Bài 7: Struct - Union

Nguyễn Hoàng Anh

Struct

Trong ngôn ngữ lập trình C, struct là một cấu trúc dữ liệu cho phép lập trình viên tự định nghĩa một kiểu dữ liệu mới bằng cách nhóm các biến có các kiểu dữ liệu khác nhau lại với nhau. struct cho phép tạo ra một thực thể dữ liệu lớn hơn và có tổ chức hơn từ các thành viên (members) của nó.

```
struct name_struct
{
    <data type 1> <member 1>;
    <data type 2> <member 2>;
    // ...
};
```

```
typedef struct name_struct
{
    <data type 1> <member 1>;
    <data type 2> <member 2>;
    // ...
} name_struct;
```

Struct

```
struct Example
{
    uint8_t  a; // 1 byte
    uint16_t b; // 2 byte
    uint32_t c; // 4 byte
};
```

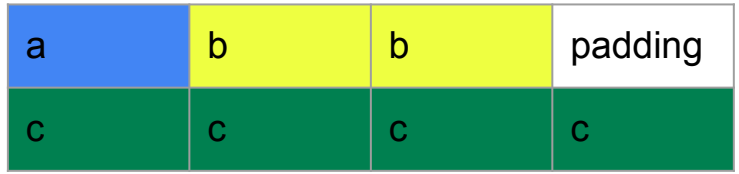
Struct

```
struct Example
{
    uint8_t  a; // 1 byte
    uint16_t b; // 2 byte
    uint32_t c; // 4 byte
};
```

uint8_t a

uint16_t b

uint32_t c



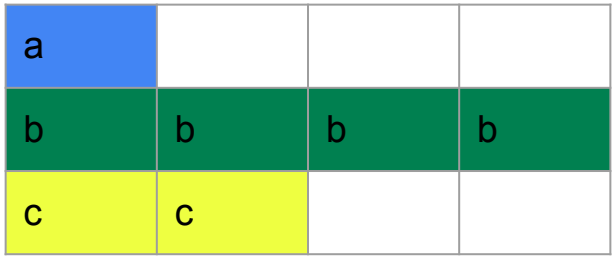
Struct

```
struct Example
{
    uint8_t  a;
    uint32_t b;
    uint16_t c;
};
```

uint8_t a

uint32_t b

uint16_t c



Struct

```
struct Example1
{
    uint8_t  arr1[5];  // 5x1
    uint16_t arr2[4];  // 4x2
    uint32_t arr3[2];  // 2x4
};
```

Struct

```
struct Example1
{
    uint8_t  arr1[0];
    uint8_t  arr1[1];
    uint8_t  arr1[2];
    uint8_t  arr1[3];
    uint8_t  arr1[4];
    uint16_t arr2[4];
    uint32_t arr3[2];
};
```

arr1[0]	arr1[1]	arr1[2]	arr1[3]
arr1[4]			

Struct

```
struct Example1
{
    uint8_t  arr1[5];
    uint16_t arr2[0];
    uint16_t arr2[1];
    uint16_t arr2[2];
    uint16_t arr2[3];
    uint32_t arr3[2];
};
```

arr1[0]	arr1[1]	arr1[2]	arr1[3]
arr1[4]	arr2[0]	arr2[0]	padding
arr2[1]	arr2[1]	arr2[2]	arr2[2]
arr2[3]	arr2[3]		

Struct

```
struct Example1
{
    uint8_t  arr1[5];
    uint16_t arr2[4];
    uint32_t arr3[0];
    uint32_t arr3[1];
};
```

arr1[0]	arr1[1]	arr1[2]	arr1[3]
arr1[4]	arr2[0]	arr2[0]	
arr2[1]	arr2[1]	arr2[2]	arr2[2]
arr2[3]	arr2[3]		
arr3[0]	arr3[0]	arr3[0]	arr3[0]
arr3[1]	arr3[1]	arr3[1]	arr3[1]

Struct

```
struct Student {
    int studentID;
    char name[50];
    double GPA;
};
```

```
struct Address {
    int number;
    char street[50];
    char city[30];
    int zipCode;
};
```

```
struct Point {
    double x;
    double y;
};
```

Bit field

Trong C, “bit field” (trường bit) là một thành phần đặc biệt của cấu trúc (struct) cho phép bạn chỉ định số lượng bit cụ thể dùng để lưu trữ một biến số nguyên. Thay vì sử dụng toàn bộ kích thước của một kiểu dữ liệu, bạn có thể “cắt nhỏ” bộ nhớ theo số bit cần thiết, giúp tiết kiệm không gian bộ nhớ và mô tả chính xác hơn ý nghĩa của dữ liệu (ví dụ: lưu trạng thái bật/tắt chỉ cần 1 bit)

```
struct name_struct
{
    <data type 1> <member 1> : <number of bits>;
    <data type 2> <member 2> : <number of bits>;
    // ...
```

Bit field

```
struct Example
{
    int32_t flag  : 1;    // chỉ sử dụng 1 trong 32 bit
    int64_t count : 4;    // chỉ sử dụng 4 trong 64 bit
};
```

- Số bit chỉ định trực tiếp giới hạn **phạm vi giá trị** có thể lưu. Ví dụ: một bit field khai báo với **: 3** có thể lưu các giá trị từ **0 đến 7** (đối với unsigned).
- Không thể sử dụng toán tử lấy địa chỉ (&) trên các thành viên bit field.

