Embedded Systems

Task 7

by Mohammed Elahmady

Task 7 Objects

- Task 7 Objects
  - What is the purpose of typedef?
  - How are bit fields declared and what are their size limitations?
  - What happens if a bit field overflows?
  - How is typedef used with complex types like structs and unions?
  - What is the default underlying type of an enum?
  - How is a union different from a struct?
  - When is using a union more memory-efficient?

What is the purpose of typedef?

How are bit fields declared and what are their size limitations?

What happens if a bit field overflows?

How is typedef used with complex types like structs and unions?

What is the default underlying type of an enum?

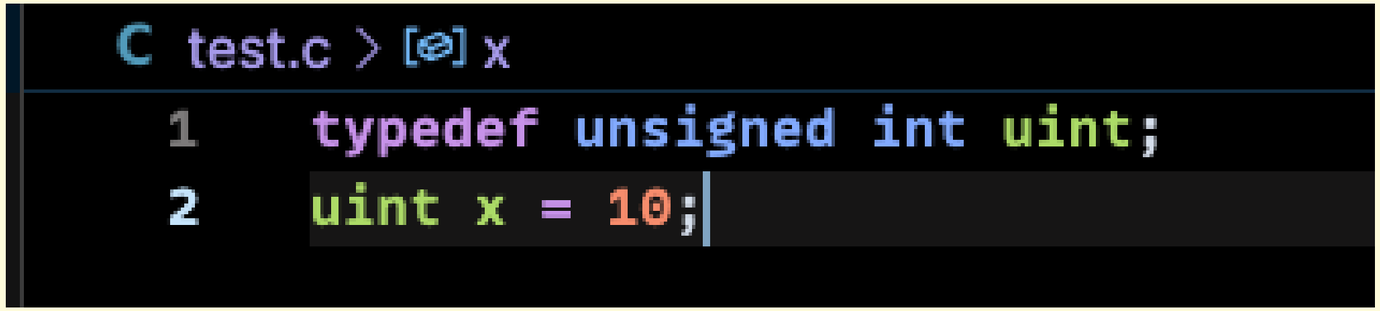
How is a union different from a struct?

When is using a union more memory-efficient?

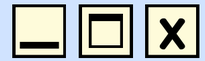
What is the purpose of typedef?

1. What is the purpose of typedef?

typedef creates a new alias (alternative name) for an existing type, improving code readability and portability.



How are bit fields declared and what are their size limitations?



2. How are bit fields declared and what are their size limitations?

Bit fields allow the packing of data in structures using a specific number of bits per member.

test.c > ...

```c
struct Flags {
    unsigned int isVisible : 1;
    unsigned int isAlive   : 1;
    unsigned int level     : 4;  // 4 bits
};
```

What happens if a bit field overflows?

3. What happens if a bit field overflows?

If a value exceeds the allocated bit size, it results in truncation—only the least significant bits are stored.

test.c > ...

```c
struct S {
    unsigned int x : 3;
};
struct S s;
s.x = 9;  // 1001 in binary → truncated to 001 → s.x becomes 1
```

How is typedef used with complex types like structs and unions?

4. How is typedef used with complex types like structs and unions?

typedef simplifies names of structs/unions for easier reuse.

test.c > s

```c
typedef struct {
    int x, y;
} Point;

Point p1 = {1, 2};
```

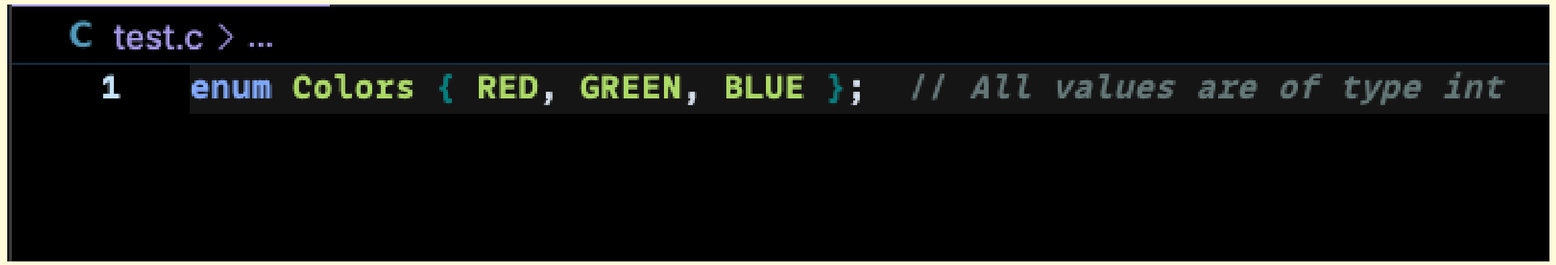
test.c > IntOrFloat

```c
typedef union {
    int i;
    float f;
} IntOrFloat;
```

What is the default underlying type of an enum?

5. What is the default underlying type of an enum?

- In C, the default underlying type of enum is int, unless otherwise specified (C99 and later allow custom underlying types in some compilers like GCC via extensions).



How is a union different from a struct?

6. How is a union different from a struct?

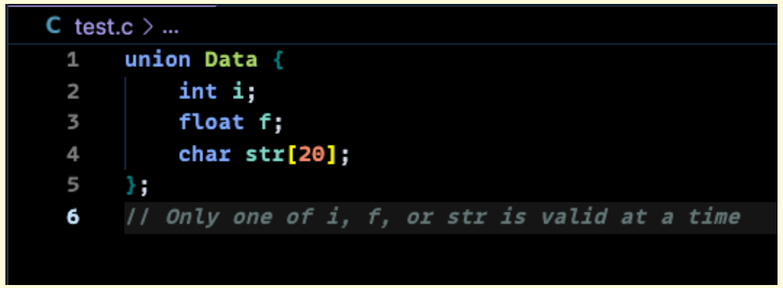
Feature	Struct	Union
Memory	Allocates **separate** memory for each member	**Shares** memory among all members
Access	All members can be used simultaneously	Only **one member** holds a value at a time
Size	Sum of all members + padding	Size of the **largest member**

When is using a union more memory-efficient?

7. When is using a union more memory-efficient?

Unions are more memory-efficient when:

- Only one member is used at a time.
- You need to store different types in the same memory space (e.g., variant types, embedded drivers, low-level protocols).



THANK YOU

Head : Tasnem Sabry

Vice : Ahmed Yasser