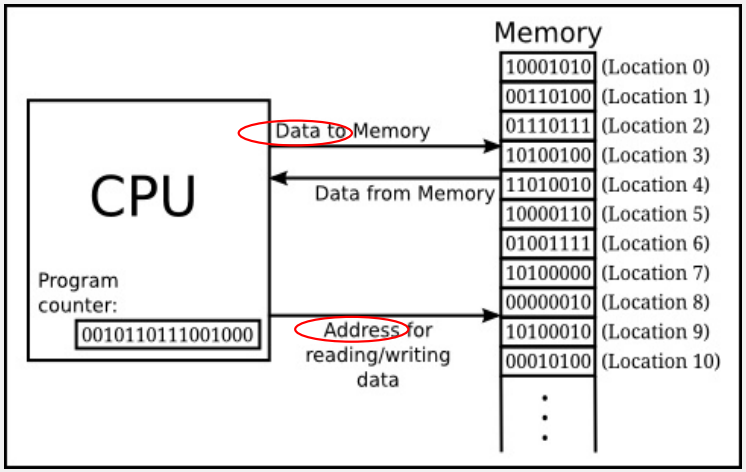
变量、常量和存储类（2）

中山大学计算机学院

讲课人：课程组

变量…值…内存…?




```
double radius = 5;
```

在计算机的内部:

- 5 变成0，1是什么样的?
- radius 与 &radius 的含义?
- double 和 int 类型的 5，在内存中一样吗?

目录

CONTENTS

中山大學
SUN YAT-SEN UNIVERSITY

整数及其表示

Integers and their representation

整数（Integers）与变量（Vars，Variables）

```c
/*unsigned int add*/
#include <stdio.h>

int main() {
    // Add two int.
    unsigned int i,j;
    i = 2;
    j = 3;
    // 详细说明见：https://zh.cppreference.com/w/c/io/fprintf
    printf("%u add %u equal %u\n",i,j,i+j);
    return 0;
}
```

无符号

unsigned int 申明（declaration）整数变量
i,j 两个变量。类似两个房间，标牌是i和j
i和j又称为变量名，是保存**值的代称**

赋值语句
= 是赋值运行，给i房间赋予数值 2

"%u add %u equal %u\n" 格式化模板
%u 其中 %是引导符，u是无符号整数
这个模板要填3个数字 i,j,i+j

不可控

将 j=3；修改为 j=-3； 请问输出结果是：

2 add 4294967293 equal 4294967295

数（Number）

- 自然数（Natural Numbers），无符号整数（unsigned integers）
  - Zero and any number obtained by repeatedly adding one to it.
  - Examples: 100, 0, 45645, 32
- 负数（Negative Numbers）
  - A value less than 0, with a – sign
  - Examples: -24, -1, -45645, -32
- 整数（Integers）
  - A natural number, a negative number, zero
  - Examples: 249, 0, - 45645, - 32
- 有理数（Rational Numbers）
  - An integer or the quotient of two integers
  - Examples: -249, -1, 0, 3/7, -2/5

自然数

用 “642”表示多少个一?

600 + 40 + 2 ?

也可能

384 + 32 + 2 ?

也可能是…

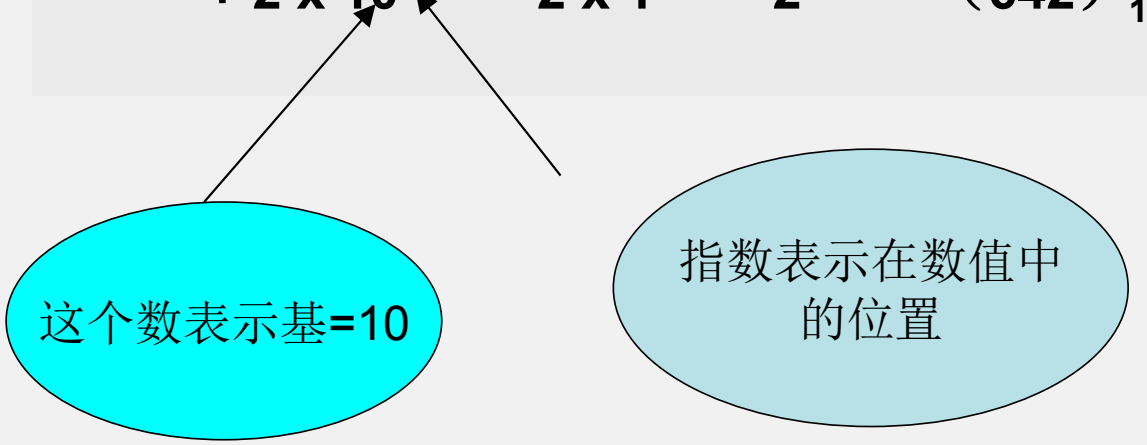
1536 + 64 + 2 ?

