知识点总结

·在C语言中，枚举（enum）是一种用户自定义的数据类型，它允许定义一组具有名称的整型常量。枚举类型为程序员在处理一组相关的整型常量值时，提供了一种显著提升代码可读性和可维护性的方式。

·定义枚举的语法格式：

```
enum 枚举名 {
    枚举常量名1,
    枚举常量名2,
    // 更多枚举常量
};
```

例：定义一个表示颜色的枚举类型。

```
enum Color {
    RED,
    GREEN,
    BLUE
};
```

· 另，可借助typedef关键字给枚举类型定义一个更简洁的别名。

例:

```
typedef enum Color {
    RED,
    GREEN,
    BLUE
} Color;
```

· 注意：

枚举常量的作用域是定义它们的封闭作用域（全局、函数或代码块），而非枚举类型内部。

这意味着：同一封闭作用域内，枚举常量名不可重复，即使它们属于不同的枚举类型。

不同封闭作用域内，允许同名枚举常量。

例：

在全局作用域下定义枚举类型Color和LedColor时，均使用了同一个枚举常量名RED。

```
enum Color { RED = 1 };
// 错误! 重复定义
enum LedColor { RED = 2 };
```

例：

在两个函数内部分别定义枚举类型Color和LedColor时，均使用了同一个枚举常量名RED。

```
void func1() {
    // 仅在 func1 内有效
    enum Color { RED = 1 };
    ...
}
void func2() {
    // 仅在 func2 内有效
    enum LedColor { RED = 2 };
    ...
}
```

· 枚举常量赋值规则

在枚举类型定义中，如果没有对枚举常量进行显式赋值（即未直接赋予具体整数值），那么第一个枚举常量会默认被赋值为0。

对于后续那些没有进行显式赋值的枚举常量，其值等于紧邻的前一个枚举常量的值加1。

例1：所有枚举常量均未进行显式赋值。

```
enum Color {
    RED,
    GREEN,
    BLUE
};
```

此时，RED = 0，GREEN = 1，BLUE = 2。

例2：对部分枚举常量进行显式赋值。

```
enum Color {
    RED,
    GREEN = 5,
    BLUE
};
```

此时，RED = 0，GREEN = 5，BLUE = 6。

例3：对全部枚举常量进行显式赋值。

```
enum Color {
    RED = 1,
    GREEN = 3,
    BLUE = 5
};
```

此时，RED = 1，GREEN = 3，BLUE = 5。

注意：枚举常量本质上是整型常量，在枚举类型定义的范围之外，无法对其重新赋值。

例：

```
enum Color {
    RED,
    GREEN,
    BLUE
};
RED = 10; ❌
```

· 定义与初始化枚举变量：

1）先定义枚举类型，再定义枚举变量，同时对枚举变量进行初始化。

例：

```
enum Color {
    RED,
    GREEN,
    BLUE
};
enum Color color = GREEN;
```

2）在定义枚举类型的同时，对枚举变量进行定义与初始化。

例：

```
enum Color {
    RED,
    GREEN,
    BLUE
} color = GREEN;
```

注意：我们在给枚举变量赋值时，优先使用枚举常量，避免直接使用整数。

```
enum Color color = GREEN; // 推荐，可读性高
enum Color color = 1;         // 不推荐
```

· 枚举的应用场景：

1）**提高代码可读性**：当需要使用**一组逻辑相关的整型常量**时，使用枚举可以显著提高代码的可读性。

例如，若要表示颜色，定义一个表示颜色的枚举，比直接使用数字 0、1、2 来表示颜色更直观。

```
enum Color {
    RED,
    GREEN,
    BLUE
};
```

2）简化条件判断：在switch语句中使用枚举可以使代码结构更清晰。

```c
#include <stdio.h>

enum Color {
    RED,
    GREEN,
    BLUE
};

int main() {
    enum Color current_color = GREEN;

    switch (current_color) {
    case RED:
        printf("The color is Red.\n");
        break;
    case GREEN:
        printf("The color is Green.\n");
        break;
    case BLUE:
        printf("The color is Blue.\n");
        break;
    default:
        break;
    }
    return 0;
}
```

注意：

1、枚举常量是整型常量（int类型），可在代码中直接使用。

在使用printf函数输出枚举常量时，枚举常量不能按字符串格式输出，应使用%d格式符。

```
printf("%d\n", RED); ✔
printf("%s\n", RED); ✘
```

2、枚举类型的大小通常与int类型相同，一般为4字节。

题目1、以下代码中，枚举常量Thu的值是多少？（C）

```
enum Week {MON, TUE, WED=5, THU, FRI};
```

A. 3

B. 4

C. 6

D. 5

题目2、以下代码的输出结果是什么？（B）

```
#include <stdio.h>

enum Direction {UP, DOWN, LEFT = 10, RIGHT};

int main() {
    printf("%d,%d", DOWN, RIGHT);
    return 0;
}
```

A. 1,10

B. 1,11

C. 2,11

D. 1,12

题目3、以下代码存在两处错误，请指出并修正。

```
#include <stdio.h>
enum Sensor {LOW = 0, HIGH = 1};

int main() {
    enum Sensor status;
    printf("输入传感器状态（0或1）: ");
    scanf("%s", &status);   scanf("%d", &status);
    if (status = LOW) {     if (status == LOW)
        printf("低电平");
    }
    else {
        printf("高电平");
    }
    return 0;
}
```