第五十九节： 全局“一键替换”功能的#define。

【59.1　#define 作用和书写格式。】

上一节讲 const 的时候，讲到了当某个常量在程序中是属于需要频繁更改的“阀值”的时候，用 const 就可以提供“一键更改”的快捷服务。本节的#define 也具有此功能，而且功能比 const 更加强大灵活，它除了可以应用在常量，还可以应用在运算式以及函数的“一键更改”中。所谓“一键更改”，其实是说，#define 内含了“替换”的功能，此“替换”跟 word 办公软件的“替换”功能几乎是一模一样的。#define 的“替换”功能，除了在某些场合起到“一键更改”的作用，还可以在某些场合，把一些在字符命名上不方便阅读理解的常量、运算式或函数先“替换”成容易理解的字符串，让程序阅读起来更加清晰更加方便维护。#define 的常见三种书写格式如下：

```
#define  字符串  常量     //注意，这里后面没有分号“;”
#define  字符串  运算式   //注意，这里后面没有分号“;”
#define  字符串  函数     //注意，这里后面没有分号“;”
```

具体一点如下：

```
#define  AA  1        //常量
#define  BB  (a+b+c)  //运算式
#define  C   add()    //函数
```

需要注意的时候，#define 后面没有分号“;”，因为它是 C 语言中的“预处理”的语句，不是单片机运行的程序指令语句。

【59.2　#define 的编译机制。】

#define 是属于“预编译”的指令，所谓“预编译”就是在“编译”之前就开始的准备工作。编译器在正式编译某个源代码的时候，先进行“预编译”的准备工作，对于#define 语句，编译器是直接把#define 要替换的内容先在“编辑层面”进行机械化替换，这个“机械化替换”纯粹是字符串的替换，可以理解成 word 办公软件的“替换”编辑功能。比如以下程序：

```
#define A 3
#define B (2+6)   //有括号
#define C  2+6     //无括号
unsigned long x=3;
unsigned long a;
unsigned long b;
unsigned long c;
void main() //主函数
{
    a=x*A;
    b=x*B;
    c=x*C;
```

```
    while(1)
    {
    }
}
```

经过编译器“预编译”的“机械化替换”后，等效于以下代码：

```
unsigned long x=3;
unsigned long a;
unsigned long b;
unsigned long c;
void main() //主函数
{
    a=x*3;
    b=x*(2+6);
    c=x*2+6;
    while(1)
    {
    }
}
```

【59.3　#define 在常量上的“一键替换”功能。】

上一节讲 const（或 code）的时候，举了一个“阀值”常量的例子，这个例子可以用#define 来替换等效。比如，原来 const(或 code)的例子如下：

```
code unsigned char Cu8Level=90;  //需要调整“阀值”时，只需更改一次这里的“90”这个数值。

unsigned char a;
unsigned char b;
void main() //主函数
{
    if(89>=Cu8Level)  //大于或者等于阀值，就输出 1。
    {
       a=1;
    }
    else  //否则输出 0。
    {
       a=0;
    }
    if(95>=Cu8Level)  //大于或者等于阀值，就输出 1。
    {
       b=1;
```

```
    }
    else  //否则输出 0。
    {
       b=0;
    }
    while(1)
    {
    }
}
```

上述程序现在用#define 来替换，等效如下：

```
#define Cu8Level 90  //需要调整“阀值”时，只需更改一次这里的“90”这个数值。

unsigned char a;
unsigned char b;
void main() //主函数
{
    if(89>=Cu8Level)  //大于或者等于阀值，就输出 1。
    {
       a=1;
    }
    else  //否则输出 0。
    {
       a=0;
    }
    if(95>=Cu8Level)  //大于或者等于阀值，就输出 1。
    {
       b=1;
    }
    else  //否则输出 0。
    {
       b=0;
    }
    while(1)
    {
    }
}
```

【59.4　#define 在运算式上的“一键替换”功能。】

#define 在运算式上应用的时候，有一个地方要特别注意，就是必须加小括号“()”，否则容易出错。因为#define 的替换是很“机械呆板”的，它只管“字符编辑层面”的机械化替换，举一个例子如下：

```
#define B (2+6)   //有括号
#define C  2+6     //无括号
unsigned long x=3;
unsigned long b;
unsigned long c;
void main() //主函数
{
    b=x*B;  //等效于 b=x*(2+6)，最终运算结果 b 等于 24。因为 3 乘以 8（2 加上 6 等于 8）。
    c=x*C;  //等效于 c=x*2+6，   最终运算结果 c 等于 12。因为 3 乘以 2 等于 6，6 再加 6 等于 12。
    while(1)
    {
    }
}
```

