapai Kandungan Tatasusunan

- Setiap elemen pada satu tatasusunan diberi satu nilai subskrip unik. Elemen tatasusunan dicapai secara individu dengan menggunakan subskrip unik ini.

- Subskrip element tatasusunan digunakan sebagai indeks penunjuk kepada elemen tertentu pada tatasusunan dan untuk bahasa C, subskrip tatasusunan bermula dengan indeks 0.

7.2.2 Mencapai Kandungan Tatasusunan

- Contoh subskrip yang digunakan untuk mencapai elemen pada tatasusunan dengan lima elemen:



- Format ungkapan untuk mencapai nilai yang disimpan di dalam elemen-elemen tatasusunan:

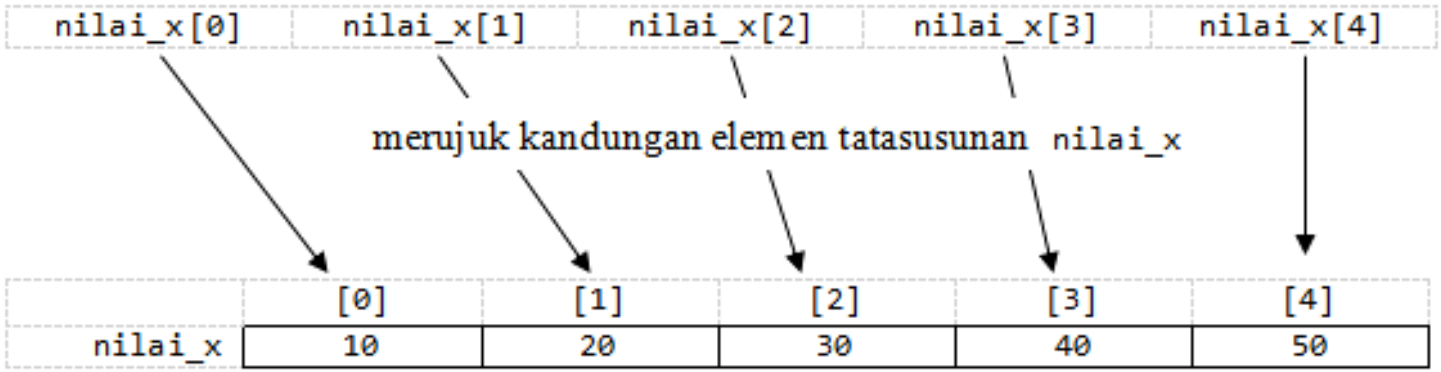
```
nama_tatasusunan[subskrip]
```

7.2.2 Mencapai Kandungan Tatasusunan

- Contoh:

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

- Kandungan tatasusunan nilai_x :



7.2.2 Mencapai Kandungan Tatasusunan

- contoh untuk mencapai elemen-elemen tatasusunan nilai_x dengan menggunakan subskrip:

```
int a = 1, b = 3;
printf("%d %d %d", x[2], x[a], x[a+b]);
```

- Hasil capaian elemen dan output daripada keratan atur cara di atas adalah seperti berikut::

```
30 20 50
```

7.2.3 Operasi Tatasusunan dengan Gelung

- Gelung digunakan untuk membuat operasi yang sama pada setiap elemen pada tatasusunan. Kebiasaannya gelung for digunakan untuk mengawal ulangan pada setiap elemen menggunakan subskrip.

- Contoh:

```
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;
```

7.2.3 Operasi Tatasusunan dengan Gelung

- Kaduangan tatasusunan kuasa2 :

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

- Contoh aturcara penggunaan gelung for untuk kiraan nilai kuasa dua

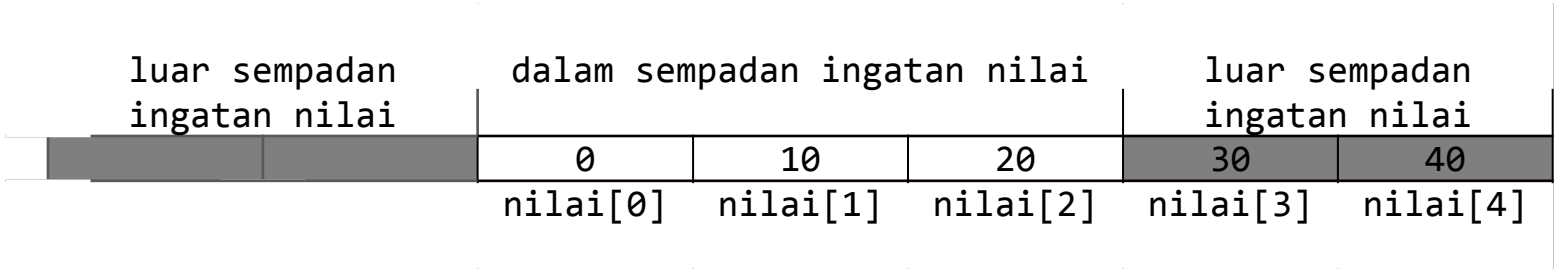
```
1:  //Mengira kuasa dua nilai indeks#
2:  include <stdio.h>
3:
4:  int main (){
5:       const int SAIZ = 11;
6:       int i;
7:       int kuasa2[SAIZ];
8:
9:       for (i=0; i< SAIZ ; i++) {
10:           kuasa2[i] = i * i;
11:           printf("%d ", kuasa2[i]);
12:          }
13:
14:          return 0;
15: }
```

7.2.3 Operasi Tatasusunan dengan Gelung

- Contoh ingatan di luar sempadan yang tidak diperuntukkan untuk tatasusunan nilai telah ditukar kepada nilai 30 dengan subskrip 3 dan 40 dengan subskrip 4 :

```
int nilai[3];
for (i=0; i< 5 ; i++) {
    nilai[i] = i*10;
}
```

- Lokasi di luar sempadan ingatan yang diganti dengan subskrip tidak sah tatasusunan nilai :



7.3 Penuding, Tatasusunan dan Aritmetik Penuding

- Penuding digunakan untuk merujuk nilai perameter sebenar yang mana alamat paremeter ini dihantar ke fungsi melalui parameter formal fungsi.

- Dengan konsep yang sama elemen tatasusunan boleh juga dicapai dengan merujuk alamat elemen tatasusunan.

7.3.1 Pemboleh ubah Penuding

- Pemboleh ubah penuding adalah pemboleh ubah yang menyimpan alamat ingatan yang merjuk kepada lokasi data disimpan.

