西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

将简单数据引入C程序中

- 2、字符常量
- 3、变量
- 4、类型定义的修饰

2.2.3 字符常量

- 1. 用字面常量表示字符常量
  - 以一对单引号（‘ ’）括起来的一个字符表示字符常量，如‘A’、‘0’、‘&’等。
  - 字符常量表示的是一个字符，存储的是该字符的ASCII码值。例如‘A’表示英文字符A，数据值是65；‘2’表示数字字符2，数据值是50。

2.2.3 字符常量

- 单引号是字符常量的边界符，它只能包括一个字符，如‘AB’的写法就是错误的。

- 字符'2'和整数2的写法是有区别的，前者是字符常量，后者是整型常量，它们的含义和在内存中的存储形式完全是不相同的。

2.2.3 字符常量

- 2．用转义字符表示字符常量

  - 以反斜线（\）开头，后跟一个或几个字符序列表示的字符称为转义字符，如\n表示换行符。

  - 转义字符中的字符序列已转换成另外的含义，故称之为“转义”。如\n中的n不代表字母n而是换行符。

  - 采用转义字符可以表示ASCII字符集中不可打印或不方便输入的控制字符和其他特定功能的字符。

2.2.3 字符常量

表2-2 转义字符及其含义

转义字符形式	含 义	ASCII码值
\a	响铃符	7
\b	退格符	8
\f	进纸符，将光标位置移到下页开头	12
\n	换行符，将光标位置移到下一行开头	10
\r	回车符，将光标位置移到本行开头	13
\t	水平制表符，光标跳到下一个TAB位置	9
\v	垂直制表符	11
\'	单引号	39
\"	双引号	34
\\	反斜线	92
\?	问号	63

2.2.3 字符常量

续表2-2 转义字符及其含义

转义字符形式	含 义	ASCII码值
\0	空字符	0
\ooo	用1～3位八进制数ooo为码值所对应的字符	ooo（八进制）
\xhh	用1～2位十六进制数hh为码值所对应的字符	hh（十六进制）

\ooo和\xhh称为通用转义字符，其中ooo表示可以用1至3位八进制数作为码值表示一个ASCII字符，hh表示可以用1至2位十六进制数作为码值表示ASCII字符。

2.2.3 字符常量

- C语言规定通用转义字符在3位或不足3位的第一个非八进制数处结束，或在2位或不足2位的第一个非十六进制数处结束。

- 示例
  - \1234　　3位　\123和4
  - \128　　2位　\12和8
  - \19　　1位　\1和9
  - \9　　错误
  - \0xhh　　错误

2.2.3 字符常量

- 由于字符型数据在内存中只占用1个字节，即使按无符号处理其最大值也仅是255（八进制为377），因此ooo的数值范围为0～377（八进制），其他值使得字符型数据溢出。同理，hh的数值范围为0～FF。

2.2.3 字符常量

【例2.3】

转义字符的使用。

2.2.3 字符常量

例2.3

```
1 #include<stdio.h>
2 int main()
3 {
4   printf("ab c\t de\rf\tg\n");
5   printf("h\ti\b\bj k\n123\'\"\\\x41\102CDE\n");
6   return 0;
7 }
```

2.2.3 字符常量

例2.3

```
1 #include<stdio.h>
2 int main()
3 {
```

程序运行屏幕

```
f␣ ␣ ␣ ␣ ␣ ␣ ␣gde
h␣ ␣ ␣ ␣ ␣ ␣ j ␣k
123'"\ABCDE
```

2.2.4 字符串常量

- 以一对双引号（“ ”）括起来的零个或多个字符组成的字符序列称为字符串常量，ASCII字符集或多字节字符集（如汉字、日韩文字等）都可以组成字符串。
- 双引号是字符串常量的边界符，不是字符串的一部分，如果在字符串中要出现双引号应使用转义字符（\“）。

2.2.4 字符串常量

- 示例

```
""      //空字符串（0个字符）
" "     //包含一个空格的字符串
"Hello,World\n" //包含Hello,World 和 换行符的字符串
"xyz\101\x42"  //包含x y z A(101) B（x42)的字符串
"\\\'\"\n“
//包含反斜线（\\） 单引号（\'）和双引号（\"）的字符串
"\"a/b\" isn\'t a\\b"  //字符串"a/b" isn't a\b
```

2.2.4 字符串常量

- 字符串常量是数组的一种常量形式，请不要将字符串常量与字符常量混淆，二者相比有很大的区别，表现在：

- （1）边界符不同。

- （2）字符数不同。

- （3）在内存中的存储形式不同。

2.2.4 字符串常量

- 书写字符串常量时，不能从“…”中间换行，例如：

```
printf("C Programming
Language");
```

- 是错误的。

2.2.4 字符串常量

- C语言允许将两个相邻的仅由空格、TAB或换行分开的字符串常量，连接成一个新字符串常量。这使得可以用多行书写长的字符串常量，如写法

```
printf("C" " Programming"
" Language");
```

