



程序设计基础 (C) ——变量

郑州大学软件学院/网络空间安全学院

Lecturer: 宋轩

Office : 行政楼-306

Email : songxuan@zzu.edu.cn

做计算

```
printf("23+43=%d\n",23+43);
```

算找零

```
printf("23+43=%d\n",23+43);
```

```
printf("100-23=%d\n",100-23);
```

需要：

- 1.有地方放输入的数字；
- 2.有办法输入数字；
- 3.输入的数字能参与计算。

◆如何能在程序运行时输入那个数字23，然后计算输出结果？

```
int price = 0;

printf("请输入金额（元）：");
scanf("%d", &price);

int change = 100- price;

printf("找您%d元。 \n", change);

return 0;
```

今后课件上的程序都只有main的{}里面的部分

如何输入

- ◆ 输入也在终端窗口中
- ◆ 输入是以行为单位进行的，行的结束标志就是你按下了回车键。
在你按下回车之前，你的程序不会读到任何东西

变量

```
int price = 0;
```

```
printf("请输入金额（元）：");  
scanf("%d", &price);
```

```
int change = 100- price;
```

```
printf("找您%d元。\\n", change);
```

- ◆ **int price = 0;**
- ◆ 这一行，定义了一个变量。变量的名字是**price**，类型是**int**，初始值是**0**。
- ◆ 变量是一个保存数据的地方，当我们需要在程序里保存数据时，比如上面的例子中要记录用户输入的价格，就需要一个变量来保存它。用一个变量保存了数据，它才能参加到后面的计算中，比如计算找零。

变量定义

◆ 变量定义的一般形式就是：

- <类型名称> <变量名称>;
- int price;
- int amount;
- int price, amount;

变量的名字

- ◆ 变量需要一个名字，变量的名字是一种“标识符”，意思是它是用来识别这个和那个的不同的名字。
- ◆ 标识符有标识符的构造规则。基本的原则是：标识符只能由字母、数字和下划线组成，数字不可以出现在第一个位置上，C语言的关键字（有的地方叫它们保留字）不可以用做标识符。

C语言的保留字

auto,break,case,char,const,
continue,default,do,double,else,
enum,extern,float,for,goto,
if,int,long,register,return,
short,signed,sizeof,static,struct,
switch,typedef,union,unsigned,void,
volatile,while,inline,restrict

不需要背诵！

赋值和初始化

```
int price = 0;

printf("请输入金额（元）：");
scanf("%d", &price);

int change = 100 - price;

printf("找您%d元。\\n", change);
```

```
int price = 0;
```

- ◆ 这一行，定义了一个变量。变量的名字是 price，类型是int，初始值是0。
- ◆ price=0是一个式子，这里的“=”是一个赋值运算符，表示将“=”右边的值赋给左边的变量。

赋值

和数学不同， $a=b$ 在数学中表示关系，即a和b的值一样；而在程序设计中， $a=b$ 表示要求计算机做一个动作：将b的值赋给a。

关系是静态的，而动作是动态的。

在数学中， $a=b$ 和 $b=a$ 是等价的，而在程序设计中，两者的意思完全相反

初始化

- ◆ 当赋值发生在定义变量的时候，就像程序1中的第7行那样，就是变量的初始化。虽然C语言并没有强制要求所有的变量都在定义的地方做初始化，但是所有的变量在第一次被使用（出现在赋值运算符的右边）之前被应该赋值一次。
- ◆ 如果没有初始化？

读整数

```
int price = 0;
```

```
printf("请输入金额（元）：");  
scanf("%d", &price);
```

```
int change = 100- price;
```

```
printf("找您%d元。\\n", change);
```

如果输入的不是整数会怎样？

- `scanf("%d", &price);`

- 要求scanf这个函数读入下一个整数，读到的结果赋值给变量price

- 小心price前面的&

表达式

“=” 是赋值运算符，有运算符的式子就叫做表达式。

- price=0;
- change=100-price;

变量类型

```
int price = 0;
```

```
printf("请输入金额（元）：");  
scanf("%d", &price);
```

```
int change = 100 - price;
```

```
printf("找您%d元。\\n", change);
```

- `int price = 0;`
- 这一行，定义了一个变量。变量的名字是`price`，类型是`int`，初始值是0。
- C是一种有类型的语言，所有的变量在使用之前必须定义或声明，所有的变量必须具有确定的数据类型。数据类型表示在变量中可以存放什么样的数据，变量中只能存放指定类型的数据，程序运行过程中也不能改变变量的类型。

变量初始化

```
int price = 0;
```

```
printf("请输入金额（元）：");  
scanf("%d", &price);
```

```
int change = 100 - price;
```

```
printf("找您%d元。\\n", change);
```

- <类型名称> <变量名称> = <初始值>;
 - int price = 0;
 - int amount = 100;
- 组合变量定义的时候，也可以在这个定义中单独给单个变量赋初值，如：
 - int price = 0, amount = 100;

第二个变量

```
int price = 0;
```

```
printf("请输入金额（元）：");  
scanf("%d", &price);
```

```
int change = 100 - price;
```

```
printf("找您%d元。\\n", change);
```

- `int change = 100 - price;`
- 定义了第二个变量 `change`
- 并且做了计算

C99!