- Alamat ialah ruang ingatan yang didefinisikan secara unik.

- Alamat sesuatu pemboleh ubah boleh dirujuk dengan menggunakan operator & dengan format seperti berikut:

```
&nama_pemboleh_ubah
```

7.3.1 Pemboleh ubah Penuding

- Contoh :

```
int p_ubah = 37;
printf("nilai pemboleh ubah %d\n", p_ubah);
printf("alamat pemboleh ubah bentuk hexa %X", & p_ubah);
```

- Output keratan kod yang mencetak nilai dan alamat satu pemboleh ubah

```
nilai pemboleh ubah 37
alamat pemboleh ubah bentuk hexa 23FE4C
```

7.3.1 Pemboleh ubah Penuding

- Alamat pemboleh ubah seperti p_ubah boleh disimpan di dalam satu pemboleh ubah penuding dari jenis yang sama. Pengisytiharan pemboleh ubah penuding perlu dibuat sebelum alamat ingatan boleh di simpan di pemboleh ubah penuding

- Format :

```
jenis_data *nama_penuding;
```

- Sintaks berikut digunakan untuk mengumpukkan satu alamat pemboleh ubah kepada pemboleh ubah penuding:

```
nama_penuding = &pemboleh_ubah;
```

7.3.1 Pemboleh ubah Penuding

- Jenis data pemboleh ubah yang hendak diumpukkan ke penuding mestilah sama dengan pemboleh ubah penuding.

- Berikut menunjukkan contoh pengisytiharan pemboleh ubah penuding `penuding1` yang diumpukkan alamat pemboleh ubah `p_ubah`:

```
int *penuding1;
penuding1 = &p_ubah;
```

- Format ungkapan untuk mencapai nilai yang dirujuk atau ditunjuk oleh pemboleh ubah penuding adalah seperti berikut:

```
*nama_penuding
```

7.3.1 Pemboleh ubah Penuding

- contoh penggunaan pemboleh ubah penuding untuk mencapai nilai satu pemboleh ubah lain:

```
 1:  #include <stdio.h>
 2:
 3:  int main (){
 4:    // pengisytiharan
 5:    int p_ubah;
 6:    int *penuding1;
 7:    int *penuding2 = NULL;
 8:
 9:    printf("Alamat pemboleh ubah\n");
10:    printf("\tpubah: %X \n\tpenuding1: %X \n\tpenuding2: %X\n\n",
11:           &p_ubah, &penuding1, &penuding2);
12:
13:    // umpukan p_ubah dan penuding1
14:    p_ubah = 37;
15:    penuding1 = &p_ubah;
16:
17:    printf("Kandungan pubah adalah %d\n", p_ubah);
18:    if (penuding1 == NULL)
19:           printf("Kandungan penuding1 adalah nol\n");
20:    else {
21:           printf("Kandungan penuding1 adalah %X\n", penuding1);
```

7.3.1 Pemboleh ubah Penuding

- contoh penggunaan pemboleh ubah penuding untuk mencapai nilai satu pemboleh ubah lain(sambungan):

```
22:       printf("Nilai yang ditunjuk penuding1 adalah %d\n",
23:           *penuding1);
24:     }
25:     if (penuding2 == NULL)
26:         printf("Kandungan penuding2 adalah nol\n");
27:     else {
28:           printf("Kandungan penuding2 adalah %X\n", penuding2);
29:           printf("Nilai yang ditunjuk penuding2 adalah %d\n",
30:                   *penuding2);
31:      }
32:      printf("\n");
33:
34:      // umpukan penuding2
35:      penuding2 = penuding1;
36:     *penuding2 = 55;
37:
38:      printf("Kandungan pubah adalah %d\n", p_ubah);
39:      if (penuding1 == NULL)
40:          printf("Kandungan penuding1 adalah nol\n");
41:      else {
```

7.3.1 Pemboleh ubah Penuding

- contoh penggunaan pemboleh ubah penuding untuk mencapai nilai satu pemboleh ubah lain: (sambungan)

```
42:     printf("Kandungan penuding1 adalah %X\n", penuding1);
43:     printf("Nilai yang ditunjuk penuding1 adalah %d\n",
44:                 *penuding1);
45:    }
46:    if (penuding2 == NULL)
47:        printf("Kandungan penuding2 adalah nol\n");
48:    else {
49:          printf("Kandungan penuding2 adalah %X\n", penuding2);
50:          printf("Nilai yang ditunjuk penuding2 adalah %d\n",
51:                 *penuding2);
52:    }
53:
54:   return 0;
55:  }
```

7.3.1 Pemboleh ubah Penuding

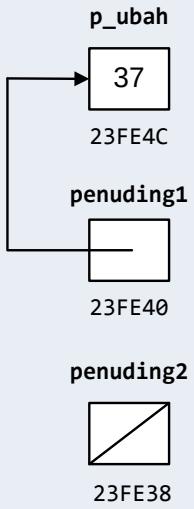
- Gambaran perubahan ruang ingatan semasa pengisytiharan dan umpukan pemboleh ubah penuding:

Kenyataan	Ruang ingatan
Pengisytiharan pemboleh ubah	

`int p_ubah;`
`int * penuding1;`
`int * penuding2 = NULL;`
	

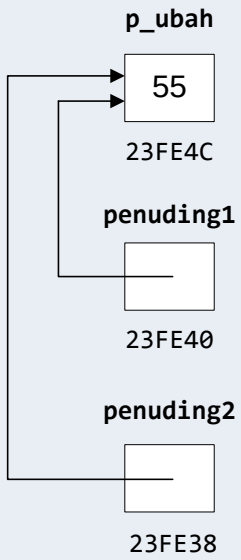
7.3.1 Pemboleh ubah Penuding

- Gambaran perubahan ruang ingatan semasa pengisytiharan dan umpukan pemboleh ubah penuding:

Kenyataan	Ruang ingatan
Umpukan p_ubah dan penuding1	
`p_ubah = 37;` `penuding1 = &p_ubah;`	

7.3.1 Pemboleh ubah Penuding

- Gambaran perubahan ruang ingatan semasa pengisytiharan dan umpukan pemboleh ubah penuding:

Kenyataan	Ruang ingatan
Umpukan penuding2	
`penuding2 = penuding1;` `*penuding2 = 55;`	

