中国石油大学信息与控制工程学院

智能仪表开发

学时：授课：32学时、实验：16学时

主讲：廖明燕 鄢志丹
邮箱：liaomy@upc.edu.cn
TEL：18253262519

2012.04

第二章　单片机程序设计C51语言概述

2.1 C51语言概述

2.2 C51的数据类型

2.3 C51的运算量

2.4 C51的运算符

2.5 C51表达式语句及复合语句

2.6 C51程序基本结构与相关语句

2.7 C51函　数

2.8 C51构造数据类型

2.1 C语言概述与最简单的C程序

2.1.1 C语言的特点及程序结构

C语言的特点

1. 语言简洁、紧凑，使用方便、灵活。
2. 运算符丰富。
3. 数据结构丰富。具有现代化语言的各种数据结构。
4. 可进行结构化程序设计。
5. 可以直接对计算机硬件进行操作。
6. 生成的目标代码质量高，程序执行效率高。
7. 可移植性好。

C语言的程序结构

C语言程序采用函数结构，每个C语言程序由一个或多个函数组成，在这些函数中至少应包含一个主函数main()，也可以包含一个main()函数和若干个其它的功能函数。不管main()函数放于何处，程序总是从main()函数开始执行，执行到main()函数结束则结束。在main()函数中调用其它函数，其它函数也可以相互调用，但main()函数只能调用其它的功能函数，而不能被其它的函数所调用。

功能函数可以是C语言编译器提供的库函数，也可以是由用户定义的自定义函数。在编制C程序时，程序的开始部分一般是预处理命令、函数说明和变量定义等。

2.1.2 C语言与MCS-51单片机

用C语言编写MCS-51单片机程序与用汇编语言编写MCS－51单片机程序不一样。

用汇编语言编写MCS－51单片机程序必须要考虑其存储器结构，尤其必须考虑其片内数据存储器与特殊功能寄存器的使用以及按实际地址处理端口数据。

用C语言编写的MCS－51单片机应用程序，则不用像汇编语言那样须具体组织、分配存储器资源和处理端口数据，但在C语言编程中，对数据类型与变量的定义，必须要与单片机的存储结构相关联，否则编译器不能正确地映射定位。

- 用C语言编写单片机应用程序与标准的C语言程序也有相应的区别：

C语言编写单片机应用程序时，需根据单片机存储结构及内部资源定义相应的数据类型和变量，而标准的C语言程序不需要考虑这些问题；

C51包含的数据类型、变量存储模式、输入输出处理、函数等方面与标准的C语言有一定的区别。其它的语法规则、程序结构及程序设计方法等与标准的C语言程序设计相同。

现在支持MCS-51系列单片机的C语言编译器有很多种，如American Automation、Avocet、BSO/TASKING、DUNFIELD SHAREWARE、KEIL/Franklin、IAR等。各种编译器的基本情况相同，但具体处理时有一定的区别。

2.1.3 C51程序结构

C51的语法规定、程序结构及程序设计方法都与标准的C语言程序设计相同，但C51程序与标准的C程序在以下几个方面不一样：

（1）C51中定义的库函数和标准C语言定义的库函数不同。标准的C语言定义的库函数是按通用微型计算机来定义的，而C51中的库函数是按MCS-51单片机相应情况来定义的；

（2）C51中的数据类型与标准C的数据类型也有一定的区别，在C51中还增加了几种针对MCS-51单片机特有的数据类型；

（3）C51变量的存储模式与标准C中变量的存储模式不一样，C51中变量的存储模式是与MCS-51单片机的存储器紧密相关；

（4）C51与标准C的输入输出处理不一样，C51中的输入输出是通过MCS-51串行口来完成的，输入输出指令执行前必须要对串行口进行初始化；

（5）C51与标准C在函数使用方面也有一定的区别，C51中有专门的中断函数。

2.2 C51的数据类型

C51的数据类型分为基本数据类型和组合数据类型，情况与标准C中的数据类型基本相同，其中char型与short型相同，float型与double型相同，另外，C51中还有专门针对于MCS-51单片机的特殊功能寄存器型和位类型。

一．字符型char

有signed char和unsigned char之分，默认为signed char。它们的长度均为一个字节，用于存放一个单字节的数据。

对于signed char，它用于定义带符号字节数据，其字节的最高位为符号位，“0”表示正数，“1”表示负数，补码表示，所能表示的数值范围是-128~+127；

对于unsigned char，它用于定义无符号字节数据或字符，可以存放一个字节的无符号数，其取值范围为0~255。unsigned char可以用来存放无符号数，也可以存放西文字符，一个西文字符占一个字节，在计算机内部用ASCII码存放。

二．int整型

分singed int和unsigned int。默认为signed int。它们的长度均为两个字节，用于存放一个双字节数据。对于signed int，用于存放两字节带符号数，补码表示，数的范畴为-32768~+32767。对于unsigned int，用于存放两字节无符号数，数的范围为0~65535。

三．long长整型

分singed long和unsigned long。默认为signed long。它们的长度均为四个字节，用于存放一个四字节数据。对于signed long，用于存放四字节带符号数，补码表示，数的范畴为-2147483648~+2147483647。对于unsigned long，用于存放四字节无符号数，数的范围为0~4294967295。

四．float浮点型

float型数据的长度为四个字节，格式符合IEEE-754标准的单精度浮点型数据，包含指数和尾数两部分，最高位为符号位，“1”表示负数，“0”表示正数，其次的8位为阶码，最后的23位为尾数的有效数位，由于尾数的整数部分隐含为“1”，所以尾数的精度为24位。

五．*指针型

指针型本身就是一个变量，在这个变量中存放的指向另一个数据的地址。这个指针变量要占用一定的内存单元，对不同的处理器其长度不一样，在C51中它的长度一般为1~3个字节。

六．特殊功能寄存器型

这是C51扩充的数据类型，用于访问MCS-51单片机中的特殊功能寄存器数据，它分sfr和sfr16两种类型。其中：

sfr为字节型特殊功能寄存器类型，占一个内存单元，利用它可以访问MCS-51内部的所有特殊功能寄存器；

sfr16为双字节型特殊功能寄存器类型，占用两个字节单元，利用它可以访问MCS-51内部的所有两个字节的特殊功能寄存器。

在C51中对特殊功能寄存器的访问必须先用sfr或sfr16进行声明。

七．位类型

这也是C51中扩充的数据类型，用于访问MCS-51单片机中的可寻址的位单元。在C51中，支持两种位类型：bit型和sbit型。它们在内存中都只占一个二进制位，其值可以是“1”或“0”。

