



Ahmad Muhardian · 26 May 2019

Belajar Pemrograman C #13: Mengenal Tipe Data Struct

#C



Tipe Data Struct pada Program C

petanikode.com

Apa itu Struct?

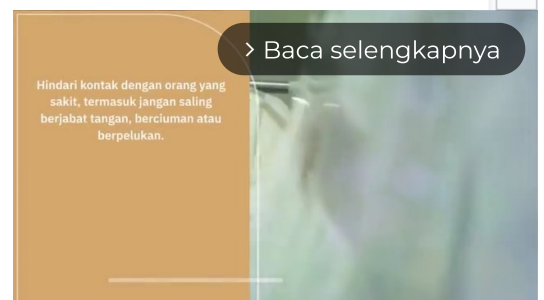
Structure atau struct adalah kumpulan dari beberapa variabel dengan beragam tipe data yang dibungkus dalam satu variabel.

Dalam bahasa pemrograman lain, struct ini bisa disamakan seperti:

- *Records* kalau di bahasa Pascal;
- *Dictionary* kalau di bahasa Python;
- *Asosiatif Array* kalau di bahasa PHP;
- *Object* kalau di bahasa Javascript.

Nah, untuk bahasa pemrograman C, Kita nyebutnya *Struct*.

Mengapa sih kita membutuhkan struct?



Sekarang coba perhatikan contoh kasus berikut:

Misalnya kita ingin menyimpan data mahasiswa. Kita bisa saja melakukannya seperti ini:

```
char name[] = "Dian";  
char address[] = "Mataram";  
int age = 22;
```

Lalu bagaimana kalau ada lebih dari satu mahasiswa?

Mungkin bisa saja kita buat seperti ini:

```
char name[] = "Dian";  
char address[] = "Mataram";  
int age = 22;  
  
char name2[] = "Bambang";  
char address2[] = "Surabaya";  
int age2 = 23;  
  
char name3[] = "Bimo";  
char address3[] = "Jakarta";  
int age3 = 23;
```

Ugh! terlihat kurang bagus.

Biar tidak membuat banyak variabel seperti ini, maka variabel-variabel yang masih dalam satu kelompok bisa kita bungkus di dalam *struct*.

Gimana caranya?

Mari kita pelajari:

Ad

Cara Membuat Struct

Struct dapat kita buat dengan kata kunci **struct** kemudian diikuti dengan nama struct dan isinya.



Contoh:

```
struct Mahasiswa
{
    char *name;
    char *address;
    int age;
};
```

Pada contoh ini, kita membuat struct dengan nama **Mahasiswa**.

Perhatikan, di sini kita menggunakan tanda ***** (pointer) untuk tipe data **char**, supaya bisa diisi dengan string.

Selain menggunakan cara di atas, kita juga bisa membuat struct dengan kata kunci **typedef**.

Contohnya:

```
typedef struct Mahasiswa
{
    char *name;
    char *address;
    int age;
};
```

Atau bisa juga seperti ini:

```
typedef struct
{
    char *name;
    char *address;
    int age;
} Mahasiswa;
```

Nanti kita bahas, gimana bedanya membuat struct yang menggunakan `typedef` dan yang tanpa `typedef`.

Untuk saat ini, kita pelajari dulu gimana cara menggunakan Struct.

Cara Menggunakan Struct

Agar struct dapat digunakan, kita harus membuat variabel untuknya. Caranya dengan menggunakan nama struct sebagai tipe data.

Contoh:

```
struct Mahasiswa mhs1;
```

Lalu melalui variabel `mhs1`, kita bisa mengakses anggota struct dengan cara seperti ini:

```
// mengisi data ke struct
mhs1.name = "Petani Kode";
mhs1.address = "Bandung";
mhs1.age = 22;
```

Tanda `.` adalah operator untuk mengakses member pada struct.

Selain itu, kita juga bisa membuat variabel untuk struct dengan mengisinya secara langsung seperti ini:

```
struct Mahasiswa mhs1 = {
    .name = "Petani Kode",
    .address = "Bandung",
    .age = 22
};
```

Cara yang seperti ini disebut *Designated initializers*.

Kita bebas menggunakan cara manapun yang diinginkan.

Oke.. biar makin paham, mari kita coba latihan.