上述例子中，“有括号”与“没括号”的运算结果差别很大，第一个是 24，第二个是 12。具体的分析已经在源代码的注释了。

【59.5　#define 在函数上的“一键替换”功能。】

#define 的应用很广，也可以应用在函数的“替换”上。例子如下：

```
void add(void);  //函数的声明。
void add(void)   //函数的定义。
{
   a++;
}

#define a_zi_jia  add()  //用字符串 a_zi_jia 来替代函数 add()。

unsigned long a=1;
void main() //主函数
{
    a_zi_jia;  //这里相当于调用函数 add()。
    while(1)
    {
    }
}
```

【59.6　#define 在常量后面添加 U 或者 L 的特殊写法。】

有些初学者今后可能在工作中遇到#define 以下这种写法：

```
#define  字符串  常量U
#define  字符串  常量L
```

具体一点如下：

```
#define  AA  6U
#define  BB  6L
```

常量加后缀“U”或者“L”有什么含义呢？字面上理解，U表示该常量是无符号整型unsigned int;L表示该常量是长整型long。但是在实际应用中这样“多此一举”地去强调某个常量的数据类型有什么意义呢？我自己私下也做了一些测试，目前我本人暂时还没有发现这个秘密的答案。所以对于这个问题，初学者现在只要知道这种写法在语法上是合法的就可以，至于它背后有什么玄机，有待大家今后更深的发掘。

【59.7　#define省略常量的特殊写法。】

有些初学者今后在多文件编程中，在某些头文件.h中，会经常遇到以下这类代码：

```
#ifndef _AAA_
#define _AAA_
#endif
```

其中第2行代码“#define _AAA_”后面居然没有常量，这样子的写法也行，到底是什么意思？在这类写法中，当字符串“_AAA_”后面省略了常量的时候，编译器默认会给_AAA_添加一个“非0”的常量，也许是1或者其它“非0”的值，多说一句，所谓“非0”值就是“肯定不是0”。上述代码等效于：

```
#ifndef _AAA_
#define _AAA_ 1  //编译器会在这类默认添加一个1或者其它“非0”的常量
#endif
```

这个知识点大家只要先有一个感性的认识即可，暂时不用深入了解。

【59.8　例程练习和分析。】

现在编一个练习程序来熟悉#define的用法。

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

    //第1个：常量的例子
    #define Cu8Level 90  //需要调整“阀值”时，只需更改一次这里的“90”这个数值。
    unsigned char a;
    unsigned char b;

    //第2个：运算式的例子
    #define C (2+6)   //有括号
    #define D  2+6    //无括号
    unsigned char x=3;
```

```
    unsigned char c;
    unsigned char d;

    //第 3 个：函数的例子
    unsigned char e=1;
    void add(void);
    void add(void)
    {
       e++;
    }
    #define a_zi_jia  add()  //用字符串 a_zi_jia 来替代函数 add()。


void main() //主函数
{

    //第 1 个：常量的例子
    if(89>=Cu8Level)  //大于或者等于阀值，就输出 1。
    {
       a=1;
    }
    else  //否则输出 0。
    {
       a=0;
    }
    if(95>=Cu8Level)  //大于或者等于阀值，就输出 1。
    {
       b=1;
    }
    else  //否则输出 0。
    {
       b=0;
    }


    //第 2 个：运算式的例子
    c=x*C;  //等效于 c=x*(2+6)，最终运算结果 c 等于 24。因为 3 乘以 8（2 加上 6 等于 8）。
    d=x*D;  //等效于 d=x*2+6,   最终运算结果 d 等于 12。因为 3 乘以 2 等于 6，6 再加 6 等于 12。


    //第 3 个：函数的例子
    a_zi_jia;  //这里相当于调用函数 add()。e 从 1 自加到 2。
    a_zi_jia;  //这里相当于调用函数 add()。e 从 2 自加到 3。
```

```
    View(a);  //把第 1 个数 a 发送到电脑端的串口助手软件上观察。
    View(b);  //把第 2 个数 b 发送到电脑端的串口助手软件上观察。
    View(c);  //把第 3 个数 c 发送到电脑端的串口助手软件上观察。
    View(d);  //把第 4 个数 d 发送到电脑端的串口助手软件上观察。
    View(e);  //把第 5 个数 e 发送到电脑端的串口助手软件上观察。

    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:24
十六进制:18
二进制:11000

第 4 个数
十进制:12
十六进制:C
二进制:1100

第 5 个数
十进制:3
十六进制:3
二进制:11
```

分析：

a 为 0。

b 为 1。
c 为 24。
d 为 12。
e 为 3。

【59.9　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。