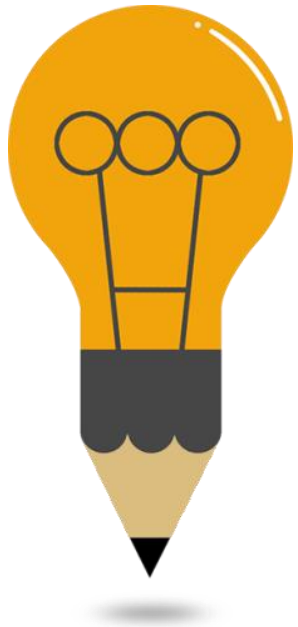
STRUKTUR DATA

Pertemuan 2

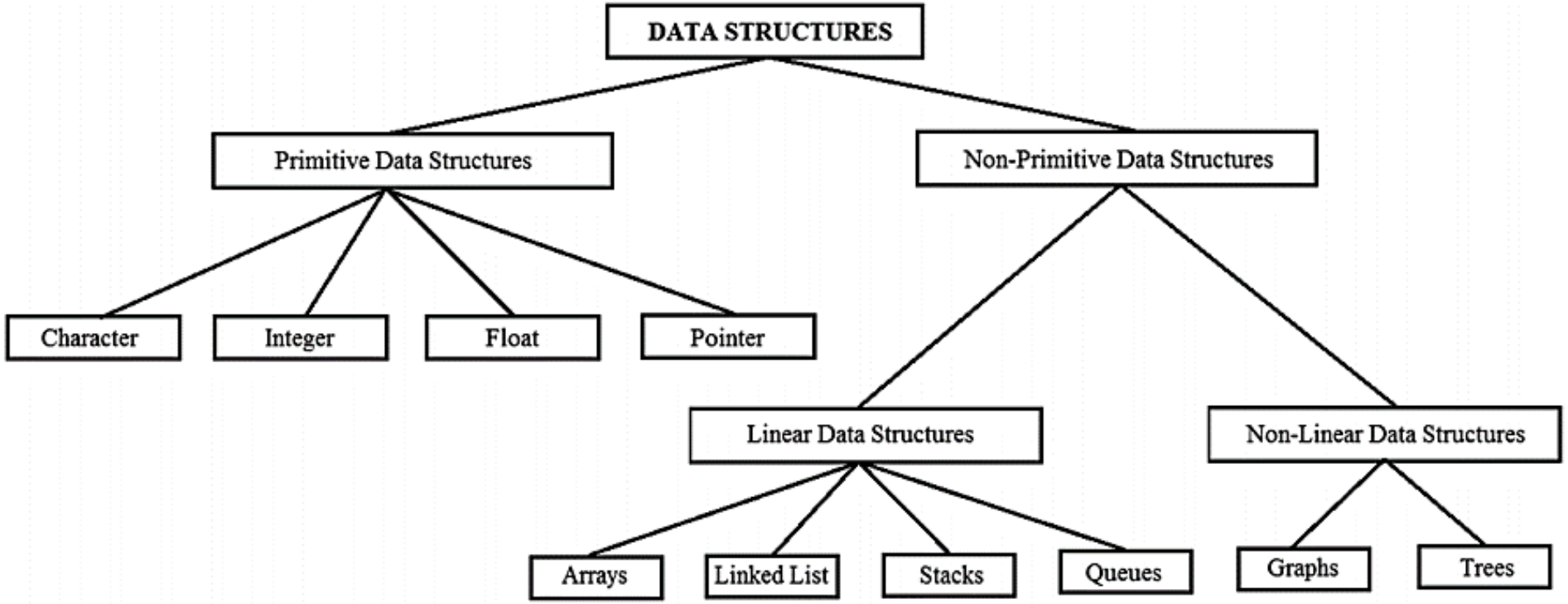
Ratih Ngestrini, Nori Wilantika, Firdaus