Latihan Menggunakan Struct

Buatlah program baru dengan nama `contoh_struct.c` kemudian isi dengan kode berikut.

```

#include <stdio.h>

// membuat struct
struct Mahasiswa {
    char *name;
    char *address;
    int age;
};

int main(){

    // menggunakan struct
    struct Mahasiswa mhs1, mhs2;

    // mengisi nilai ke struct
    mhs1.name = "Dian";
    mhs1.address = "Mataram";
    mhs1.age = 22;

    // mengisi nilai ke struct
    mhs2 = {
        .name = "Bambang",
        .address = "Surabaya",
        .age = 23
    };

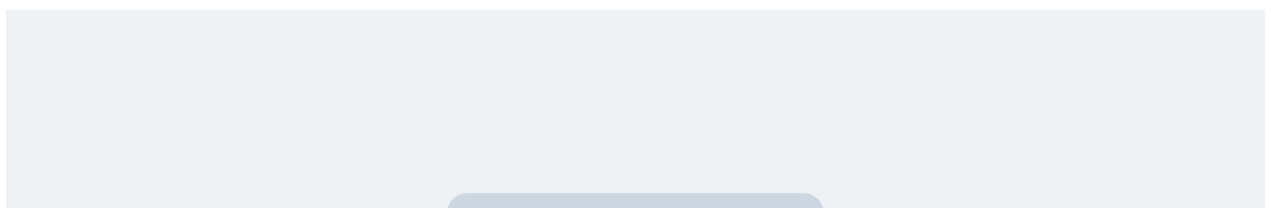
    // mencetak isi struct
    printf("## Mahasiswa 1 ##\n");
    printf("Nama: %s\n", mhs1.name);
    printf("Alamat: %s\n", mhs1.address);
    printf("Umur: %d\n", mhs1.age);

    printf("## Mahasiswa 2 ##\n");
    printf("Nama: %s\n", mhs2.name);
    printf("Alamat: %s\n", mhs2.address);
    printf("Umur: %d\n", mhs2.age);

    return 0;
}

```

Hasilnya:





Lanjut kita bahas:

Gimana bedanya Struct yang pakai `typedef` dengan yang biasa?

Seperti yang sudah kita pelajari di atas, struct bisa dibuat dengan dua cara. Yakni menggunakan `typedef` dan tanpa `typedef`.

Apa bedanya?

Kata kunci `typedef` adalah kata kunci untuk membuat tipe data baru di C. Saat kita menggunakan `typedef` untuk struct, maka struct tersebut akan dikenali sebagai tipe data.

Sehingga saat menggunakan struct, kita tidak perlu lagi pakai kata kunci `struct`.

Contohnya:

```
typedef struct Phone {  
    char* name;  
    char* cpu;  
    int* memory;  
};  
  
struct Laptop {  
    char* name;  
    char* cpu;  
    int memory;  
};
```

Ada dua struct yang kita buat di contoh ini, yang pertama struct `Phone` dengan menggunakan `typedef` dan yang kedua struct `Laptop` yang tidak menggunakan `typedef`.

Cara pakai kedua struct ini akan berbeda.

Kalau struct **Phone** dipakai langsung tanpa perlu **struct**, seperti ini:

```
Phone iphone = {  
    .name = "iPhone Pro max",  
    .cpu = "ARM",  
    .memory = 8  
}
```

Sementara untuk struct **Laptop** harus menggunakan struct di depannya.

```
struct Laptop lenovo = {  
    .name = "ThinkPad X",  
    .cpu = "AMD",  
    .memory = 16  
}
```

Mengapa harus pakai **struct** di struct **Laptop**?

Ya karena struct **Laptop** tidak pakai **typedef**, sehingga dia tidak dianggap sebagai tipe data.

Sampai sini apa kamu sudah paham?

Kalau gitu, kita lanjut bahas struct di dalam struct.



Struct Bersarang

Struct bisa juga kita buat di dalam struct. Ini disebut dengan *nested struct* atau struct bersarang.

Contoh:

```
struct Weapon
{
    char* name;
    int attackPower;
    int range;
};

struct Player
{
    char* name;
    int healthPoin;
    Weapon weapon;
};
```

Lalu cara menggunakannya akan seperti ini:

```
Player player1;

player1.name = "Petani Kode";
player1.healthPoin = 100;
player1.weapon.name = "Katana";
player1.weapon.attackPower = 30;
player1.weapon.range = 100;
```

Atau bisa juga seperti ini:


```
Player player1 = {
    .name = "Petani Kode",
    .healthPoin = 100, // 100%
    .weapon = {
        .name = "Katana",
        .attackPower = 30,
        .range = 100, // 1 meter
    }
};
```

Supaya makin paham, mari kita coba latihan.

Latihan Nested Struct

Buatlah file baru dengan nama `nested_struct.c`, kemudian isi dengan kode berikut:

```
#include <stdio.h>

int main () {
    // mendefinisikan struct Weapon
    struct Weapon {
        char* name;
        int attackPower;
        int range;
    };

    // mendefinisikan struct Player
    struct Player {
        char* name;
        int healthPoin;
        struct Weapon weapon;
    };

    // membuat object struct
    struct Player player1 = {
        .name = "Petani Kode",
        .healthPoin = 100,
        .weapon = {
            .name = "Katana",
            .attackPower = 30,
            .range = 100
        }
    };

    // print player dan weapon
```

```

printf("== Player Status ==\n");
printf("Player: %s\n", player1.name);
printf("HP: %d\n", player1.healthPoin);
printf("-- 🗡️ Weapon --\n");
printf("    Name: %s\n", player1.weapon.name);
printf("    Attack: %d\n", player1.weapon.attackPower);
printf("    Range: %d\n", player1.weapon.range);

return 0;
}

```

Setelah itu, compile dan jalankan.

