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.

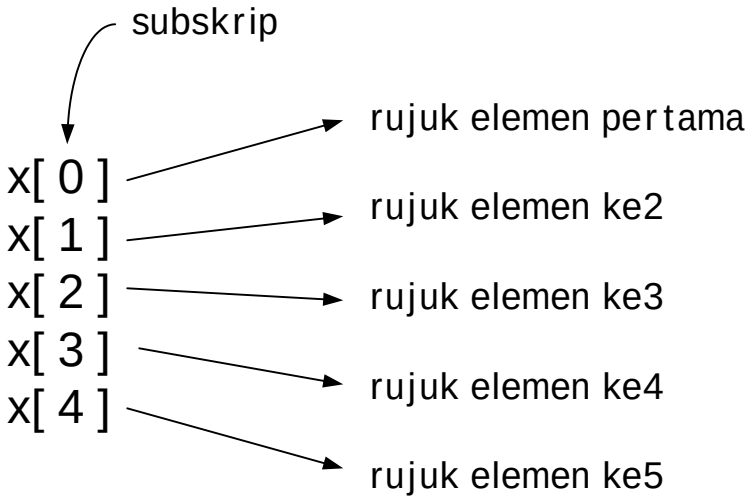
```
int x[] = { 10,  20, 30, 40, 50 };
          atau
int x[5] = { 10,  20, 30, 40, 50 };
```

- 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....)

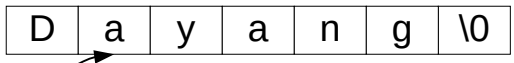
- Perhatikan struktur tts di bawah:

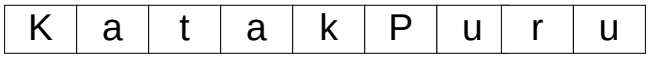
	y[0]	y[1]	y[2]	y[3]	y[4]	y[5]	y[6]	y[7]
y	7.7	5.6	4.1	2.5	1.0	7.7	18.7	9.9

```
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

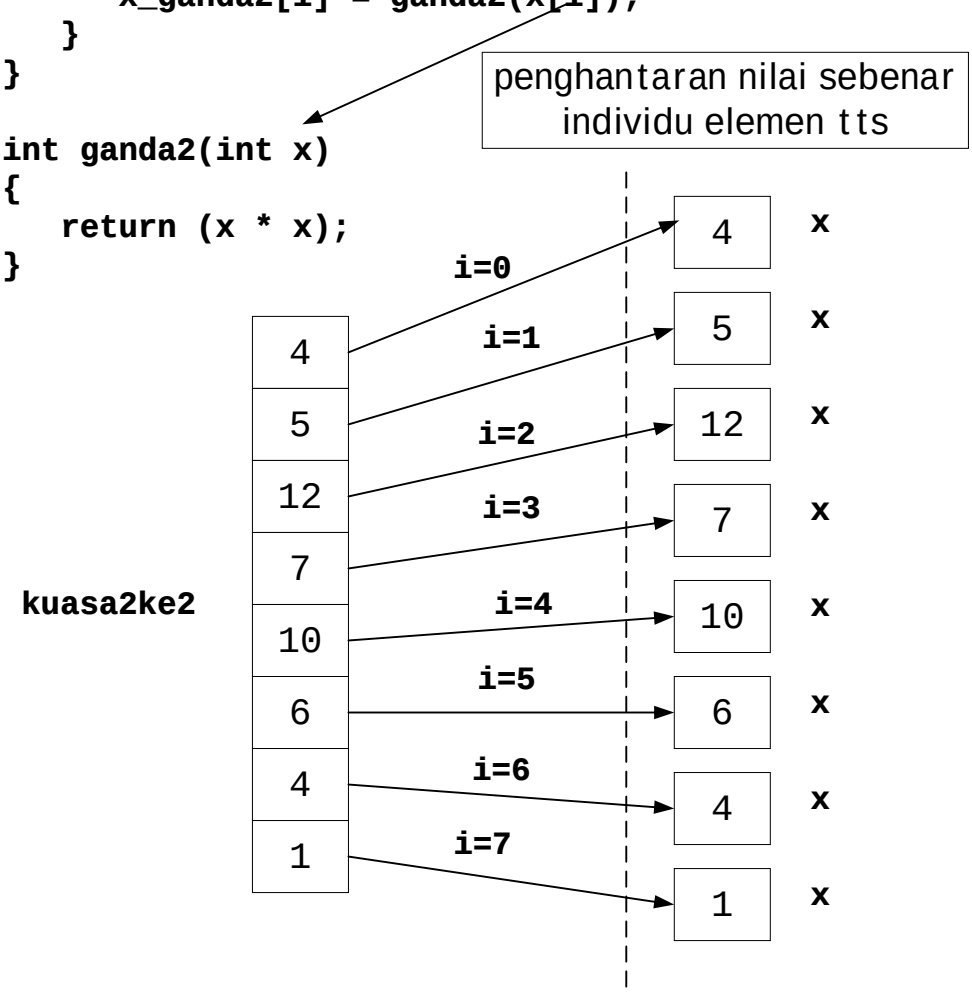
- Tatasusunan boleh dihantar ke fungsi samada:-
  - o individu elemen
  - o kesemua elemen tatasusunan

- 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

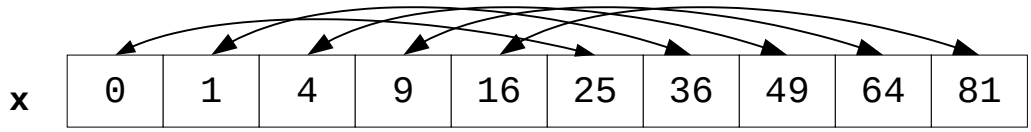
Contoh 1: Ganda 2 nombor dlm tts x

