模块 3

数据类型、变量、常量及格式声明

通过模块 2 的详细描述和学习，读者对 C 程序结构有了一定的具体认识，学习了变量的命名规则和定义规则等相关知识。下面进一步来学习并掌握 C 语言的数据类型、变量、常量及格式声明等基础知识。

在模块 1 中我们了解了 C 语言的数据类型，其中，基本类型是 C 语言数据类型的“地基”。本模块我们重点学习 C 语言的基本数据类型，主要是整型、实型（也称浮点型）和字符型，其他数据类型留到以后的相关模块中学习。

首先通过下例来重温模块 2 的相关知识点。

```
int a;
float b=3.14;
a=12;
printf("%d\n",a);
printf("%f\n",b);
```

(1)变量的定义方法：

```
数据类型 变量名;
```

如“int a;”定义了一个名为 a 的整型变量；“float b=3.14;”定义了一个浮点型变量 b，并初始化其值为实型常量 3.14。

(2)“a=12;”把常量 12 赋给变量 a。这里 12 是一个十进制整型常量。

(3)“printf("%d\n",a);”输出变量 a 的值。这里“%d”就是格式声明，声明输出格式为十进制整数(因为 a 是整型变量)。

(4)“printf("%f\n",b);”输出 b 的值。这里“%f”同样是格式声明，只不过输出的是浮点数(因为 b 是浮点型变量)。

由此可见，数据类型、变量、常量及格式声明是密不可分的关系，它们必须保持一致才能保证程序的正常运行。那么，何谓“保持一致”呢？即什么样数据类型的变量对应什么样的格式声明和常量。再具体举例来说，就是整型变量对应整型形式的格式声明和整型常量，实型变量对应实型形式的格式声明和实型常量。

下面就通过实例来具体学习各类型变量所对应的格式声明及其常量形式等相关知识。

3.1 常　量

3.1.1 常量的类型

常量可分为整型常量、实型常量、字符常量和字符串常量。

1. 整型常量

1)十进制整型常量

由正负号和数字 0～9 组成，如－36、25 等。

2)八进制整型常量

由正负号和数字 0～7 组成，且必须以 0 开头，如 025(对应十进制数为：$2\times8^1+5\times8^0=21$)等。

3)十六进制整型常量

由正负号、数字 0～9 和字母 A～F(不区分大小写)组成，且必须以 0x 开头，其中字符 A～F依次表示 10～15，如 0xd、0x15(对应十进制数为：$1\times16^1+5\times16^0=21$)等。

2. 实型常量

注意：小数点是实数的标志。

1)十进制实型常量

由正负号、数字 0～9 和小数点组成，如 1.23、－3.4 等。在 C 语言中，5/2 和 5.0/2 结果是不同的，前者被认为是两个整型数据相除，结果仍为整型，后者 5.0 为实型常量，则结果为实型。因此，5/2 结果为 2(取整)，5.0/2 结果为 2.5。

2)指数形式的实型常量

由正负号、数字 0～9、小数点和字母 E(不区分大小写)组成。其一般形式是 aEn。

其中，a 为 1～9 的实数，默认小数位为 6 位；E 为阶码标志，代表底数 10；n 为阶码，只能是十进制整数。例如：

0.11 写成指数形式就是 1.100000E－01，即 1.100000×10^{-1}。

125.6 写成指数形式就是 1.256000E＋02，即 1.256000×10^{2}。

3. 字符常量

1)普通字符常量

由单引号 '' 括起来的单个字符就是普通字符常量，如'A','a' 等。字符常量在内存中占 1 个字节(B)的内存空间。

2)转义字符

由一个斜杠“\”开头，不同于字符原有意义，故称为转义字符。

- \'：表示单引号字符，即 '。
- \"：表示双引号字符，即"。
- \\：表示反斜杠字符，即\。
- \a：响铃。
- \b：退格。
- \n：换行。

- \f：换页。
- \r：回车。
- \t：到下一个制表位。
- \ddd：表示 3 位八进制数所代表的字符，如\101＝$1\times8^2+0\times8^1+1\times8^0$＝65，对应字符 A(A 的 ASCII 码值为 65)。
- \xhh：表示 2 位十六进制数所代表的字符，如\x42＝$4\times16^1+2\times16^0$＝66，对应字符 B(B 的 ASCII 码值为 66)。