Agenda Pertemuan



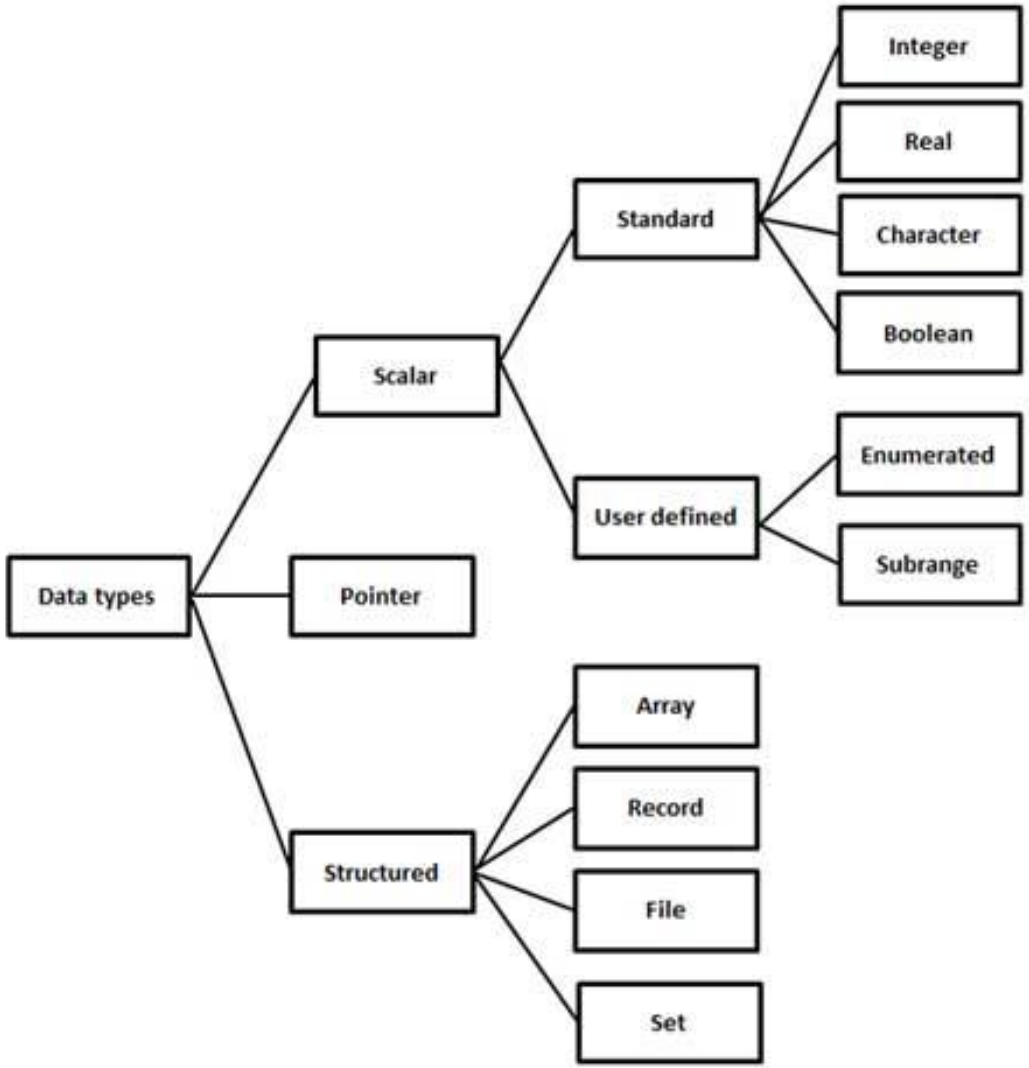
1. Tipe Data dalam Bahasa C
2. Tipe Data Bentukan (ADT)
3. Typedef dan Structure

