DAYANG NORHAYATI ABANG JAWAWI
ROSBI MAMAT

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Bab 8: Jenis Data Berstruktur

8.1 Pengenalan

- Jenis data berstruktur membolehkan pengkompil memperuntukkan lebih daripada satu ruang ingatan untuk nilai-nilai data yang berkaitan dengan merujuk kepada satu pemboleh ubah yang sama.

- Struktur dan kesatuan adalah contoh jenis data berstruktur takrifan pengguna yang ditakrif daripada jenis-jenis data asas seperti aksara, integer dan nombor nyata.

- Penomboran pula adalah jenis data lebih mudah berbading struktur dan kesatuan, yang ianya juga ditakrif daripada jenis data asas.

8.2 Rekod dengan Struktur

- Rekod adalah himpunan maklumat berkaitan satu data objek yang disimpan dalam ingatan komputer untuk membentuk pangkalan data.

- Struktur adalah jenis data takrifan pengguna yang mewakilkan himpunan data yang berkaitan untuk membentuk maklumat mengenai sesuatu data, dan tidak semestinya ahli-ahli data tersebut daripada jenis yang sama.

- Contoh struktur rekod pelajar :

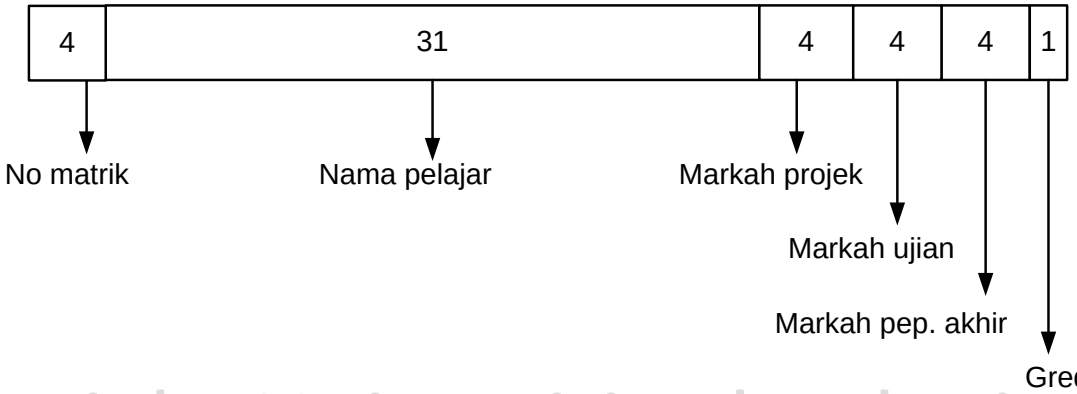
Nama Data	Jenis Data	Jenis data asas C
No matrik	Nilai integer	`int`
Nama pelajar	Tatasusunan aksara 31	`char [31]`
Markah projek	Nilai titik terapung	`float`
Markah ujian	Nilai integer	`float`
Markah peperiksaan akhir	Nilai integer	`Float`
Gred	Aksara	`Char`

8.2.1 Penakrifan dan Pengisytiharan Struktur

- Penakrifan jenis struktur adalah untuk menentukan bentuk ahli pada satu struktur, format takrifan struktur adalah:

```
struct nama_jenis_struktur
{
   jenis_data nama_ahli_1;
   jenis_data nama_ahli_2;
   :
   jenis_data nama_ahli_n;
};
```

- Saiz ingatan yang diperlukan utk struktur ini dalambait :



8.2.1 Penakrifan dan Pengisytiharan Struktur

- Contoh pengisytiharan (a) :

```
struct pelajar
{
int       no_metrik;
char      nama[31];
float     markah_projek;
float     markah_ujian;
float     markah_final;
char      gred;
};
struct pelajar rekod_pelajar;
```

- Contoh pengisytiharan (b) :

```
struct pelajar
{
int       no_metrik;
char      nama[31];
float     markah_projek;
float     markah_ujian;
float     markah_final;
char      gred;
} rekod_pelajar;
```

8.2.1 Penakrifan dan Pengisytiharan Struktur

- Contoh pengisytiharan (c) :

```
struct
{
int       no_metrik;
char      nama[31];
float     markah_projek;
float     markah_ujian;
float     markah_final;
char      gred;
} rekod_pelajar;
```