Bit field

```c
#include <stdio.h>
#include <stdint.h>
#define COLOR_RED   0
#define COLOR_BLUE  1
#define COLOR_BLACK 2
#define COLOR_WHITE 3
#define POWER_100HP 0
#define POWER_150HP 1
#define POWER_200HP 2
#define ENGINE_1_5L 0
#define ENGINE_2_0L 1

typedef uint8_t CarColor;
typedef uint8_t CarPower;
typedef uint8_t CarEngine;

#define SUNROOF_MASK 1 << 0     // 0001
#define PREMIUM_AUDIO_MASK 1 << 1 // 0010
#define SPORTS_PACKAGE_MASK 1 << 2 // 0100
// Thêm các bit masks khác tùy thuộc vào tùy chọn
```

Struct

Ứng dụng của struct trong:

- cấu hình (GPIO, UART, SPI, v.v)
- json
- list

Union

Trong ngôn ngữ lập trình C, union là một cấu trúc dữ liệu giúp lập trình viên kết hợp nhiều kiểu dữ liệu khác nhau vào cùng một vùng nhớ. Mục đích chính của union là tiết kiệm bộ nhớ bằng cách **chia sẻ cùng một vùng nhớ** cho các thành viên của nó. Điều này có nghĩa là, trong một thời điểm, chỉ một thành viên của union có thể được sử dụng. Điều này được ứng dụng nhằm tiết kiệm bộ nhớ.

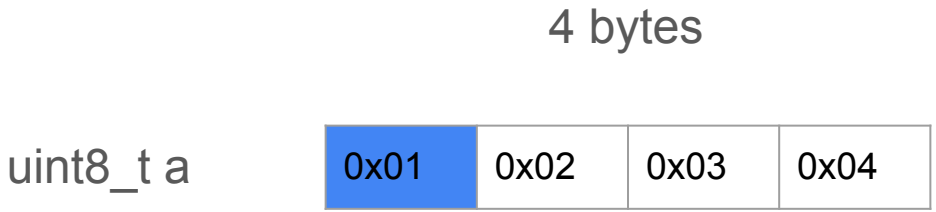
```
union name_union
{
    kieuDuLieu1 thanhVien1;
    kieuDuLieu2 thanhVien2;
    // ...
};
```

Union

```
union Data
{
    uint8_t  a;
    uint16_t b;
    uint32_t c;
};
```

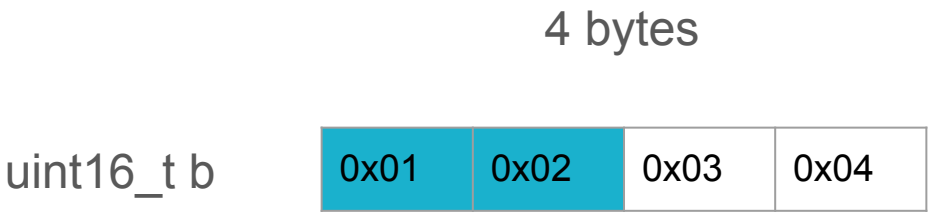
Union

```
union Data
{
    uint8_t  a;
    uint16_t b;
    uint32_t c;
};
```



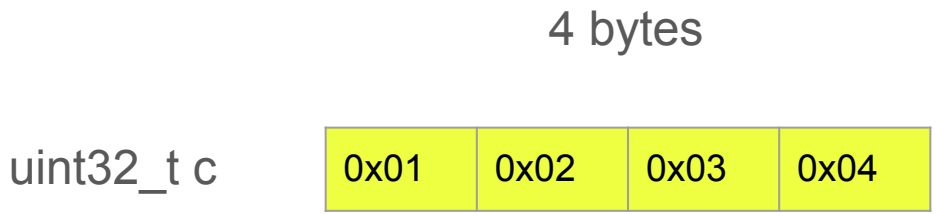
Union

```
union Data
{
    uint8_t  a;
    uint16_t b;
    uint32_t c;
};
```



Union

```
union Data
{
    uint8_t  a;
    uint16_t b;
    uint32_t c;
};
```



Union

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

typedef union Data
{
    uint8_t arr1[5];
    uint8_t arr2[3];
    uint8_t arr3[6];
} Data;
```

