# 目录

[目录 1](#_Toc13761031)

[第3章 C语言数据类型 3](#_Toc13761032)

[3.1 C语言的数据类型 3](#_Toc13761033)

[3.2 标识符 3](#_Toc13761034)

[3.2.1 标识符概念 3](#_Toc13761035)

[3.2.2 标识符的分类 3](#_Toc13761036)

[3.2.3 标识符的命名规则 3](#_Toc13761037)

[3.3 关键字 4](#_Toc13761038)

[3.3.1 关键字概念 4](#_Toc13761039)

[3.3.2 ANSI C 关键字列表 4](#_Toc13761040)

[3.4 常量 5](#_Toc13761041)

[3.4.1 常量概念 5](#_Toc13761042)

[3.4.2 常量分类 5](#_Toc13761043)

[3.5 变量 6](#_Toc13761044)

[3.5.1 变量概念 6](#_Toc13761045)

[3.5.2 变量声明 6](#_Toc13761046)

[3.5.3 变量初始化 6](#_Toc13761047)

[3.5.4 变量分类： 6](#_Toc13761048)

[3.6 整型数据 7](#_Toc13761049)

[3.6.1 整型数据类型的声明 7](#_Toc13761050)

[3.6.2 整型数据类型的存储 7](#_Toc13761051)

[3.6.3 整型常量 9](#_Toc13761052)

[3.6.4 整型数据的输入和输出 10](#_Toc13761053)

[3.7 字符型数据 11](#_Toc13761054)

[3.7.1 字符型变量声明 11](#_Toc13761055)

[3.7.2 字符型的存储 11](#_Toc13761056)

[3.7.3 字符常量 11](#_Toc13761057)

[3.7.4 转义字符 12](#_Toc13761058)

[3.7.5 字符类型数据输入/输出 scanf printf 13](#_Toc13761059)

[3.7.6 字符变量 14](#_Toc13761060)

[3.7.7 字符串常量 15](#_Toc13761061)

[3.8 浮点型 16](#_Toc13761062)

[3.8.1 浮点类型声明 16](#_Toc13761063)

[3.8.2 浮点型数据的存储 17](#_Toc13761064)

[3.8.3 浮点常量的表示方法 19](#_Toc13761065)

[3.8.4 浮点型数据输入/输出 19](#_Toc13761066)

[3.9 类型转换 20](#_Toc13761067)

[3.9.1 自动转换（隐式转换） 20](#_Toc13761068)

[3.9.2 强制类型转换（显式转换） 22](#_Toc13761069)

[3.10 格式化输出函数 （输出的本质都是字符） 22](#_Toc13761070)

[3.10.1 输出函数 22](#_Toc13761071)

[3.10.2 整型 22](#_Toc13761072)

[3.10.3 浮点型 23](#_Toc13761073)

[3.10.4 字符型 24](#_Toc13761074)

[3.10.5 输入函数 24](#_Toc13761075)

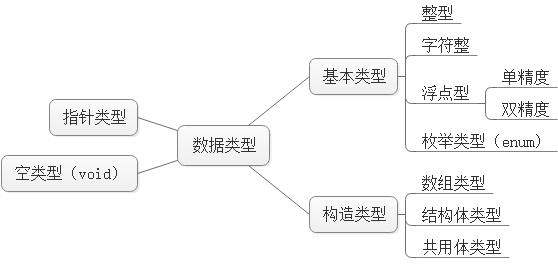
[3.10.6 转换说明符使用 25](#_Toc13761076)

# C语言数据类型

## C语言的数据类型

所谓数据类型是按被定义变量的性质，表示形式，占据存储空间的多少，构造特点来划分的。

在Ｃ语言中，数据类型可分为：基本数据类型，构造数据类型，指针类型，空类型四大类。



1. 基本数据类型：基本数据类型最主要的特点是，其值不可以再分解为其它类型。
2. 构造数据类型：构造数据类型是根据已定义的一个或多个数据类型使用构造的方法来定义的。构造类型的值可以分解成若干个“成员”或“元素”。每个“成员” 都是一个基本数据类型或又是一个构造类型。在 C 语言中，构造类型有以下几种：

* 数组类型
* 结构体类型
* 共用体（联合）类型

1. 指针类型：指针是一种特殊的类型，其值用来表示某个变量在内存储器中的地址。
2. 空类型：在调用函数值时，用于做不需要返回结果的函数返回值类型。

## 标识符

### 标识符概念

概念：用来标识C源程序中某个对象名称的符号。

### 标识符的分类

对象：常量，变量，语句标号（goto语句后面的标号），数据类型，自定义函数名，数组等。

### 标识符的命名规则

避免与编译器的默认编译规则产生冲突，导致编译时出现二义性。

1. 组成成分：字母、数字、下换线（\_）、美元符($)组成。 A$B
2. 开头部分：不能以数字开头；
3. 标识符号长度：C89规定编译器至少要支持31（包含31字符），C99至少支持63（63字符）
4. 不能与C关键字相同或C库函数同名；
5. 最好能够体现出标识符所具有的含义。（要做到望文生义） ---- C标准不要求，但是最好是实现