8.2.1 Penakrifan dan Pengisytiharan Struktur

- Satu lagi pilihan penakrifan dan pengisytiharan struktur adalah dengan menggunakan `typedef`. Contoh:

```
typedef struct
{
int      no_metrik;
char     nama[31];
float    markah_projek;
float    markah_ujian;
float    markah_final;
char     gred;
} JenisDataPelajar;
JenisDataPelajar rekod_pelajar;
```

- Penggunaan typedef lebih memudahkan pengisytiharan pemboleh ubah struktur kerana kata kunci struct boleh diabaikan.

8.2.2 Pengawalan Ahli Struktur

- Nilai awal boleh diberikan kepada ahli struktur, formatnya adalah hampir sama dengan cara pengawalan tatasusunan, rujuk format pengawalan ahli stuktur berikut:

```
struct nama_jenis_struktur nama_struktur = {senarai_data};
```

- Jika menggunakan typedef untuk penakrifan jenis struktur, maka format pengawalan ahli stuktur adalah seperti berikut:

```
jenis_struktur nama_struktur = {senarai_data};
```

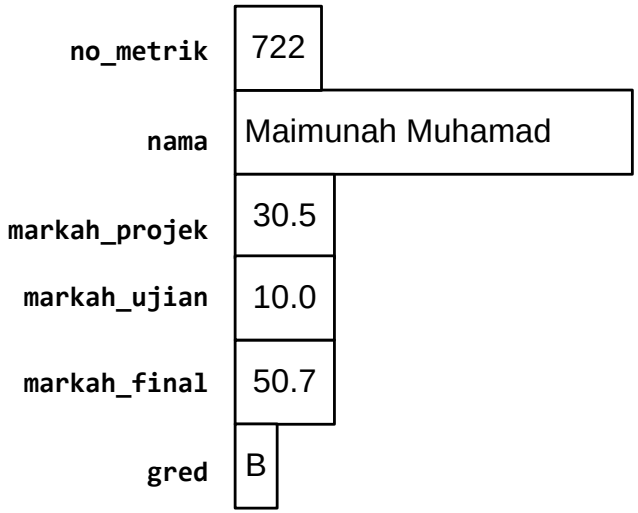
8.2.2 Pengawalan Ahli Struktur

- contoh mengumpukkan nilai awal :

```
struct pelajar rekod_pelajar = {722, "Maimunah Muhamad", 30.5, 10.0,
                                50.7, 'B'};
```

- Ruang ingatan untuk pemboleh ubah `rekod_pelajar` :

```
struct pelajar rekod_pelajar772 = {722, "Maimunah Muhamad" };
```



8.2.2 Pengawalan Ahli Struktur

- Pengawalan boleh dibuat untuk subset ahli struktur tetapi nilai pengawalan mestilah mula dengan nilai pertama sehingga ahli tertentu tanpa mengabaikan mana-mana nilai antara dua nilai.

- Contoh pengawalan :

```
struct pelajar {
int              no_metrik;
char     nama[31];
float    markah_projek;
float    markah_ujian;
float    markah_final;
char             gred;
} rekod_pelajar = {722, “Maimunah Muhamad”, 30.5, 10.0,
                   50.7, ‘B’};
```

8.2.3 Operasi ke atas Ahli Struktur

- Setelah diisytiharkan, ahli struktur boleh dirujuk dengan menggunakan operator pemilihan komponen terus.

- Format sintaks untuk merujuk ahli struktur adalah seperti berikut:

```
nama_struktur.nama_ahli
```

- Contoh mencetak nama pelajar:

```
printf("%s", rekod_pelajar.nama);
```

- contoh mengumpukan nilai kepada ahli `no_metrik`:

```
rekod_pelajar.no_metrik = 5443;
```

8.2.3 Operasi ke atas Ahli Struktur

- Contoh atur cara operasi ke atas ahli struktur

```
 1:  #include <stdio.h>
 2:
 3:  struct nombor{
 4:            int n1;
 5:            double n2;
 6:            float n3;
 7:  };
 8:
 9:  int main() {
10:           struct nombor nom = {4, 6.5, 15.0};
11:           nom.n2 += 5;
12:           nom.n3 = nom.n2 - nom.n1;
13:           nom.n1 = 8;
14:           printf("%0.2f %d %0.2f", nom.n2 ,nom.n1, nom.n3);
15:
16:         return 0;
17:  }
```