ANSI C

■只能在代码开头的地方定义变量

```
int price = 0;
```

```
printf("请输入金额（元）：");  
scanf("%d", &price);
```

```
int change = 100- price;
```

```
printf("找您%d元。\\n", change);
```

C99

```
int price = 0;  
Int change=0;
```

```
printf("请输入金额（元）：");  
scanf("%d", &price);
```

```
int change = 100- price;
```

```
printf("找您%d元。\\n", change);
```

ANSI C

常量

```
int price = 0;
```

```
printf("请输入金额（元）：");  
scanf("%d", &price);
```

```
int change = 100 - price;
```

```
printf("找您%d元。\\n", change);
```

- `int change = 100 - price;`

- 固定不变的数，是常数。直接写在程序里，我们称作直接量（`literal`）。

- 更好的方式，是定义一个常量：

- `Const int AMOUNT = 100;`

常量

```
int price = 0;

printf("请输入金额（元）：");
scanf("%d", &price);

int change = 100 - price;

printf("找您%d元。\\n", change);
```

```
const int AMOUNT = 100;
int price = 0;

printf("请输入金额（元）：");
scanf("%d", &price);

int change = AMOUNT - price;

printf("找您%d元。\\n", change);
```

- `int change = 100 - price;`
- 100这个固定不变的数，叫做常数。直接写在程序里，我们称作直接量（literal，字面量）。
- 更好的方式，是定义一个常量变量：
 - `const int AMOUNT = 100;`

C99!

const

- `const`是一个修饰符，加在`int`的前面，用来给这个变量加上一个`const`（不变的）的属性。这个`const`的属性表示这个变量的值一旦初始化，就不能再修改了。
- `int change = AMOUNT - price;`
- 如果你试图对常量做修改，把它放在赋值运算符的左边，就会被编译器发现，指出为一个错误。

C语言的字符集

- C语言源程序中使用的字符来自C语言的字符集。

- 字符集 (Character set) ——来自ASCII表

- 52个大小写字母 (Letters)

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

- 10个数字 (Digits)

0 1 2 3 4 5 6 7 8 9

- 空白符 (Blanks)

空格符、制表符、回车符、换行符

- 图形符号 (Graphic characters)

! # % ^ & * (_) - + = ~ [] ' | \ ; : " { } , . < > / ?

C语言的关键字

- 关键字 (Keywords) 是C语言中的词汇。
 - 也称为保留字 (Reserved words)
 - 类型说明
 - int、long、short、float、double、char、unsigned、signed、const、void、volatile、enum、struct、union**
 - 语句定义
 - if、else、goto、switch、case、do、while、for、continue、break、return、default、typedef**
 - 存储类别说明
 - auto、register、extern、static**
 - 长度运算符
 - sizeof**

auto: 指定变量的存储类型，是默认值

break: 跳出循环或switch语句

case: 定义switch中的case子句

char: 定义字符型变量或指针

const: 定义常量或参数

continue: 在循环语句中，回到循环体的开始处重新执行循环

default: 定义switch中的default子句

do: 定义do-while语句

double: 定义双精度浮点数变量

else定义枚举类型

enum声明外部变量或函数

extern声明外部变量或函数

float定义浮点型变量或指针

for定义for语句

goto定义goto语句

if定义if语句或if-else语句

int 定义整型变量或指针

long 定义长整型变量或指针

register 指定变量的存储类型是寄存器变量，Turbo c 中用自动变量代替

return 从函数返回

short 定义短整型变量或指针

signed 定义有符号的整型变量或指针

sizeof 获取某种类型的变量或数据所占内存的大小，是运算符

static 指定变量的存储类型是静态变量，或指定函数是静态函数

struct 定义结构体类型

switch 定义switch语句

typedef 为数据类型定义别名

union 定义无符号的整型或字符型变量或指针

unsigned 定义无符号的整型变量或数据

void 定义空类型变量或空类型指针，或指定函数没有返回值

volatile 变量的值可能在程序的外部被改变

while 定义while或do-while语句

C语言的标识符

- 标识符 (Identifiers) 是程序中引用对象的名称。
 - 用来标识变量、符号常量、数组、函数、结构体、共用体、自定义类型等。
 - 命名规则
 - 只能包括大小写字母、数字和下划线;
 - 首字符必须是字母或下划线;
 - 一般标识符的前31个字符有效; **(视具体编译器的规定)**
 - 不能与关键字相同。