Maka hasilnya:



Pada contoh ini, kita membuat dua Struct yakni struct **Player** dan struct **Weapon**. Di dalam struct **Player** ada struct **Weapon**.

Perhatikanlah cara mengakses struct **Weapon** yang ada di dalam **Player**, kita harus mengakses **player1** terlebih dahulu.. baru setelah itu kita akses **weapon** yang merupakan member dari **Player**.

```

printf("    Name: %s\n", player1.weapon.name);
printf("    Attack: %d\n", player1.weapon.attackPower);
printf("    Range: %d\n", player1.weapon.range);

```

Kalau ada struct di dalam struct, lalu di dalam struct lagi gimana?

Ya caranya sama aja kita harus akses dari struct terluar, lalu masuk ke dalam.

```
structLuar.struct.struct.struct;
```

Tapi umumnya, nested struct dibuat sampai maksimal tiga level. Kalau lebih dari itu, akan terlihat kompleks.

Passing Struct ke dalam Fungsi

Struct dapat kita buat sebagai parameter untuk fungsi.

Contoh:

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

int main() {
    struct student s1;

    printf("Enter name: ");
    scanf("%[^\n]*c", s1.name);

    printf("Enter age: ");
    scanf("%d", &s1.age);

    display(s1); // passing structure as an argument

    return 0;
}

// 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);
}
```

Hasilnya:



Apa Selanjutnya?

Sejauh ini kita sudah mengenal dan menggunakan Struct.

Intinya:

Struct itu buat menyimpan beberapa data yang masih berkaitan dalam satu variabel.

Selanjutnya:

Pelajari tentang tipe data union. Tipe data ini baru dikenalkan di versi C11.

- [Belajar C #13: Tipe Data Union](#)

Kalau ada yang kurang dimengerti, silakan tanyakan di komentar.

Selamat belajar!

19 Komentar

📖 Daftar isi tutorial


📖 Belajar Pemrograman C #01: Pengenalan Bahasa Pemrograman C

📖 Belajar Pemrograman C #02: Persiapan Pemrograman C di Linux

 Belajar Pemrograman C #02: Persiapan Pemrograman C di Windows


 Belajar Pemrograman C #03: Struktur Dasar Penulisan Program C

 Belajar Pemrograman C #04: Fungsi Input dan Output

 Belajar Pemrograman C #05: Variabel, Konstanta, dan Tipe Data


 Belajar Pemrograman C #06: Operator

 Belajar Pemrograman C #07: Percabangan


 Belajar Pemrograman C #08: Perulangan


 Belajar Pemrograman C #19: Struktur Data Array

 Belajar Pemrograman C #10: Prosedur dan Fungsi


 Belajar Pemrograman C #11: Tipe Data Enum

 Belajar Pemrograman C #12: Tipe Data Structure

 Belajar Pemrograman C #13: Tipe Data Union

 Belajar Pemrograman C #14: Tipe Data String

 Belajar Pemrograman C #15: Apa itu Pointer?

 Belajar Pemrograman C #16: Fungsi untuk Alokasi Memori

 Belajar Pemrograman C #17: Cara Membaca dan Menulis File di C

 Belajar Pemrograman C #18: Memahami Preprocessor dan Macro

 Belajar Pemrograman C #19: Header File pada C



Newsletter..

Mau dapat tips belajar coding, info teknologi, dan perkembangan karir sebagai programmer?

 [Subscribe](#)

Artikel Terbaru

Belajar C++ #13: Mengenal Tipe Data Union

03 May 2024 • baca 5 menit



Belajar C++ #12: Mengenal Tipe Data Struct

01 May 2024 • baca 8 menit



Belajar C++ #11: Tipe Data Enum di C++

28 Apr 2024 • baca 6 menit



Belajar C++ #14: Memahami Pointer di C++

23 Nov 2023 • baca 10 menit



Cara Install dan Setup Unity Engine di Mac dengan Benar

23 Nov 2023 • baca 5 menit



Tutorial CSS: Menentukan Ukuran Elemen dengan Satuan yang Tepat

28 Mar 2023 • baca 8 menit



Powered by **GliaStudio**



Tempat belajar budidaya kode (coding)
dengan tutorial yang gampang dipahami.

Belajar

[Artikel](#)

[Tutorial](#)

[Buku](#)

Popular Tutorial

[Tutorial Bahasa C](#)

[Tutorial Javascript](#)

[Tutorial Java](#)

[Tutorial PHP](#)

[Tutorial Python](#)

Social Media

[Facebook Page](#)

[Instagram](#)

[Twitter](#)

[Youtube Channel](#)

[Telegram Channel](#)

Petani Kode

[About](#)

[FAQs](#)

[Contact](#)

Ikuti Kami di



© 2024 **Petani Kode** · Made with ❤️ using Hugo 0.123.8