- 与写法

```
printf("C Programming Language");
```

- 效果完全相同。

2.2.5 符号常量

- 符号常量定义形式为：

```
#define 标识符 常量
```

- 其中#define是宏定义命令，作用是将标识符定义为常量值，在程序中所有出现该标识符的地方均用常量替换。

2.2.5 符号常量

【例2.4】

编程计算圆的周长和面积。

2.2.5 符号常量

例2.4

```
1 #include<stdio.h>
2 #define PI 3.1415926 //3.1415926即为圆周率π
3 int main()
4 {
5   double r=5.0;
6   printf("L=%f,S=%f\n",2*PI*r,PI*r*r); //PI替换为3.1415926
7   return 0;
8 }
```

2.3 变量

- 在程序运行期间其值可以改变的量称为变量（variable）。
- 变量实际上就是计算机中的一个内存单元。

2.3.1 变量的概念

- C语言规定变量应该有一个名字，用变量名代表内存单元。
- C语言通过定义变量时指定其数据类型来确定内存单元的大小，不同的数据类型有不同的数据形式和存储形式，需要一定数量（单位为字节）的内存单元。
- 除变量名和数据类型之外，变量还有地址、作用域、生命周期等属性。

2.3.2 定义变量

- C语言变量必须“先定义，后使用”，定义变量的一般形式是：

```
变量类型  变量名列表;
```

- 示例

```
double a , b , c , d; //定义变量
int i , j , k;        //一个定义定义多个int型变量
char m , n;           //不同类型需要多个定义
int a, char c;        //错误
```

2.3.3 使用变量

- 在变量定义的同时给变量一个初值，称为变量初始化（initialized），一般形式为：

```
变量类型  变量名=初值;

变量类型  变量名1=初值1，  变量名2=初值2，  ......;
```

2.3.3 使用变量

- 示例

```
double pi=3.1415926; //正确，初始化pi为3.1415926
int x , y , k=10;    //正确，可以只对部分变量初始化
int a=1 , b=1 , c=1; //正确，可以同时初始化多个变量
int d=a , e=a+b;     //错误，初值不能是变量或表达式
int m=n=z=5;         //错误，不能对变量连续初始化
```

2.3.3 使用变量

- 定义变量后，可以通过赋值语句为变量赋予新的数据，一般形式为：

```
变量名=表达式 ;
```

- 赋值后，无论变量原来的值是多少，都将被新值替代。

```
int k;
k=5;    //给k赋值5
...... //k保持不变
k=10;  //重新给k赋值10，k已改变不再是5
```

2.3.4 存储类别

- auto是变量默认的存储类别，称为自动变量；
- static是静态存储类别的变量，称为静态变量；
- register称为寄存器变量。

```
auto 类型 变量名[=初值], ......
register 类型 变量名[=初值], ......
static 类型 变量名[=初值], ......
extern 类型 变量名, ......
```

2.3.5 类型限定

- 1．const限定
- 在变量定义前加上const修饰，这样的变量称为只读变量（read-only variable）或常变量（constant variable）
- 它在程序运行期间不能被修改。其定义的一般形式为

```
const 变量类型 变量名列表;
```

2.3.5 类型限定

- 示例

```
int x;
const int i=6 , j=10;
x=i+1; //正确，可以使用const变量
i=10;  //错误，不可以给const变量赋值
j++;   //错误，不可以修改const变量
```

```
const int i=6; //正确
const int m;   //错误
```

2.3.5 类型限定

- const限定过的变量在编译过程中若发现有修改的操作时会报编译错误，从而“阻止”对变量的修改。

- 使用const限定强制实现对象最低访问权限，是现代软件开发的设计原则之一。

2.3.5 类型限定

- 2．volatile限定
- 在变量定义前加上volatile修饰，这样的变量称为隐式存取变量，它表示变量在程序运行期间会隐式地（不明显地）被修改。其定义的一般形式为：

```
volatile 变量类型 变量名列表;
```

- 在变量定义前加上volatile修饰，“阻止”编译器对这样的变量进行优化。

2.3.5 类型限定

- 示例

```
int x=5 , m , n;
volatile int y=6;
m=x*x; //两次读取x被编译器优化为只读一次，m是x的平方。
n=y*y; //不允许优化，则先取一次y，再取一次y，
       //若取之间y发生变化，n不一定是y的平方。
```


程序设计