## 第七十八节： typedef 和#define 和 enum。

### 【78.1　typedef 和#define 和 enum。】

typedef 称为“类型定义”，#define 称为“宏定义”，enum 称为“枚举”。三者都有“一键替换”的能力，但是应用的侧重点各有不同。请看下面的例子，要写一个函数，把学生的分数分为 3 个等级，第 1 等级是“优”（范围：“优”>=90 分），第 2 等级是“中”（范围：70 分<=“中”<90 分），第 3 等级是“差”（范围：“差”<70 分），实现此算法的函数需要一个输入口和一个输出口，用来输入分数和输出判断结果，判断的结果用三个数字常量 0,1,2 来表示，0 代表“优”，1 代表“中”，2 代表“差”。代码如下：

```
unsigned char GetGrade(unsigned char u8Score)
{
     if(u8Score<70)
      {
         return 2;  //2 代表“差”
      }
      else if(u8Score>=70&&u8Score<90)
      {
         return 1;  //1 代表“中”
      }
      else
      {
         return 0;  //0 代表“优”
      }
}
```

上述代码没有添加任何“typedef，#define，enum”，是“素颜照”级别的原始代码。现在对上述代码做一些美容，加入“typedef，#define，enum”的元素，代码如下：

```
#define BAD_MEDIUM   70  //宏定义。用 BAD_MEDIUM 来表示“差”和“中”分数的分界线
#define MEDIUM_GOOD  90  //宏定义。用 MEDIUM_GOOD 来表示“中”和“优”分数的分界线

typedef unsigned  char u8;  //用 typedef 为类型“unsigned char”增加一个名为“u8”的代言人

enum {GOOD = 0,MEDIUM,BAD}; //用 enum 把“0,1,2”三个常量转换为“GOOD,MEDIUM,BAD”

u8 GetGrade(u8 u8Score)
{
      if(u8Score<BAD_MEDIUM) //等级分数分界线的判断
      {
         return BAD;    //BAD 就是常量 2，代表“差”。
      }
      else if(u8Score>=BAD_MEDIUM&&u8Score<MEDIUM_GOOD)  //等级分数分界线的判断
      {
```

```
            return MEDIUM;  //MEDIUM 就是常量 1，代表“中”
        }
        else
        {
            return GOOD;    //GOOD 就是常量 0，代表“优”
        }
    }
```

代码赏析:

赏析片段一：

```
#define BAD_MEDIUM   70  //宏定义。用 BAD_MEDIUM 来表示“差”和“中”分数的分界线
#define MEDIUM_GOOD  90  //宏定义。用 MEDIUM_GOOD 来表示“良”和“优”分数的分界线
```

这里，用宏定义#define 来关联分界线判断的分数，给后续代码的升级维护带来了便捷，因为用户有可能会要求把“差”“中”“优”三者的分数线进行调整，这时直接更改 70 和 90 这个数值就可以实现分数线的调整。可见，宏定义#define 经常用在涉及“分界线”判断的场合。

赏析片段二：

```
typedef unsigned  char u8;  //用 typedef 为类型“unsigned char”增加一个名为“u8”的代言人
```

用类型定义 typedef 为类型“unsigned char”增加一个名为“u8”的代言人，u 代表 unsigned 的 u，8 代表此类型占用 8 位，比如 unsigned  char 就是占用 8 位的 unsigned 类型，所以用 u8。如果是 16 位的 unsigned 类型就用 u16，32 位则用 u32，这都是单片机界的常用命名习惯。上述代码用了类型定义，今后代码中凡是想定义一个 unsigned char 变量，都可以直接用 u8 来替代。这样有两个好处：第一个好处，u8 的字符个数明显比 unsigned char 少，省了敲代码的力气。第二个好处，方便代码在各种不同硬件平台上的移植，因为不同的单片机不同的编译器对 unsigned char，unsigned int,unsigned long 翻译所得的结果是不一样的，比如，51 单片机的 unsigned int 是占用 16 位的，而很多 32 位单片机的 unsigned int 是占用 32 位的，它们的 16 位则用 unsigned short int 类型，而不是 unsigned int。

当我们用 51 单片机写代码的时候，可以如下类型定义：

```
typedef unsigned  char     u8;
typedef unsigned  int      u16;
typedef unsigned  long     u32;
```

当我们用 32 位的单片机写代码的时候，可以如下类型定义：

```
typedef unsigned char          u8;
typedef unsigned short int     u16;
typedef unsigned int           u32;
```

这样，当我们想把 51 单片机的代码移到 32 位的单片机上时，只需要修改类型定义 typedef 这部分的代码，就可以快速做到代码在不同编译器平台上的类型兼容。

赏析片段三：

```
enum {GOOD = 0,MEDIUM,BAD}; //用 enum 把“0,1,2”三个常量转换为“GOOD,MEDIUM,BAD”
```

用枚举 enum 把“0,1,2”三个常量转换为“GOOD,MEDIUM,BAD”英文单词，最大的好处就是方便代码的阅读和修改。再多补充一点枚举的基础知识，上述代码中，第一个英文单词 GOOD，经过“GOOD = 0”这条初始化的语句后，等效于常量 0，后面的 MEDIUM 和 BAD 则 C 编译器自动对它们进行“累加 1”排序，所以 MEDIUM 和 BAD 分别为常量 1,2，这是 C 语言的语法规则。枚举 enum 的应用侧重在某些涉及到“状态”的数据类型，但是也不绝对。

