



PERTEMUAN IX

DATA TINGKAT LANJUT



SASARAN

Setelah menyelesaikan bab ini,
anda diharapkan dapat:

- Menjelaskan tentang tipe data union
- Menjelaskan penggunaan bitfield
- Menjelaskan tentang tipe data enumerasi
- Menjelaskan penggunaan typedef
- Menjelaskan penggunaan ternary operator
- Menjelaskan tentang konversi tipe data (*type casting*)

DASAR UNION

- Memungkinkan suatu lokasi memori ditempati oleh dua atau lebih variabel yang bisa saja tipenya berlainan.
- Contoh definisi union yang menyatakan data bertipe int & karakter.

```
union bil_bulat {  
    unsigned int di;  
    unsigned char dc[2];  
};
```

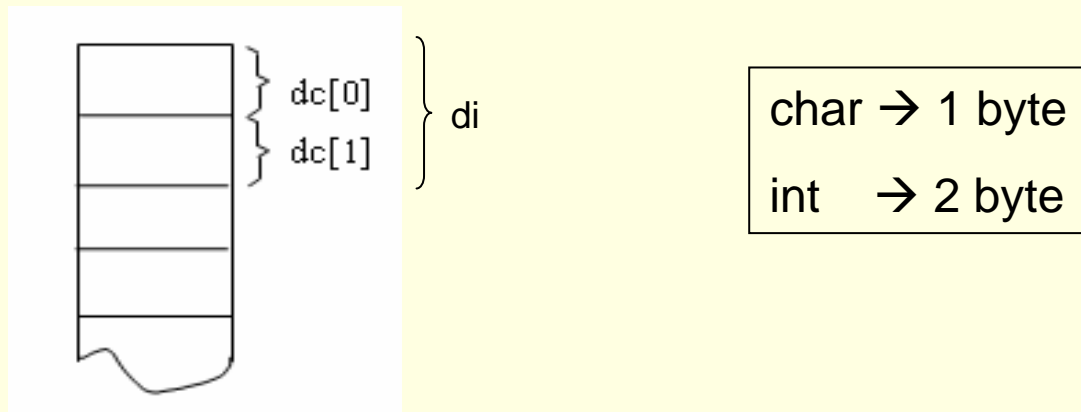
- Cara pendeklarasian union :

```
union bil_bulat bil_x;
```

Catatan : Cara untuk definisi & deklarasi union sama dengan struktur.

Dasar Struktur – Cont. 1

- Dari definisi sebelumnya, terlihat variabel `bil_x` dengan elemen `di` & `dc` mempunyai alamat memori yang sama.



- Cara akses elemen dari sebuah union :

```
variabel_union.nama_element
```

Contoh : `bil_x.di = 321;`

Artinya : Mengisikan nilai 321 ke elemen union **di**. Biner 321 adalah 101000001. Sehingga **dc[0]** akan bernilai byte ke-0 dari **di**, dan **dc[1]** bernilai byte ke-1 dari **di**.

CONTOH PROGRAM UNION

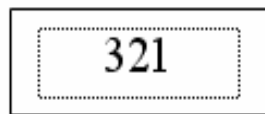
```
#include <stdio.h>
main()
{
    union
    {
        unsigned int di;
        unsigned char dc[2];
    } bil_x;    /* variabel union */

    bil_x.di = 321;
    printf("di = %d\n", bil_x.di);
    printf("dc[0] = %d dc[1] = %d\n", bil_x.dc[0],
    bil_x.dc[1]);
}
```

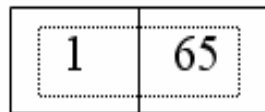
di = 321	
dc[0] = 65	dc[1] = 1

ANALISA PROGRAM

- Program untuk menjelaskan cara mengakses byte ke-0 atau byte ke-1 dari **di**, **dc[0]** atau **dc[1]** yang digunakan.