Kenyataan di baris 14 akan mencetak nilai-nilai berikut:

```
11.50 8 7.50
```

8.2.3 Operasi ke atas Ahli Struktur

- Jadual untuk menjejak perubahan nilai ahli struktur pemboleh ubah nom selepas setiap operasi :

Kenyataan	n1	n2	n3
Pengawalan	4	6.5	15.0
nom.n2 += 5;	4	11.5	15.0
nom.n3 = nom.n2 - nom.n1;	4	11.5	7.5
nom.n1 = 8;	8	11.5	7.5

8.3 Struktur Bersarang

- Struktur bersarang terhasil jika ahli struktur boleh terdiri daripada struktur yang lain.

- Pertimbangkan penakrifan dan pengisytiharan struktur berikut:

```
struct pelajar
{
        char    nama[31];
        int     no_metrik;
        char    kod_fakulti[5];
        int     umur;
        char    jantina;
        int     markah_projek1;
        int     markah_projek2;
        int     markah_projek3;
        float   markah_ujian1;
        float   markah_ujian2
        float   markah_final;
        char    gred;
} rekod_pelajar;
```

8.3 Struktur Bersarang

- Mengstrukturkan semula rekod pelajar di atas menggunakan struktur bersarang memudahkan rujukan data kerana data boleh diklasifikasi mengikut kategori berikut:

1. maklumat peribadi pelajar
2. markah pelajar

8.3 Struktur Bersarang

- Struktur pelajar di atas boleh ditakrif semula dengan menggunakan struktur bersarang dengan cara berikut:

```
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;
        float markah_ujian1;
        float markah_ujian2
        float markah_final;
}

struct pelajar
{
        struct peribadi_pelajar     peribadi;
        struct markah_pelajar       markah;
        char     gred;
};
```

8.3 Struktur Bersarang

- Cara lain:

```
struct pelajar
{
  struct
   {
         char      nama[31];
         int       no_metrik;
         char      kod_fakulti[5];
         int       umur;
         char      jantina;
    } peribadi;

   struct
   ‘{
         int       markah_projek1;
         int       markah_projek2;
         int       markah_projek3;
         float     markah_ujian1;
         float     markah_ujian2
         float     markah_final;
    } markah;
    char   gred;
};
```

8.4 Tatasusunan Berstruktur

- Tatasusunan berstuktur terbentuk bila pengisytiharan struktur melibatkan tatasusunan yang mana struktur merupakan elemen kepada tatasusunan.

- Pengisytiharan tatasusunan satu dimensi berstruktur menggunakan sintaks berikut:

```
jenis_struktur nama_struktur[saiz_elemen];
```

8.4 Tatasusunan Berstruktur

- Contoh penakrifan :

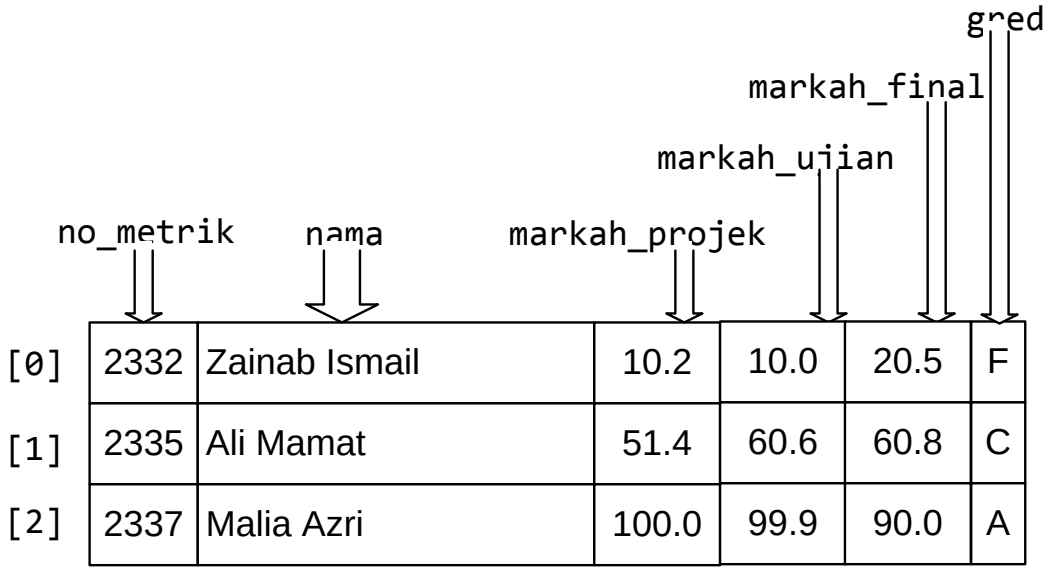
```
struct pelajar
{
        int     no_metrik;
        char    nama[31];
        float markah_projek;
        float   markah_ujian;
        float   markah_final;
        char    gred;
};
struct pelajar rekod_pelajar[3];
```