7.3.1 Pemboleh ubah Penuding

- Output atur cara yang menunjukkan contoh penggunaan pemboleh ubah:

```
Alamat pemboleh ubah
        pubah: 23FE4C
        penuding1: 23FE40
        penuding2: 23FE38

Kandungan pubah adalah 37
Kandungan penuding1 adalah 23FE4C
Nilai yang ditunjuk penuding1 adalah 37
Kandungan penuding2 adalah nol

Kandungan pubah adalah 55
Kandungan penuding1 adalah 23FE4C
Nilai yang ditunjuk penuding1 adalah 55
Kandungan penuding2 adalah 23FE4C
Nilai yang ditunjuk penuding2 adalah 55
```

7.3.2 Penuding kepada Tatasusunan

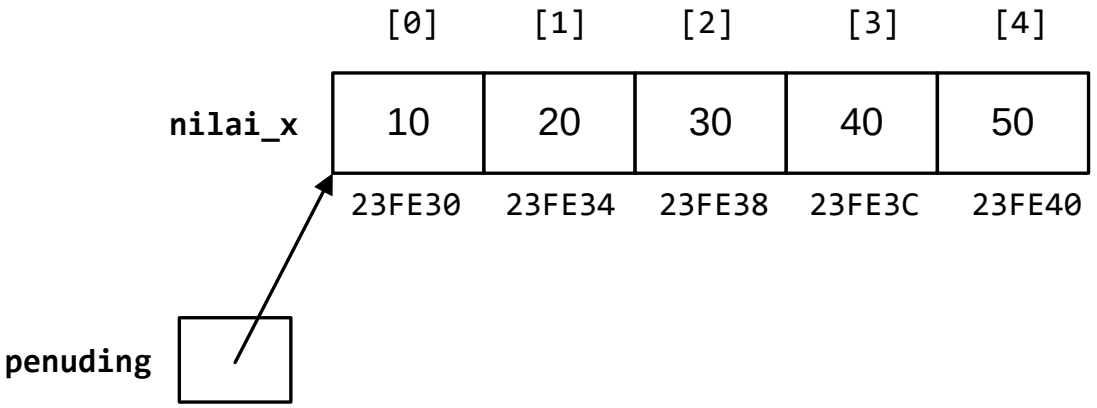
- Penuding kepada tatasusunan : satu pemboleh ubah penuding boleh menyimpan alamat elemen pertama tatasusunan, maka ini membolehkan pemboleh ubah penuding merujuk kepada nilai elemen pertama tatasusunan.

- Contoh: `nilai_x[0]` dan `*penuding` merujuk kepada nilai yang sama iaitu 10.

```
int nilai_x [5] = { 10, 20, 30, 40, 50 };
int *penuding = nilai_x;
```

7.3.2 Penuding kepada Tatasusunan

- Gambaran situasi ruangan ingatan yang mana pemboleh ubah `penuding` dan `nilai_x` merujuk kepada lokasi ingatan yang sama iaitu `23FE30`:



7.3.2 Penuding kepada Tatasusunan

- Elemen pertama tatasusunan boleh dicapai dengan tiga cara iaitu dengan menggunakan:

1. Pemboleh ubah tatasusuanan dan subskrip, contoh `nilai_x[0]`
2. Pemboleh ubah tatasusunan menggunakan penuding, contoh `*nilai_x`
3. Pemboleh ubah penuding kepada tatasusunan, contoh `*penuding.`

7.3.2 Penuding kepada Tatasusunan

- Jujukan alamat kepada elemen tatasusunan ini membolehkan elemen-elemen tatasusunan dirujuk secara mudah dengan menggunakan penuding. Ia boleh dirujuk dengan menggunakan aritmetik penuding.

- **Aritmetik penuding** menggunakan ungkapan aritmetik kepada pemboleh ubah penuding untuk merujuk kepada elemen-elemen tatasusunan.

7.3.2 Penuding kepada Tatasusunan

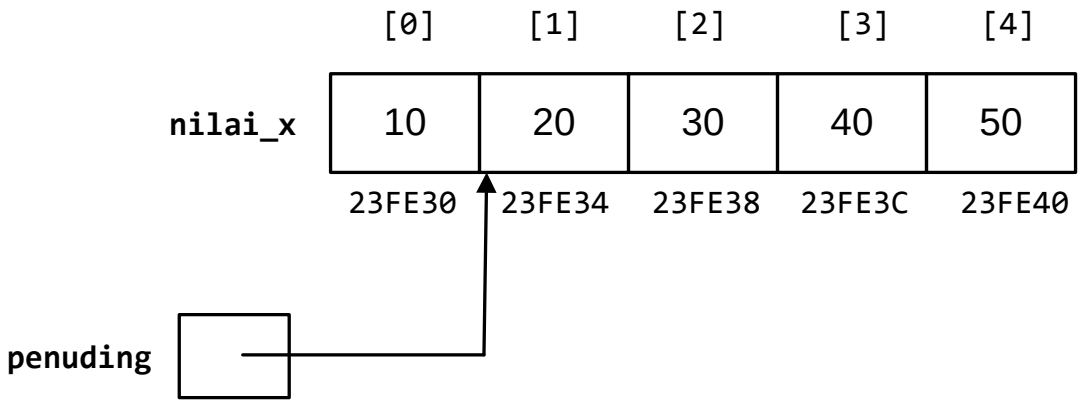
- Nilai dan alamat elemen tatasusunan dicapai dengan menggunakan pemboleh ubah penuding:

Pemalar	penuding pada elemen pertama tatasusunan	penuding pada elemen kedua tatasusunan
*penuding	10	20
penuding	23FE30	23FE34
penuding + 3	23FE3C	23FE40
*(penuding + 3)	40	50

- lajur kedua yang merujuk kepada capaian nilai bila penuding pada lokasi seperti di gambaran di slide 29 sebelum ini

7.3.2 Penuding kepada Tatasusunan

- Gambaran ruang ingatan pemboleh ubah penuding setelah kenyataan `penuding = penuding + 1` :



7.3.2 Penuding kepada Tatasusunan

- Satu elemen tatasusunan boleh dicapai sama ada dengan mengunakan pemboleh ubah tatasusunan bersama subskrip atau penuding kepada tatasusunan bersama aritmetik penuding.

- Contoh atur cara akan menggunakan dua gelung untuk merujuk kepada elemen-elemen tatasusunan nilai_x.

- Gelung pertama menggunakan pemboleh ubah penuding dan aritmetik penuding untuk menukar nilai setiap elemen tatasusunan, manakala gelung kedua menggunakan pemboleh ubah tatasusunan dan subskrip untuk mencetak setiap elemen tatasusunan.