其中：用bit定义的位变量在C51编译器编译时，在不同的时候位地址是可以变化的，而用sbit定义的位变量必须与MCS-51单片机的一个可以寻址位单元或可位寻址的字节单元中的某一位联系在一起，在C51编译器编译时，其对应的位地址是不可变化的。

C51编译器能够识别的基本数据类型：

基本数据类型	长度	取值范围
unsigned char	1字节	0~255
signed char	1字节	-128~+127
unsigned int	2字节	0~65535
signed int	2字节	-32768~+32767
unsigned long	4字节	0~4294967295
signed long	4字节	-2147483648~+2147483647
float	4字节	±1.175494E-38~±3.402823E+38
bit	1位	0或1
sbit	1位	0或1
sfr	1字节	0~255
sfr16	2字节	0~65535

在C51语言程序中，有可能会出现在运算中数据类型不一致的情况。C51允许任何标准数据类型的隐式转换，隐式转换的优先级顺序如下：

bit→char→int→long→float

signed→unsigned

也就是说，当char型与int型进行运算时，先自动对char型扩展为int型，然后与int型进行运算，运算结果为int型。C51除了支持隐式类型转换外，还可以通过强制类型转换符“（）”对数据类型进行人为的强制转换。

C51编译器除了能支持以上这些基本数据类型之外，还能支持一些复杂的组合型数据类型，如数组类型、指针类型、结构类型、联合类型等这些复杂的数据类型，在后面将相继介绍。

2.3 C51的运算量

2.3.1 常量

常量是指在程序执行过程中其值不能改变的量。在C51中支持整型常量、浮点型常量、字符型常量和字符串型常量。

一．整型常量

整型常量也就是整型常数，根据其值范围在计算机中分配不同的字节数来存放。在C51中它可以表示成以下几种形式：

十进制整数。如234、-56、0等。

十六进制整数。以0x开头表示，如0x12表示十六进制数12H。

长整数。在C51中当一个整数的值达到长整型的范围，则该数按长整型存放，在存储器中占四个字节，另外，如一个整数后面加一个字母L，这个数在存储器中也按长整型存放。如123L在存储器中占四个字节。

二．浮点型常量

浮点型常量也就是实型常数。有十进制表示形式和指数表示形式。

十进制表示形式又称定点表示形式，由数字和小数点组成。如 0.123、34.645等都是十进制数表示形式的浮点型常量。

指数表示形式为： [±] 数字 [.数字] e [±]数字

例如：123.456e-3、-3.123e2等都是指数形式的浮点型常量。

三．字符型常量

字符型常量是用单引号引起的字符，如‘a’、‘1’、‘F’等。可以是可显示的ASCII字符，也可以是不可显示的控制字符。对不可显示的控制字符须在前面加上反斜杠“\”组成转义字符。利用它可以完成一些特殊功能和输出时的格式控制。常用的转义字符如下表所示。

转义字符	含 义	ASCII码（十六进制数）
\ o	空字符（null）	00H
\ n	换行符（LF）	0AH
\ r	回车符（CR）	0DH
\ t	水平制表符（HT）	09H
\ b	退格符（BS）	08H
\ f	换页符（FF）	0CH
\ ‘	单引号	27H
\ ”	双引号	22H
\ \	反斜杠	5CH

四．字符串型常量

字符串型常量由双引号“”括起的字符组成。如“D”、“1234”、“ABCD”等。注意字符串常量与字符常量是不一样，一个字符常量在计算机内只用一个字节存放，而一个字符串常量在内存中存放时不仅双引号内的字符一个占一个字节，而且系统会自动的在后面加一个转义字符“\o”作为字符串结束符。因此不要将字符常量和字符串常量混淆，如字符常量‘A’和字符串常量“A”是不一样的。

2.3.2 变量

变量是在程序运行过程中其值可以改变的量。一个变量由两部分组成：变量名和变量值。

在C51中，变量在使用前必须对变量进行定义，指出变量的数据类型和存储模式。以便编译系统为它分配相应的存储单元。定义的格式如下：

[存储种类] 数据类型说明符 [存储器类型] 变量名1[=初值]，变量名2[初值]…；

一．数据类型说明符

在定义变量时，必须通过数据类型说明符指明变量的数据类型，指明变量在存储器中占用的字节数。可以是基本数据类型说明符，也可以是组合数据类型说明符，还可以是用typedef定义的类型别名。

在C51中，为了增加程序的可读性，允许用户为系统固有的数据类型说明符用typedef起别名，格式如下：

typedef c51固有的数据类型说明符 别名;

定义别名后，就可以用别名代替数据类型说明符对变量进行定义。别名可以用大写，也可以用小写，为了区别一般用大写字母表示。

【例】 typedef的使用。

```
typedef  unsigned  int   WORD;
typedef  unsigned  char  BYTE;
BYTE  a1=0x12;
WORD  a2=0x1234;
```

二．变量名

变量名是C51区分不同变量，为不同变量取的名称。在C51中规定变量名可以由字母、数字和下划线三种字符组成，且第一个字符必须为字母或下划线。变量名有两种：普通变量名和指针变量名。它们的区别是指针变量名前面要带“*”号。

六．位变量

在C51中，允许用户通过位类型符定义位变量。位类型符有两个：bit和sbit。可以定义两种位变量。

bit位类型符用于定义一般的可位处理位变量。它的格式如下：

bit 位变量名；

【例】 bit型变量的定义。

```
bit a1;      /*正确*/
bit a2;      /*正确*/
```

sbit位类型符用于定义在可位寻址字节或特殊功能寄存器中的位，定义时须指明其位地址，可以是位直接地址，可以是可位寻址变量带位号，也可以是特殊功能寄存器名带位号。格式如下：

sbit 位变量名=位地址;

如位地址为位直接地址，其取值范围为0x00~0xff；如位地址是可位寻址变量带位号或特殊功能寄存器名带位号，则在它前面须对可位寻址变量或特殊功能寄存器进行定义。字节地址与位号之间、特殊功能寄存器与位号之间一般用“^”作间隔。