➀文件名 文件名要体现出文件的内容

➁函数名 函数名要体现出函数的功能（eg）

➂变量名 变量名要体现出变量的含义（eg）

简单版本《深圳信盈达C程序编程规范.pdf》

具体可以《华为软件编程规范》 ----

建议使用英文表示，多个单词使用下划线连接或大小写分隔。

addFunc

AddFun

add\_func

## 关键字

### 关键字概念

关键字是被C编译器已经定义保留的专用特殊标识符。关键字都是有固定的名称和含义的，用户自定义的标识符不能与关键字相同。

### ANSI C 关键字列表



|  |  |  |
| --- | --- | --- |
| **ANSI C的32个关键字** | | |
| **关键字** | **用途** | **说明** |
| **auto** | 储存类声明 | 用以声明局部变量，默认值为此 |
| **break** | 程序语句 | 退出最内层循环体 |
| **case** | 程序语句 | switch语句中的选择项 |
| **char** | 数据类型声明 | 单字节整型数或字符型数据 |
| **const** | 储存类声明 | 在程序执行过程中不可修改的变量值 |
| **continue** | 程序语句 | 转向下一次循环 |
| **default** | 程序语句 | switch语句中的缺省选择项 |
| **do** | 程序语句 | 构成do…while循环结构 |
| **double** | 数据类型声明 | 双精度浮点数 |
| **else** | 程序语句 | 构成if…else选择结构 |
| **enum** | 数据类型声明 | 枚举 |
| **extern** | 数据类型声明 | 在其他程序模块中声明了的全局变量 |
| **float** | 数据类型声明 | 单精度浮点数 |
| **for** | 程序语句 | 构成for循环结构 |
| **goto** | 程序语句 | 构成goto转移结构 |
| **if** | 程序语句 | 构成if…else选择结构 |
| **int** | 数据类型声明 | 基本整型数 |
| **long** | 数据类型声明 | 长整型数 |
| **register** | 储存类声明 | 使用CPU内部寄存器的变量 |
| **return** | 程序语句 | 函数返回 |
| **short** | 数据类型声明 | 短整型数 |
| **signed** | 数据类型声明 | 有符号数，二进制数据的最高位为符号位 |
| **sizeof** | 运算符 | 计算表达式或数据类型的字节数 |
| **static** | 储存类声明 | 静态变量 |
| **struct** | 数据类型声明 | 结构类型数据 |
| **switch** | 程序语句 | 构成switch选择结构 |
| **typedef** | 数据类型声明 | 重新进行数据类型定义 |
| **union** | 数据类型声明 | 联合类型数据 |
| **unsigned** | 数据类型声明 | 无符号数据 |
| **void** | 数据类型声明 | 无类型数据 |
| **volatile** | 数据类型声明 | 声明该变量在程序执行中可被隐含地改变 |
| **while** | 程序语句 | 构成while和do…while循环结构 |

## 常量

### 常量概念

在程序执行过程中，不能被改变的数据量称为常量。

### 常量分类

#### 直接常量(字面常量)

直接常量就是各种基本数据类型具体的数值。

* 整型常量：12、0、-3；
* 浮点型常量：4.6、-1.23；
* 字符常量：'a'、'b'。

#### 符号常量（#define，const）

1. 概念：使用标识符表示的常量，称为符号常量。
2. 使用符号常量优点：含义清楚；能做到“一改全改”。
3. 定义符号常用两种形式：
4. 宏定义： #define <符号常量> 常量

宏名规则：**一般是全部大写。**

示例：

#define PI 3.1

1. const 语句 ：const <类型说明符> <常量名> = <常量值>

说明：const 修饰符主要用来定义常量或变量。C语言中，const修饰的是一类特殊的常量，一般称为符号常量或const变量。

示例：

|  |
| --- |
| int main(int argc, char \*argv[])  {  //const变量定义时必须初始化，否则后面不能修改  const int i = 123;  printf("i=%d\r\n",i);  //不能修改 const 变量，编译会报错 ,可以打开编译测试效果  //i = 456;  printf("i=%d\r\n",i);  return 0;  } |

**思考：**宏定义和const定义的常量有什么区别？

宏定义没有类型，const有类型的，编译期间会进行语法检查。

## 变量

### 变量概念

1. 程序运行的时候，可以动态修改的数据量称为变量。
2. 变量在内存中占据一定的存储单元。
3. 变量名命名规则符合标识符的命名规则。

12kddfj , G+h ， o#23f

g$sd12 , \_\_1\_\_\_2\_\_ , a$b

### 变量声明

所有的C变量使用前都必须先定义，也就是变量的声明。

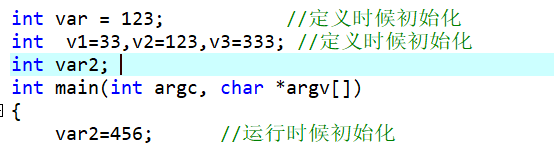
没有定义就使用：

语法：类型说明符号 变量标识符1, 变量标识符2,……,变量标识符n;

int var1,var2,var3;s

### 变量初始化

1. 赋值初始化（eg）
2. 声明初始化（eg）

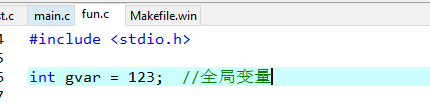


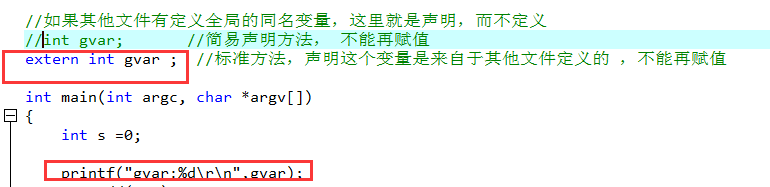
### 变量分类：

全局变量：定义在任何函数之外，存储在全局静态存储区中，整个命令周期中地址不变，也不会释放。整个应用程序（其他c文件也可以使用）都可以使用的变量。没有初始化值，默认值是0.