- hasil atur cara ini akan mencetak nilai-nilai “8 18 28 38 48”.

7.3.2 Penuding kepada Tatasusunan

- Contoh atur cara:

```
 1:  #include <stdio.h>
 2:
 3:  int main (){
 4:            int nilai_x [5] = { 10, 20, 30, 40, 50 };
 5:            int *penuding = nilai_x;
 6:
 7:           int i;
 8:
 9:            for (i=0; i<5; i++)
10:                     *(penuding+i) -= 2;
11:
12:           for (i=0; i<5; i++)
13:                     printf("%d  ", nilai_x[i]);
14:
15:           return 0;
16:  }
```

7.4 Menghulur Tatasusunan kepada Fungsi

- Tatasusunan boleh dihulur ke fungsi samada :
  1. individu elemen
  2. kesemua elemen tatasusunan

- Penghuluran satu persatu elemen kepada tatasusunan pula boleh dibuat melalui nilai sebenar satu alamat nilai.

7.4.1 Individu Elemen melalui Nilai Sebenar

- Penghuluran individu elemen tatasusunan ke fungsi adalah seperti penghuluran pemboleh ubah secara nilai ke fungsi.

- Ini bermaksud satu nilai elemen tatasusunan tersebut disalin ke lokasi ingatan lain untuk diproses oleh fungsi tersebut.

7.4.1 Individu Elemen melalui Nilai Sebenar

- Contoh atur cara yang menghulur satu elemen ke fungsi ganda2( ).

```
1:  #include<stdio.h>
2:  #define BILMAX 8
3:
4:  int ganda2(int elemen);
5:
6:  int main (void)
7:  {
8:     int elemen[BILMAX] = {4,5,12,7,10,6,4,1};
9:     int elemen_ganda2[BILMAX], i;
10:
11:     for (i=0; i<BILMAX; i++) {
12:        elemen_ganda2[i] = ganda2(elemen[i]);
13:    }
14:    printf("Elemen tatasusunan elemen_ganda\n");
15:    for (i=0; i<BILMAX; i++){
16:           printf("%d\t", elemen_ganda2[i]); // hulur satu nilai
17: elemen
18:    }
19:    printf("\n\nElemen tatasusunan elemen\n");
```

7.4.1 Individu Elemen melalui Nilai Sebenar

- Contoh atur cara yang menghulur satu elemen ke fungsi ganda2( ). (Sambungan)

```
20:     for (i=0; i<BILMAX; i++){
21:         printf("%d\t", elemen[i]);
22:     }
23:      return 0;
24: }
25:
26: int ganda2(int elemen) // salin satu nilai ke elemen
27: {
28:     elemen = elemen * elemen;
29:     return (elemen);
30: }
```

7.4.1 Individu Elemen melalui Nilai Sebenar

- Output:

```
Elemen tatasusunan elemen_ganda
16      25      144     49      100     36      16      1

Elemen tatasusunan elemen
4       5       12      7       10      6       4       1
```

- setelah kembali dari memproses setiap elemen tatasusunan dengan memanggil fungsi `ganda2()` cetakan elemen tatasusunan elemen masih mengekalkan nilai asal tatasusunan elemen.

7.4.2 Individu Elemen melalui Alamat Nilai

- Untuk mencapai elemen tatasusunan secara terus, pemboleh ubah rujukan perlu dihulur kepada parameter fungsi dengan cara penghuluran induvidu elemen tatasusunan melalui alamat nilai.

- Maka, parameter sebenar akan menghantar alamat satu elemen tatasusunan dan parameter formal pula mengisytiharkan pemboleh ubah penuding untuk menerima penghuluran alamat ini.

7.4.2 Individu Elemen melalui Alamat Nilai

- Contoh aturcara menunjukkan bagaimana elemen tatasusunan dihulur sebagai pemboleh ubah rujukan:

```
 1:  #include<stdio.h>
 2:  void tukartempat(int *depan, int *belakang);
 3:
 4:  int main (void)
 5:  {
 6:     int tts[] = {0,1,4,9,16,25,36,49,64,81};
 7:     int i;
 8:
 9:     for (i=0; i<5; i++) {
10:        tukartempat(&tts[i], &tts[5+i]); // hulur alamat satu nilai
11:     }
12:
13:     for (i=0; i<10; i++){
14:           printf("%d\t", tts[i]);
15:      }
16:
```

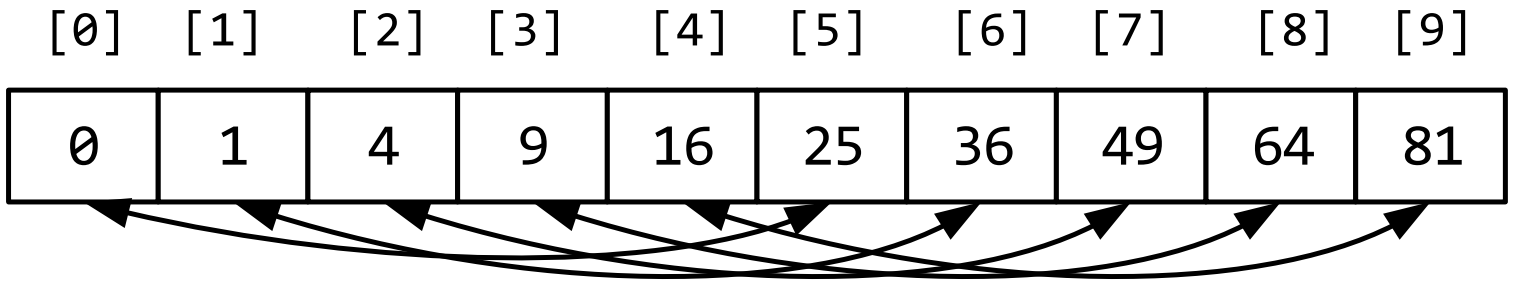
7.4.2 Individu Elemen melalui Alamat Nilai

- Contoh aturcara menunjukkan bagaimana elemen tatasusunan dihulur sebagai pemboleh ubah rujukan (Sambungan):

```
17:     return 0;
18: }
19: // simpan alamat pada pemboleh ubah penuding
20: void tukartempat(int *depan, int *belakang)
21: {
22:    int salinan;
23:
24:    salinan = *depan;
25:    *depan = *belakang;
26:    *belakang = salinan;
27: }
```