Jenis-Jenis Struktur Data

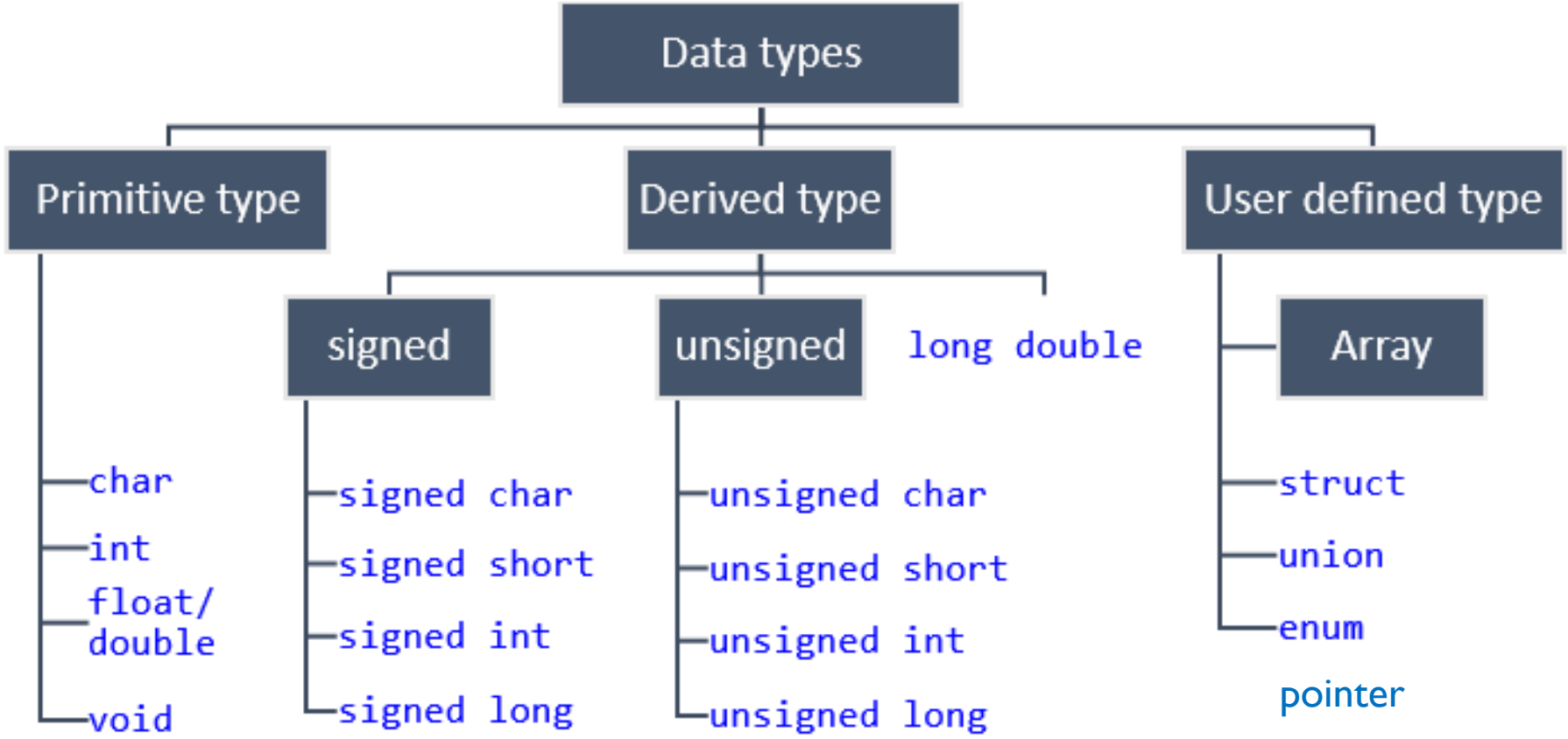


TIPE DATA DALAM BAHASA C

Tipe Data dalam Pascal



Tipe Data dalam Bahasa C



1. Tipe Data Dasar (*Primitive Data Type*)

- Bulit-in dari bahasa pemrograman. Tidak berorientasi pada persoalan yang dihadapi.
- hanya mampu menyimpan satu nilai pada setiap satu variabel.
- mempunyai besaran memori yang tetap dan pasti