- 【例】sbit型变量的定义:
- sbit OV=0xd2;
- sbit CY=oxd7;
- unsigned char bdata flag;
- sbit flag0=flag^0;
- sfr P1=0x90;
- sbit P1_0=P1^0;
- sbit P1_1=P1^1;
- sbit P1_2=P1^2;
- sbit P1_3=P1^3;
- sbit P1_4=P1^4;
- sbit P1_5=P1^5;
- sbit P1_6=P1^6;
- sbit P1_7=P1^7;

特别注意：

在C51中，为了用户处理方便，C51编译器把MCS-51单片机的常用的特殊功能寄存器和特殊位进行了定义，放在一个“reg51.h”或“reg52.h”的头文件中，当用户要使用时，只须要在使用之前用一条预处理命令#include <reg52.h>把这个头文件包含到程序中，然后就可使用殊功能寄存器名和特殊位名称。

2.3.3 存储模式

C51编译器支持三种存储模式：SMALL模式、COMPACT模式和LARGE模式。不同的存储模式对变量默认的存储器类型不一样。

（1）SMALL模式。SMALL模式称为小编译模式，在SMALL模式下，编译时，函数参数和变量被默认在片内RAM中，存储器类型为data。

（2）COMPACT模式。COMPACT模式称为紧凑编译模式，在COMPACT模式下，编译时，函数参数和变量被默认在片外RAM的低256字节空间，存储器类型为pdata。

（3）LARGE模式。LARGE模式称为大编译模式，在LARGE模式下，编译时函数参数和变量被默认在片外RAM的64K字节空间，存储器类型为xdata。

2.4 C51的运算符及表达式

2.4.1 赋值运算符

赋值运算符“=”，在C51中，它的功能是将一个数据的值赋给一个变量，如x=10。利用赋值运算符将一个变量与一个表达式连接起来的式子称为赋值表达式，在赋值表达式的后面加一个分号“；”就构成了赋值语句，一个赋值语句的格式如下：

变量=表达式;

执行时先计算出右边表达式的值，然后赋给左边的变量。例如：

```
x=8+9;    /*将8+9的值赋给变量x*/
x=y=5;    /*将常数5同时赋给变量x和y*/
```

在C51中，允许在一个语句中同时给多个变量赋值，赋值顺序自右向左。

2.4.2 算术运算符

C51中支持的算术运算符有:

+ 加或取正值运算符

- 减或取负值运算符

* 乘运算符

/ 除运算符

% 取余运算符

加、减、乘运算相对比较简单，而对于除运算，如相除的两个数为浮点数，则运算的结果也为浮点数，如相除的两个数为整数，则运算的结果也为整数，即为整除。如25.0/20.0结果为1.25，而25/20结果为1。

对于取余运算，则要求参加运算的两个数必须为整数，运算结果为它们的余数。例如：x=5%3，结果x的值为2。

2.4.3 关系运算符

C51中有6种关系运算符：

> 大于　　< 小于

>= 大于等于　　<= 小于等于

== 等于　　!= 不等于

关系运算用于比较两个数的大小，用关系运算符将两个表达式连接起来形成的式子称为关系表达式。关系表达式通常用来作为判别条件构造分支或循环程序。关系表达式的一般形式如下：

表达式1 关系运算符 表达式2

关系运算的结果为逻辑量，成立为真（1），不成立为假（0）。其结果可以作为一个逻辑量参与逻辑运算。例如：5>3，结果为真（1），而10==100，结果为假（0）。

注意：关系运算符等于“==”是由两个“=”组成。

2.4.4 逻辑运算符

C51有3种逻辑运算符：

|| 逻辑或　　&& 逻辑与　　！ 逻辑非

逻辑与，格式：

条件式1 && 条件式2

当条件式1与条件式2都为真时结果为真（非0值），否则为假（0值）。

逻辑或，格式：

条件式1 || 条件式2

当条件式1与条件式2都为假时结果为假（0值），否则为真（非0值）。

逻辑非，格式：

！条件式

当条件式原来为真（非0值），逻辑非后结果为假（0值）。当条件式原来为假（0值），逻辑非后结果为真（非0值）。

例如：若a=8，b=3，c=0，则！a为假，a && b为真，b && c为假。

2.4.5 位运算符

C51语言能对运算对象按位进行操作，它与汇编语言使用一样方便。位运算是按位对变量进行运算，但并不改变参与运算的变量的值。如果要求按位改变变量的值，则要利用相应的赋值运算。C51中位运算符只能对整数进行操作，不能对浮点数进行操作。C51中的位运算符有：

&	按位与
|	按位或
^	按位异或
~	按位取反
<<	左移
>>	右移

【例】设a=0x45=01010100B，b=0x3b=00111011B，则a&b、a|b、a^b、~a、a<<2、b>>2分别为多少？

a&b=00010000b=0x10。

a|b=01111111B=0x7f。

a^b=01101111B=0x6f。

~a=10101011B=0xab。

a<<2=01010000B=0x50。

b>>2=00001110B=0x0e。

2.4.6 复合赋值运算符

C51语言中支持在赋值运算符“=”的前面加上其它运算符，组成复合赋值运算符。下面是C51中支持的复合赋值运算符

+=	加法赋值	-+	减法赋值
*=	乘法赋值	/=	除法赋值
%=	取模赋值	&=	逻辑与赋值
|=	逻辑或赋值	^=	逻辑异或赋值
~=	逻辑非赋值	>>=	右移位赋值
<<=	左移位赋值		

复合赋值运算的一般格式如下：

变量 复合运算赋值符 表达式

它的处理过程：先把变量与后面的表达式进行某种运算，然后将运算的结果赋给前面的变量。其实这是C51语言中简化程序的一种方法，大多数二目运算都可以用复合赋值运算符简化表示。例如：a+=6相当于a=a+6；a*=5相当于a=a*5；b&=0x55相当于b=b&0x55；x>>=2相当于x=x>>2。

2.4.7 逗号运算符

在C51语言中，逗号“，”是一个特殊的运算符，可以用它将两个或两个以上的表达式连接起来，称为逗号表达式。逗号表达式的一般格式为：

表达式1，表达式2，……，表达式n

程序执行时对逗号表达式的处理：按从左至右的顺序依次计算出各个表达式的值，而整个逗号表达式的值是最右边的表达式（表达式n）的值。例如：x=(a=3,6*3)结果x的值为18。