7.4.2 Individu Elemen melalui Alamat Nilai

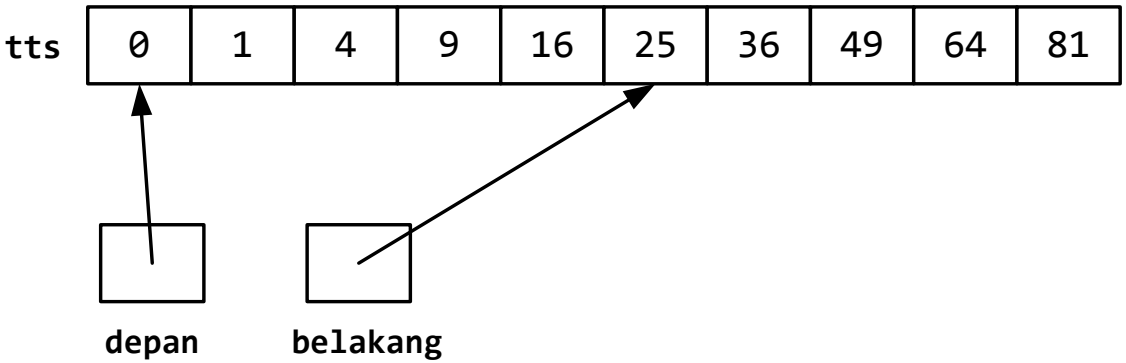
- Proses penukaran elemen pada tatasusunan `tts` :



proses menukar lima elemen hadapan dengan lima elemen belakang

7.4.2 Individu Elemen melalui Alamat Nilai

- Contohnya bila nilai i di baris 9 adalah 0, penuding depan dan belakang merujuk kepada alamat yang dihantar kepada fungsi seperti di bawah:



- Kenyataan mencetak nilai elemen dengan gelung di baris 13 hingga baris 15 akan memberi output "25 36 49 64 81 0 1 4 9 16".

7.4.3 Semua Elemen Tatasusunan

- Satu fungsi boleh merujuk kepada kesemua elemen tatasusunan dengan satu argumen dan parameter formal. Cara ini fungsi membolehkan semua elemen tatasusunan dirujuk hanya dengan sekali panggilan fungsi.

- dilakukan dengan menghulur nama tatasusunan pada parameter fungsi yang merujuk kepada alamat lokasi pertama tatasusunan

- Apabila alamat lokasi pertama dihulur, ini bermakna keseluruhan elemen tatasusunan dapat dicapai.

- jika tatasusunan dihulur ke fungsi dengan cara ini, secara automatik ia adalah penghuluran melalui alamat iaitu alamat pertama tatasusunan.

7.4.3 Semua Elemen Tatasusunan

- Contoh aturcara menghulur keseluruhan tatasusunan ke fungsiMenghulur keseluruhan tatasusunan ke fungsi

```
1:  #include<stdio.h>
2:  void tukartempat(int, int []);
3:
4:  int main (void)
5:  {
6:     int i, tts[] = {0,1,4,9,16,25,36,49,64,81};
7:
8:     tukartempat(10, tts);
9:     for (i=0; i<10; i++){
10:       printf("%d ", tts[i]); // hulur alamat elemen pertama
11:    }
12:
```

7.4.3 Semua Elemen Tatasusunan

- Contoh aturcara menghulur keseluruhan tatasusunan ke fungsiMenghulur keseluruhan tatasusunan ke fungsi (Sambungan)

```
13:  return 0;
14: }
15:
16: void tukartempat(int saiz, int elemen[]) // terima alamat
17: {
18:    int i, salinan;
19:
20:     for ( i=0; i<(saiz/2); i++ ) {
21:       salinan = elemen[i];
22:       elemen[i] = elemen[(saiz/2)+i];
23:       elemen[(saiz/2)+i] = salinan;
24:    }
25: }
```

7.4.3 Semua Elemen Tatasusunan

- Oleh kerana penghuluran tatasusunan ke fungsi ini dilakukan dengan alamat lokasi pertama tatasusunan, maka fungsi memanggil boleh menggunakan penuding untuk merujuk elemen tatatsusunan dalam fungsi.
- Contoh fungsi `tukartempat()` merujuk elemen-elemen tatasusunan dengan penuding adalah seperti berikut:

```
void tukartempat(int saiz, int *depan) // terima alamat
tatasusunan
{
   int i, salinan;
   int tengah = saiz/2;
   int *belakang = depan + tengah;

   for ( i=0; i<tengah; i++ ) {
                salinan = *(depan+i);
                *(depan+i) = *(belakang+i);
                *(belakang+i) = salinan;
   }
}
```

7.4.3 Semua Elemen Tatasusunan

- contoh `const` digunakan pada atur cara di baris 2 fungsi prototaip dan baris 14 kepala fungsi untuk memastikan elemen tatasusunan x tidak diubah secara tidak sengaja di dalam fungsi `terbesar()`.

- Contoh aturcara mencari nombor terbesar di dalam elemen tatasusunan menggunakan fungsi:

```
 1: #include<stdio.h>
 2: int terbesar(const int nom[8], int saiz);
 3:
 4: int main (void) {
 5:    int max,
 6:        tts[] = {25, 36, 49, 64, 81, 0, 1, 4, 9, 100};
 7:
 8:    max = terbesar(tts, 10);
 9:
10:    printf("Nombor terbesar %d", max);
11:    return 0;
12: }
13:
14: int terbesar(const int nom[8], int saiz){
15:    int i, besar, salinan;
```