8.4 Tatasusunan Berstruktur

- Saiz tatasusunan boleh diabaikan jika nilai awal diberi, contohnya :

```
struct pelajar rekod_pelajar[] = {
      {2332, "Zainab Ismail", 10.2, 10.0, 20.5, 'F'},
      {2335, "Ali Mamat", 51.4, 60.6, 60.8, 'C'},
      {2337, "Malia Azri", 100.0, 99.0, 90.0, 'A'}};
```

- Ruang ingatan tatasusunan berstuktur `rekod_pelajar` :



8.4 Tatasusunan Berstruktur

- Contoh atur cara operasi ke atas ahli tatasusunan berstruktur :

```
 1:  #include <stdio.h>
 2:
 3:  typedef struct {
 4:     char fakulti[50];
 5:     char universiti[50];
 6:     char singkatan[10];
 7:     int poskod;
 8:     char bandar[30];
 9:     char negeri[30];
10:  } Alamat;
11:
12:  int main(){
13:           Alamat utmFC = {"Fakulti Komputeran",
14:                   "Universiti Teknologi Malaysia", "UTM", 81310,
15:                  "Skudai", "Johor"};
```

8.4 Tatasusunan Berstruktur

- Contoh atur cara operasi ke atas ahli tatasusunan berstruktur (Sambungan) :

```
16:       Alamat fakultiKomputer[20] = {
17:                   {"Fakulti Teknologi & Sains Maklumat",
18:                   "Universiti Kebangsaan Malaysia", "UKM", 43600,
19:                   "Bangi", "Selangor"},
20:                   {"Pusat Pengajian Sains Komputer",
21:                   "Universiti Sains Malaysia", "USM", 11800,
22:                   "\0", "Pulau Pinang"},
23:                   {"Fakulti Komputer & Sains Matematik",
24:                   "Universiti Teknologi Mara", "UiTM", 40450,
25:                   "Shah Alam", "Selangor"}
26:     };
27:     int j, bil=3;
28:     fakultiKomputer[bil++] = utmFC;
29:     for (j=0; j<bil; j++){
30:                   printf("\n%s\n", fakultiKomputer[j].fakulti);
31:                   printf("%s\n", fakultiKomputer[j].universiti);
```

8.4 Tatasusunan Berstruktur

- Contoh atur cara operasi ke atas ahli tatasusunan berstruktur

  (Sambungan) :

```
32:                     printf("%d ", fakultiKomputer[j].poskod);
33:                     printf("%s \n", fakultiKomputer[j].singkatan);
34:                     printf("%s, ", fakultiKomputer[j].bandar);
35:                     printf("%s\n",fakultiKomputer[j].negeri);
36:             }
37:             return 0;
38: }
```

8.4 Tatasusunan Berstruktur

- Output:

```
Fakulti Teknologi & Sains Maklumat
Universiti Kebangsaan Malaysia
43600 UKM
Bangi, Selangor

Pusat Pengajian Sains Komputer
Universiti Sains Malaysia
11800 USM
, Pulau Pinang

Fakulti Komputer & Sains Matematik
Universiti Teknologi Mara
40450 UiTM
Shah Alam, Selangor

Fakulti Komputeran
Universiti Teknologi Malaysia
81310 UTM Skudai, Johor
```

8.5 Penuding kepada Struktur

- Alamat ahli-ahli pemboleh ubah struktur boleh dicapai seperti alamat pemboleh ubah lain dengan penuding kepada struktur.