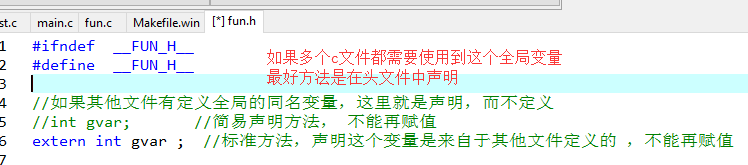
局部变量：定义在函数体之中，存储在栈空间（公用内存）中，用完就释放，没有初始化，默认值是随机，是上一次使用后留下的值

多文件使用全局变量：

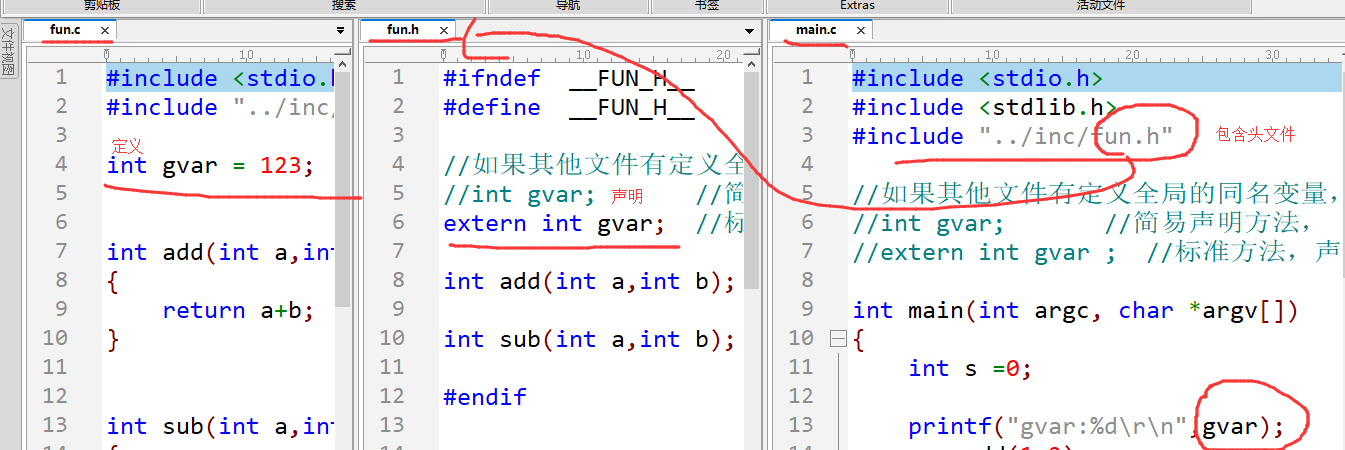




标准用法：：



**三个文件关系：**



## 整型数据

整型数据是整数类型数据，是最常用的数据类型。类型声明符是 int ,short ,long 。

### 整型数据类型的声明

整型数据类型：整型变量和整型常量，其前面可以加上不同的修饰符，整型数据有以下几种类型：

unsigned short int == unsigned short 只有大于等于0数值

signed short int == signed short 有正数有负数

signed long int == signed long 有正数有负数

unsigned long int == unsigned long 只有大于等于0数值

### 整型数据类型的存储

C语言提供多种整数类型，分别表示不同范围的整数。整型数据的大小一般与系统的字长有关，其存储一般采用补码的形式。

#### 字长概念

CPU在单位时间内(同一时间)能一次处理的二进制数的位数叫字长。

能处理字长为8位数据的CPU通常就叫8位的CPU。同理32位的CPU就能在单位时间内处理字长为32位的二进制数据。

二进制数据的一个0或1是组成二进制的最小单元，称为比特（bit）。

**说明： 变量的地址（指针）占用的空间大小一定等于计算机的字长。**

数值表示

两个开关 AB有多少种组合？

A 0 1

B 0 1

A B

0 0

0 1

1 0

1 1

C语言最小访问单位是字节：

1字节 == 8个二进制位（8bit） , 最多可以表示 256种组合。

如果是无符号 0~255 ,

有符号 -128 ~127

如果是16位的数据：

无符号：0~(2^16)-1 , 0~65535

有符号：-(2^15) ~ (2^15)-1 ，-32768 ~ 32767

n位有符号数值： -2^(n-1) ~ (2^(n-1)) -1

#### 整型的存储空间

在C语言中，根据计算机的字长不同，为int 类型分配的存储空间也不一样。如果不确定当前的计算的int 类型占用的存储空间大小，可以使用sizeof 来计算（eg）。

各种整型取值范围：

SHRT\_MAX: 32767 short

SHRT\_MIN: -32768

USHRT\_MAX: 65535 unsigned short

INT\_MAX: 2147483647

INT\_MIN: -2147483648

UINT\_MAX: 4294967295

LONG\_MAX: 9223372036854775807

LONG\_MIN: -9223372036854775808

ULLONG\_MAX: 18446744073709551615 （8字节，无符号时）

LLONG\_MAX: 9223372036854775807 （8字节，有符号时）

**大小测试示例：**

|  |
| --- |
| #include <stdio.h>  int main(void)  {  int k = 123;  char \*pc;  int \*pi =&k;  short \* ps;  long long \* pll;  printf("pc:%d\r\n", sizeof(pc)); //4  printf("ps:%d\r\n", sizeof(ps)); //4  printf("pi:%d\r\n", sizeof(pi)); //4  printf("pll:%d\r\n", sizeof(pll)); //4    printf("&printf:%d\r\n", sizeof(&printf)); //4  printf("k:%d\r\n", sizeof(k)); //4  //[整型字面常量]默认是int类型  printf("0:%d\r\n", sizeof(0)); //? 4  printf("short:%d\r\n", sizeof(short)); //2  printf("unsigned short:%d\r\n", sizeof(unsigned short));//2  printf("int:%d\r\n", sizeof(int));//4  printf("unsigned int:%d\r\n", sizeof(unsigned int));//4  printf("long:%d\r\n", sizeof(long));//4  printf("unsigned long:%d\r\n", sizeof(unsigned long));//4  printf("long long:%d\r\n", sizeof(long long));//8  printf("unsigned long long:%d\r\n", sizeof(unsigned long long));//8  getchar();  return 0;  } |