## 【78.2　enum 和 typedef 的相结合。】

enum 一旦搭载上 typedef 后，可以把各自的特性发挥得淋漓尽致，产生另外一种常见的用途，那就是“人造”数据类型的用途，这里的“人造”解读为“人为制造”之意。比如上述 78.1 的函数 u8 GetGrade(u8 u8Score)，输出接口接收的是 u8 类型，但是内部 return 返回的是枚举类型的“GOOD，MEDIUM，BAD”其中之一，而 u8 虽然也能接收和兼容常量“GOOD，MEDIUM，BAD”，但是总是感觉有点“类型不匹配”的“不适感”，如果想消除这点“不适感”，可以用 enum 和 typedef 相结合的办法，修改后代码如下：

```
#define BAD_MEDIUM   70  //宏定义。用 BAD_MEDIUM 来表示“差”和“中”分数的分界线
#define MEDIUM_GOOD  90  //宏定义。用 MEDIUM_GOOD 来表示“良”和“优”分数的分界线

typedef unsigned  char u8;  //用 typedef 为类型“unsigned char”增加一个名为“u8”的代言人

typedef enum {
    GOOD = 0,
    MEDIUM,
    BAD
  } Grade;  //通过 typedef 和 enum 的相结合，“人造”出一个新的数据类型 Grade。

Grade GetGrade(u8 u8Score)  //这里返回的类型是 Grade，而“GOOD,MEDIUM,BAD”就是属于 Grade
{
      if(u8Score<BAD_MEDIUM) //等级分数分界线的判断
      {
         return BAD;    //BAD 就是常量 2，代表“差”。
      }
      else if(u8Score>=BAD_MEDIUM&&u8Score<MEDIUM_GOOD)  //等级分数分界线的判断
      {
         return MEDIUM;  //MEDIUM 就是常量 1，代表“中”
      }
      else
      {
         return GOOD;    //GOOD 就是常量 0，代表“优”
      }
}
```

## 【78.3　例程练习和分析。】

为了熟悉 typedef，#define，enum 的用法，现在要写一个函数，把学生的分数分为 3 个等级，第 1 等级是“优”（范围：“优”>=90 分），第 2 等级是“中”（范围：70 分<=“中”<90 分），第 3 等级是“差”（范围：“差”<70 分），实现此算法的函数需要一个输入口和一个输出口，用来输入分数和输出判断结果，判断的结果用三个数字常量 0,1,2 来表示，0 代表“优”，1 代表“中”，2 代表“差”。

```
/*---C 语言学习区域的开始。-----------------------------------------------*/


    #define BAD_MEDIUM   70  //宏定义。用 BAD_MEDIUM 来表示“差”和“中”分数的分界线
    #define MEDIUM_GOOD  90  //宏定义。用 MEDIUM_GOOD 来表示“良”和“优”分数的分界线

    typedef unsigned  char u8;  //用 typedef 为类型“unsigned char”增加一个名为“u8”的代言人

    typedef enum {
        GOOD = 0,
        MEDIUM,
        BAD
      } Grade;  //通过 typedef 和 enum 的相结合，“人造”出一个新的数据类型 Grade。

    Grade GetGrade(u8 u8Score);  //函数声明

    Grade a;  //“人造”出 Grade 类型的变量 a，用来接收函数的判断结果。
    Grade b;  //“人造”出 Grade 类型的变量 b，用来接收函数的判断结果。
    Grade c;  //“人造”出 Grade 类型的变量 c，用来接收函数的判断结果。

    Grade GetGrade(u8 u8Score)  //这里返回的类型是 Grade，而“GOOD,MEDIUM,BAD”就是属于 Grade
    {
          if(u8Score<BAD_MEDIUM) //等级分数分界线的判断
          {
             return BAD;    //BAD 就是常量 2，代表“差”。
          }
          else if(u8Score>=BAD_MEDIUM&&u8Score<MEDIUM_GOOD)  //等级分数分界线的判断
          {
             return MEDIUM;  //MEDIUM 就是常量 1，代表“中”
          }
          else
          {
             return GOOD;    //GOOD 就是常量 0，代表“优”
          }
    }

void main() //主函数
```

```
{
    a=GetGrade(98);  //输入 98 分，a 来接收判断的结果
    b=GetGrade(88);  //输入 88 分，b 来接收判断的结果
    c=GetGrade(68);  //输入 68 分，c 来接收判断的结果

    View(a);  //在电脑端观察 98 分的判断结果 a
    View(b);  //在电脑端观察 88 分的判断结果 b
    View(c);  //在电脑端观察 68 分的判断结果 c
    while(1)
    {
    }
}
/*---C 语言学习区域的结束。----------------------------------------------*/
```

在电脑串口助手软件上观察到的程序执行现象如下：

```
开始...

第 1 个数
十进制:0
十六进制:0
二进制:0

第 2 个数
十进制:1
十六进制:1
二进制:1

第 3 个数
十进制:2
十六进制:2
二进制:10
```

分析：

98 分的判断结果 a 为 0,0 代表“优”。

88 分的判断结果 b 为 1,1 代表“中”。

68 分的判断结果 c 为 2,2 代表“差”。

## 【78.4　如何在单片机上练习本章节 C 语言程序？】

直接复制前面章节中第十一节的模板程序，练习代码时只需要更改“C 语言学习区域”的代码就可以了，其它部分的代码不要动。编译后，把程序下载进带串口的 51 学习板，通过电脑端的串口助手软件就可以观察到不同的变量数值，详细方法请看第十一节内容。