进位制/位值计数法（Positional Notation）

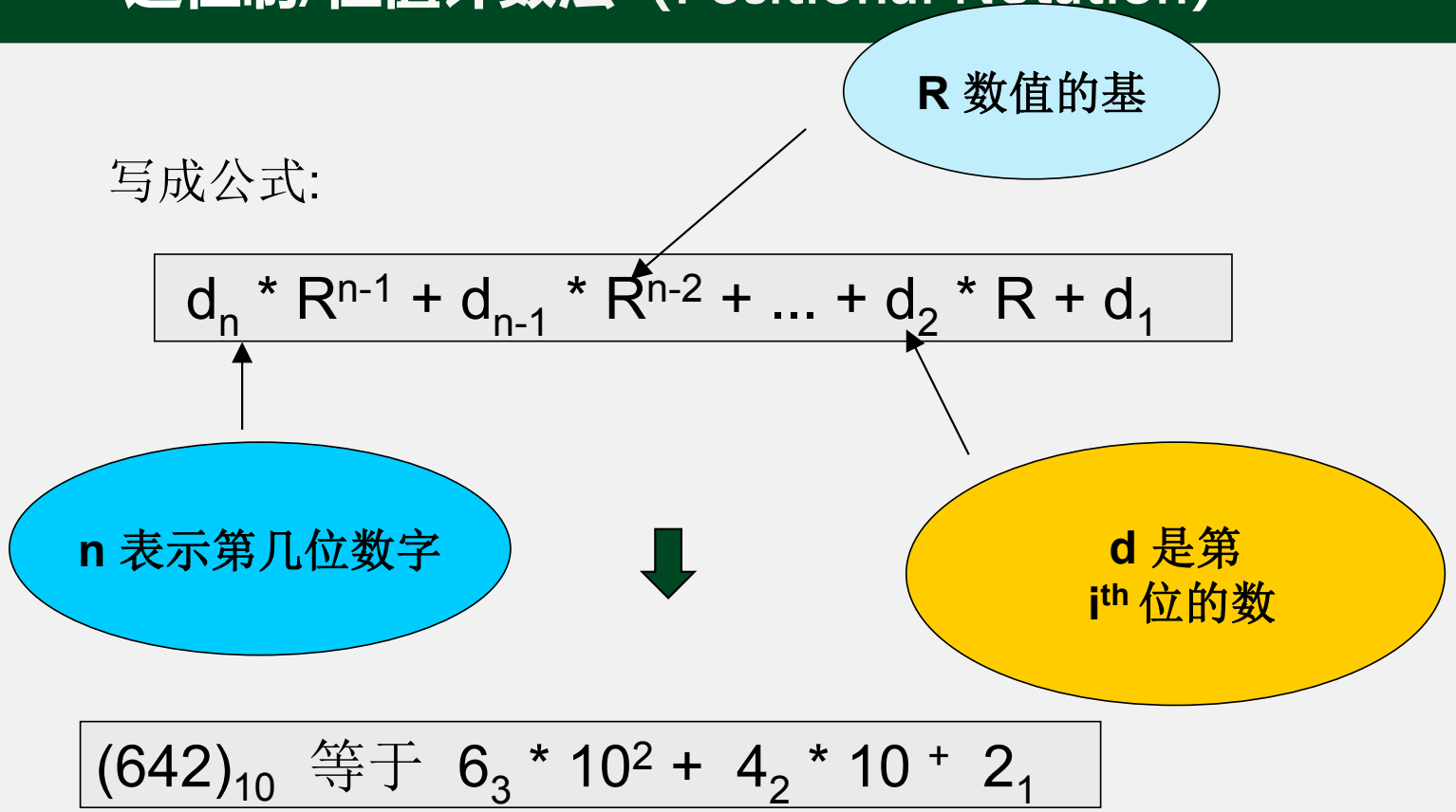
为什么…

642 用基（Base）为 10位值计数法:

$$
\begin{aligned}
6 \times 10^2 &= 6 \times 100 &&= 600 \\
+\ 4 \times 10^1 &= 4 \times 10 &&= 40 \\
+\ 2 \times 10^0 &= 2 \times 1 &&= 2 \quad = (642)_{10}
\end{aligned}
$$