count, student_name, sum,
test13, _number, Sum ✓

M. John, \$123, hi!,
12xyz, void ✗

存储单元的基本概念

- 存储单元：计算机内部存储器的部分空间，一般规定了相应的字节长度。
 - 赋值：破坏性的
 - 读值：非破坏性的

基本数据类型

- 引例
 - 问题
 - 计算任意一个圆的面积。
 - 源程序 (cw02-01.c)

```
#include <stdio.h>

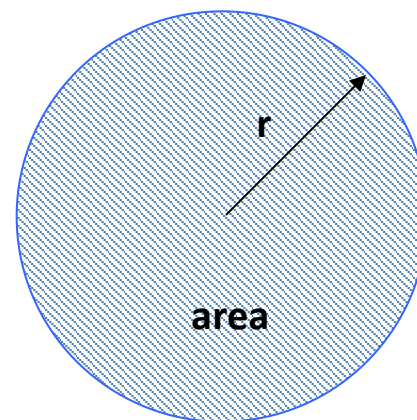
void main() {
    float r, area; //定义变量

    scanf("%f",&r); //读取输入的半径

    area = 3.14*r*r; //计算圆的面积

    printf("area=%f",area); //输出
}
```

$$\text{area} = \pi r^2$$



```
1
area=3.140000
```

变量与常量

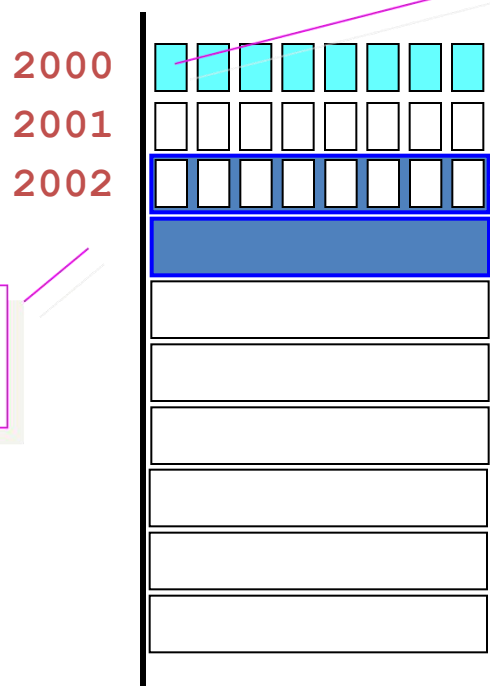
- 数据 (Data)
 - 程序需要使用数据。
 - 数据是信息的载体。
 - 数据有多种形式：数、字符、图片等。
- 常量 (Constants)
 - 在程序运行之前可以预先设定，并在整个运行过程中没有变化的数据。
 - 例如引例中的圆周率3.14。
- 变量 (Variables)
 - 在程序运行过程中可能变化或被赋值的数据。
 - 例如引例中的半径和面积。

数据类型

- 数据类型的作用
 - 决定数据的存储方式和占用的存储空间的大小。
 - 决定可以进行的操作。
- C语言的数据类型
 - 基本类型
 - 整型 (integer) , 字符型 (character) , 浮点型 (floating-point) , 枚举类型 (enumeration)
 - 构造类型
 - 结构体 (structure) , 共用体 (union) , 数组 (array)
 - 指针类型 (pointer)
 - 空类型 (void)

位、字节和字

• 内存储器的组织



地址：以字节为单位从0开始编号。

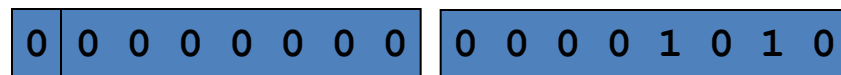
位 (bit)：最小的存储单位，可以容纳两个值之一，即0或1。

字节 (Byte)：基本的存储单位，8位。

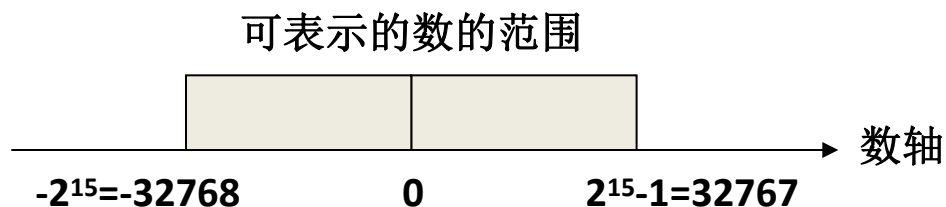
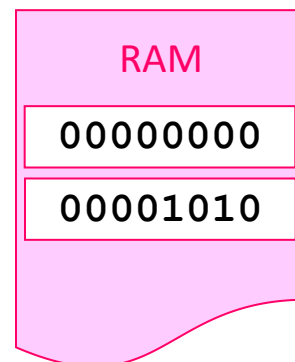
字 (word)：自然的存储单位，包含若干个字节。例如32位机的一个字就是32位。

整数的存储方式


- 有符号的正整数
 - 在内存中以二进制补码形式存放。
 - 正整数的补码与原码相同。
 - 例如：10



符号位



- | | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |



A diagram of a RAM memory block. It is a light blue rectangle with a darker blue border. Inside, the word "RAM" is written in blue at the top. Below it, there are two horizontal white boxes with blue borders. The first box contains the binary string "11111111" and the second box contains "11110110".

