第十二节：变量的定义和赋值。

【12.1　学习 C 语言的建议和方法。】

先提一些学 C 语言的建议和方法，帮大家删繁就简，去掉一些初学者常见的思想包袱。现阶段我们的学习是使用单片机，把单片机当做一个成品，把单片机当做一个忠诚的士兵，学习 C 语言就是学习如何使用单片机，如何命令单片机，如何让单片机听懂我们的话并且听我们指挥。单片机内部太细节的构造原理暂时不用过多去关注，只要知道跟我们使用相关的几个特征就可以，这样初学者的学习包袱就没有那么重，就可以把重点放在使用上的，而不是好奇于根本原理的死磕到底。学 C 语言跟学习英语的性质是一样的，都是在学习一门外语，只是 C 语言比英语的语法要简单很多，非常容易上手，词汇量也没有英语那么多，C 语言常用单词才几十个而已。学习任何一门语言的秘诀在于练习，学习 C 语言的秘诀是多在单片机上练习编程。本教程后面几乎每个章节都有例程，这个例程很重要，初学者即使看懂了，我也强烈建议要把“C 语言学习区域”的那部分代码亲自上机敲键盘练习一遍，并且看看实验现象是否如你所愿。

【12.2　变量定义和赋值的感性认识。】

这些年我用过很多单片机，比如 51，PIC，LPC17 系列，STM8，STM32 等单片机。尽管各类单片机有一些差异，但是在选单片机时有 3 个参数我们一定会关注的，它们分别是：工作频率，数据存储器 RAM，程序存储器 ROM。工作频率跟晶振和倍频有关，决定了每条指令所要损耗的时间，从而决定了运算速度。RAM 跟代码里所定义变量的数量有关。ROM 跟程序代码量的大小有关。程序是什么？程序就是由对象和行为两者构成的。对象就是变量，就是变量的定义，就是 RAM，RAM 的大小决定了一个程序允许的对象数量。行为就是赋值，判断，跳转，运算等语法，就是 ROM，ROM 的大小决定了一个程序允许的行为程度。本节的标题是“变量的定义和赋值”，其中“定义”就是对象，“赋值”就是行为。

【12.3　变量的定义。】

变量的定义。一个程序最大允许有多少个对象，是由 RAM 的字节数决定的(字节是一种单位，后面章节会讲到)。本教程的编译环境是以 AT89C52 芯片为准，AT89C52 这个单片机有 256 个字节的 RAM，但是并不意味着程序就一定要全部占用这些 RAM。程序需要占用多少 RAM，完全是根据程序的实际情况来决定，需要多少就申请多少。这里的“对象”就是变量，这里的“申请”就是变量的定义。

定义变量的关键字。常用有 3 种容量的变量，每种变量的取值范围不一样。第一种是”unsigned char”变量，取值范围从 0 到 255，占用 RAM 一个字节，比喻成一房一厅。第二种是”unsigned int”变量，取值范围从 0 到 65535，占用 RAM 两个字节，比喻成两房一厅。第三种是“unsigned long”变量，取值范围从 0 到 4294967295，占用 RAM 四个字节，比喻成四房一厅。unsigned char,unsigned int 和 unsigned long 都是定义变量的关键字，所谓关键字也可以看成是某门外语的单词，需要大家记忆的，当然不用死记硬背，只要多上机练习就自然熟记于心，出口成章。多说一句，上述的变量范围是针对本教程所用的单片机，当针对不同的单片机时上述变量的范围可能会有一些小差异，比如在 stm32 单片机中，unsigned int 的字节数就不是两个字节，而是四个字节，这些都是由所选的编译器决定的，大家暂时有个初步了解就可以。

定义变量的语法格式。定义变量的语法格式由 3 部分组成：关键字，变量名，分号。比如：

```
unsigned char a;
```

其中 unsigned char 就是关键字，a 就是变量名，分号”;”就是一条语句的结束符号。

变量名的命名规则。变量名的第一个字符不能是数字，必须是字母或者下划线，字母或者下划线后面可以带数字，一个变量名之间的字符不能带空格，两个独立变量名之间也不能用空格隔开（但是两个独立变量名之间可以用逗号隔开）。变量名不能跟编译器已征用的关键字重名，不能跟函数名重名，这个现象跟古代

要求臣民避讳皇帝的名字有点像。哪些名字是合法的，哪些名字是不合法的？现在举一些例子说明：

```
unsigned char 3a; //不合法，第一个字符不能是数字。
unsigned char char; //不合法，char 是编译器已征用的关键字。
unsigned char a b; //不合法，ab 是一个变量名，a 与 b 的中间不能有空格。
unsigned char a,b; //合法，a 和 b 分别是一个独立的变量名，a 与 b 的中间可以用逗号隔开。
unsigned char a; //合法。
unsigned char abc; //合法。
unsigned char _ab; //合法。
unsigned char _3ab; //合法。
unsigned char a123; //合法。
unsigned char a12ced; //合法。
```

定义变量与 RAM 的内在关系。当我们定义一个变量时，相当于向单片机申请了一个 RAM 空间。C 编译器会自动为这个变量名分配一个 RAM 空间，每个字节的 RAM 空间都有一个固定唯一的地址。把每个字节的 RAM 空间比喻成房间，这个地址就是房号。地址是纯数字编号，不利于我们记忆，C 语言编译器为了降低我们的工作难度，不用我们记每个变量的地址，只需要记住这个变量的名称就可以了。操作某个变量名，就相当于操作某个对应地址的 RAM 空间。变量名与对应地址 RAM 空间的映射关系是 C 编译器暗中悄悄帮我们分配好的。比如：

```
unsigned char a;    //a 占用一个字节的 RAM 空间，这个空间的地址由 C 编译自动分配。
unsigned char b;    //b 占用一个字节的 RAM 空间，这个空间的地址由 C 编译自动分配。
unsigned char c;    //c 占用一个字节的 RAM 空间，这个空间的地址由 C 编译自动分配。
```