4. 字符串常量

由一对双引号括起来的字符序列就是字符串常量，如"English"，"A"等。系统会自动给字符串常量加一个结尾符 '\0'，因而字符串常量在内存中所占的空间为实际长度＋1，故 "A" 和 'A' 是完全不同的，"A"是字符串常量，在内存中占 2 B 的空间；'A' 是字符常量，在内存中占 1 B 的空间。

3.1.2　常量的进制转换

1. 数制的表示

(1)十进制(D)。例如，$(35)_{10}$或 35D。

(2)二进制(B)。例如，$(10000001)_2$ 或 10000001B。

(3)八进制(O)。例如，$(145.65)_8$ 或 145.65O。

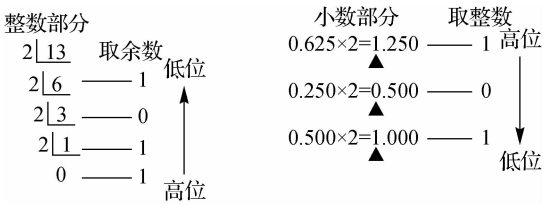
(4)十六进制(H)。例如，$(3C)_{16}$或 3CH。

2. 进制相互转换

(1)十进制转二进制(基数 2)。整数部分“除 2 取余”，小数部分“乘 2 取整”。

例如：

13.625D＝_____B?



所以，13.625D＝1101.101B。

以此类推，十进制转八进制(基数 8)、十六进制(基数 16)方法同上。

(2)二进制转十进制。例如：

1101.101B＝$1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$＝13.625D，即数码(如 1)乘以位权(如 2^3)之和。

(3)二进制与八进制互转。八进制范围内的有效数字与其二进制的对应如下：

0	1	2	3	4	5	6	7
000	001	010	101	100	101	110	111

①二进制转八进制——“三位一并”法，即整数部分自低(右)至高(左)三位一并，小数部分自高(左)至低(右)三位一并，不足部分补 0。

1100101.1101B=001 100 101.110 100B
1 4 5 6 4 O

即 1100101.1101B=145.64O。

②八进制转二进制——“一分为三”法。

011 010
65 7.32O=110 101 111.011 01B
110 101 111

(4)二进制与十六进制互转。十六进制范围内的有效数字与其二进制的对应如下：

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

测试

①二进制转十六进制——“四位一并”法，方法同二进制转八进制类似。

1011101010.11101B=0010 1110 1010.1110 1000B=2EA.E8H

②十六进制转二进制——“一分为四”法，方法同八进制转二进制类似。

3C9.7AH=0011 1100 1001.0111 1010B=11111001001.0111101B

3.2 变 量

基本类型的变量包括整型变量、实型变量、字符变量和枚举变量。本模块主要学习整型变量、实型变量及字符变量的相关知识。

在模块 2 中已经学习了变量的定义、变量的命名规则及变量的初始化(详见模块 2.2.6)，这里不再赘述。

3.2.1 整型变量

1. 基本型整型(int)和短整型(short)

例如，“int x_1=128;”声明一个名为 x_1 的 int 型变量并初始化其值为 128。

“short x_2=-1;”声明一个名为 x_2 的 short 型变量并初始化其值为-1。

(1)内存空间长度：2 B(16 bit)。

(2)所表数的范围：-32 768～32 767(-2^{15}～$2^{15}-1$)。

(3)对应的格式声明：

①%d 或%i：有符号十进制整数，若为正，则省略“+”号。

②%u：无符号十进制整数。

③%o：八进制整数。

④%x：十六进制整数。

【例 3-1】

```
int x_1=128;
printf("%d\n",x_1);   /*输出 128*/
printf("%i\n",x_1);   /*输出 128*/
```

```
printf("%u\n",x_1);   /*输出 128*/
printf("%o\n",x_1);   /*输出 200(2×8^2+0×8^1+0×8^0=128)*/
printf("%x\n",x_1);   /*输出 80 (8×16^1+0×8^0=128)*/
```

【例 3-2】

```
short x_2=-1;
printf("%d\n",x_2);   /*输出-1*/
printf("%i\n",x_2);   /*输出-1*/
printf("%u\n",x_2);   /*输出 65535*/
printf("%o\n",x_2);   /*输出 177777*/
printf("%x\n",x_2);   /*输出 ffff*/
```