7.4.3 Semua Elemen Tatasusunan

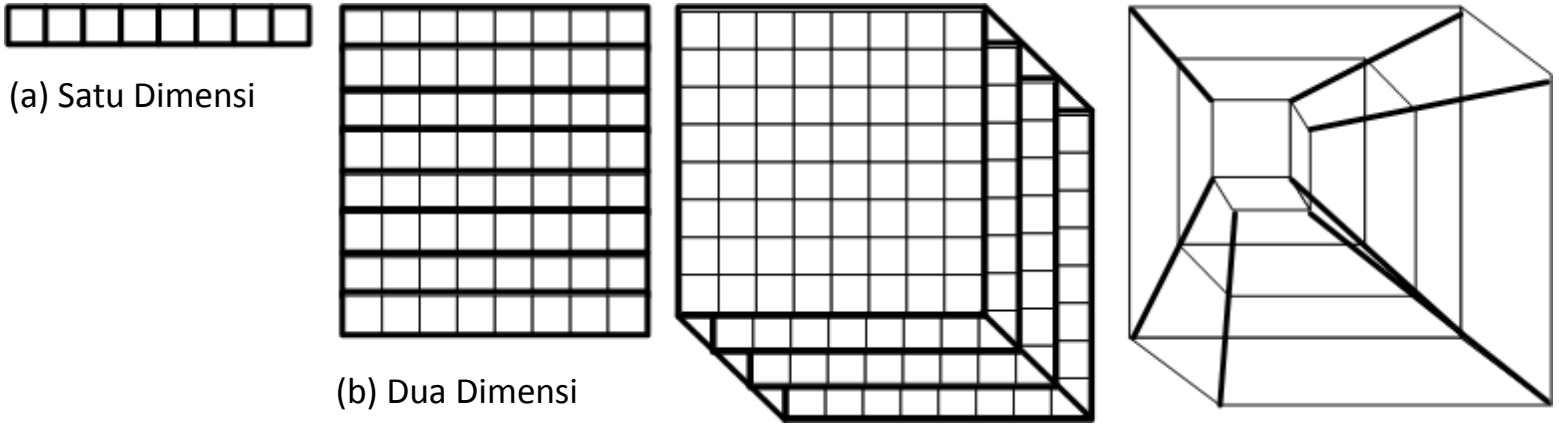
- Contoh aturcara (Sambungan)

```
16:
17:     besar = nom[0];
18:     for (i=1; i<saiz; i++){
19:        if (nom[i]>besar){
20:           besar = nom[i];
21:        }
22:     }
23:     return besar;
24: }
```

7.5 Tatasusunan Berbilang dimensi

- Tatasusunan berbilang dimensi adalah tatasusunan yang terbentuk daripada jujukan tatsusunan dimensi sebelumnya, contohnya jujukan tatasusunan satu dimensi membentuk tatasusunan dua dimensi; jujukan tatasusunan dua dimensi pula membentuk tatasusunan tiga dimensi dan seterusnya.

- Gambaran struktur grafik tatasusunan berbilang dimensi



(a) Satu Dimensi

(b) Dua Dimensi

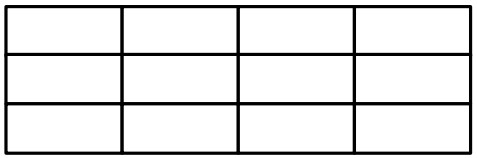
(c) Tiga Dimensi

(d) Empat Dimensi

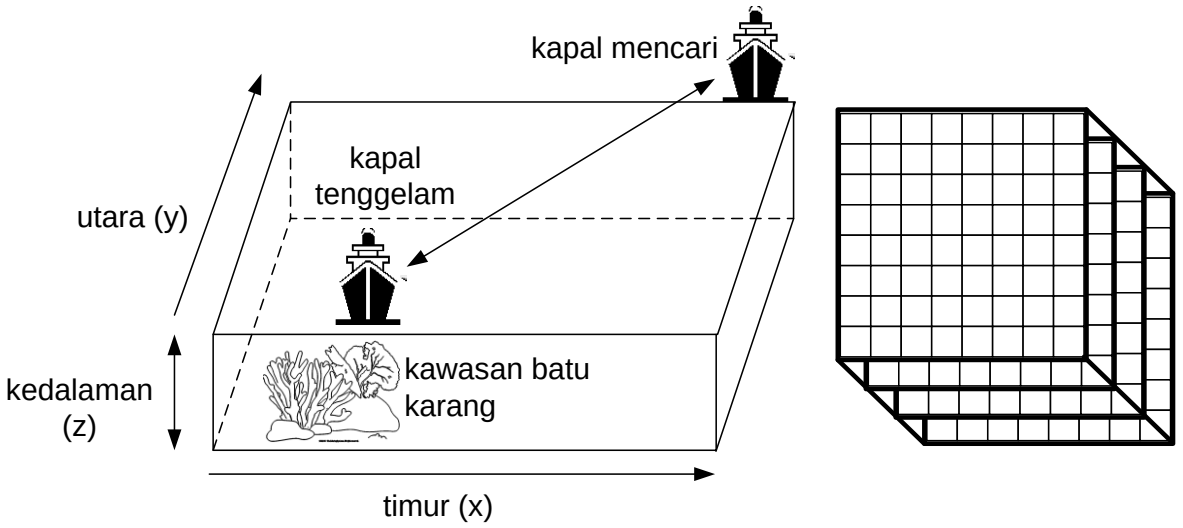
7.5 Tatasusunan Berbilang dimensi

- Contoh perwakilan Matrik A menggunakan struktur tatasusunan dua dimensi :

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



- Contoh perwakilan sistem radar untuk mencari kapal tenggelam di dasar lautan menggunakan struktur tatasusunan tiga dimensi



7.5 Tatasusunan Berbilang dimensi

- format pengisytiharan tatasusunan berbilang :

```
jenis_data nama_tatasusunan [saiz_ dimensi1] [saiz_ dimensi2]. . .
                                                [saiz_ dimensin];
```

- Contoh pengisytiharan tatasusunan dua dimensi :

```
int A[3][4];
```

- Contoh pengisytiharan tatasusunan tiga dimensi

```
float radar[X][Y][Z];;
```

7.5.1 Tatasusunan Dua Dimensi

- Elemen-elemen pada tatasusunan Matrik A boleh dirujuk dengan baris dan lajur subskrip seperti di bawah :

	[0]	[1]	[2]	[3]
[0]	A[0][0]	A[0][1]	A[0][2]	A[0][3]
[1]	A[1][0]	A[1][1]	A[1][2]	A[1][3]
[2]	A[2][0]	A[2][1]	A[2][2]	A[2][3]

- Pengisytiharan tatasusunan dua dimensi :

```
jenis_data nama_tatasusunan [baris][lajur];
```

- Format pengawalan nilai semasa pengisytiharan untuk tatasusunan dua dimensi

```
jenis_data nama_tatasusunan[baris][ lajur] = {senarai_data};
```

7.5.1 Tatasusunan Dua Dimensi

- Contoh pengisytiharan bagi matrik A :

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

- Contoh bagi yang lebih mudah untuk dibaca :

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

- Contoh bagi saiz baris boleh diabaikan

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

7.5.1 Tatasusunan Dua Dimensi

- Gambarajah tatasusunan dua dimensi yang terhasil daripada pengisytiharan matrik A :

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

- tatasusunan Matrik A boleh dicapai dengan kenyataan berikut:

```
A[1][2] = A[1][1] + 5; // elemen A[1][2] diumpuk nilai 9 (4 + 5)
tambah = A[1][2] + A[2][3]; // tambah diumpuk nilai 4 (3 + -7)
tambah += A[1][3]; // tambah diumpuk nilai 9 (4 + 5)
```

7.5.1 Tatasusunan Dua Dimensi

- Gelung bersarang boleh digunakan untuk mencapai dan memproses satu persatu elemen tatasusunan dua dimensi. Elemen-elemen tatasusunan boleh dicapai baris demi baris atau lajur demi lajur.