```c
void display(uint8_t arr[], int size)
{
```

Union

```
union Data
{
    uint8_t arr1[5]; // 5 byte
    uint8_t arr2[3]; // 3 byte
    uint8_t arr3[6]; // 6 byte
};
```

a3[0]	a3[1]	a3[2]	a3[3]	a3[4]	a3[5]
0x9a	0x9b	0x9c	0x9d	0x9e	0x9f

Union

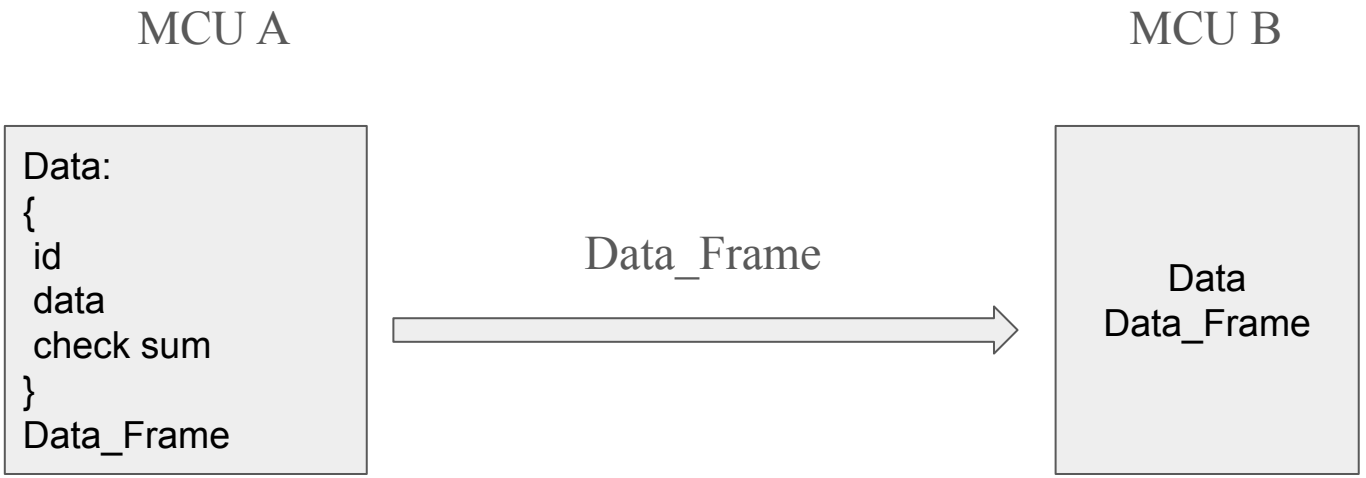
```
union Data
{
    uint8_t  arr1[5];
    uint16_t arr2[9];
    uint32_t arr3[3];
};
```

Union

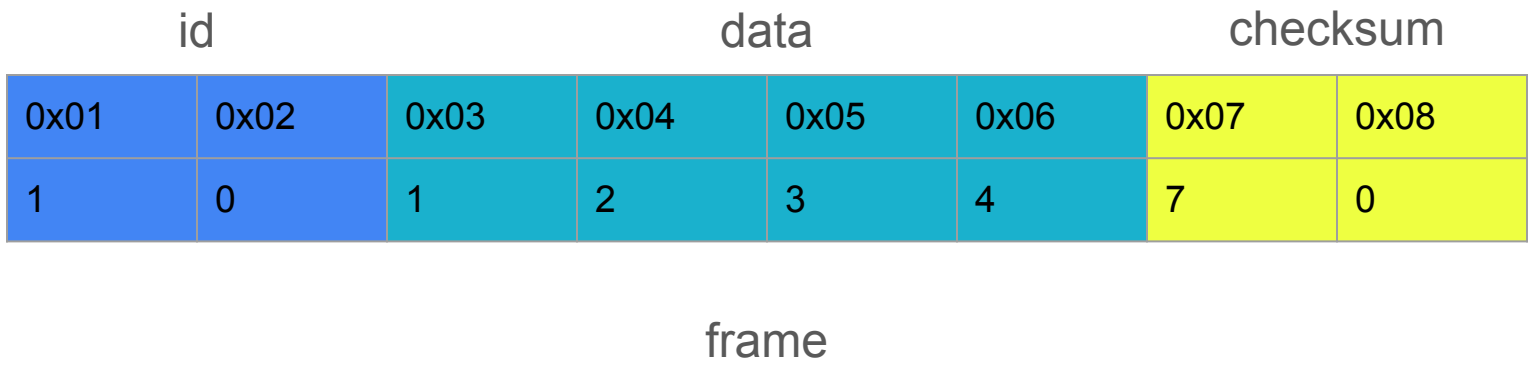
```
union SensorData
{
    int temperature;
    float humidity;
    char motionStatus;
};
```

```
union Number
{
    int intValue;
    float floatValue;
};
```

Ứng dụng kết hợp struct và union



Ứng dụng kết hợp struct và union



Ứng dụng kết hợp struct và union

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

typedef union
{
    struct
    {
        uint8_t id[2];
        uint8_t data[4];
        uint8_t check_sum[2];
    } data;

    uint8_t frame[8];
} Data_Frame;

int main(int argc, char const *argv[])
{
```