- **Pascal**:

Tipe Data Pascal	Karakteristik	Contoh
`String`	Teks	‘New York’, ‘My Name’
`Integer`	Bilangan bulat	23, 16595, 0, -632
`Real`	Bilangan desimal	3.14, 503.2
`Boolean`	True atau False	TRUE, FALSE
`Character`	Satu karakter	A, e, u, 9, o, 3

Contoh jangkauan (range) variabel:

Tipe Data	Minimum	Maximum
`Integer`	-32,768	32,767
`LongInt`	-2,147,483,648	2,147,487,647
`ShortInt`	-128	127
`Real`	2.9 x 10e-39	1.7 x 10e+38

1. Tipe Data Dasar (*Primitive Data Type*)

- **Bahasa C**:

Tipe Data C	Range	Kode Format
`char`	-128 to 127 atau 0 to 255	**%c**
`int`	-32,768 to 32,767 atau -2,147,483,648 to 2,147,483,647	**%d**
`float`	3.4e-038 to 3.4e+038 (6 angka di belakang koma)	**%f**
`double`	2.3e-308 to 1.7e+308 (15 angka di belakang koma)	**%f**
`void`	(tidak bertipe, tidak menyimpan apapun, biasanya untuk tipe fungsi yang tidak return value apapun)	

*Ukuran dan range dari setiap tipe data tergantung mesin (misal mesin 32bit bisa memberikan hasil yang berbeda dari 64bit) dan variasi compiler

2. Tipe data turunan (*Derived Data Type*)

- Kombinasi ***qualifiers*** dan **tipe data primitive**

```
[sign-qualifier] [size-qualifier] <basic-data-type>
```

  - **[…]** : optional (boleh ada atau tidak)
  - Contoh: **`unsigned short int`**
  - **Size qualifier** digunakan untuk mengubah ukuran : **`short`** atau **`long`**
  - **Sign qualifier** digunakan untuk menentukan apakah variabel tersebut dapat menampung nilai negative atau tidak: **`signed`** **(bisa +/-)** atau **`unsigned`** **(hanya +)**

2. Tipe data turunan (*Derived Data Type*)

- Contoh:

Tipe Data	Ukuran (byte)	Range
`int` (atau `signed int`)	2 atau 4*	-32,768 to 32,767 atau -2,147,483,648 to 2,147,483,647
`unsigned int`	2 atau 4*	0 to 65,535 atau 0 to 4,294,967,295
`long int` (biasa ditulis `long` saja)	8	-2,147,483,648 to 2,147,483,647
`double`	8	2.3e-308 to 1.7e+308
`long double`	10	3.4e-4932 to 1.1e+4932

**tergantung prosesor (32bit / 64bit)*

- Ukuran dan range tergantung mesin dan compiler

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <float.h>

int main(int argc, char** argv) {

    printf("CHAR_BIT    :   %d\n", CHAR_BIT);
    printf("CHAR_MAX    :   %d\n", CHAR_MAX);
    printf("CHAR_MIN    :   %d\n", CHAR_MIN);
    printf("INT_MAX     :   %d\n", INT_MAX);
    printf("INT_MIN     :   %d\n", INT_MIN);
    printf("LONG_MAX    :   %ld\n", (long) LONG_MAX);
    printf("LONG_MIN    :   %ld\n", (long) LONG_MIN);
    printf("SCHAR_MAX   :   %d\n", SCHAR_MAX);
    printf("SCHAR_MIN   :   %d\n", SCHAR_MIN);
    printf("SHRT_MAX    :   %d\n", SHRT_MAX);
    printf("SHRT_MIN    :   %d\n", SHRT_MIN);
    printf("UCHAR_MAX   :   %d\n", UCHAR_MAX);
    printf("UINT_MAX    :   %u\n", (unsigned int) UINT_MAX);
    printf("ULONG_MAX   :   %lu\n", (unsigned long) ULONG_MAX);
    printf("USHRT_MAX   :   %d\n", (unsigned short) USHRT_MAX);

    return 0;
}
```

```
Output:

