知识点总结

· 在C语言中，共用体（union）是一种特殊的数据类型，它允许不同的数据类型共用同一块内存空间，使得同一内存区域在不同时刻可被解释为不同的数据类型。

· 定义一个共用体的语法格式：

```
union 共用体标签 {
    数据类型 成员1;
    数据类型 成员2;
    ... // 可以有多个不同类型的成员
};
```

例：定义一个包含int、float和char类型成员，可在不同时刻存储不同类型数据的共用体。

```
union MultiType {
    int int_value;
    float float_value;
    char char_value;
};
```

·定义与初始化共用体变量：

1）在定义好共用体之后，再定义共用体变量，同时对共用体变量进行初始化。

例：

```
union MultiType {
    int int_value;
    float float_value;
    char char_value;
};
union MultiType data = {10};
```

2）在定义共用体类型的同时，对共用体变量进行定义与初始化。

例：

```
union MultiType {
    int int_value;
    float float_value;
    char char_value;
} data = {10};
```

注意：只能初始化共用体的第一个成员。上述代码均是将data里的成员int_value的值初始化为10。

· 通过成员访问符（. 或 ->）访问共用体变量中的成员：

①直接操作共用体变量本身：共用体变量名.成员名

例：直接操作共用体变量访问并打印出data里的int_value成员的值。

```
union MultiType data = {10};
printf("Int value: %d\n", data.int_value);
```

②使用指向共用体变量的指针：指针名->成员名

例：使用指向共用体变量的指针访问并打印出data里的int_value成员的值。

```
union MultiType data = {10};
union MultiType* p = &data;
printf("Int value: %d\n", p->int_value);
```

·若定义出共用体变量，后续再对变量里的成员进行赋值时，需要借助成员访问符来实现。因为一旦完成共用体变量定义后，我们就不能再进行整体赋值了。

例：

```
union MultiType data;

data.int_value = 10;✔
```

```
union MultiType data;

data = {10};❌
```

· 注意：

1、由于共用体的所有成员共用一块内存，所以同一时间只能储存其中一个成员的值。当修改共用体里一个成员的值时，会覆盖之前储存在这块内存的其它成员的值。

```c
#include <stdio.h>

union MultiType {
    int int_value;
    float float_value;
    char char_value;
};

int main() {
    union MultiType data;

    data.char_value = 'A';
    printf("Char value: %c\n", data.char_value);
    // 覆盖了char_value的值
    data.int_value = 70;
    printf("\n");
    printf("Char value: %c\n", data.char_value);
    printf("Int value: %d\n", data.int_value);
    // 覆盖了int_value的值
    data.float_value = 12.34;
    printf("\n");
    printf("Int value: %d\n", data.int_value);
    printf("Float value: %.2f\n", data.float_value);
}
```

运行结果如下：

```
Char value: A

Char value: F
Int value: 70

Int value: 1095069860
Float value: 12.34
```

· 注意：

1、由于共用体的所有成员共用一块内存，所以同一时间只能储存其中一个成员的值。当修改共用体里一个成员的值时，会覆盖之前储存在这块内存的其它成员的值。

2、在共用体中，由于所有成员共享同一块内存空间，这就要求该内存空间必须能够容纳占用内存最大的成员，因此，共用体的大小是由其最大成员的大小所决定。

- 补充：可以用typedef关键字给共用体类型定义一个更简洁的别名。
- 语法格式如下：

```
typedef union 共用体标签 {
    数据类型 成员名1;
    数据类型 成员名2;
    ... // 可以有更多成员
} 别名;
```

例：给union MultiType共用体定义一个别名MultiType。

```
typedef union MultiType {
    int int_value;
    float float_value;
    char char_value;
} MultiType;
MultiType data;
```

共用体的所有成员共享同一块内存，所以这块内存得装得下占用内存最大的成员

题目1、以下共用体的大小是（B）

```
union Test {
    int a;        4 byte
    char b[10];   10 byte
    double c;     8 byte
};
```

A. 4

B. 10

C. 16

D. 8

题目2、简述共用体与结构体的主要区别。

答：

（1）内存分配：

共用体所有成员共用同一块内存，同一时间只能存储一个成员的值，修改任一成员会覆盖其他成员的值。

结构体每个成员拥有独立内存，修改某一成员不影响其他成员。

（2）大小计算：

共用体大小由最大成员决定。

结构体大小由所有成员的大小共同决定。

（3）应用场景：

共用体适用于节省内存、灵活解析数据。

结构体适用于将逻辑相关的数据组合为整体。