2.4.8 条件运算符

条件运算符“？：”是C51语言中唯一的一个三目运算符，它要求有三个运算对象，用它可以将三个表达式连接在一起构成一个条件表达式。条件表达式的一般格式为：

逻辑表达式？表达式1：表达式2

其功能是先计算逻辑表达式的值，当逻辑表达式的值为真（非0值）时，将计算的表达式1的值作为整个条件表达式的值；当逻辑表达式的值为假（0值）时，将计算的表达式2的值作为整个条件表达式的值。例如：条件表达式max=(a>b)?a:b的执行结果是将a和b中较大的数赋值给变量max。

2.4.9 指针与地址运算符

指针是C51语言中的一个十分重要的概念，在C51中的数据类型中专门有一种指针类型。指针为变量的访问提供了另一种方式，变量的指针就是该变量的地址，还可以定义一个专门指向某个变量的地址的指针变量。

为了表示指针变量和它所指向的变量地址之间的关系，C51中提供了两个专门的运算符：

* 指针运算符

& 取地址运算符

指针运算符“*”放在指针变量前面，通过它实现访问以指针变量的内容为地址所指向的存储单元。例如：指针变量p中的地址为2000H，则*p所访问的是地址为2000H的存储单元，x=*p，实现把地址为2000H的存储单元的内容送给变量x。

取地址运算符“&”放在变量的前面，通过它取得变量的地址，变量的地址通常送给指针变量。例如：设变量x的内容为12H，地址为2000H，则&x的值为2000H，如有一指针变量p，则通常用p=&x，实现将x变量的地址送给指针变量p，指针变量p指向变量x，以后可以通过*p访问变量x。

2.5 表达式语句及复合语句

2.5.1 表达式语句

在表达式的后边加一个分号“；”就构成了表达式语句，如:

```
a=++b*9;
x=8；y=7;
++k;
```

可以一行放一个表达式形成表达式语句，也可以一行放多个表达式形成表达式语句，这时每个表达式后面都必须带“；”号，另外，还可以仅由一个分号“；”占一行形成一个表达式语句，这种语句称为空语句。

空语句在程序设计中通常用于两种情况：

（1）在程序中为有关语句提供标号，用以标记程序执行的位置。例如采用下面的语句可以构成一个循环。

```
repeat: ;
    ;
    goto repeat;
```

（2）在用while语句构成的循环语句后面加一个分号，形成一个不执行其它操作的空循环体。这种结构通常用于对某位进行判断，当不满足条件则等待，满足条件则执行。

【例】下面这段子程序用于读取8051单片机的串行口的数据，当没有接收到则等待，当接收到，接收数据后返回，返回值为接收的数据。

```
#include  <reg51.h>
char  getchar()
{
char  c;
while(!RI);              //当接收中断标志位RI为0则等待，
                                  当接收中断标志位为1则;
等待结束
c=SBUF;
RI=0;
return(c);
}
```

2.5.2 复合语句

复合语句是由若干条语句组合而成的一种语句，在C51中，用一个大括号“{ }”将若干条语句括在一起就形成了一个复合语句，复合语句最后不需要以分号“；”结束，但它内部的各条语句仍需以分号“；”结束。复合语句的一般形式为：

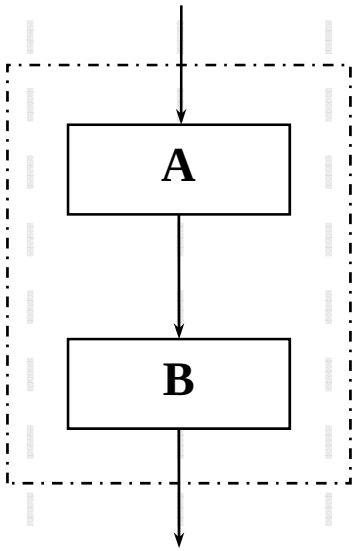
```
{
局部变量定义;
语句1;
语句2;
}
```

2.6 C51程序基本结构与相关语句

2.6.1 C51的基本结构

一．顺序结构

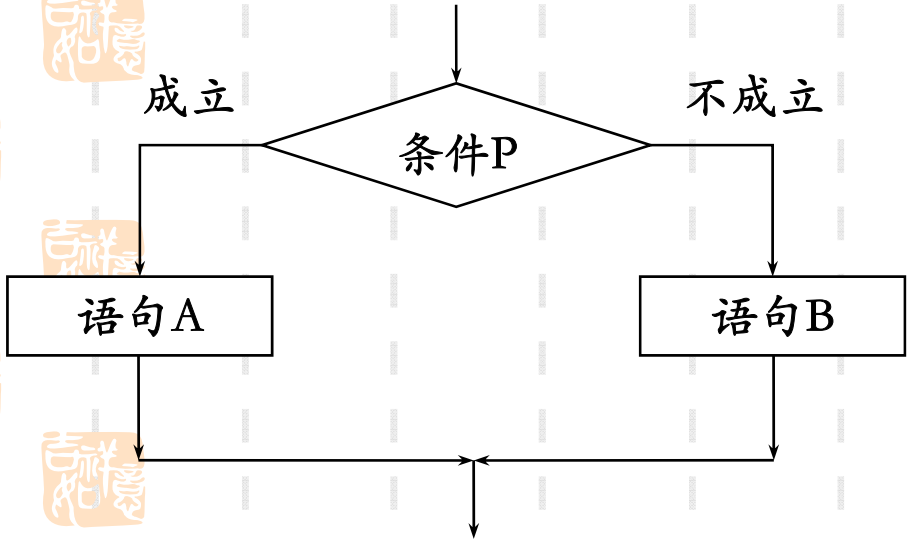
顺序结构是最基本、最简单的结构，在这种结构中，程序由低地址到高地址依次执行，如图给出顺序结构流程图，程序先执行A操作，然后再执行B操作。



顺序结构流程图

二．选择结构

选择结构可使程序根据不同的情况，选择执行不同的分支，在选择结构中，程序先都对一个条件进行判断。当条件成立，即条件语句为“真”时，执行一个分支，当条件不成立时，即条件语句为“假”时，执行另一个分支。



在C51中，实现选择结构的语句为if/else，if/else if语句。另外在C51中还支持多分支结构，多分支结构既可以通过if和else if语句嵌套实现，可用swith/case语句实现。

2.6.2 if语句

if语句是C51中的一个基本条件选择语句，它通常有三种格式:

（1）if（表达式）{语句；}

（2）if（表达式）{语句1；} else {语句2；}

（3）if（表达式1）{语句1；}
else if（表达式2）（语句2；）
else if（表达式3）（语句3；）
.....
else if（表达式n-1）（语句n-1；）
else {语句n}