(4)说明:关于负数的%u(无符号)、%o(八进制)、%x(十六进制)输出。

整数在内存中都是以补码形式存在的,因而对负数的运算必须先求其补码,然后按指定的格式输出,如-1(长度 2 B,16 bit,即 16 位)。

提示:①原码:最高位为符号位,1 表示负,0 表示正。正数的原码、反码、补码一致。②反码:保持最高位不变,其余按位取反。③补码:在反码的基础上加 1,即反码+1(二进制 1)。

原码	10000000 00000001	⇨	按%u 输出,最高位变数据位,输出 65535
反码	11111111 11111110		按%o 输出,三位一并,输出 177777
补码	11111111 11111111		按%x 输出,四位一并,输出 FFFF

思考:-2 的补码是 11111111 11111110,按%u 输出时,输出 65 534,那么-3、-4、……呢?

2. 无符号整型(unsigned int)和无符号短整型(unsigned short)

例如:

```
unsigned int x_3=16;
unsigned short x_4;
```

(1)内存空间长度:2 B(16 bit)。

(2)所表数的范围:0～65 535(0～$2^{16}-1$)。

(3)对应格式声明:%d、%u、%o、%x。

unsigned 的数据范围是 0～65 535,-1 按 %u 输出时为 65 535,即 65 536-1。由此可知:-2 按%u 输出对应 65 536-2=65 534;-3 按%u 输出对应 65 536-3=65 533;……以此类推。

3. 长整型(long)

例如:

```
long x_5;
```

(1)内存空间长度:4 B(32 bit)。

(2)所表数的范围:-2 147 483 648～2 147 483 647(-2^{31}～$2^{31}-1$)。

(3)对应格式声明:%ld、%lu。

4. 无符号长整型(unsigned long)

例如:

```
unsigned long x_5;
```

(1)内存空间长度:4 B(32 bit)。

(2)所表数的范围:0～4 294 967 295(0～$2^{32}-1$)。

(3)对应格式声明:%ld、%lu。

3.2.2 实型变量

1. 单精度浮点型(float)

例如:

```
float y_1=3.14;
```

声明一个名为 y_1 的单精度浮点型变量并初始化其值为 3.14。

(1)内存空间长度:4 B(32 bit)。

(2)所表数的范围:10^{-37}～10^{38}。有效数字:6～7 位。

(3)对应格式声明:

①%f:单精度浮点数,默认保留 6 位小数。

②%e:指数形式。

③%g:浮点数形式,输出时省略无意义的 0。

例如:

```
float y_1=13.040;
printf("%f\n",y_1);   /*输出 13.040000*/
printf("%e\n",y_1);   /*输出 1.304000e+01*/
printf("%g\n",y_1);   /*输出 13.04*/
```

2. 双精度浮点型(double)

例如:

```
double y_2;
```

(1)内存空间长度:8 B(64 bit)。

(2)所表数的范围:10^{-307}～10^{308}。有效数字:15～16 位。

(3)对应格式声明:%f、%e、%g。

3. 长双精度浮点数(long double)

例如:

```
long double y_3;
```

(1)内存空间长度:16 B(128 bit)。

(2)所表数的范围:$10^{-4\,931}$～$10^{4\,932}$。有效数字 18～19 位。

(3)对应格式声明:%lf、%le、%lg。

3.2.3 字符变量

1. 字符型(char)

例如:

```
char c='A';
```

声明一个名为 c 的字符变量并初始化其值为字符 A。

(1)内存长度:1 B(8 bit)。

(2)格式声明:

①%c:单个字符。

②%d:该字符的 ASCII 码值。

例如:

```
char c='A',ch='\102';
printf("%c\n",c); /*输出字符 A*/
prntf("%d\n",c); /*输出 A 的 ASCII 码值 65*/
prntf("%c\n",'c'); /*输出字符 c*/
printf("%c—%d\n",ch,ch); /*输出 B—66*/
```

注意:字符变量只能保存单个字符,不能把字符串常量赋给字符变量,如“char ch="A";”错误,因为"A"是字符串。

2. 字符串只能赋给字符数组或对字符指针变量初始化

“char ch[]="How are you";”声明一个字符数组 ch 并初始化其值为字符串"How are you",“char *p="Welcome to China!";”定义一个字符指针变量并初始化其值为字符串"Welcome to China!"的首地址。