#### 整型的存储形式

在计算机中，整型数据是以二进制形式存储的，且一般使用二进制补码的形式表示。（对于有符号的变量最高位是符号位，0表示正数，1表示负数。）

补码规则：

1. 正数补码是其自身（原码）
2. 负数的补码，除符号位外逐位取反（0变成1，1变成0），再加1 。（原码(绝对值)求反加1）

思考： -15 的补码： 11110001 ，如何求得？

-15 原码 15 。

原码（-15绝对值--15） 0000 1111 ----全部取反--》 1111 0000 ---- 加1--- -》 1111 0001



### 整型常量

在C语言中，整型常量和整型常数可以表示八进制、十六进制和十进制数值，编译器是根据数值前缀区分各种进制的。 12 0x12 012

#### 十进制整常数

十进制整常数没有前缀。其数码为 0～9。

以下各数是合法的十进制整常数：

237、-568、65535、1627；

不合法的十进制整常数：

023 (不能有前导 0)、23D (含有非十进制数码)。

#### 八进制整常数

八进制整常数必须以 0 开头，即以 0 作为八进制数的前缀。

数码取值为 0～7，八进制数通常是无符号数。

合法的八进制数：

015(十进制为 13)、0101(十进制为 65)、0177777(十进制为 65535)；

不合法的八进制数：

256(无前缀 0)、092(包含了非八进制数码)、-0127(出现了负号)。

#### 十六进制整常数

十六进制整常数的前缀为 0X 或 0x。其数码取值为 0~9，A~F 或 a~f。

合法的十六进制整常数：

0X2A(十进制为 42)、0XA0 (十进制为 160)、0XFFFF (十进制为 65535)；

不合法的十六进制数：

FF(无前缀)、0x5HB(包含了非16进制数码)。

#### 整型常数的后缀 --- 重点

在 32 位字长的机器上，基本整型的长度也为32 位，因此表示的数的范围也是有限定的。

十进制无符号整常数的范围为 0～2^32，有符号数为-2^31～+2^31-1。

int可以表示的最大范围是：-2147483648<---->2147483647

2147483649UL

2147483647ul

2147483647Ul

2147483647uL

2147483647LL

如果使用的数超过了上述范围添加后缀

l ，L 、ul,UL 、ll，LL、ull, ULL :

面试题目：定义宏表示10000年的秒。

分 1\*60UL 小时 ： 分\*60UL ……

### 整型数据的输入和输出

C语言中，可使用scanf 函数来接收键盘输入的整数，使用printf函数输出整数，根据整型数据类型不同，格式控制符也不相同。

1. 输入/输出不同类型的整型数据

%ld long

%hd short

%lld long long

%u unsigned int

%lu unsigned long

%llu unsigned long long

示例：

int ii;

scanf("%d",&ii); //输入

printf("ii:%d\r\n",ii); //输出

1. 输入不同进制的整数数据

在C语言中，可按10进制，16进制，8进制三种方式书写整数。使用scanf函数也允许按这三种进制输入整数。默认情况下输入10进制数据，但是可以通过scanf函数中的控制符号来输入/输出不同进制的数据。

%x

%o (字母o)

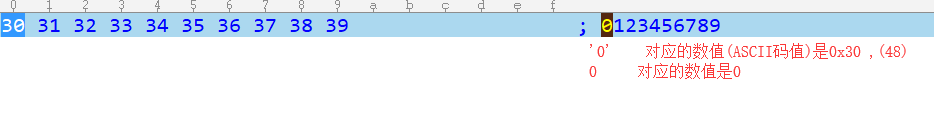
## 字符型数据

字符型数是用来存放字符的变量类型。字符型变量只能存放一个字符。

### 字符型变量声明

分类：有符号字符型和无符号字符

C语言中，字符型数据在操作时是按整型数据处理，计算机中的字符是以ASCII码方式表示，长度为1字节。

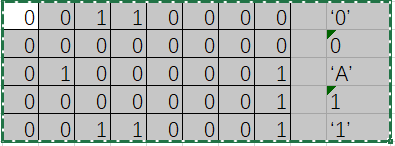


类型符号：char



### 字符型的存储

字符型数据存储很简单，占用1字节数据，直接存放字符的ASCII对应的二进制数值。



### 字符常量

概念：字符常量是用单引号括起来的一个字符。

例如：'a'、'b'、'='、'+'、'?' 都是合法字符常量。

**字符常量有以下特点：**

1. 字符常量只能用单引号括起来，不能用双引号或其它括号。
2. 字符常量只能是单个字符，不能是字符串。

 错误的

1. 字符可以是字符集中任意字符。
2. 单引号中的字符不能是单引号(') 和反斜杠（\）

char var = '''; //错误的 ，因为 配对错误

char var2 = '\'; //错误的 ，因为 \ 有特殊含义，表示转义

正确做法：

char var = '\''; //

char var2 = '\\'; //

1. 对于一些不能使用符号表示的控制符号，也可以使用ASCII值表示。（13-回车，13-换行）

不能显示ASCII字符，如tab， 回车 ， 退格 ， 响铃 。。。。。。

### 转义字符

转义字符是一种特殊的字符常量。转义字符以反斜线"\"开头，后跟一个或几个字符。

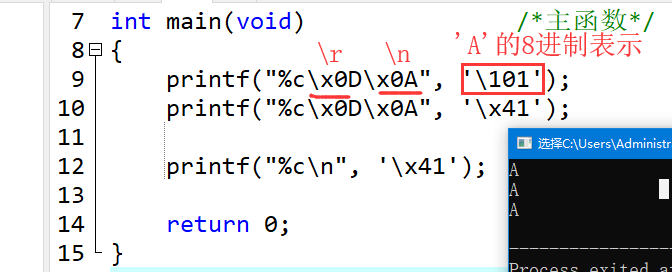
转义字符具有特定的含义，不同于字符原有的意义，故称“转义”字符。例如，在前面各例题 printf 函数的格式串中用到的“\n”就是一个转义字符，其意义是“回车换行”。转义字符主要用来表示那些用一般字符不便于表示的控制代码。