【例】 if语句的用法。

（1）if (x!=y) printf(“x=%d,y=%d\n”,x,y);

执行上面语句时，如果x不等于y，则输出x的值和y的值。

（2）if (x>y) max=x;

else max=y;

（3）if (score>=90) printf(“Your result is an A\n”);

else if (score>=80) printf(“Your result is an B\n”);

else if (score>=70) printf(“Your result is an C\n”);

else if (score>=60) printf(“Your result is an D\n”);

else printf(“Your result is an E\n”);

执行上面语句后，能够根据分数score分别打出A、B、C、D、E五个等级。

2.6.3 switch/case语句

if语句通过嵌套可以实现多分支结构，但结构复杂。switch是C51中提供的专门处理多分支结构的多分支选择语句。它的格式如下：

```
switch （表达式）
{case 常量表达式1：{语句1；}break;
case 常量表达式2：{语句2；}break;
……
case 常量表达式n：{语句n；}break;
default：{语句n+1；}
```

说明如下：

（1）switch后面括号内的表达式，可以是整型或字符型表达式。

（2）当该表达式的值与某一“case”后面的常量表达式的值相等时，就执行该“case”后面的语句，然后遇到break语句退出switch语句。若表达式的值与所有case后的常量表达式的值都不相同，则执行default后面的语句，然后退出switch结构。

（3）每一个case常量表达式的值必须不同否则会出现自相矛盾的现象。

（4）case语句和default语句的出现次序对执行过程没有影响。

（5）每个case语句后面可以有“break”，也可以没有。有break语句，执行到break则退出switch结构，若没有，则会顺次执行后面的语句，直到遇到break或结束。

（6）每一个case语句后面可以带一个语句，也可以带多个语句，还可以不带。语句可以用花括号括起，也可以不括。

（7）多个case可以共用一组执行语句。

【例】 switch/case语句的用法。

对学生成绩划分为A~D，对应不同的百分制分数，要求根据不同的等级打印出它的对应百分数。可以通过下面的switch/case语句实现。

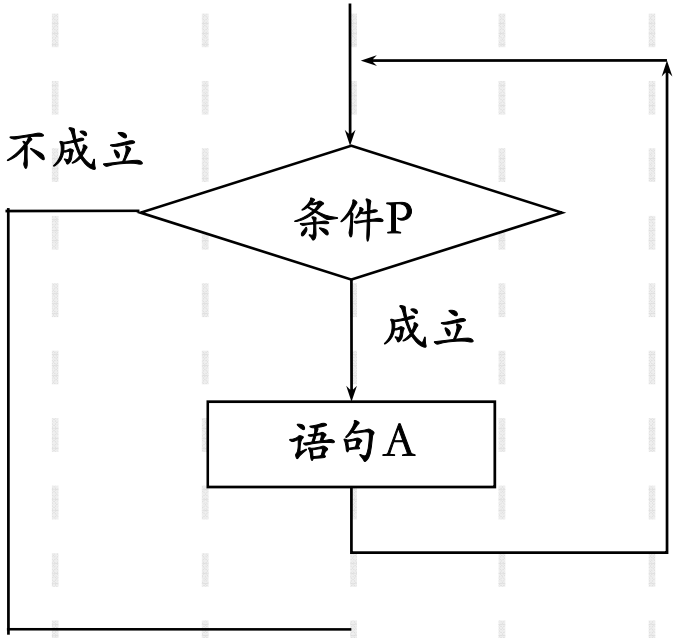
```
switch（grade）
{
case ‘A’；printf（”90~100\n”）；break;
case ‘B’；printf（”80~90\n”）；break;
case ‘C’；printf（”70~80\n”）；break;
case ‘D’；printf（”60~70\n”）；break;
case ‘E’；printf（”<60\n”）；break;
default; printf（”error”\n)
}
```

三．循环结构

循环结构就是能够使程序段重复执行的结构。循环结构又分为两种：当（while）型循环结构和直到（do...while）型循环结构。

（1）当型循环结构

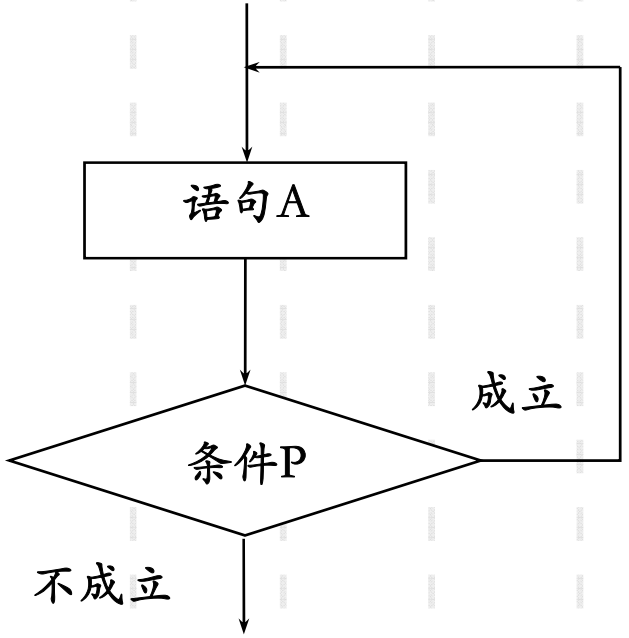
当型循环结构如右图所示，当条件P成立（为“真”）时，重复执行语句A，当条件不成立（为“假”）时才停止重复，执行后面的程序。



当型循环结构

（2）直到型循环结构

直到型循环结构如右图所示，先执行语句A，再判断条件P，当条件成立（为“真”）时，再重复执行语句A，直到条件不成立（为“假”）时才停止重复，执行后面的程序。



直到型循环结构

构成循环结构的语句主要有：while、do while、for、goto等。

2.6.4 while语句

while语句在C51中用于实现当型循环结构，它的格式如下：

```
while（表达式）
  {语句；} /*循环体*/
```

while语句后面的表达式是能否循环的条件，后面的语句是循环体。当表达式为非0（真）时，就重复执行循环体内的语句；当表达式为0（假），则中止while循环，程序将执行循环结构之外的下一条语句。它的特点是：先判断条件，后执行循环体。在循环体中对条件进行改变，然后再判断条件，如条件成立，则再执行循环体，如条件不成立，则退出循环。如条件第一次就不成立，则循环体一次也不执行。

```
#include  <reg52.h>  //包含特殊功能寄存器库
#include  <stdio.h>  //包含I/O函数库
void main(void)    //主函数
{
int  i,s=0;      //定义整型变量x和y
i=1;
while  (i<=100)    //累加1~100之和在s中
{
s=s+i;
i++;
}
printf( “1+2+3……+100=%d\n” ,s);
while(1);
}
```