CHAR_BIT    :   8
CHAR_MAX    :   127
CHAR_MIN    :   -128
INT_MAX     :   2147483647
INT_MIN     :   -2147483648
LONG_MAX    :   9223372036854775807
LONG_MIN    :   -9223372036854775808
SCHAR_MAX   :   127
SCHAR_MIN   :   -128
SHRT_MAX    :   32767
SHRT_MIN    :   -32768
UCHAR_MAX   :   255
UINT_MAX    :   4294967295
ULONG_MAX   :   18446744073709551615
USHRT_MAX   :   65535
```

3. Tipe data yang didefinisikan user *(User-defined Data Type / UDT)*

Tipe data yang didefinisikan oleh pemrogram (*programmer*). Mendekati penyelesaian persoalan yang dihadapi.

- **Array**: Tipe data yang terdiri dari kumpulan tipe data dasar. Tipe data tersebut harus 1 jenis.
- **Structure**: Tipe data yang terdiri dari kumpulan tipe data dasar. Tipe data tersebut bisa lebih dari 1 jenis.
- **Pointer**: Tipe data untuk mengakses alamat memory suatu variabel secara langsung.
- **Union**: Tipe data yang menyimpan beberapa tipe data berbeda dalam lokasi memory yang sama.

TIPE DATA BENTUKAN - *ABSTRACT DATA TYPE (ADT)*

- Perluasan dari konsep UDT
- Bukan hanya mendefinisikan tipe data, tetapi juga menyediakan operasi-operasi dalam tipe data tersebut.

- Bagaiamana jika kita ingin membuat tipe data baru?
- Untuk pembuatan tipe data baru digunakan keyword **`typedef`**
- Selain itu kita membutuhkan **`structure`**

TYPEDEF

Apabila kita ingin membuat sebuah tipe data baru dari tipe data yang sudah ada atau tipe data lama, gunakan keyword **`typedef:`**

```
typedef <tipe_data_lama> <nama_tipe_data_baru>
```

Contoh:

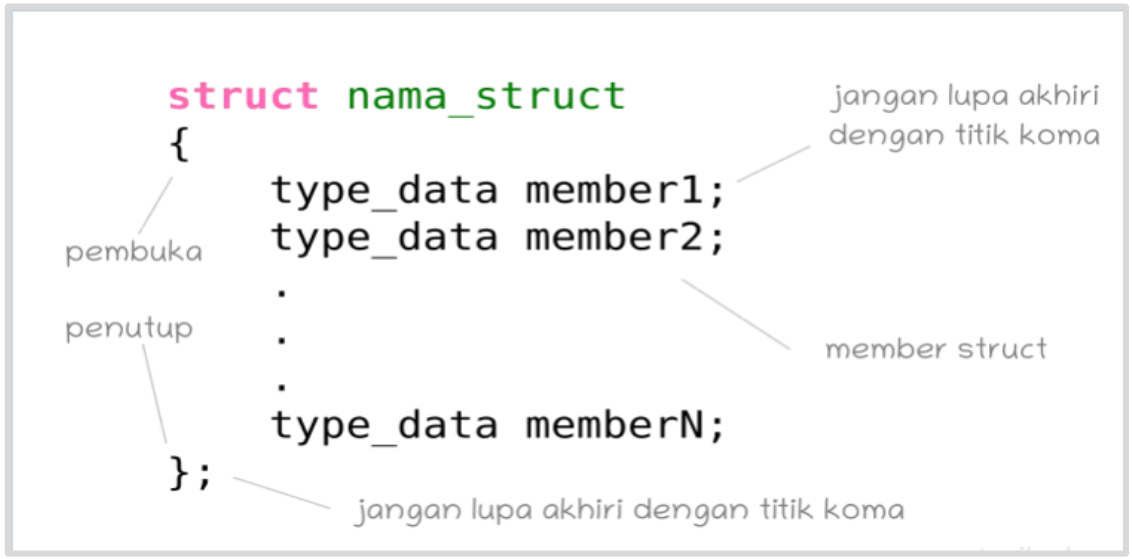
```c
#include <stdio.h>
typedef int bulat;