- Contoh aturcara:

```
 1: #include <stdio.h>
 2:
 3: int main ()
 4: {
 5:    int A[][4]= {{1, 2, -1, 4},{2, 4, 3, 5},
 6:                 {1, -2, 6, -7}};
 7:    int baris, lajur;
 8:
 9:    printf("Baris demi baris\n");
10:    for (baris=0; baris<3; baris++){
11:       for (lajur=0; lajur<4; lajur++)
12:          printf("%d\t",A[baris][lajur]);
```

7.5.1 Tatasusunan Dua Dimensi

- Contoh aturcara (Sambungan):

```
13:     printf("\n\n");
14:   }
15:
16:    printf("Lajur demi lajur\n");
17:    for (lajur=0; lajur<4; lajur++){
18:        for (baris=0; baris<3; baris++)
19:           printf("%d\t",A[baris][lajur]);
20:         printf("\n");
21:     }
22:     return 0;
23: }
```

- 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
```

7.5.1 Tatasusunan Dua Dimensi

- Contoh atur cara mengisytihar saiz lajur dan baris sebagai pemalar sejagat. Atur cara membaca nilai-nilai dari fail untuk dimasukkan ke dalam tatasusunan matrik dan mencetak elemen-elemen tatsusunan matrik dengan memanggil fungsi `cetakMatrik()`.

- Contoh fail `matrik3x4.dat` yang dibaca sebagai input Atur cara :

```
10 12 11 14
22 24 23 25
31 32 36 37
```

7.5.1 Tatasusunan Dua Dimensi

❖ Contoh atur cara :

```
 1:  #include <stdio.h>
 2:  #include <process.h>
 3:
 4:  #define BARIS  3
 5:  #define LAJUR  4
 6:
 7:  void cetakMatrik(const int tts[][LAJUR]);
 8:
 9:  int main ()
10:  {
11:        int matrik[BARIS][LAJUR];
12:        int baris, lajur;
13:           FILE  *fmatrik;
14:
15:           fmatrik = fopen("matrik3x4.dat", "r");
16:
```

7.5.1 Tatasusunan Dua Dimensi

❖ Contoh atur cara (Sambungan) :

```
17:     if (fmatrik == NULL)
18:     {
19:          printf("Ralat dalam pembukaan fail.\n");
20:          exit(-1);
21:     }
22:
23:    for (baris=0; baris<BARIS; baris++){
24:      for (lajur=0; lajur<LAJUR; lajur++)
25:        fscanf(fmatrik, "%d", &matrik[baris][lajur]);
26:  }
27:
28:    cetakMatrik(matrik);
29:
30:   return 0;
31:  }
32:
33:  void cetakMatrik(const int tts[][LAJUR])
34:  {
```

7.5.1 Tatasusunan Dua Dimensi

- Contoh atur cara (Sambungan) :

```
35:     int baris, lajur;
36:     printf("Baris demi baris\n");
37:     for (baris=0; baris<BARIS; baris++){
38:       for (lajur=0; lajur<LAJUR; lajur++)
39:           printf("%d\t", tts[baris][lajur]);
40:         printf("\n");
41:     }
42: }
43:
44:
```

7.6 Rentetan dan Tatasusunan Rentetan

- Rentetan adalah jujukan aksara. Jenis data rentetan tidak disediakan oleh bahasa C, oleh itu rentetan diwakili oleh tatasusunan aksara.

- Oleh kerana rentetan adalah tatasusunan aksara maka pengisyiharan rentetan memerlukan saiz rentetan seperti berikut:

```
char nama_rentetan[saiz];
```

7.6 Rentetan dan Tatasusunan Rentetan

- Contoh pengisytiharan dua tatasusunan aksara:

```
char nama1[]=”Muhammad”;
char nama2[]={‘I’, ‘s’, ‘m’, ‘a’, ‘e’, ‘l’, ‘\0’};;
```

- Contoh struktur rentetan :

nama1

M	U	h	a	m	m	a	d	\0

nama2

I	s	m	a	e	l	\0

7.6 Rentetan dan Tatasusunan Rentetan

- Cetakan jujukan elemen tatasusunan biasanya menggunakan gelung, tetapi cetakan rentetan boleh dilakukan tanpa gelung.

- Contoh untuk mencetak kandungan tatasusunan aksara :

```
puts(nama1);
printf("%s\n", nama2);
```

7.6 Rentetan dan Tatasusunan Rentetan

- Tatasusunan rentetan adalah jujukan rentetan yang strukturnya boleh diwakili dengan tatsusunan aksara dua dimensi.

- Contoh tatasusunan rentetan `senaraiNama` yang diwakili oleh tatasusunan dua dimensi :

```
char senaraiNama[][10] = {“Adam”, “Ismail”, “Isa”, “Muhammad”};
```

senaraiNama

[0]	A	d	a	m	\0					
[1]	I	s	m	a	i	l	\0			
[2]	I	s	a	\0						
[3]	M	u	h	a	m	m	a	d	\0	

7.6 Rentetan dan Tatasusunan Rentetan

- Untuk memudahkan operasi-operasi ke atas rentetan, beberapa fungsi disediakan dalam perpustakaan C `string.h` iaitu fungsi `strcpy()`, `strcmp()` dan `strlen()`.

- Fungsi `strcpy()` menyalin elemen-elemen rentetan ke dalam tatasusunan aksara dan `strlen()` mengira bilangan aksara pada satu rentetan.

- Fungsi `strcmp()` pula membandingkan nilai dua rentetan berdasarkan kepada nilai ASCII, ia akan memulangkan integer 0 jika kedua renteran adalah sama dan memulangkan nilai perbezaan jika berbeza.

7.6 Rentetan dan Tatasusunan Rentetan

❖ Contoh aturcara operasi ke atas tatasusunan rentetan :