常用的转义字符及其含义

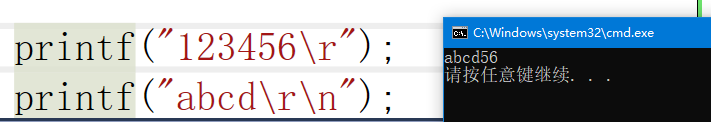
|  |  |  |
| --- | --- | --- |
| 转义字符 | 意义 | ASCII码值（十进制） |
| \a | 响铃(BEL) | 007 |
| \b | 退格(BS) ，将当前位置移到前一列 | 008 |
| \f | 换页(FF)，将当前位置移到下页开头  在屏幕中的显示没有任何影响，但是会影响打印机操作。 | 012 |
| **\n** | **换行(LF) ，将当前位置移到下一行开头** | **010** |
| **\r** | **回车(CR) ，将当前位置移到本行开头** | **013** |
| \t | 水平制表(HT) （跳到下一个TAB位置） | 009 |
| \v | 垂直制表(VT) | 011 |
| \\ | 代表一个反斜线字符''\' | 092 |
| \' | 代表一个单引号（撇号）字符 | 039 |
| \" | 代表一个双引号字符 | 034 |
| \0 | 空字符(NULL) | 000 |
| \ddd | 1到3位八进制数所代表的任意字符 | 三位八进制 |
| \xhh | 1到2位十六进制所代表的任意字符 | 二位十六进制 |

Ｃ语言字符集中的任何一个字符均可用转义字符来表示。表中的 \ddd 和 \xhh正是为此而提出的。ddd 和 hh 分别为八进制和十六进制的 ASCII 代码。如 '\101表示字母 "A" ，'\102' 表示字母"B"，'\134' 表示反斜线，'\x0A' 表示换行等。

printf("%c\n", '\102');



示例： **\r**\n 应用示例：



【例 3.8】转义字符的使用。

#include<stdio.h>

#include<math.h>

int main(void) /\*主函数\*/

{

int a, b, c;

a = 5;

b = 6;

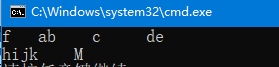
c = 7;

printf(" ab c\tde\rf\n");

printf("hijk\tL\bM\n");

return 0;

}



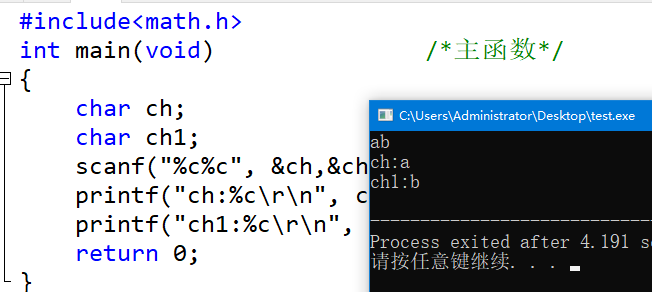
### 字符类型数据输入/输出 scanf printf

C语言中，可使用scanf 函数来接收键盘输入的字符，使用printf函数输出字符，格式控制符为 %c 。

示例：输出

|  |
| --- |
| int main(void) /\*主函数\*/  {  char ch=65; //  printf("ch-d:%d\r\n", ch); // 65  printf("ch-c:%c\r\n", ch); // A  ch = '@';// @ == 64  printf("ch-d:%d\r\n", ch); // 64  printf("ch-c:%c\r\n", ch); // @  return 0;  } |

示例：输入



其他输入输出函数：

输入函数：getchar,getch, gets

int getchar(void) 从键盘读取一个字符，并且带回显功能，需要等待回车才读入

int getch(void) 从键盘读取一个字符，不带回显功能，不等待回车直接读入

char \*gets(char \*str) 从键盘读取一个字符串，回车结束输入。

输出函数：putchar,putch,puts

示例：

|  |
| --- |
| #include<math.h>  int main(void) /\*主函数\*/  {  char ch;  printf("请输入一个字符,回车结束:");//要自己加换行功能  //puts("请输入一个字符,回车结束:"); //带换行功能  ch = getchar(); //从键盘中读入一个字符 ，有回显功能  printf("ch:%c\r\n", ch); //输出  //putchar(ch); //输出    //puts("请输入一个字符,回车结束:"); //带换行功能  printf("请输入一个字符,直接就可以读取，不需要回车:"); //要自己加换行功能    ch = getch(); //从键盘 中读入一个字符,没有回显功能  printf("\r\n你输入的字符是:\r\n", ch);  printf("%c\r\n", ch); //输出  //putchar(ch); //输出    return 0;  } |

### 字符变量

字符变量用来存储字符常量，即单个字符，字符值是以 ASCII码的形式存放在变量的内存单元之中的。

类型说明符: char。

内存大小：一个字节的内存空间，

示例：%c , %d 输出形式

示例：把小写字母a，b换成大写字母A,B （a，b）。

### 字符串常量

字符串常量是由一对双引号括起的字符序列。

例如： "CHINA" ， "C program" ，"$12.5" 等都是合法的字符串常量。

注意：字符串常量和字符常量是不同的量。

它们之间主要有以下区别：

* 字符常量由单引号括起来，字符串常量由双引号括起来。
* 字符常量只能是单个字符，字符串常量则可以含一个或多个字符。
* 可以把一个字符常量赋予一个字符变量，但不能把一个字符串常量赋予一个字符变量。
* 字符常量占一个字节的内存空间。字符串常量占的内存字节数等于字符串中字节数加 1。增加的一个字节中存放字符 '\0' (ASCII 码为 0)，这是字符串结束的标志。

例如：

字符串 "C program" 在内存中所占的字节为：



字符常量'a'和字符串常量"a"虽然都只有一个字符，但在内存中的情况是不同的。

'a'在内存中占一个字节，可表示为：



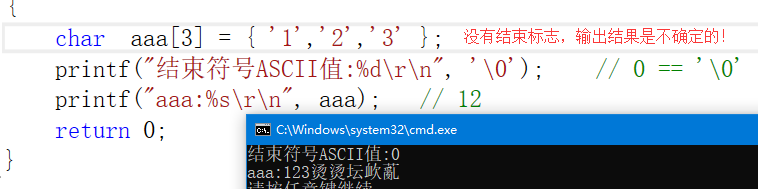
"a"在内存中占二个字节，可表示为：



示例：

|  |
| --- |
| #include<stdio.h>  #include<math.h>  #include<stdio.h>  #include<math.h>  int main(void) /\*主函数\*/  {  char aaa[ ]={'1','2',0,'3','\0'};  printf("结束符号ASCII值:%d\r\n",'\0') ; // 0 == '\0'  printf("aaa:%s\r\n",aaa) ; // 12  return 0;  } |