bulat main(){
    bulat a=10;
    printf("%d",a);
    return 0;
}
```

```c
typedef int bulat;
typedef int desimal;
typedef char huruf;
typedef char String[100];
```

STRUCTURE

- Kumpulan dari beberapa variabel (elemen) dengan beragam tipe data yang dibungkus dalam satu nama
- Masing-masing elemen data tersebut dikenal juga dengan sebutan **field**. Elemen data tersebut dapat memiliki tipe data yang sama ataupun berbeda.
- Dalam bahasa pemrograman lain seperti Pascal, dikenal dengan records




```
struct mahasiswa
{
 char nim[11];
 char nama[30];
 char alamat[50];
 float ipk;
};
```

VARIABLE STRUCTURE

- Variabel structure dapat dideklarasikan bersamaan dengan deklarasi structure, atau
- sebagai deklarasi terpisah seperti mendeklarasikan variabel dengan tipe data dasar.

```
struct data_tanggal
{
 int tahun;
 int bulan;
 int tanggal;
} ultah;
```

```
struct data_tanggal
{
 int tahun;
 int bulan;
 int tanggal;
};

struct data_tanggal ultah;
```

```
struct data_tanggal
{
 int tahun;
 int bulan;
 int tanggal;
};

int main()
{
   struct data_tanggal tgl_masuk;
   struct data_tanggal tgl_ultah;
}
```

- **`typedef`** sendiri: hanya rename saja
- **`structure`** sendiri: hanya pengelompokan variabel
- **`typedef + structure`** + operasi-operasi = Tipe Data Bentukan / Abstract Data Type

Operation	Description
`Creation`	Allocation of memory for the data structure, the creation of data structure may take place either during compile-time or during run-time.
`Insertion`	Insert a data item in the data structure.
`Deletion`	Delete a data item from the data structure.
`Traversing`	Accessing and processing each data item of the data structure exactly once.
`Searching`	Find the location of the key value within the data structure.
`Sorting`	Arranging all the data items in a data structure either in ascending or in descending order or in lexicographical order (for Strings).
`Merging`	Combining the data items of two different sorted lists into a single sorted list.

TYPEDEF + STRUCTURE

```
// membuat struct dengan typedef
typedef struct Distance{
    int feet;
    float inch;
} distances;

void main() {
    // menggunakan struct
    distances dist1, dist2, sum;
}
```

- Perhatikan perbedaan penggunaan structure di atas dengan penggunaan structure tanpa typedef.

Contoh Penggunaan Structure

```c
#include <stdio.h>
#include <string.h>

//membuat struct
typedef struct Mahasiswa {
   char  name[20];
   char  address[20];
   int   age;
} db_mahasiswa;

int main( ) {
    //menggunakan struct
   db_mahasiswa mhs1;

//mengisi nilai elemen struct
   strcpy( mhs1.name, "Dian");
   strcpy( mhs1.address, "Mataram");
   mhs1.age = 22;

   // Mencetak isi elemen pada struct
   printf( "## Mahasiswa ##\n");
   printf( "Nama: %s\n", mhs1.name);
   printf( "Alamat: %s\n", mhs1.address);
   printf( "Umur: %d\n\n", mhs1.age);

   return 0;
}
```

Contoh Penggunaan Structure (2)

```c
#include <stdio.h>
#include <string.h>

typedef struct Buku {
   char  judul[50];
   char  pengarang[50];
   int   id;
} db_buku;

int main( ) {
   db_buku Buku1;
   db_buku Buku2;