- Format pengisyiharan pemboleh ubah penuding kepada struktur :

```
struct nama_jenis_struktur *nama_penuding;
```

- jika penakrifan dengan menggunakan `typedef`.

```
jenis_struktur *nama_penuding;
```

8.5 Penuding kepada Struktur

- Contoh kenyataan pengisytiharan penuding kepada jenis struktur yang ditakrif di Atur cara sebelum ini dan kenyataan umpukan alamat pemboleh ubah utmFC ke penuding tersebut adalah seperti berikut:

```
Alamat *dayang;
dayang = &utmFC;
```

- Setiap ahli struktur boleh dirujuk menggunakan penuding kepada struktur dengan mana-mana dua format sintaks berikut:

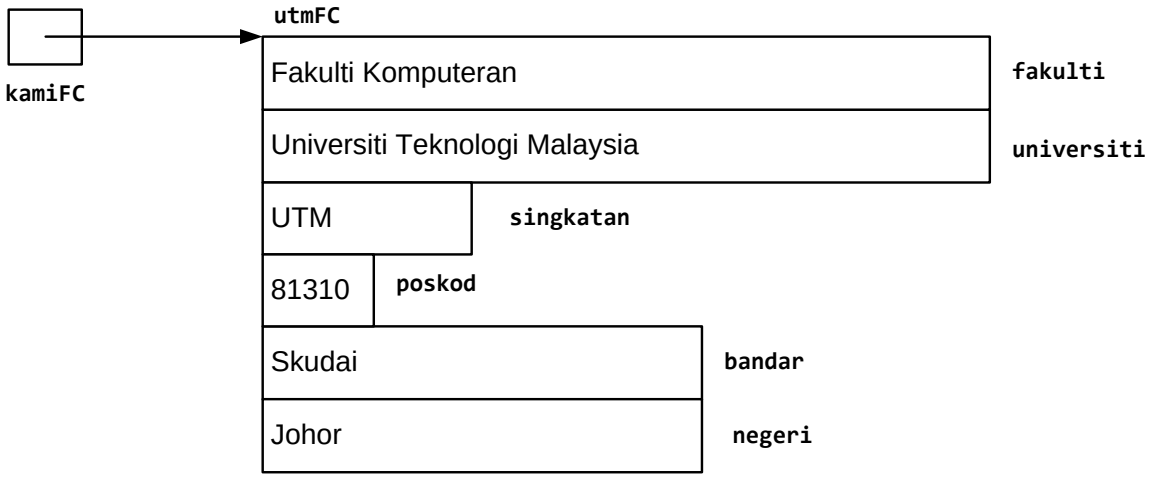
```
(*nama_penuding).nama_ahli
nama_penuding->nama_ahli
```

8.5 Penuding kepada Struktur

- Contoh:

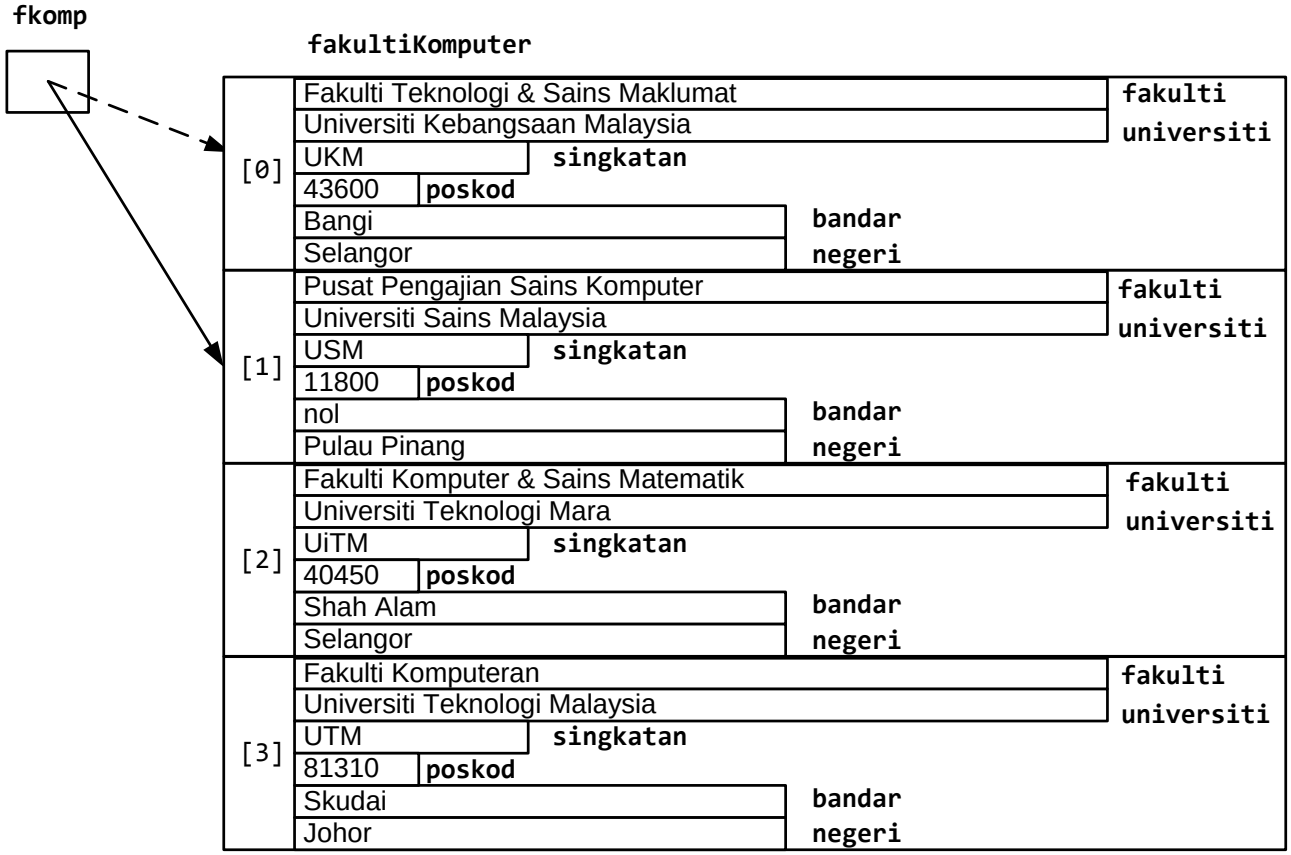
```
Alamat *kamiFC, *fkomp;
kamiFC = &utmFC;
fkomp = fakultiKomputer; fkomp++;
```

- Gambaran ruang ingatan hasil pengisytiharan dan umpukan pemboleh ubah penuding kamiFC



8.5 Penuding kepada Struktur

- Gambaran ruang ingatan hasil pengisytiharan dan umpukan pemboleh ubah penuding `fkomp` :



8.5 Penuding kepada Struktur

- Contoh kenyataan berikut digunakan untuk mencetak nilai beberapa ahli struktur `utmFC` dan tatasusunan berstuktur `fakultiKomputer`.

```
printf("Lokasi %s di %s\n", (*kamiFC).universiti, kamiFC->negeri);
printf("Lokasi %s di %s\n", (*fkomp).universiti, fkomp->negeri);
```

- Output :

```
Lokasi Universiti Teknologi Malaysia di Johor
Lokasi Universiti Sains Malaysia di Pulau Pinang
```

8.5 Penuding kepada Struktur

- Langkah demi langkah maksud `(*kamiFC).universiti`

Rujukan	Jenis	Nilai
kamiFC	Alamat *	Alamat utmFC
*kamiFC	Alamat	Struktur yang juga dirujuk sebagai utmFC
(*kamiFC).universiti	char[]	"Universiti Teknologi Malaysia"

8.6 Struktur dan Fungsi

- Seperti juga pemboleh ubah tatasusunan, ahli-ahli struktur boleh dihulurkan ke fungsi secara individu melalui nilai sebenar atau nilai alamat.