**注意：没有结束符号的数组不能直接使用%s输出，否则可能导致输出乱码或内存异常。**



//扩展应用：

//UART字符串发送函数

// char aaa[ ]={'1','2',0,'3','\0'};

//不能发送 0

void UART\_send\_string(char \*str)

{

while(\*str) {

UART\_send\_byte(\*str);

str++;

}

}

//发送N字节函数

// char aaa[ ]={'1','2',0,'3','\0'};

//能发送 0

//UART\_send\_bytes(aaa,5)

void UART\_send\_bytes(char \*data,int num)

{

int i;

for(i=0;i<num;i++) {

UART\_send\_byte(data[i]);

}

}

## 浮点型

浮点类型就是用于表示包含小数点的数据类型，也称为实数类型。

浮点数据二进制存储结构和整型不相同， 其存储空间划分为三部分：**符号位 指数位 尾数位**

**尾**数部分：使用2进制数来表示此浮点数的实际值；

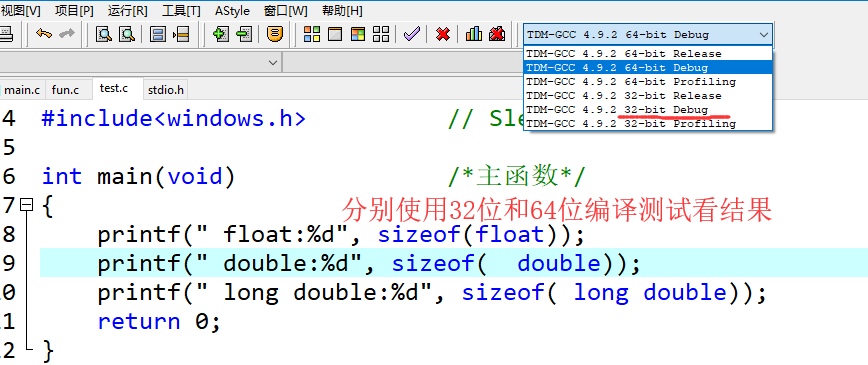
指数部分：表示数值范围：

### 浮点类型声明

C语言中支持三种浮点数据：

在32位平台下：float (占4B), double(占8B) , long double(占12B) 类型。

在64位平台下：float (占4B), double(占8B) , long double(占16B) 类型。



### 浮点型数据的存储

浮点型数据与整型数据存储方式不同，浮点数在内存中是按科学计数法来存储的，分为三部分：符号位部分、指数部分、小数部分。

符号位：表示正数还是负数

指数部分：10的N次方

尾数部分：也称为小数部分

#### float

1bit（符号位） 8bits（指数位） 23bits（尾数位）

单精度(32位)浮点数的结构：

名称                                           长度        比特位置

符号位    Sign  （S）            : 1bit       （b31）

指数部分Exponent （E）      : 8bit       （b30-b23）

尾数部分Mantissa   （M）    : 23bit     （b22-b0）

其中的指数部分（E）采用的偏置码（biased）的形式来表示正负指数，若E<127则为负的指数，否则为非负的指数。

10.65 --- > 1010. 101001 ……..

0.5 \*2 =1.0

0.0\*2 = 0.0

0.65 \*2 = 1.3 ---- 1

0.3\*2 = 0.6 ---0

0.6\*2 = 1.2 --- 1

0.2\*2 = 0.4 ---0

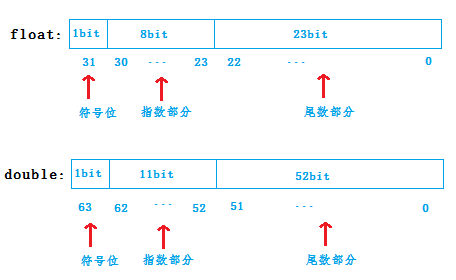
0.4\*2 = 0.8 -- 0

0.8\*2 = 1.6 -- 1

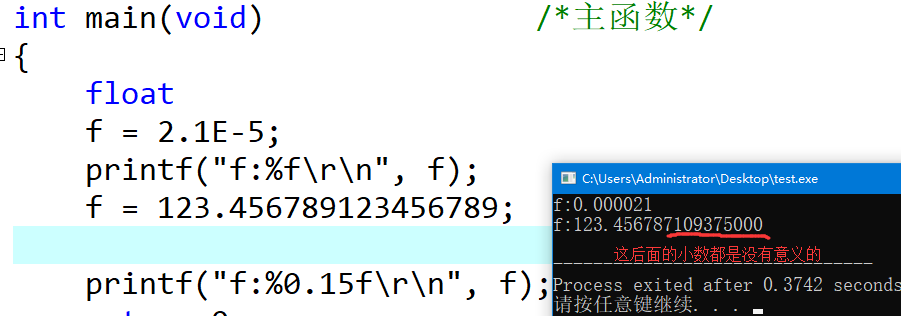
0.6\*2

另外尾数部分M存储的是当把一个浮点数规范化表示后的1.zozooz...（二进制的）形式的zozooz的部分的比特串，共23位。这部分最高位固定是1，因此可存放数值只有23位，2^23 = 8,388,60，共七位，这意味着最多能有7位有效数字，但绝对能保证的为6位，也即float的精度为6~7位有效数字；。

求值方法：(-1)^S \* (1.M)\* 2^(E-127)