【例】 下面程序是通过while语句实现计算并输出1~100的累加和。

程序执行的结果:
1+2+3……+100
=5050

2.6.5 do while语句

do while语句在C51中用于实现直到型循环结构，它的格式如下：

```
do
    {语句；}        /*循环体*/
while（表达式）；
```

它的特点是：先执行循环体中的语句，后判断表达式。如表达式成立（真），则再执行循环体，然后又判断，直到有表达式不成立（假）时，退出循环，执行do while结构的下一条语句。do while语句在执行时，循环体内的语句至少会被执行一次。

```
#include  <reg52.h>  //包含特殊功能寄存器库
#include  <stdio.h>   //包含I/O函数库
void main(void)       //主函数
{
int  i,s=0;                 //定义整型变量x和y
i=1;
do                    //累加1~100之和在s中
{
s=s+i;
i++;
}
while  (i<=100);
printf( “1+2+3……+100=%d\n” ,s);
while(1);
}
```

【例】 通过do while语句实现计算并输出1~100的累加和。

程序执行的结果:
1+2+3……+100
=5050

2.6.6 for语句

for（表达式1；表达式2；表达式3）

{语句；} /*循环体*/

for语句后面带三个表达式，它的执行过程如下：

（1）先求解表达式1的值。

（2）求解表达式2的值，如表达式2的值为真，则执行循环休中的语句，然后执行下一步（3）的操作，如表达式2的值为假，则结束for循环，转到最后一步。

（3）若表达式2的值为真，则执行完循环体中的语句后，求解表达式3，然后转到第四步。

（4）转到（2）继续执行。

（5）退出for循环，执行下面的一条语句。

【例】 用for语句实现计算并输出1~100的累加和。

```
#include  <reg52.h>    //包含特殊功能寄存器库
#include  <stdio.h>    //包含I/O函数库
void main(void)        //主函数
{
int  i,s=0;                    //定义整型变量x和y
for (i=1;i<=100;i++) s=s+i;    //累加1~100之和在s中
printf( “1+2+3……+100=%d\n” ,s);
while(1);
}
```

程序执行的结果：

1+2+3……+100=5050

2.6.7 循环的嵌套

在一个循环的循环体中允许又包含一个完整的循环结构，这种结构称为循环的嵌套。外面的循环称为外循环，里面的循环称为内循环，如果在内循环的循环体内又包含循环结构，就构成了多重循环。在C51中，允许三种循环结构相互嵌套。

【例】用嵌套结构构造一个延时程序。

```
void  delay(unsigned  int  x)
{
unsigned  char j;
while(x--)
{for (j=0;j<125;j++);}
}
```

2.6.8 break和continue语句

break和continue语句通常用于循环结构中，用来跳出循环结构。但是二者又有所不同，下面分别介绍。

1．break语句

前面已介绍过用break语句可以跳出switch结构，使程序继续执行switch结构后面的一个语句。使用break语句还可以从循环体中跳出循环，提前结束循环而接着执行循环结构下面的语句。它不能用在除了循环语句和switch语句之外的任何其它语句中。

【例】下面一段程序用于计算圆的面积，当计算到面积大于100时，由break语句跳出循环。

```
for (r=1; r<=10; r++)
{
area=pi*r*r;
if (area>100) break;
printf("%f\n", area);
}
```

2．continue语句

continue语句用在循环结构中，用于结束本次循环，跳过循环体中continue下面尚未执行的语句，直接进行下一次是否执行循环的判定。

continue语句和break语句的区别在于：continue语句只是结束本次循环而不是终止整个循环；break语句则是结束循环，不再进行条件判断。

【例】 输出100~200间不能被3整除的数。

```
for (i=100; i<=200; i++)
{
if (i%3= =0) continue;
printf("%d "; i);
}
```

在程序中，当i能被3整除时，执行continue语句，结束本次循环，跳过printf（）函数，只有能被3整除时才执行printf（）函数。

2.8 函　数

2.8.1 函数的定义

函数定义的一般格式如下：

函数类型 函数名（形式参数表） [reentrant][interrupt m][using n]

形式参数说明

{

局部变量定义

函数体

}

前面部件称为函数的首部，后面称为函数的尾部，格式说明：

1．函数类型

函数类型说明了函数返回值的类型。

2．函数名

函数名是用户为自定义函数取的名字以便调用函数时使用。

3．形式参数表

形式参数表用于列录在主调函数与被调用函数之间进行数据传递的形式参数。

【例】定义一个返回两个整数的最大值的函数max()。

```
int  max(int  x,int  y)
{
int  z;
z=x>y?x:  y;
return  (z)  ;
}
```

也可以用成这样：

```
int  max(x,y)
int  x,y;
{
int  z;
z=x>y?x:  y;
return  (z)  ;
}
```

return语句

return语句一般放在函数的最后位置，用于终止函数的执行，并控制程序返回调用该函数时所处的位置。返回时还可以通过return语句带回返回值。return语句格式有两种：

（1）return;

（2）return (表达式);

如果return语句后面带有表达式，则要计算表达式的值，并将表达式的值作为函数的返回值。若不带表达式，则函数返回时将返回一个不确定的值。通常我们用return语句把调用函数取得的值返回给主调用函数。

interrupt m修饰符（中断函数）

interrupt m是C51函数中非常重要的一个修饰符，这是因为中断函数必须通过它进行修饰。在C51程序设计中，当函数定义时用了interrupt m修饰符，系统编译时把对应函数转化为中断函数，自动加上程序头段和尾段，并按MCS-51系统中断的处理方式自动把它安排在程序存储器中的相应位置。

在该修饰符中，m的取值为0~31，对应的中断情况如下：

0——外部中断0

1——定时/计数器T0

2——外部中断1

3——定时/计数器T1

4——串行口中断

5——定时/计数器T2

其它值预留。

编写MCS-51中断函数注意如下：

（1）中断函数不能进行参数传递，如果中断函数中包含任何参数声明都将导致编译出错。

（2）中断函数没有返回值，如果企图定义一个返回值将得不到正确的结果，建议在定义中断函数时将其定义为void类型，以明确说明没有返回值。

（3）在任何情况下都不能直接调用中断函数，否则会产生编译错误。

（4）C51编译器对中断函数编译时会自动在程序开始和结束处加上相应的内容，具体如下：在程序开始处对ACC、B、DPH、DPL和PSW入栈，结束时出栈。中断函数未加using n修饰符的，开始时还要将R0~R1入栈，结束时出栈。