- Terdapat juga situasi di mana fungsi perlu memproses sebahagian besar atau kesemua ahli fungsi, dalam keadaan ini, adalah lebih mudah jika keseluruhan pemboleh ubah struktur dihulurkan ke fungsi.

- Pemboleh ubah struktur setempat membolehkan pemprosesan ahli-ahli struktur dibuat pada fungsi dan pemboleh ubah struktur tersebut boleh dikembalikan sebagai hasil output fungsi.

8.6.1 Penghuluran Struktur ke Fungsi

- Jika terdapat struktur dengan penakrifan struktur dan pemboleh ubah berikut di dalam satu atur cara;

```
struct rekod_pelajar {
          int no_id;
          float tugasan;
          float kuiz;
          float ujian;
          char gred;
};
```

- dan fungsi berikut juga terdapat di dalam atur cara yang sama.

```
void cetak(char gred, int id)
{
          printf("Gred pelajar %d ialah %c\n", id,
gred);
}
```

- Maka, ahli pemboleh ubah struktur `gred` dan `id` boleh dihulurkan ke fungsi `cetak()`.

8.6.1 Penghuluran Struktur ke Fungsi

- Panggilan fungsi `cetak()` dengan dua argumen yang merupakan ahli kepada pemboleh ubah struktur `rekod_pelajar` iaitu `no_id` dan `gred` adalah seperti berikut:

```
cetak(amir.gred, amir.no_id);
```

8.6.1 Penghuluran Struktur ke Fungsi

- Penghuluran alamat nilai ahli struktur perlu digunakan untuk membolehkan perubahan nilai ahli-ahli di dalam fungsi akan mengubah nilai ahli-ahli struktur di luar fungsi. Perhatikan contoh:

```
void tentu_gred(float jum_mar, char *gred)
{
        if (jum_mar >= 60)
                *gred = 'L';
        else
                *gred = 'G';
}
```

- Panggilan fungsi `tentu_gred()` :

```
tentu_gred(jum_mar, &amir.gred);
```

8.6.1 Penghuluran Struktur ke Fungsi

- Penghuluran keseluruhan pemboleh ubah struktur fungsi melibatkan penghuluran nilai sebenar dan alamat nilai pemboleh ubah struktur. Contoh :

```
float kira_jumlah(struct rekod_pelajar p)
{
        float jum;
        jum = p.tugasan + p.kuiz + p.ujian;
        return jum;
}
```

- Dengan pemboleh ubah amir, fungsi ini boleh dipanggil dengan menggunakan kenyataan berikut:

```
jum_mar=kira_jumlah(amir);
```

8.6.2 Mengembali Struktur dari Fungsi

- Satu fungsi boleh kembalikan pemboleh ubah berjenis struktur. Contoh :

```
struct rekod_pelajar baca_data()
{
   FILE *data;
   struct rekod_pelajar p;

   if ((data = fopen("markah.dat", "r"))== NULL)
   {
          puts("Ralat dalam pembukaan fail.");
          exit(-1); /*Tamatkan Atur cara*/
   }
           fscanf(data, "%d", &p.no_id);
           fscanf(data, "%f", &p.tugasan);
           fscanf(data, "%f", &p.kuiz);
           fscanf(data, "%f", &p.ujian);
           close(p);
           return p;
}
```

- contoh kenyataan panggilan untuk unpukkan nilai ke pemboleh ubah struktur lain:

```
amir = baca_data();
```

8.7 Kesatuan

- 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

8.7 Kesatuan

- Contoh atur cara :

```
 1: #include <stdio.h>
 2: union nombor {
 3:          char x;
 4:          int y;
 5: };
 6:
 7:  int main( ){
 8:         union nombor nilai;
 9:
10:          printf("saiz union ialah %d\n\n", sizeof(nilai));
11:          nilai.x = 0x41; /*ASCII 'A'*/
12:          printf("char:  %X \nint:   %X\n", nilai.x, nilai.y);
13:          printf("alamat bermula pada  %d\n\n", &nilai.x);
14:          nilai.y = 0x1234;
15:          printf("char:  %X \nint:   %X\n", nilai.x, nilai.y);
16:          printf("alamat bermula pada  %d\n", &nilai.y);
17:
18:          return 0;
19: }
```