RAM

11111111

11110110

-10的原码

1 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0

按位取反

1 1 1 1 1 1 1 1

1 1 1 1 0 1 0 1

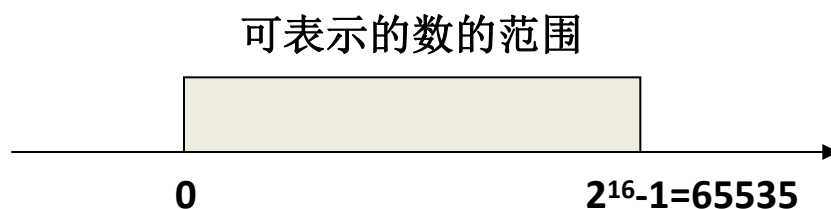
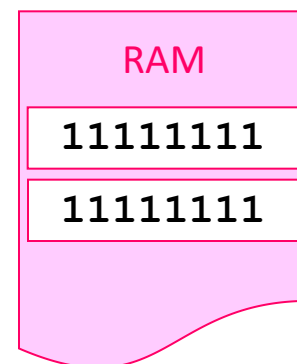
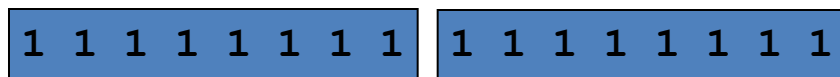
加一后得到 -10的补码

1 1 1 1 1 1 1 1

1 1 1 1 0 1 1 0

整数的存储方式

- 无符号整数
 - 所有二进制位都存放数值。
 - 例如：65535



整数的类型

- C语言提供多种整数类型
 - 为程序员提供了针对不同用途的多种选择。

类型名称	类型说明符	字节数	数值范围
基本整型	[signed] int	4	-2147483648~ 2147483647
短整型	[signed] short [int]	2	-32768~32767
长整型	[signed] long [int]	4	-2147483648~ 2147483647
无符号基本整型	unsigned [int]	4	0~4294967295
无符号短整型	unsigned short [int]	2	0~65535
无符号长整型	unsigned long [int]	4	0~4294967295

c标准只规定: $\text{short} \leq \text{int} \leq \text{long}$

最大最小值参考<limits.h>

仅供参考，实际值与所使用的操作系统、编译系统、机器有关。

整数常量

- 整型常量有三种形式：

- 十进制 (decimal) 整数

12 65 65535

- 八进制 (octal) 整数：带前缀 0 (zero)

014 0101 0177777

- 十六进制 (hexadecimal) 整数：带前缀 0x 或 0X

0xc 0x41 0xffff

- 默认类型是int，即有符号的基本整型。

- 可以加上后缀 u 或 U 表示无符号整数，或者 l 或 L 表示长整数。

0xb5Lu

整型变量

- 声明变量 (Declaration)

- 变量在使用之前必须被声明。
- 声明语句的格式:

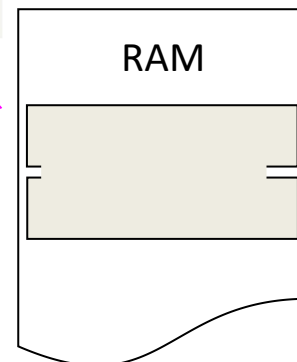
<类型说明符> <变量名>[, <变量名>[, ...]];

- 举例

```
int counter;  
int width, height;  
short x, y;  
long number;
```

变量声明创建了变量：为变量分配了存储空间。

height



整型变量

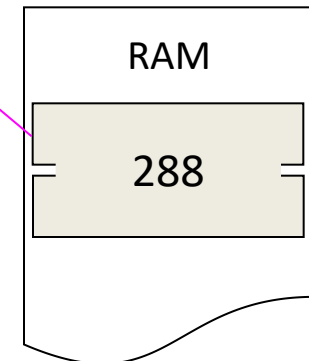
- 初始化 (Initialize) 变量
 - 为变量赋一个初始值。
 - 可以在声明语句中初始化变量。
 - 举例

```
int counter = 0;  
int width = 352, height = 288;
```

- 变量获得值的方法
 - 直接赋值
 - 输入
 - 初始化

初始化式

height



整型变量

- 输出变量的值
 - 可以使用 `printf()` 函数。
 - 与 `int` 类型对应的格式说明符是 `%d`。
 - 举例 (`cw02-02a.c`)

```
#include <stdio.h>

void main() {
    int a, b;

    a=32767; b=-32768;

    printf("a=%d,b=%d\n", a, b);
}
```

```
a=32767,b=-32768
```