```
#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

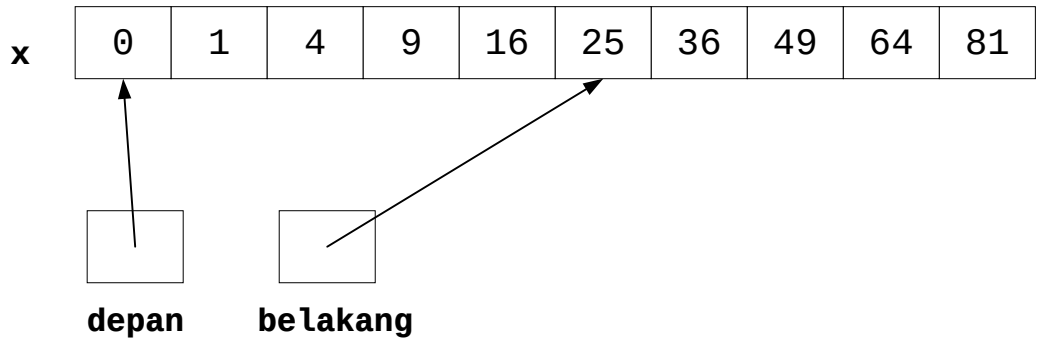
Contoh 2 : 10 nombor dlm tts, menukar 5 elemen hadapan dgn 5 elemen belakang. Operasi spt berikut:




```
#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).

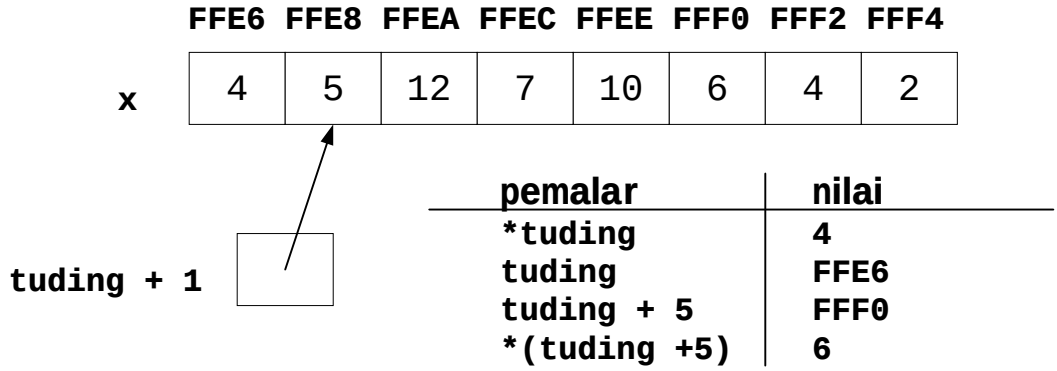
Contoh 5: Aturcara tolak 2 dari setiap elemen tts