```
 1:  #include <stdio.h>
 2:  #include <string.h>
 3:
 4:  int main ()
 5:  {
 6:     char senaraiNama[][10] = {"Adam", "Ismail",
 7:                                  "Isa", "Muhammad"};
 8:     int i;
 9:
10:     printf("ANALISA NAMA\n");
11:     for (i=0; i<4; i++) {
12:        printf(" %d\n",strlen(senaraiNama[i]));
13:        puts(senaraiNama[i]);
14:     }
15:
```

7.6 Rentetan dan Tatasusunan Rentetan

❖ Contoh aturcara operasi ke atas tatasusunan rentetan (Sambungan):

```
16:     if (strcmp(senaraiNama[1], senaraiNama[2]) == 0)
17:        printf("\nNama %s & %s ialah sama\n",
18:                 senaraiNama[1], senaraiNama[2]);
19:     else
20:        printf("\nNama %s & %s ialah berbeza\n",
21:                 senaraiNama[1], senaraiNama[2]);
22:
23:     strcpy(senaraiNama[2], "Cahaya");
24:
25:     printf("\nSENARAI BARU\n");
26:     for (i=0; i<4; i++) {
27:         puts(senaraiNama[i]);
28:     }
29:
30:  return 0;
31: }
32:
```

7.6 Rentetan dan Tatasusunan Rentetan

- Output hasil operasi ke atas tatasusunan rentetan :

```
ANALISA NAMA
 4
Adam
 6
Ismail
 3
Isa
 8
Muhammad

 Nama Ismail & Isa ialah berbeza
 SENARAI BARU
Adam
Ismail
Cahaya
Muhammad
```

7.7 Tatasusunan Selari

- Terdapat keperluan untuk menghubungkan dua atau lebih tatasusunan dalam pemprosesan data pada tatasusunan.

- Hubungan ini boleh dilakukan dengan tatasusunan selari di mana dua atau lebih tatasusunan yang mengandungi data yang berkaitan. Jenis data tatasusunan yang selari tidak semestinya jenis yang sama.

- Contoh Tatasusunan selari kelas matematik Tingkatan 4 :

Kelas

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
A	B	C	D	E	F	G	H

bilP

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
23	25	18	29	24	30	24	26

purataM

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
58.5	78.6	58.6	60.4	62.1	80.8	73.7	71.9

7.7 Tatasusunan Selari

- Contoh atur cara menunjukkan contoh penggunaan tatasusunan selari yang digunakan untuk mengira matrik C dan matrik D dari matrik A dan B seperti slide sebelum ini.

- Matrik C adalah hasil tambah matrik A dan B, sementara matrik D menggunakan persamaan $D = 2A + B$.

- Nilai matrik A dan matrik B:

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

7.7 Tatasusunan Selari

- Contoh atur cara :

```
 1: #include<stdio.h>
 2:
 3: int main (void)
 4: {
 5:    int A[] = {4,5,12,7,10,6,4},
 6:        B[] = {3,7,2,2,11,10,1},
 7:        C[7], D[7];
 8:    int i;
 9:
10:    for (i=0; i<7; i++){
11:       C[i] = A[i] + B[i];
12:       D[i] = 2 * A[i] + B[i];
13:    }
14:
15:     printf("MATRIK C");
```

7.7 Tatasusunan Selari

- Contoh atur cara (Sambungan):

```
16:     for (i=0; i<7; i++){
17:     printf("\n|%4d|", C[i]);
18:     }
19:
20:     printf("\n\n MATRIK D");
21:     for (i=0; i<7; i++){
22:        printf("\n|%4d|", D[i]);
23:     }
24:     return 0;
25: }
```

7.7 Tatasusunan Selari

- Operasi untuk membandingkan kandungan dua tatasusunan memerlukan manipulasi subskrip yang sama. Penggunaan operator hubungan == terus dengan pemboleh ubah tatasusunan tidak boleh dilakukan.

- Contoh segmen kod untuk membandingkan dua matrik A dan matrik B:

```
int matrikSama = 1;         // pemboleh ubah bendera
int subskrip = 0;           // mengawal gelung
while (matrikSama && subskrip < 7){
  if (A[subskrip] != B[subskrip])   //banding elemen demi elemen
        matrikSama = 0;
     subskrip++;
}
if (matrikSama) //semak bendera
  printf("Matrik A sama dengan Matrik B.\n");
else
  printf("Matrik A tidak sama dengan Matrik B.\n");
```

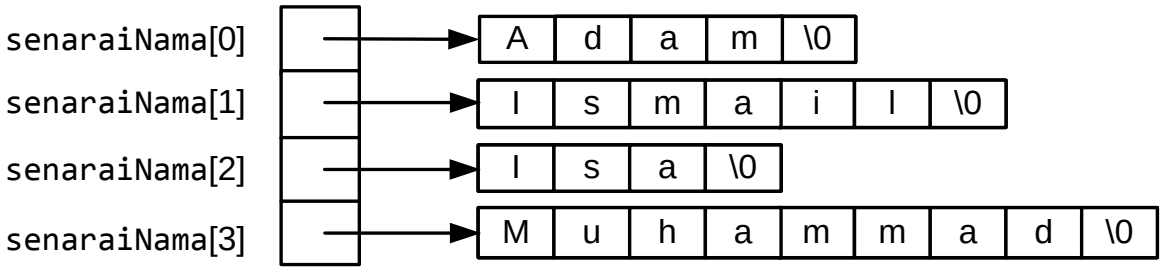
7.8 Tatasusunan Penuding

- Tatasusunan penuding merujuk kepada tatasusunan berbilang dimensi yang diwakili oleh penuding. Sintaks pengisytiharan tatasusunan penuding adalah:

```
jenis_data *nama_tatasusunan [saiz_ dimensi1] [saiz_ dimensi2]. . .
                                                    [saiz_ dimensin];
```

- Contoh:

```
char *senaraiNama[4] = {“Adam”, “Ismail”, “Isa”, “Muhammad”};
```



7.8 Tatasusunan Penuding

- Mencapai elemen tatasusunan penuding masih menggunakan cara yang sama dengan tatasusunan rentetan, gelung berikut boleh digunakan untuk mencetak setiap baris tatasusunan penuding.

- Satu elemen aksara tatasusunan penuding juga boleh dicapai dengan cara yang sama untuk mencapai tatasusuanan rentetan, perhatikan contoh berikut:

```
char *p;
p = senaraiNama[3];
printf("%c ", senaraiNama[1][1]);
printf("%c %c", *p, *(p+4));
```

- Keratan atur cara di atas akan mencetak aksara “s M m”. Cetakan aksara ‘s’ dibuat dengan format tatasusunan aksara dua dimensi dan aksara ‘M’ dan ‘m’ dicapai dengan menggunanakan penuding p.