整数的溢出

- 溢出

- 整数太大，超出了整数类型的数值范围。

- 使用printf() 时与unsigned int类型对应的格式说明符是%u。

- 举例 (cw02-02b.c)

```
#include <stdio.h>

void main() {

    unsigned c, d;

    c = 4294967295; d = c+1;

    printf("c=%u,d=%u", c, d);
}
```


字符的存储方式

- 字符编码

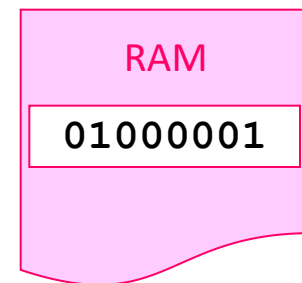
- 计算机使用一种数字编码（整数）来表示字符，每一个字符都对应一个特定的整数。
- 常用的编码是ASCII（美国信息交换用标准码）。
 - 7位二进制数，十进制码值范围从0到127。
 - 一般用一个字节保存，最高位为0。

- 字符的存储方式与整数相同

- 举例

- 字母A的ASCII码值为65，
- 那么在内存中以65的二进制形式存储，
- 且占一个字节。

01000001



字符的类型和字符变量

- C语言的字符类型：char
 - 占一个字节；
 - 可视为一个有符号的整数。
 - 举例 (cw02-03.c)

```
#include <stdio.h>

void main() {
    char c1, c2; //声明字符变量

    c1 = 97; //把一个整数赋值给字符变量
    c2 = c1-32; //字符变量可以进行算术运算

    printf("c1=%c,c2=%c\n", c1, c2);
    printf("c1=%d,c2=%d\n", c1, c2);
}
```

```
c1=a,c2=A
c1=97,c2=65
```

- 字符常量

- 用单引号括起来的一个字符。

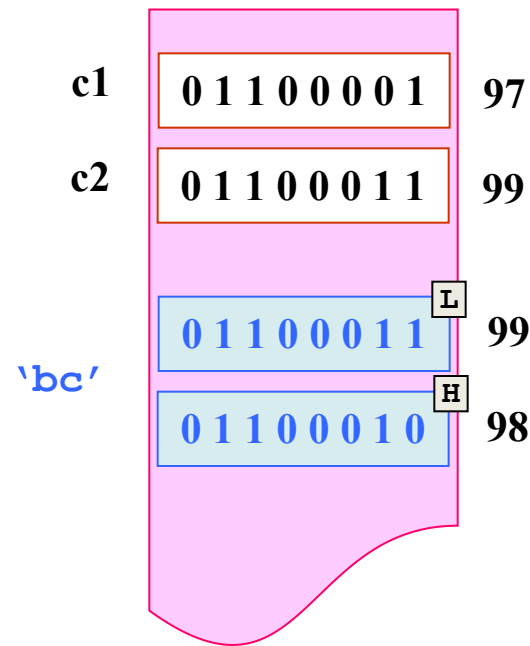
'x' '9' '+'

- C语言将字符常量视为int类型。

- 举例

```
char c1, c2;  
c1 = 'a';  
c2 = 'bc';
```

- char类型为8位,一个字节,
- 那么对于'bc', 将把'b'和'c'的ASCII码值存储在两个字节中, 并把'c'赋值给变量c2。
- 注意: 不同系统处理方式不同, 结果不同。



字符常量

- 转义字符 (escape character)
 - 指代一些特殊的字符。 (打印不出来的字符)

\a	警报	\\	反斜杠 (\)
\b	退格	\?	问号 (?)
\f	走纸	\'	单引号 (')
\n	换行	\"	双引号 (")
\r	回车	\ooo	八进制值 (o表示一个八进制数字)
\t	水平制表符	\xhh	十六进制值 (h表示一个十六进制数字)
\v	垂直制表符		

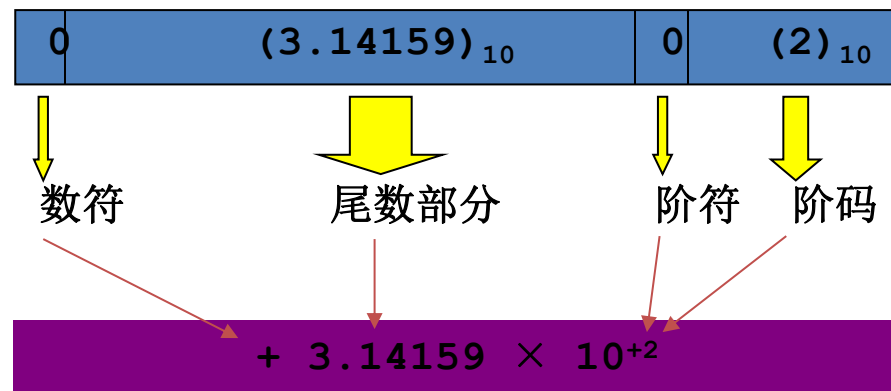
– 举例 (cw02-05.c)

```
#include <stdio.h>
void main()
{
    printf("a\tb\nc\bd\100\x40\n");
}
```

```
a______b
d@@
```