Nilai 321 dalam kesatuan *unsigned int*



Nilai 321 jika dinyatakan dalam dua buah *unsigned char*

Byte rendah (byte ke-0)

Byte rendah (byte ke-0)

CONTOH PROGRAM

-- Union dalam Fungsi --

```
#include <stdio.h>
union bil_bulat{          /* definisi tipe union */
    unsigned int di;
    unsigned char dc[2];
};
void beri_nilai(union bil_bulat *x); /*prototype fungsi */
main()
{
    union bil_bulat bil_x;      /* deklarasi var union */
    beri_nilai(&bil_x);        /* melewati alamat union */
    printf("di = %d\n", bil_x.di);
    printf("dc[0] = %d dc[1] = %d \n", bil_x.dc[0],
    bil_x.dc[1]);
}
void beri_nilai(union bil_bulat *x)
{
    x -> di = 321;  /*(*x).di      /* elemen di yang ditunjuk */
}
/* oleh x diberi nilai 321 */
```

di = 321
dc[0] = 65 dc[1] = 1

DASAR BITFIELD

- Suatu bit atau beberapa bit dalam sebuah data berukuran satu byte atau dua byte dapat diakses dengan mudah melalui *bitfield*.
- Suatu bit atau beberapa bit dapat diakses tanpa melibatkan operator manipulasi bit (seperti &, |).
- Satu atau dua byte memori dapat dipakai untuk menyimpan sejumlah informasi.

PENDEFINISIAN BITFIELD

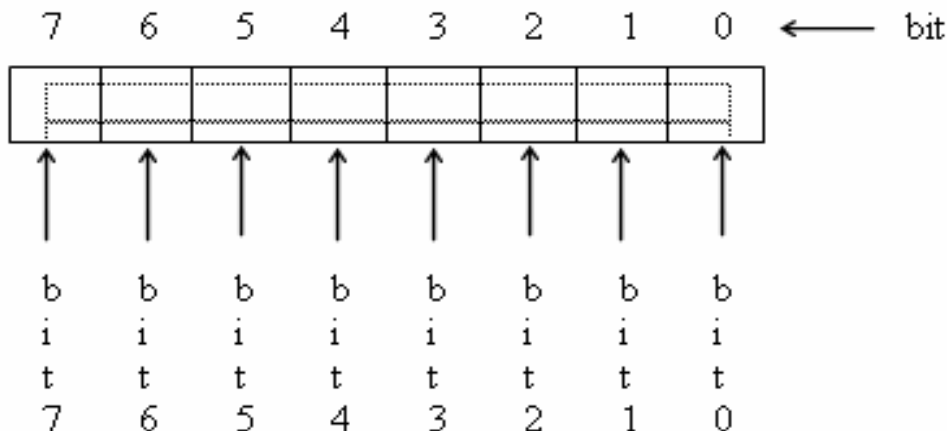
↙ nama tipe struktur yang terdiri atas sejumlah *bitfield*

```
struct info_byte
{ unsigned bit0:1;
  unsigned bit1:1;
  unsigned bit2:1;
  unsigned bit3:1;
  unsigned bit4:1;
  unsigned bit5:1;
  unsigned bit6:1;
  unsigned bit7:1;
};
```

↑
↑
panjang/jumlah bit
nama variabel *bitfield*

Definisi Bitfield untuk memperoleh informasi masing-masing bit dari suatu data satu byte.

unsigned → panjang bit = 1 (0 & 1)



Susunan bit dari memori sebuah data bertipe info_byte

CONTOH PROGRAM BITFIELD

```
#include <stdio.h>
main()
{
    struct info_byte /* definisi tipe bitfield */
    {
        unsigned bit0:1; /* bit ke-0 */
        unsigned bit1:1; /* bit ke-1 */
        unsigned bit2:1; /* bit ke-2 */
        unsigned bit3:1; /* bit ke-3 */
        unsigned bit4:1; /* bit ke-4 */
        unsigned bit5:1; /* bit ke-5 */
        unsigned bit6:1; /* bit ke-6 */
        unsigned bit7:1; /* bit ke-7 */
    };
    /* deklarasi variabel union dan elemen bitfield */
    union
    {
        unsigned char karakter;
        struct info_byte byte;
    } ascii;

    printf("Masukkan sebuah karakter : ");
    scanf("%c", &ascii.karakter);
    printf("\nKode ASCII dari karakter %c adalah %d\n", ascii.karakter, ascii.karakter);
    printf("Bentuk biner dari nilai %d adalah ", ascii.karakter);
    printf("%d%d%d%d%d%d%d%d\n", ascii.byte.bit7, ascii.byte.bit6, ascii.byte.bit5,
    ascii.byte.bit4, ascii.byte.bit3, ascii.byte.bit2, ascii.byte.bit1, ascii.byte.bit0);
}
```