6位整数.123457898998999999129129192



#### double

1bit（符号位） 11bits（指数位） 52bits（尾数位）

双精度(64位)浮点数的结构与单精度相仿

名称                                           长度        比特位置

符号位    Sign  （S）             : 1bit        （b63）

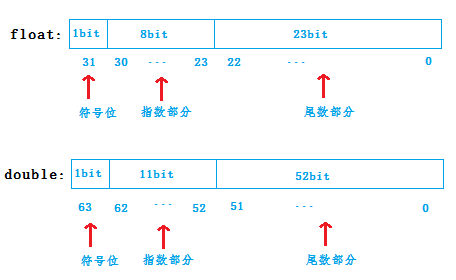
指数部分Exponent （E）     : 11bit      （b62-b52）

尾数部分Mantissa   （M）   : 52bit      （b51-b0）

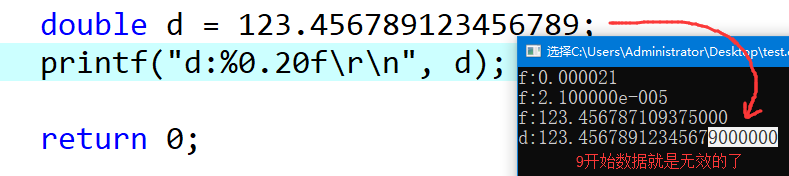
双精度的指数部分（E）采用的偏置码为1023。

求值方法：(-1)^S\*(1.M)\*2^(E-1023)

double：2^52 = 4,503,599,627,370,496‬，一共16位，同理，double的精度为15~16位。（能绝对保证15位没有问题）



18位整数.1~~00000999888~~



#### long double 12 字节(devcpp 32位模式)

可以表示的数值更大，有效数位是18位。

### 浮点常量的表示方法

实型也称为浮点型。实型常量也称为实数或者浮点数，实数只使用十进制表示。

它有二种形式：十进制小数形式，指数形式。

#### 十进制数形式

由数码 0~ 9 和小数点组成，

例如：0.0、25.0、5.789、0.13、5.0、300.、-267.8230 等均为合法的实数。

#### 指数形式

由十进制数，加阶码标志“e”或“E”以及阶码（只能为整数，可以带符号）组成。

指数表示形式：aEn（a 为十进制数，n 为十进制整数，可正可负） 其值为 a\*10n。

如：

2.1E5 (等于 2.1\*105)

3.7E-2 (等于 3.7\*10-2)

0.5E7 (等于 0.5\*107)

-2.8E-2 (等于-2.8\*10-2)

**不合法的**指数表示形式**：**

E7 (阶码标志 E 之前无数字)

-5 (无阶码标志) //赋值给一个变量是可以的。

53.**-**E3 (负号位置不对)

2.7E (无阶码)

示例：356. 356 -2.8E-2

### 浮点型数据输入/输出

在C语言中，可以使用scanf函数向程序输入数据，使用printf函数输出。对于浮点型数据，可采用以下控制符：

%f： 对应float类型数据，表10进制形式表示

%e： 对应float类型数据，以指数形式表示

%Lf：对应double 和long double，以10进制形式

%Le: 对应double 和long double，以指数形式表示

## 类型转换

C语言中，数据类型是可以相互转换的。转换的方法有两种，一种是自动转换，一种是强制转换。

### 自动转换（隐式转换）

char c = 3;

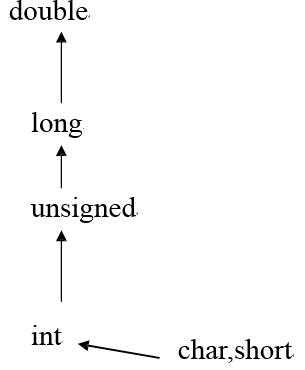
1.23 + 5 \* c --🡪

C语言中，当不同的数据类型混合在一起运算时，就会出现类型转换。

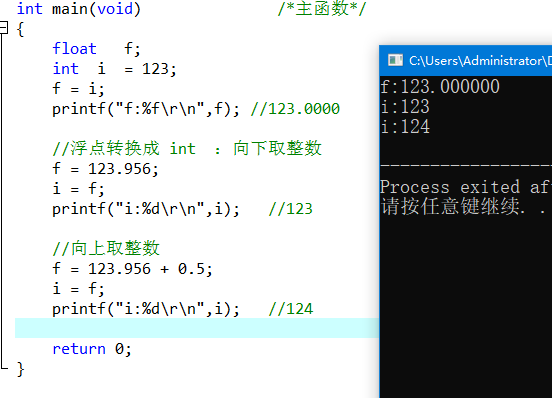
自动转换发生在不同数据类型的量混合运算时，由编译系统自动完成。

自动转换遵循以下规则：

1. 若参与运算量的类型不同，则先转换成同一类型，然后进行运算。
2. 转换按数据长度增加的方向进行，以保证精度不降低。如 int 型和 long 型运算时， 先把 int 量转成 long 型后再进行运算。
3. 所有的浮点运算都是以双精度进行的，即使仅含 float 单精度量运算的表达式，也要先转换成 double 型，再作运算。
4. **char 型和 short 型参与运算时，必须先转换成 int 型。**
5. 在赋值运算中，赋值号两边量的数据类型不同时，赋值号右边量的类型将转换为左边量的类型。如果右边量的数据类型长度左边长时，将丢失一部分数据，这样会降低精度，丢失的部分按四舍五入向前舍入。
6. 下图表示了类型自动转换的规则。



【例 3.12】使用int 类型和float 类型转换效果



|  |
| --- |
| #include<stdio.h>  #include<math.h>  #include<windows.h> // Sleep(500); 500ms  int main(void) /\*主函数\*/  {  float f;  int i = 123;  f = i;  printf("f:%f\r\n",f); //123.0000    //浮点转换成 int ：向下取整数  f = 123.956;  i = f;  printf("i:%d\r\n",i); //123    //向上取整数  f = 123.956 + 0.5;  i = f;  printf("i:%d\r\n",i); //124    return 0;  } |

### 强制类型转换（显式转换）

强制类型转换是通过类型转换运算来实现的。

语法形式：(类型说明符) (表达式)