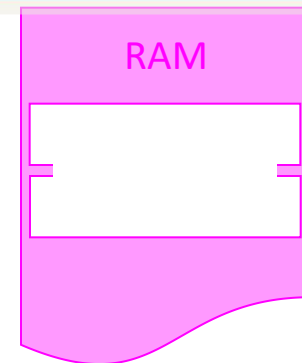
浮点数的存储方式

- 浮点数
 - 浮点型数据在内存中按指数形式存放。
 - 例如： $314.15 = 3.1415 \times 10^2$



由此可见，尾数部分的宽度决定了有效数字的个数（即精度），阶码部分的宽度决定了数值范围。

科学计数法允许使用少量的数字表示很大范围的数和很小的数。

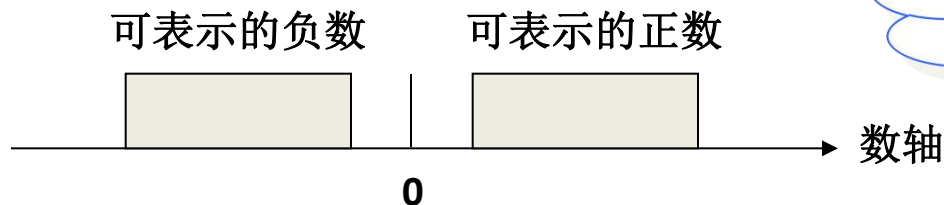


浮点数的类型

- 浮点数也有多种类型
 - 类型名称及典型大小

类型名称	类型说明符	字节数	有效数字	数值范围
单精度	float	4	6~7	(s)$10^{-37} \sim 10^{38}$
双精度	double	8	15~16	(s)$10^{-307} \sim 10^{308}$
长双精度	long double	8	15~16	(s)$10^{-307} \sim 10^{308}$

s = ± 1



仅供参考，实际值与所使用的操作系统、编译系统、机器有关。

浮点型常量

- 浮点型常量有两种形式：

- 十进制形式

12.3 .65 0.

- 指数形式：< 小数 > < e | E > < 整数 >

1.2e-2

.1E5

7E0

1.2×10^{-2}

0.1×10^5

7.0×10^0

- 默认类型是double。
- 可以加上后缀 **f** 或 **F** 表示float类型，或者 **l** 或 **L** 表示long double类型，否则该常量是double类型。

2.3f 1.2L .1E5f

浮点型变量

- 浮点型变量的声明和初始化
 - 举例

```
float radius;  
double x = 0.0, y = 0.0;
```

不能写成:

```
double x = y = 0.0;
```


浮点型变量

- 浮点数的输出
 - 使用printf()函数
 - float和double对应的格式说明符为%f、%e。
 - 举例

```
#include <stdio.h>

void main() {
    float f;
    double d;

    f=33333.33333f;
    d=33333.3333333333;

    printf("f=%f\nd=%f", f, d);
}
```

有效数字位数是有限的，在可表示的有效位之外的数字被舍去。因此可能会产生误差。

```
f=33333.332031
d=33333.333333
```

浮点数的舍入误差

- 浮点数的舍入误差
 - 举例

```
#include <stdio.h>

void main() {
    float a, b;

    a=123456.789e5;
    b=a+20;

    printf("a=%f\nb=%f", a, b);
}
```

```
a=12345678848.000000
b=12345678848.000000
```

???

浮点数的舍入误差

- 浮点数的舍入误差
 - 结果分析

a+20的理论值应该是：**12345678920**

但是，一个实型变量能够保证的有效数字是7位，后面的数字将被舍去，是没有意义的。

因此，最后得到

b=12345678848.000000

应当避免一个很大的数和一个很小的数直接相加或相减，否则就会“丢失”较小的数。

浮点数的溢出

- 上溢

- 举例

- 若某系统中的最大float值为 $3.4e38$ ，进行如下操作

```
float toobig = 3.4e38 * 100.0f;  
printf("toobig=%f", toobig);
```

- 得到结果

```
...inf...
```

无穷大 (infinity)

- 下溢

- 举例：假设-10是最小的指数，能够保留四位有效数字

- 如果把数 $0.1234e-10$ 除以10，将得到结果 $0.0123e-10$ ，但损失了一位有效数字。

数据类型小结

- C语言有多种数据类型。
- 基本的数据类型包括两大类：
 - 整数类型
 - 浮点类型
- 开发程序时，应当注意所需变量及其类型的选择。
 - 一般使用int和float表示数，用char表示字符。
 - 在使用变量的函数的可执行语句之前声明该变量，并为它选择有意义的名字。
 - 初始化变量使用的常量应当与变量的类型相匹配。

格式输入输出

- `printf()` 函数
- `scanf()` 函数
- 使用说明
 - 程序中可以不明确指定包含 `stdio.h` 头文件

```
#include <stdio.h>
```