```
#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

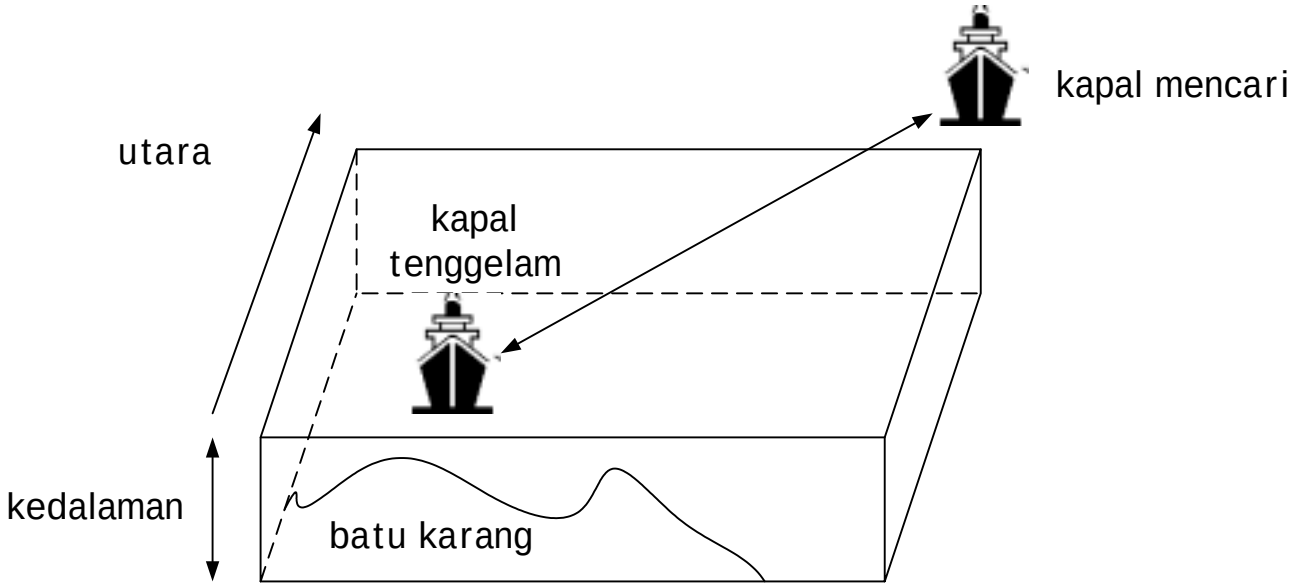
8.4 Tts Multi-Dimensi

- 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



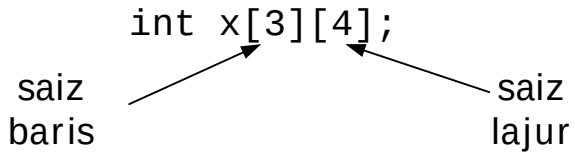
- Dlm contoh 1 tts 2 dimensi diperlukan & dlm contoh 2 tts 3 dimensi diperlukan.

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");
   }
}
```

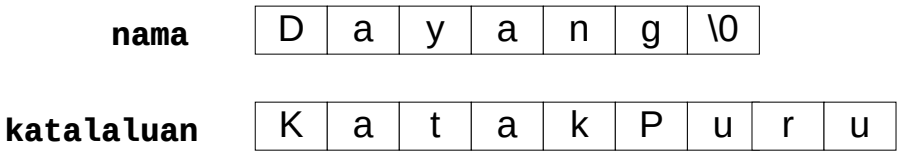
Output:

```
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);
```

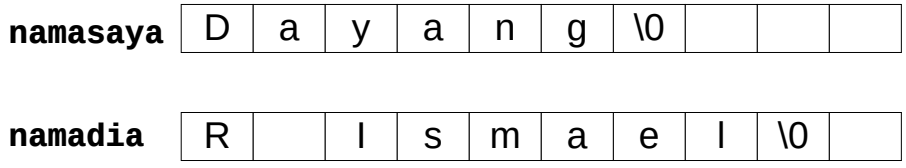
Utk memudahkan operasi[2] ke atas rentetan, beberapa fungsi disediakan dlm perpustakaan C **`string.h`**. Kita akan lihat fungsi:

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");
```