上述 a,b,c 三个变量各自占用一个字节的 RAM 空间，同时被 C 编译器分配了 3 个不同的 RAM 空间地址。

变量定义的初始化。变量定义之后，等于被 C 编译器分配了一个 RAM 空间，那么这个空间里面存储的数据是什么？如果没有刻意给它初始化，RAM 空间里面存储的数据是不太确定的，是默认的。有些场合，需要在给变量分配 RAM 空间时就给它一个固定的初始值，这就是变量定义的初始化。变量初始化的语法格式由 3 部分组成：关键字，变量名赋值，分号。比如：

```
unsigned char a=9;
```

其中 unsigned char 就是关键字。

其中 a=9 就是变量名赋值。a 从被 C 编译器分配 RAM 空间那一刻起，就默认是预存了一个 9 的数据。

分号“;”就是一条语句的结束符号。

【12.4　变量的赋值。】

赋值语句的含义。把右边对象的内容复制一份给左边对象。赋值语句有一个很重要的特性，就是覆盖性，左边对象原来的内容会被右边对象复制过来的新内容所覆盖。比如，左边对象是变量 a，假设原来 a 里面存的数据是 3，右边对象是数据 6，执行赋值语句后，会把右边的 6 赋值给了对象 a，那么 a 原来的数据 3 就被覆盖丢失了，变成了 6。

赋值语句的格式。赋值语句的语法格式由 4 部分组成：左边对象，关键字，右边对象，分号。比如：

```
a=b;
```

其中 a 就是左边对象。

其中“=”就是关键字。写法跟我们平时用的等于号是一样，但是在 C 语言里不是等于的意思，而是代表赋值的意思，它是代表中文含义的“给”，而不是用于判断的“等于”，跟等于号是两码事（C 语言的等于号是“==”，这个后面章节会讲到）。

其中 b 就是右边对象。

其中分号“;”代表一条语句的结束符。

赋值语句与 ROM 的关系。赋值语句是行为的一种，所以编译会把赋值这个行为翻译成对应的指令，这些指令在下载程序时最终也是以数据的形式存储在 ROM 里，指令也是以字节为单位(字节是一种单位，后面章节会讲到)。本教程的编译环境是以 AT89C52 芯片为准，AT89C52 这个单片机有 8K 的 ROM 容量，也就是有 8192 个字节的 ROM（8 乘以 1024 等于 8192），但是并不意味着程序就一定要全部占用这些 ROM。程序需要占用多少 ROM，完全是根据程序的行为程度决定，也就是通常所说的你的程序容量有多大，有多少行代码。多说一句，在单片机或者我们常说的计算机领域里，存储容量是以字节为单位，而每 K 之间的进制不是我们日常所用的 1000，而是 1024，所以刚才所说的 8K 不是 8000，而是 8192，这个是初学者很容易迷惑的地方。刚才提到，赋值语句是行为，凡是程序的行为指令都存储在单片机的 ROM 区。C 编译器会把一条赋值语句翻译成对应的一条或者几条机器码，机器码指令也是以字节为单位的。下载程序的时候，这些机器码就会被下载进单片机的 ROM 区。比如以下这行赋值语句：

```
unsigned char a;
unsigned char b=3;
a=b;
```

经过 C 编译器编译后会生成以字节为单位的机器码。这些机器码记录着这些信息：变量 a 的 RAM 地址，变量 b 的 RAM 地址和初始化时的预存数据 3，以及把 b 变量的内容赋值给 a 变量的这个行为。所有这些信息，不管是“数据”还是“行为”，本质都是以“数据”（或称数字，数码都可以）的形式存储记录的，单位是字节。

【12.5　例程的分析和练习。】

接下来练习一个程序实例。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。本章节在“C 语言学习区域”练习的代码如下：

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

void main() //主函数
{
   unsigned char a;  //定义的变量 a 被分配了一个字节的 RAM 空间，保存的数据是不确定的默认值。
   unsigned char b;   //定义的变量 b 被分配了一个字节的 RAM 空间，保存的数据是不确定的默认值。
   unsigned char c;   //定义的变量 c 被分配了一个字节的 RAM 空间，保存的数据是不确定的默认值。
   unsigned char d=9; //定义的变量 d 被分配了一个字节的 RAM 空间，保存的数据被初始化成 9.

   b=3;  //把 3 赋值给变量 b，b 由原来不确定的默认数据变成了 3。
   c=b;  //把变量 b 的内容复制一份赋值给左边的变量 c，c 从不确定的默认值变成了 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。
   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:255
十六进制:FF
二进制:11111111

第 2 个数
十进制:3
十六进制:3
二进制:11

第 3 个数
十进制:3
十六进制:3
二进制:11

第 4 个数
十进制:9
十六进制:9
二进制:1001
```

分析：

第 1 个数 a 居然是 255，这个 255 从哪来？因为 a 我们一直没有给它初始值，也没有给它赋值，所以它是不确定的默认值，这个 255 就是所谓的不确定的默认值，是编译器在定义变量 a 时分配的，带有不确定的随机性，不同的编译器可能分配的默认值都会存在差异。根据我的经验，unsigned char 类型定义的默认值往往是 0 或者 255（255 是十六进制的 0xff，十六进制的内容后续章节会讲到）。