可以省略此命令

格式输出

- 格式控制字符串

- 举例

```
printf( "n=%5d, f=%5.2f\n" , 3, 6.235)
```

普通字符

原样输出

格式说明符（转换规则）

% [修饰符] 格式字符

指定数据的输出格式

```
n=   3, f= 6.24
```

格式输出

- 格式字符

格式字符	功 能
d, i	以十进制有符号形式输出整数（正数不输出符号）
o	以八进制无符号形式输出整数（不输出前缀）
x, X	以十六进制无符号形式输出整数（不输出前缀）
u	以十进制无符号形式输出整数
f	以小数形式输出单、双精度实数
e, E	以指数形式输出单、双精度实数
g, G	选用%f和%e格式中输出宽度较短的一种，不输出无意义的零
c	以字符形式输出，输出一个字符
s	输出字符串

格式输出

- 举例

```
int main()
```

```
{ int a=97,b=-1;
```

```
float f=123.4;
```

```
printf("%d,%c\n",a,a);
```

```
printf("%d,%o,%x,%u\n",b,b,b,b);
```

```
printf("%f,%e,%gEND",f,f,f);
```

```
}
```

1 1 1 1 1 1 1 1 1 1 1 1 1 1

```
97, a
```

```
-1, 3777777777, ffffffff, 4294967295
```

```
123.400002, 1.234000e+002, 123.4END请按任意键继续. . .
```

格式输出

- 举例

```
int main()
{ int a=1,b=2,c=3;
  printf("%d,%d,%d,%d\n",a,b,c);
  printf("%d,%d,%d\n",a,b,c,a+b+c);
}
```

格式说明符和输出项
在数量和类型上应该
一一对应。

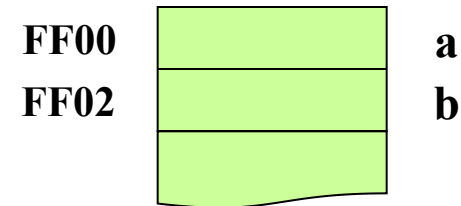
格式输入

- scanf
 - 使用形式
scanf(<格式控制字符串>, <地址列表>);
 - 按格式控制字符串规定的格式，从指定的输入设备读入数据，并存放到地址列表中的各地址项指定的变量中
- 使用说明
 - 格式控制字符串
 - 由双引号括起来的字符串，用于指定输入格式
 - 地址列表
 - 由若干个变量的地址组成

格式输入

- 地址列表

```
scanf ("%d, %d", &a, &b)
```



- 取地址运算符：&

- &<变量>

- 得到变量在内存中的地址。

格式输入

- 格式字符

格式字符	功 能
d, i	以十进制形式输入有符号整数
o	以八进制形式输入无符号整数
x, X	以十六进制形式输入无符号整数
u	以十进制形式输入无符号整数
f	以小数形式或指数形式输入实数
e, E, g, G	同f，它们之间可以互换
c	输入单个字符
s	输入字符串

格式输入

```
int main()  
{ char a,b,c;  
  scanf("%c%c%c", &a, &b, &c);  
  printf("a=%c,b=%c,c=%c", a,b,c);  
}
```

用 **c** 格式字符输入字符时，若格式控制字符串中无普通字符，那么认为所有输入的字符（包括空格、制表符、换行符）均为有效字符。

abc

1

a=a, b=b, c=c

a b c

2

a=a, b= , c=b

a b b b b b b c

3

a=a, b= b , c=b

a

4

b

a=a, b=

, c=b

- 算术运算的特点
 - 整数除法的结果一定是整数
 - 求余运算符的操作数必须是整数
 - 用0去除一个数会引起程序的致命错误
 - 算术表达式用直线形式的一行字符表示
 - 用圆括号区分分子表达式

C语言中的算术运算

- 算术运算符的优先级
 - $()$: 最高
 - $*$ $/$ $%$: 次之
 - $+$ $-$: 最低
- 算术运算符的结合律
 - 从左到右
 - 从右到左
 - 嵌套时: 从内到外

C语言中的算术运算

- 代数表达式和C语言表达式的例子
 - 求数的平均值 : $m = (a + b + c + d + e) / 5;$
 - 直线方程 : $y = m * x + b;$
 - 求余等 : $z = p * r \% q + w / x - y;$