   /* Spesifikasi Buku 1 */
   strcpy( Buku1.judul, "C Programming");
   strcpy( Buku1.pengarang, "Nuha Ali");
   Buku1.id = 6495407;

   /* Spesifikasi Buku 2 */
   strcpy( Buku2.judul, "Telecom Billing");
   strcpy( Buku2.pengarang, "Zara Ali");
   Buku2.id = 6495700;

   /* Cetak informasi Buku 1 */
   printf( "Judul Buku 1  : %s\n", Buku1.judul);
   printf( "Pengarang Buku 1  : %s\n", Buku1.pengarang);
   printf( "Id Buku 1 : %d\n\n", Buku1.id);

   /* Cetak informasi Buku 2 */
   printf( "Judul Buku 2  : %s\n", Buku2.judul);
   printf( "Pengarang Buku 2  : %s\n", Buku2.pengarang);
   printf( "Id Buku 2 : %d\n", Buku2.id);

   return 0;
}
```

Field Bertipe Array

```c
#include <stdio.h>
#include <string.h>

typedef struct Buku {
   char  judul[50];
   char  pengarang[50];
   int   id;
} db_buku;

int main( ) {
   db_buku Buku[10];

   /* Spesifikasi Buku 1 */
   strcpy( Buku[0].judul, "C Programming");
   strcpy( Buku[0].pengarang, "Nuha Ali");
   Buku[0].id = 6495407;

   /* Spesifikasi Buku 2 */
   strcpy( Buku[1].judul, "Telecom Billing");
   strcpy( Buku[1].pengarang, "Zara Ali");
   Buku[1].id = 6495700;

   /* Cetak informasi Buku 1 */
   printf( "Judul Buku 1  : %s\n", Buku[0].judul);
   printf( "Pengarang Buku 1  : %s\n", Buku[0].pengarang);
   printf( "Id Buku 1 : %d\n\n", Buku[0].id);

   /* Cetak informasi Buku 2 */
   printf( "Judul Buku 2  : %s\n", Buku[1].judul);
   printf( "Pengarang Buku 2  : %s\n", Buku[1].pengarang);
   printf( "Id Buku 2 : %d\n", Buku[1].id);

   return 0;
}
```

Field Bertipe Structure (Nested Structure)

```c
#include <stdio.h>
#include <string.h>

typedef struct Address {
   char  street[18];
   char  city[18];
   char  state[2];
   char  zip[5];
} address_template;

typedef struct Mahasiswa {
   char name[20];
   address_template  address;
   int   age;
} mahasiswa_template;
```

```c
int main( ) {

    //menggunakan struct
   mahasiswa_template mhs1;


   //mengisi nilai elemen struct
   strcpy( mhs1.address.city, "Mataram");


   // Mencetak isi elemen pada struct
   printf( "## Mahasiswa ##\n");
   printf( "Kota: %s\n", mhs1.address.city);

   return 0;
}
```

Passing Structure sebagai Parameter

```c
#include <stdio.h>
struct student
{
    char name[50];
    int age;
};

void main() {
    struct student s1;
    printf("Enter name: ");
    gets(s1.name);
    printf("Enter age: ");
    scanf("%d", &s1.age);
    // passing structure as an argument
    display(s1);
}
```

```c
// membuat fungsi dengan struct sebagai parameter
void display(struct student s) {
  printf("\nDisplaying information\n");
  printf("Name: %s", s.name);
  printf("\nRoll: %d", s.age);
}
```

Latihan

- Apa sajakah perbedaan tipe data float dengan double?
- Apa sajakah perbedaan tipe data signed dengan unsigned char?
- Bagaimana caranya menggunakan tipe data boolean di C?
- Cobalah untuk mengecek panjang tipe data dasar pada laptop/compiler Anda masing-masing, menggunakan library **limits.h**. Apakah hasilnya? Bandingkan dengan hasil pengecekan teman-teman Anda.

TERIMA KASIH