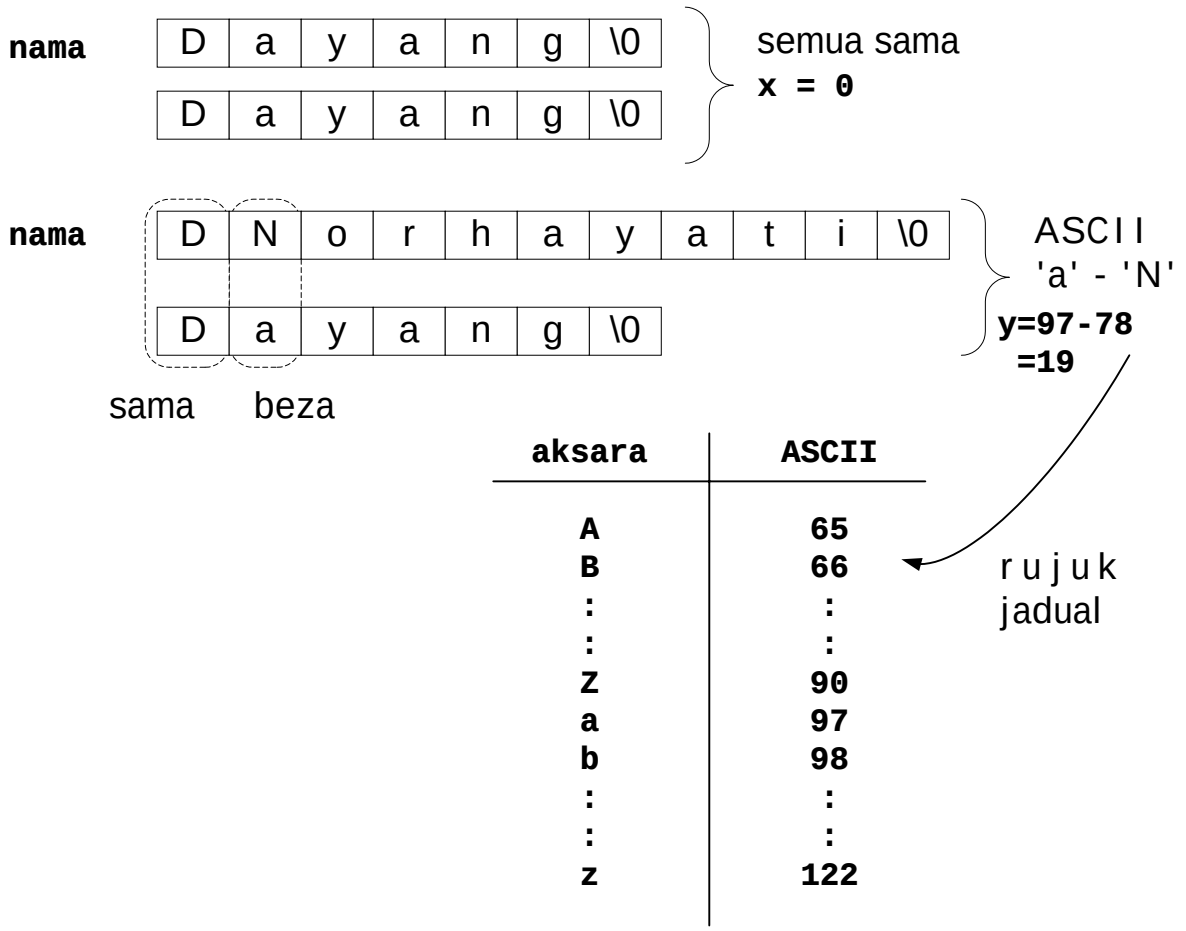


8.6 Tts Rentetan (samb....)

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

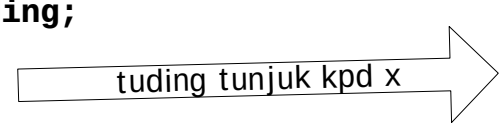
I	s	m	a	e	l	\0	
M	u	n	i	r	a	h	\0
c	a	h	a	y	a	\0	
A	d	i	b	a	h	\0	

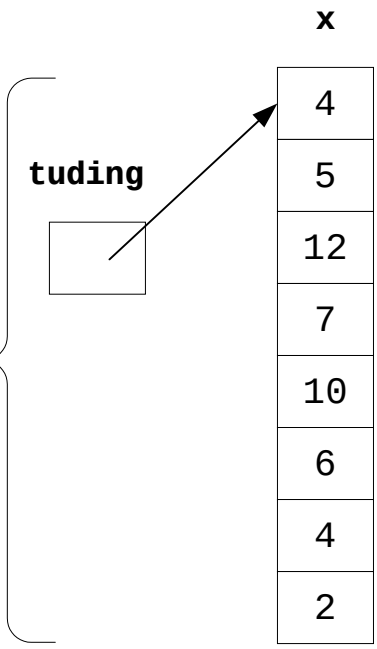
8.6 Tts & Penuding

- 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]);
   }
}
```

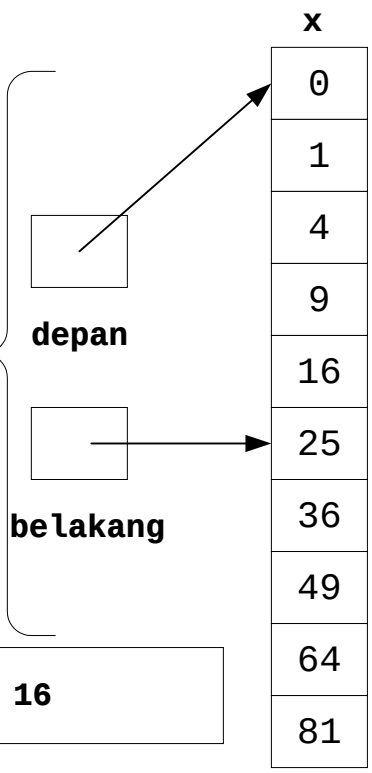




Output:

```
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
```