Masukkan sebuah karakter : A
Kode ASCII karakter A adalah 65
Bentuk biner dari nilai 65 adalah 01000001

ANALISA PROGRAM

- Jika dimasukkan karakter 'A', maka nilai `ascii.karakter = 'A'`, sehingga elemen-elemen byte akan bernilai sama dengan karakter, karena pada alamat memori yang sama.

`ascii.byte = 'A';` → tidak diijinkan

- Cara mengakses nilai :

```
printf ("%d", ascii.byte.bit7);
```

untuk mengambil nilai dari *bitfield* **bit 7**.

```
ascii.byte.bit7 = 0;
```

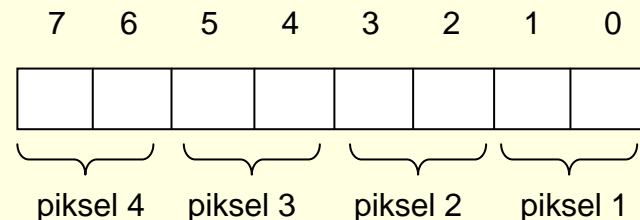
untuk mengubah **bit7** agar bernilai 0

BITFIELD PANJANG 2 BIT

Cara dalam pendefinisian dan pendeklarasian bitfield 2 bit

```
struct data_gambar
{
    unsigned piksel1:2;
    unsigned piksel2:2;
    unsigned piksel3:2;
    unsigned piksel4:2;
} koord;
```

Variabel koord yang bertipe data_gambar akan menempati memori 1 byte (8 bit).



Masing-masing piksel punya range nilai dari 0 – 3

CONTOH PROGRAM

```
#include <stdio.h>
main()
{
    struct data_gambar
    {
        unsigned piksel1:2;
        unsigned piksel2:2;
        unsigned piksel3:2;
        unsigned piksel4:2;
    } koord;
    koord.piksel1=3;
    printf ("Nilainya = %d",koord.piksel1);
}
```

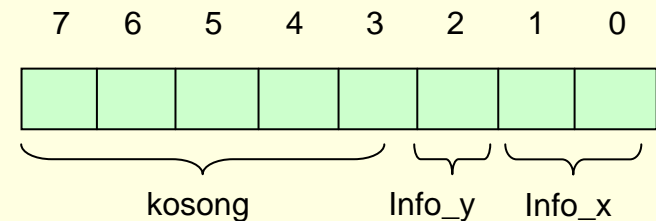
Nilainya = 3

Bagaimana bila koord.piksel dimasukkan nilai 5 ???

APLIKASI BITFIELD

- Dipakai untuk menghemat memori.
- Misal ada 2 informasi sbb :
 1. Info pertama (**info_x**) memiliki nilai 0 - 3
 2. Info kedua (**info_y**) memiliki nilai 0 - 1Hanya akan memakan memori 1 byte saja.
- Cara deklarasi :

```
struct info
{
    unsigned info_x:2;
    unsigned info_y:1;
} status;
```



CONTOH PROGRAM LENGKAP

```
#include <stdio.h>
main()
{
    /* definisi tipe bitfield */
    struct info
    {
        unsigned info_x:2;
        unsigned info_y:1;
        unsigned kosong:5; /* bisa dihilangkan */
    } status;

    status.info_x = 3;
    status.info_y = 1;
    printf("info_x = %d\n", status.info_x);
    printf("info_y = %d\n", status.info_y);
}
```

info_x = 3
info_y = 1