例如:

```
char ch[]="How are you";
char *p="Welcome to China!";
printf("%s\n",ch); /*输出字符串"How are you"*/
printf("%s\n",p); /*输出字符串"Welcome to China!"*/
```

关于字符指针变量和字符数组,以后再讲。

3.3　格式声明补充说明

“—”为左对齐;无“—”则为右对齐。

(1)例如:

```
printf("%—10d",12);
```

10 为输出宽度,不足部分补空格。此处输出时为:左对齐,右补 8 个空格,即 12uuuuuuuu(12 占两个字符,u 表示空格);“%10d”则为右对齐,左补空格,即 uuuuuuuu12。

(2)例如:

```
printf("%#x",37);
```

#x 表示输出时加前缀 0x,输出 0x25;用于标注数字为十六进制。

(3)例如:

```
printf("%#o",37);
```

#o 表示输出时加前缀 0,输出 045;用于标注数字为八进制。

(4)例如:

```
printf("%10.3s","chinese");
```

表示右对齐,位宽 10,从左往右取前 3 个字符,输出 chi,左边补 7 个空格。

(5)例如：

```
printf("%10.3f",1234.67890);
```

表示右对齐，位宽 10，四舍五入保留 3 位小数，输出 1234.679，左边再补 2 个空格，即输出 uu1234.679。

(6)例如：

```
printf("%3d",1234);
```

指定了位宽，在位宽不足时按实际长度输出，输出 1234。

(7)例如：

```
float x; ……printf("%d",x);
```

在进行输入/输出时，最常见的错误就是格式声明和所定义的变量类型不一致，如“%d”格式不是 float 类型的格式，出错。

(8)例如：

```
printf("%%%.2f",3.145678);
```

输出%3.15。

3.4 附　　表

常量的类型见表 3-1，变量的数据类型见表 3-2。

表 3-1　常量的类型

类　型	分　类	常量说明	常量举例	格式声明
整型常量	十进制整型常量	由正负号和 0～9 数字组成	－36，128，－256	%d，%i，%u
	八进制整型常量	由正负号和 0～7 数字组成，且必须以 0 开头	034，025，017	%o
	十六进制整型常量	由正负号、数字 0～9 和字符 a～f 组成，且必须以 0x 开头。其中 a，b，c，d，e，f 分别表示 10，11，12，13，14，15	0xd，0xae，0x28	%x
实型常量（小数点是实数的标志）	十进制小数形式	由正负号、数字 0～9 和小数点组成	1.23，－2.001	%f，%g
	指数形式	由正负号、数字 0～9、小数点和字母 E(或 e)组成。其一般形式为：aEn。其中 a 为介于 1 和 10 之间的十进制小数，E 为阶码标志，n 为阶码，只能为十进制整数	2.1E－5，－2.8E2	%e，%g
字符常量	字符常量	由单引号括起来的单个字符，占 1 B 内存空间	'a'，'3'，'\101'	%c，%d
字符常量	转义字符	以“\”开头，不同于字符原有意义，故称为转义字符	\'　单引号字符 \"　双引号字符 \\　反斜杠字符\ \a　响铃 \b　退格 \ddd　3 位八进制代表的字符 \f　换页 \n　换行 \r　回车 \t　到下一个制表位 \xhh　2 位十六进制数代表的字符	
字符串常量	由一对双引号括起来的字符序列	如"a"，"ab"，"1"，"123"，字符串常量不能赋给单个变量，可以用字符数组来存放字符串常量		%s