8.7 Kesatuan

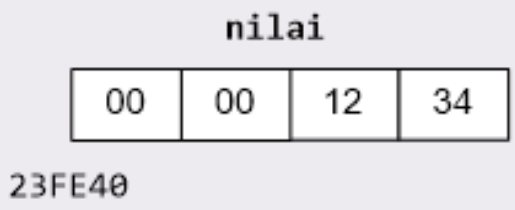
- Output:

```
saiz union ialah 4

char:  41
int:   41
alamat bermula pada  23FE40

char:  34
int:   1234
alamat bermula pada  23FE40
```

- Gambaran perubahan ruang ingatan setelah umpukan ahli pemboleh ubah kesatuan :

8.7 Kesatuan

- Gambaran perubahan ruang ingatan setelah umpukan ahli pemboleh ubah kesatuan :

Kenyataan	Ruang ingatan
Umpukan nilai.x	
`nilai.x = 0x41;`	
Umpukan nilai.y	
`nilai.y = 0x1234;`	

8.8 Penomboran

- Jenis data penomboran adalah satu set nilai pemalar integer yang diwakili oleh pengenal pasti.

- Pengisytiharan :

```
enum nama_jenis_enum
{
        pengenal_pasti_1, pengenal_pasti_2,. .. pengenal_pasti_n
};
enum nama_jenis_penomboran pemboleh_ubah_penomboran;
```

8.8 Penomboran

- Perhatikan contoh penakrifan dan pengisytiharan penomboran berikut:

```
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.

8.8 Penomboran

- Pengaturcara boleh membuat pengubahsuaian berdasarkan peraturan-peraturan berikut:
  1. boleh mengumpukkan nilai pemalar (constant) kepada senarai pernomboran
  2. boleh mengumpukkan nilai kepada senarai pernomboran dengan menggunakan pencam yang sebelumnya dalam ungkapan aritmetik
  3. boleh mengumpukkan nilai yang sama kepada lebih dari satu pencam dalam senarai pernomboran

8.8 Penomboran

- Contoh operasi ke atas pemboleh ubah penomboran :

```
 1:  #include <stdio.h>
 2:
 3:  enum duit
 4:  {
 5:        sen5 = 5, sen10 = 10, sen20 = 20, sen50 = sen5*sen10, rm1
 6:        = 2*sen50, rm5 = 5*rm1
 7:  } duit_masuk;
 8:
 9:  int main( ){
10:     int jumlah = 0;
11:     printf("**Mesin Air Minuman**\n");
12:     while (jumlah<160) {
13:           printf("Masukkan duit dalam nilai sen: ");
14:           scanf("%d", &duit_masuk);
15:           switch(duit_masuk) {
16:              case sen5:
17:                    jumlah+=sen5; break;
18:              case sen10:
19:                    jumlah+=sen10; break;
```

8.8 Penomboran

- Contoh operasi ke atas pemboleh ubah penomboran (Sambungan):

```
20:                 case sen20:
21:                       jumlah+=sen20; break;
22:                 case sen50:
23:                       jumlah+=sen50; break;
24:                 case rm1:
25:                       jumlah+=rm1; break;
26:                case rm5:
27:                       jumlah+=rm5; break;
28:             default:
29:                       printf("Duit yang diterima hanya Syiling: ”);
30:                       printf("5 10 20 50 dan Wang kertas RM1 RM5\n");
31:             }
32:             printf("Jumlah Duit masuk: %d\n", jumlah);
33:       }
34:       if (jumlah>0)
35:             printf("Duit baki %dsen\n", jumlah-160);
36:       printf("**Terima Kasih**");
37:       return 0;
38:   }
```

8.8 Penomboran

- Hasil perlaksanaan Atur cara 8.5 dengan input satu bilangan duit kertas RM1 dan dua bilangan syiling 50 sen :

```
**Mesin Air Minuman**
Masukkan duit dalam nilai sen: 100
Jumlah Duit masuk: 100
Masukkan duit dalam nilai sen: 50
Jumlah Duit masuk: 150
Masukkan duit dalam nilai sen: 50
Jumlah Duit masuk: 200
Duit baki 40sen
**Terima Kasih**
```