- 求二次多项式的值

- $y = a * x * x + b * x + c;$

- 可加冗余圆括号：

- $y = (a * x * x) + (b * x) + c;$

C语言中的算术运算

例： 仅使用本章所学习的技术，编写一个计算0-10各个数的平方和立方的程序，并使用水平制表符（\t）按照如下表格形式打印结果。

number	square	cube
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

C语言中的算术运算

- 例：制表/* hw2_31.c 制表 */

```
#include <stdio.h>
```

```
/* function main begins program execution */
```

```
int main( void )
```

```
{
```

```
    printf( "number\tsquare\tcube\n" );
```

```
    printf( "%d\t%d\t%d\n", 0, 0 * 0, 0 * 0 * 0 );
```

```
    printf( "%d\t%d\t%d\n", 1, 1 * 1, 1 * 1 * 1 );
```

```
    printf( "%d\t%d\t%d\n", 2, 2 * 2, 2 * 2 * 2 );
```

```
    printf( "%d\t%d\t%d\n", 3, 3 * 3, 3 * 3 * 3 );
```

```
    printf( "%d\t%d\t%d\n", 4, 4 * 4, 4 * 4 * 4 );
```

C语言中的算术运算

```
printf( "%d\t%d\t%d\n", 5, 5 * 5, 5 * 5 * 5 );  
printf( "%d\t%d\t%d\n", 6, 6 * 6, 6 * 6 * 6 );  
printf( "%d\t%d\t%d\n", 7, 7 * 7, 7 * 7 * 7 );  
printf( "%d\t%d\t%d\n", 8, 8 * 8, 8 * 8 * 8 );  
printf( "%d\t%d\t%d\n", 9, 9 * 9, 9 * 9 * 9 );  
printf( "%d\t%d\t%d\n", 10, 10 * 10, 10 * 10 * 10 );  
  
return 0;  
}
```

C语言中的算术运算

- 例：运行结果



```
C:\WINDOWS\sy...  
number  square  cube  
0       0      0  
1       1      1  
2       4      8  
3       9     27  
4      16     64  
5      25    125  
6      36    216  
7      49    343  
8      64    512  
9      81    729  
10     100   1000  
请按任意键继续. . .
```

做出判断：相等和关系运算符

- 相等运算符：优先级高于赋值运算符（=）
 - ==：相等
 - !=：不等
- 关系运算符：优先级高于相等运算符，小于算术运算符
 - >：大于
 - <：小于
 - >=：大于或者等于
 - <=：小于或者等于

做出判断：相等和关系运算符

- **if语句：** 允许程序通过判断一个被称为条件的表述式的真假来做出是否执行某个操作的决定
 - **if(condition){**
 - 语句;
 - **}**

2.6 做出判断：相等和关系运算符

- 1 /* Fig. 2.13 : fig02_13.c
- 2 using if statements, relational operators, and equality operators */
- 4 #include <stdio.h>
- 5
- 6 /* function main begins program execution */
- 7 int main(void)
- 8 {
- 9 int num1;
- 10 int num2;
- 11
- 12 printf("Enter two integers, and I will tell you\n");
- 13 printf("the relationships they satisfy: ");
- 15 scanf("%d%d", &num1, &num2);
- 16

做出判断：相等和关系运算符

- 17 `if(num1 == num2) {`
- 18 `printf("%d is equal to %d\n", num1, num2);`
- 19 `}`
- 21 `if(num1 != num2) {`
- 22 `printf("%d is not equal to %d\n", num1, num2);`
- 23 `}`
- 24
- 25 `if(num1 < num2) {`
- 26 `printf("%d is less than %d\n", num1, num2);`
- 27 `}`
- 28
- 29 `if(num1 > num2) {`
- 30 `printf("%d is greater than %d\n", num1, num2);`
- 31 `}`

做出判断：相等和关系运算符

- 32
- 33 `if(num1 <= num2) {`
- 34 `printf("%d is less than or equal to %d\n", num1, num2);`
- 35 `}`
- 29 `if(num1 > num2) {`
- 30 `printf("%d is greater than %d\n", num1, num2);`
- 31 `}`
- 32
- 33 `if(num1 <= num2) {`
- 34 `printf("%d is less than or equal to %d\n", num1, num2);`
- 35 `}`

做出判断：相等和关系运算符

- 图2.13执行结果（一）
 - Enter two integers, and I will tell you
 - the relationships they satisfy : **3 7**
 - 3 is not equal to 7
 - 3 is less than 7
 - 3 is less than or equal to 7

做出判断：相等和关系运算符

- 执行结果(二)
 - Enter two integers, and I will tell you
 - the relationships they satisfy : 22 12
 - 22 is not equal to 12
 - 22 is greater than 12
 - 22 is greater than or equal to 12

做出判断：相等和关系运算符

- 执行结果（三）
 - Enter two integers, and I will tell you
 - the relationships they satisfy : 7 7
 - 7 is equal to 7
 - 7 is less than or equal to 7
 - 7 is greater than or equal to 7

本章小结

- **C语言支持结构化和规范的程序设计方法。**
- **在程序执行时，注释不会引发计算机的任何操作。**
- **C程序都是从主函数main()开始执行的。**
- **凡是以#开头的行都是在程序被编译之前由预处理程序进行处理的。**
- **函数被调用执行时，可能要接收一些信息。函数执行后可以返回一些信息。**

- 课后作业:
- **第二章p48-49页2.19-2.31**

THANKS

C