（5）C51编译器从绝对地址8m+3处产生一个中断向量，其中m为中断号，也即interrupt后面的数字。该向量包含一个到中断函数入口地址的绝对跳转。

编号	中断源	入口地址
0	外部中断 0	0003H
1	定时器／计数器中断 0	000BH
2	外部中断 1	0013H
3	定时器／计数器中断 1	001BH
4	串行口中断	0023H

（6）中断函数最好写在文件的尾部，并且禁止使用extern存储类型说明。防止其它程序调用。

【例】编写一个用于统计外中断0的中断次数的中断服务程序

```
extern  int  x;
void  int0()  interrupt 0
{
  x++;
}
```

2.8.2 函数的调用与声明

一．函数的调用

函数调用的一般形式如下：

函数名（实参列表）；

对于有参数的函数调用，若实参列表包含多个实参，则各个实参之间用逗号隔开。

按照函数调用在主调函数中出现的位置，函数调用方式有以下三种：

（1）函数语句。把被调用函数作为主调用函数的一个语句。

（2）函数表达式。函数被放在一个表达式中，以一个运算对象的方式出现。这时的被调用函数要求带有返回语句，以返回一个明确的数值参加表达式的运算。

（3）函数参数。被调用函数作为另一个函数的参数。

二．自定义函数的声明

在C51中，函数原型一般形式如下：

[extern] 函数类型 函数名（形式参数表）；

函数的声明是把函数的名字、函数类型以及形参的类型、个数和顺序通知编译系统，以便调用函数时系统进行对照检查。函数的声明后面要加分号。

如果声明的函数在文件内部，则声明时不用extern，如果声明的函数不在文件内部，而在另一个文件中，声明时须带extern，指明使用的函数在另一个文件中。

【例】函数的使用

```
#include  <reg52.h>      //包含特殊功能寄存器库
#include  <stdio.h>      //包含I/O函数库
int  max(int  x,int  y);    //对max函数进行声明
void main(void)           //主函数
{
int  a,b;
scanf( “please input a,b:%d,%d” ,&a,&b);
printf( “\n” );
printf( “max is:%d\n” ,max(a,b));
while(1);
}
int  max(int  x,int  y)
{int  z;
z=(x>=y?x:y);
return(z);
}
```

【例】 外部函数的使用

程序serial_initial.c

```
#include <reg52.h> //包含特殊功能寄存器库
#include <stdio.h> //包含I/O函数库
void serial_initial(void)   //主函数
{
SCON=0x52;     //串口初始化
TMOD=0x20;
TH1=0XF3;
TR1=1;
}
```

```
程序y1.c
#include  <reg52.h>  //包含特殊功能
                                寄存器库
#include  <stdio.h>  //包含I/O函数库
extern  serial_initial();
void  main(void)
{
int  a,b;
serial_initial();
scanf( “please input a,b:%d,%d” ,&a,&b);
printf( “\n” );
printf( “max is:%d\n” ,a>=b?a:b);
while(1);
}
```

2.8.3 函数的嵌套与递归

一．函数的嵌套

在一个函数的调用过程中调用另一个函数。C51编译器通常依靠堆栈来进行参数传递，堆栈设在片内RAM中，而片内RAM的空间有限，因而嵌套的深度比较有限，一般在几层以内。如果层数过多，就会导致堆栈空间不够而出错。

【例】 函数的嵌套调用

```
#include <reg52.h> //包含特殊功能寄存器库
#include <stdio.h> //包含I/O函数库
extern serial_initial();
int max(int a,int b)
{
int z;
z=a>=b?a:b;
return(z);
}
int add(int c,int d,int e,int f)
{
int result;
result=max(c,d)+max(e,f);    //调用函数max
return(result);
}
```

```
main()
{
int final;
serial_initial();
final=add(7,5,2,8);
printf("%d",final);
while(1);
}
```

2.9 C51构造数据类型

2.9.1 数组

一．一维数组

一维数组只有一个下标，定义的形式如下：

数据类型说明符 数组名[常量表达式][={初值，初值……}]

各部分说明如下：

(1)“数据类型说明符”说明了数组中各个元素存储的数据的类型。

(2)“数组名”是整个数组的标识符，它的取名方法与变量的取名方法相同。

（3）“常量表达式”，常量表达式要求取值要为整型常量，必须用方括号“[]”括起来。用于说明该数组的长度，即该数组元素的个数。

（4）“初值部分”用于给数组元素赋初值，这部分在数组定义时属于可选项。对数组元素赋值，可以在定义时赋值，也可以定义之后赋值。在定义时赋值，后面须带等号，初值须用花括号括起来，括号内的初值两两之间用逗号间隔，可以对数组的全部元素赋值，也可以只对部分元素赋值。初值为0的元素可以只用逗号占位而不写初值0。

例如：下面是定义数组的两个例子。

```
unsigned char x[5];
unsigned int y[3]={1,2,3};
```

第一句定义了一个无符号字符数组，数组名为x，数组中的元素个数为5。

第二句定义了一个无符号整型数组，数组名为y，数组中元素个数为3，定义的同时给数组中的三个元素赋初值，赋初值分别为1、2、3。

需要注意的是，C51语言中数组的下标是从0开始的，因此上面第一句定义的5个元素分别是：x[0]、x[1]、x[2]、x[3]、x[4]。第二句定义的3个元素分别是：y[0]、y[1]、y[2]。赋值情况为：y[0]=1；y[1]=2；y[2]=3。

【例】用数组计算并输出Fibonacci数列的前20项。

Fibonacci数列在数学和计算机算法中十分有用。Fibonacci数列是这样的一组数：第一个数字为0，第二个数字为1，之后每一个数字都是前两个数字之和。设计时通过数组存放Fibonacci数列，从第三项开始可通过累加的方法计算得到。

程序如下：

```
#include <reg52.h> //包含特殊功能寄存器库
#include <stdio.h> //包含I/O函数库
main()
{
```

```
int  fib[20],i;
fib[0]=0;
fib[1]=1;
for (i=2;i<20;i++)  fib[i]=fib[i-2]+fib[i-1];
for (i=0;i<20;i++)
{
if (i%5= =0) printf( “\n” );
printf( “%6d” ,fib[i]);
}
while(1);
}
```