表 3-2　变量的数据类型

类型	名称	数据类型简写（位宽）	字节数（Byte）	数的范围		变量定义示例	对应的输入/输出格式声明
整型变量	整型	int (16 bit)	2 B	−32 768～32 767，即-2^{15}～$(2^{15}-1)$		int x_1;	%d,%u,%o,%x 等
	短整型	short (16 bit)	2 B	−32 768～32 767，即-2^{15}～$(2^{15}-1)$		short y_1;	%d,%u,%o,%x 等
	无符号整型	unsigned int (16 bit)	2 B	0～65 535，即 0～$(2^{16}-1)$		unsigned int x;	%d,%u,%o,%x 等
	无符号短整型	unsigned short (16 bit)	2 B	0～65 535，即 0～$(2^{16}-1)$		unsigned short y;	%d,%u,%o,%x 等
	长整型	long (32 bit)	4 B	−2 147 483 648～2 147 483 647，即-2^{31}～$(2^{31}-1)$		long x_2;	%ld 等
	无符号长整型	unsigned long (32 bit)	4 B	0～4 294 967 295，即 0～$(2^{32}-1)$		unsigned long y_2;	%lu 等
实型变量	单精度浮点型	float (32 bit)	4 B	10^{-37}～10^{38}	有效数字 6～7	float x;	%f,%e,%g 等
	双精度浮点型	double (64 bit)	8 B	10^{-307}～10^{308}	15～16	double y;	%f,%e,%g 等
	长双精度浮点型	long double (128 bit)	16 B	$10^{-4\,931}$～$10^{4\,932}$	18～19	long double x_3;	%lf,%le,%lg 等
字符变量		char(8 bit)	1 B	字符的 ASCII 码值		char c;	%c,%d
枚举变量		enum (16 bit)	2 B	方法：enum 枚举名{枚举值表}变量名； enum weekday{Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday}a,b,c;			

类型名称	定义举例		所占空间计算
数组	一维数组	int a[5];	5×2=10 B
	二维数组	float a[2][3];	2×3×4=24 B
结构体（struct）	结构体变量 a 和 b：“struct student{char xm[8];int age; float score;}a,b;”或者“struct student{char xm[8];int age; float score;}; struct student a,b;”		1×8+2+4=14 B（结构体成员长度之和）
联合体（union）	联合体变量 a：“union student{char xm[8]; int class; float score;}a;”或者“union student{char xm[8]; int class; float score;}; union student a;”		取最大的联合体成员长度 8 B
文件(file)	FILE * fp（通过文件指针指向文件，对文件进行操作）		指针为 int 型，2 B

续表

类型名称			定义举例	所占空间计算
指针类型	指针变量	*	int * p1; float * p2; double * p3; char * p4;	指针固定长度为 2 B,C++中为 4 B

说明:int 是 C 语言的基本型,其内存长度 16 bit 软件环境下为 2 B,32 bit 下为 4 B。

3.5 自我测试

一、选择题

1. 下列变量定义及初始化正确的是(　　)。

A. short a_1=2.8;　　B. long enum=31415;

C. float jg=5;　　D. char xm="Jack";

2. 下列定义的变量中,在内存中所占空间最大的是(　　)。

A. long x_1　　B. float x_2　　C. double x_3　　D. unsigned long x_4

3. 字符串"\\a\10123\x66\a"所占的内存空间是(　　)B。

A. 8　　B. 9　　C. 10　　D. 11

4. 下列各数中最大的是(　　)。

A. 0x80　　B. 0174　　C. 11111011B　　D. 125D

5. 有变量声明"char come='g';",则"printf("%s","come");"的输出结果是(　　)。

A. 103　　B. g　　C. come　　D. s

二、根据前面所学的关于变量定义和格式控制等的相关知识,指出下面程序的错误,修改后上机调试

该段程序的功能是:根据 Lucy 输入的 x_1 变量值来计算圆的面积,输入数大于或等于 5,就取变量 x_2 为半径,否则就取变量 x_3 为半径计算圆的面积,最后输出:Lucy's result is:圆的面积。

```
1. include <stdio.h>
2. define N 3.14
3. int main()
4. {   char name="Lucy";
5.      int x_1,float x_2=3,x_3=4;short s;
6.      printf("Please input x_1\n");
7.      scanf("%d",x_1);
8.      if(x_1>=5)s=N*x_2*x_2;
9.      else s=N*x_3*x_3;
10.     printf("%c's result is:",name);
11.     printf("x_1=%f,s=%f\n",x_1,s);
12. }
```

三、指出下列 printf()函数的输出结果

```
printf("%u,%d",-1,-1);
printf("%x",10);
printf("%x",-1);
printf("%e",2.8);
printf("%c",'c');
printf("%c",97);
printf("%o",10);
printf("%x",10.00);
printf("%f",2.1);
printf("%g",2.8000);
printf("%d",'A');
printf("%s","abcd");
printf("\123\164\165\x64\x65\x6e\164");
int a=21;printf("%#x",a);
int b=21;printf("%#o",b);
char ch[15]="goodluck"; printf("%.4s",ch);
float P=3.1415926; printf("%5.4f",p);
```