其功能是把表达式的运算结果强制转换成类型说明符所表示的类型。

【例 3.13】强制转换为float 为int

|  |
| --- |
| #include<stdio.h>  #include<math.h>  int main(void) /\*主函数\*/  {  float f;  int i = 123;  f = (float)i;  printf("f:%f\r\n",f); //123.0000    //浮点转换成 int ：向下取整数  f = 123.956;  i = (int)f; //强制转换  printf("i:%d\r\n",i); //123    //向上取整数  f = 123.956 + 0.5;  i = (int)f; //强制转换  printf("i:%d\r\n",i); //124    //如果要得到正确浮点计算数值，必须有其中一个是浮点类型  printf("\r\n10/3.0:%f\r\n",10/3.0); //  return 0;  } |

## 格式化输出函数 （输出的本质都是字符）

### 输出函数

函数名：printf()

函数功能：输出函数，用于向终端输出提示信息（输出双引号里面的内容）

函数返回值：输出字符的个数

函数原型：printf("格式控制"，输出列表)

格式控制：如果是可知的则原样输出内容，通常是作为提示信息的，起到解释说明的作用

如果是不可知的需要用格式说明符去代替（例如: %d %c %f）

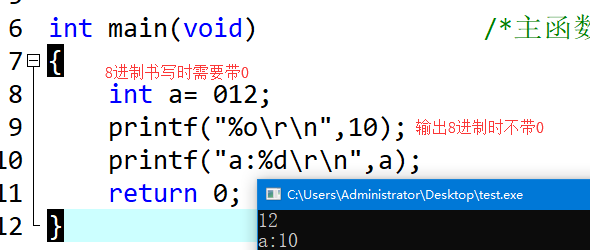
格式说明符 ：控制数据输出的格式，将数据转化成对应的格式显示出来

### 整型

目前整形数据的计数形式通常可分为八进制、十进制和十六进制

八进制：可识别的数据为：0， 1， 2， 3， 4， 5， 6， 7

以0开头的数据都是八进制形式。对应的格式说明符为 %o 八进制



十进制：默认的是十进制形式。如：123、89、34。可识别的数据为：0 ~ 9。

对应的格式说明符为 %d，%i，%u。

%[整数]d：整数用来控制数据输出的宽度(数据的位数)

%5d： 正数：表示右对齐左补空格，并且该数据总共占有5列。

%-5d：负数：表示左对齐右补空格，并且该数据总共占有5列。

printf("a:%10d\r\n",a); 



注意：当原有的数据宽度超出了所控制的数据的宽度，数据会原样输出

该宽度修饰的是数据本身已有的数据的位数和修饰的数据的宽度比较

十六进制：如：0X63，0x40；可识别的字符为：0~9，a~f，A ~ F。

以0x或0X开头的数据都是十六进制形式。

对应的格式说明符为：

%x 无符号十六进制数, 用小写字母

%X 无符号十六进制数, 用大写字母



### 浮点型

双精度实型：double -- %lf

单精度实型：float -- %f ，不加宽度控制，默认输出6位

%[.小数位数量]f：整数表示所要保留的小数的位数

注意：在保留小数点位数的同时，还会进行四舍五入。

%e 科学计数法, 使用小写"e"

%E 科学计数法, 使用大写"E"

注意：整型数据和实型数据之间不能共用格式说明符。（因为数据的存储原理不一致）。

char a = 12;

char %d

short %d

int %d

long %d

### 字符型

char 型，对应的格式说明符为 %c。 字符型数据必须用单引号括起来。

转义字符：

#### 转义成为功能符

\0 空字符

\a 响铃

\b 后退

\n 换行

\r 回车

\t 水平制表符

#### 消除特殊含义

\' 单引号

\" 双引号 “\“”

\\ 反斜杠

\反斜杠本身是没有输出效应的。

如：printf("小明说:\ "你今天很漂亮！\"\n");

3.转义为其他的数据。

'\101' ---- 8进制ASCII表示

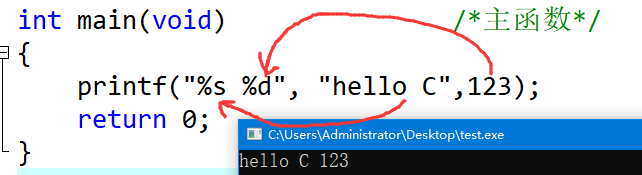
'\103' -------- 8进制ASCII表示

'\x0a' ----- 16进制ASCII表示

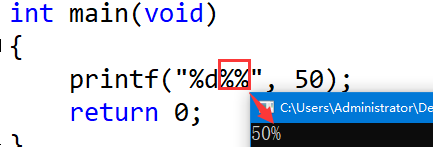
如：printf("a = %d\n", '\101'); --- 65(A的ASCII码值)

#### 其他的格式说明符

%s 一串字符



%% 一个'%'符号



### 输入函数

函数名：scanf()

函数功能：输入函数，把输入的数据保存在相应的地址空间中

函数原型：scanf("格式控制"，地址列表)

格式控制：

要有格式符，控制数据的输入格式

可有空格，逗号，和其他字符，但是在输入数据的时候要原样输入

不能加转义字符，转义会影响函数的跳出，必须原样输入转义字符才能结束函数

地址列表：

&：取地址符，功能就是取出数据所占空间的地址编号，如果不加取地址符造则会造成成程序代码崩溃

返回值：返回的是成功赋值的变量的数量

### 转换说明符使用

#### 整型 与 字符型

因为整型与字符型的位排列方式相同，所有他们可以彼此共享转换说明符，因为对字符型数据来说， 0--255的字符型都可以和整型相互转化。

%d: 取数据的后32位

%c: 取数据的后 8位

#### 整型 与 实型

因为整型与实型的位排列方式不同，所有他们不可以彼此共享转换说明符

整型：就用整型说明符

实型：就用实型说明符

注意：显示并不一定就是实际的结果或存储