程序执行结果：

0	1	1	2	3
5	8	13	21	34
55	89	144	233	377
610	987	1597	2584	4148

二．字符数组

用来存放字符数据的数组称为字符数组，它是C语言中常用的一种数组。字符数组中的每一个元素都用来存放一个字符，也可用字符数组来存放字符串。字符数组的定义下一般数组相同，只是在定义时把数据类型定义为char型。

例如：char string1[10];

char string2[20];

上面定义了两个字符数组，分别定义了10个元素和20个元素。

在C51语言中，字符数组用于存放一组字符或字符串，字符串以“\0”作为结束符，只存放一般字符的字符数组的赋值与使用和一般的数组完全相同。对于存放字符串的字符数组。既可以对字符数组的元素逐个进行访问，也可以对整个数组按字符串的方式进行处理。

【例】对字符数组进行输入和输出。

```
#include <reg52.h> //包含特殊功能寄存器库
#include <stdio.h> //包含I/O函数库
main()
{
char string[20];
serial_initial();
printf( “please type any character:” );
scanf( “%s” ,string);
printf( “%s\n” ,string);
while(1);
}
```

2.9.3 结构

结构是一种组合数据类型，它是将若干个不同类型的变量结合在一起而形成的一种数据的集合体。组成该集合体的各个变量称为结构元素或成员。整个集合体使用一个单独的结构变量名。

一．结构与结构变量的定义

结构与结构变量是两个不同的概念，结构是一种组合数据类型，结构变量是取值为结构这种组合数据类型的变量，相当于整型数据类型与整型变量的关系。对于结构与结构变量的定义有两种方法:

1．先定义结构类型再定义结构变量

结构的定义形式如下：

struct 结构名

{结构元素表};

结构变量的定义如下：

struct 结构名 结构变量名1，结构变量名2，……；

其中，“结构元素表”为结构中的各个成员，它可以由不同的数据类型组成。在定义时须指明各个成员的数据类型。

例如，定义一个日期结构类型date，它由三个结构元素year、month、day组成，定义结构变量d1和d2，定义如下：

```
struct date
{
int year;
char month,day;
}
struct date d1,d2;
```

2．定义结构类型的同时定义结构变量名

这种方法是将两个步骤合在一起，格式如下：

struct 结构名

{结构元素表} 结构变量名1，结构变量名2，……；

例如对于上面的日期结构变量d1和d2可以按以下格式定义：

```
struct date
{
int year;
char month,day;
}d1,d2;
```

二．结构变量的引用

结构元素的引用一般格式如下：

结构变量名.结构元素名

或

结构变量名->结构元素名

其中，“.”是结构的成员运算符，例如：d1.year表示结构变量d1中的元素year，d2.day表示结构变量d2中的元素day等。如果一个结构变量中结构元素又是另一个结构变量，即结构的嵌套，则需要用到若干个成员运算符，一级一级找到最低一级的结构元素，而且只能对这个最低级的结构元素进行引用，形如d1.time.hour的形式。

【例】输入3个学生的语文、数学、英语的成绩，分别统计他们的总成绩并输出。程序如下：

```
#include <reg52.h> //包含特殊功能寄存器库
#include <stdio.h> //包含I/O函数库
struct student
{
unsigned char name[10];
unsigned int chinese;
unsigned int math;
unsigned int english;
unsigned int total;
}p1[3];
```

```
main()
{
unsigned char i;
serial_initial();
printf( “input 3 studend name and result:\n” );
for (i=0;i<3;i++)
{
printf( “input name:\n” );
scanf( “%s” ,p1[i].name);
printf( “input result:\n” );
scanf( “%d,%d,%d” ,&p1[i].chinese,&p1[i].math,&p1[i].english);
}
```

```
for  (i=0;i<3;i++)
{
p1[i].total=p1[i].chinese+p1[i].math+p1[i].english;
}
for  (i=0;i<3;i++)
{
printf( “%s total is %d” ,p1[i].name,p1[i].total);
printf( “\n” );
}
while(1);
}
```

2.9.4 联合

前面介绍的结构能够把不同类型的数据组合在一起使用，另外，在C51语言中，还提供一种组合类型——联合，也能够把不同类型的数据组合在一起使用，但它与结构又不一样，结构中定义的各个变量在内存中占用不同的内存单元，在位置上是分开的，而联合中定义的各个变量在内存中都是从同一个地址开始存放，即采用了所谓的“覆盖技术”。这种技术可使不同的变量分时使用同一内存空间，提高内存的利用效率。

一．联合的定义

1．先定义联合类型再定义联合变量

定义联合类型，格式如下：

union 联合类型名

{成员列表};

定义联合变量，格式如下：

union 联合类型名 变量列表;

例如：

```
union data
{
float i;
int j;
char k;
}
union data a,b,c;
```

2．定义联合类型的同时定义联合变量

格式如下：

union 联合类型名

{成员列表}变量列表;

例如：

```
union data
{
float i;
int j;
char k;
}data a,b,c;
```

二．联合变量的引用

联合变量中元素的引用与结构变量中元素的引用格式相同，形式如下：

联合变量名.联合元素

或

联合变量名->联合元素

例如：对于前面定义的联合变量a、b、c中的元素可以通过下面形式引用。

```
a.i;
b.j;
c.k;
```

2.9.5 枚举

枚举数据类型是一个有名字的某些整型常量的集合。这些整型常量是该类型变量可取的所有的合法值。枚举定义时应当列出该类型变量的所有可取值。

枚举定义的格式与结构和联合基本相同，也有两种方法。

先定义枚举类型，再定义枚举变量，格式如下：

enum 枚举名 {枚举值列表};

enum 枚举名 枚举变量列表;

或在定义枚举类型的同时定义枚举变量，格式如下：

enum 枚举名 {枚举值列表}枚举变量列表;

例如：定义一个取值为星期几的枚举变量d1。

```
enum week
{Sun,Mon,Tue,Wed,Thu,Fri,Sat};
enum week d1;
```

或

```
enum week
{Sun,Mon,Tue,Wed,Thu,Fri,Sat} d1;
```

下面的示例将 FirstDayOfWeek 枚举的 Saturday 成员分配给变量 DayValue：

```
DayValue = FirstDayOfWeek.Saturday
```