进位制/位值计数法（Positional Notation）



进位制/位值计数法（Positional Notation）

假如 642 表示的数的基是 13?（13进制）

$$
\begin{aligned}
&+ 6 \times 13^2 = 6 \times 169 = 1014 \\
&+ 4 \times 13^1 = 4 \times 13 = 52 \\
&+ 2 \times 13^0 = 2 \times 1 = 2 \\
&= （1068）_{10}
\end{aligned}
$$

642 在基13下 等于 1068在基 10下的值
即同一数值在不同进制下的表示

二进制数（Binary Number）

- **十进制（Decimal）**的基（Base）是10，有10个数字：
  - 0,1,2,3,4,5,6,7,8,9
- **二进制（Binary）**的基是2，有2个数字：0b
  - 0,1
- **十六进制（Hexadecimal）**的基是16，有16个数字：0x
  - 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- **八进制（Octal）**的基是？，有？个数字：0
  - ?

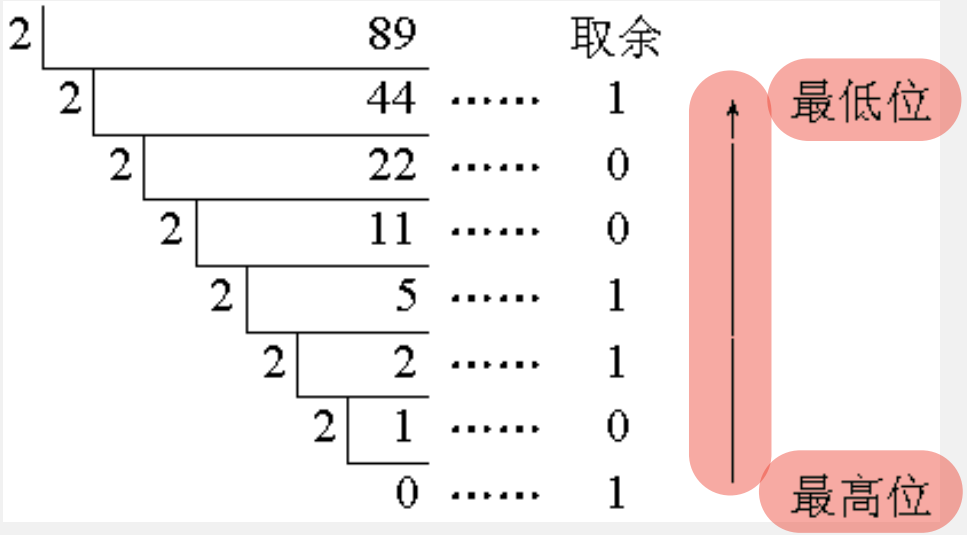
前缀

练习

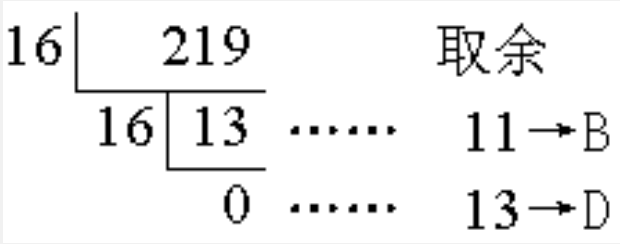
- 计算以下数的十进制值
- $(143)_8$

十进制转R进制

- **除R取余数倒排法**
- 例如：$(89)_{10} = (1011001)_2$



- 例如：$(219)_{10} = (DB)_{16}$



十进制转十六，八进制

- 十六进制数 DEF 等于？十进制数?

$$
\begin{aligned}
&D \times 16^2 = 13 \times 256 = 3328 \\
+\ &E \times 16^1 = 14 \times 16 = 224 \\
+\ &F \times 16^0 = 15 \times 1 = 15 \\
&\qquad\qquad = 3567
\end{aligned}
$$

in base 10

```c
/*int-hex-oct*/
#include <stdio.h>

int main() {
    unsigned int i = 3567;
    printf("Hex %X, Oct %o\n",i,i);
    return 0;
}
```

```
Hex DEF, Oct 6757
```

为什么我们不能用 “%b” 直接打印出二进制?

二进制和十六进制与八进制的关系

Binary	Octal	Hexa
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

Binary	Octal	Hexa
1000		8
1001		9
1010		A
1011		B
1100		C
1101		D
1110		E
1111		F

$(DEF)_{16} = (1101\ 1110\ 1111)_{2}$
$= (110\ 111\ 101\ 111)_{2}$
$= (6757)_{8}$

背，熟记！

练习

- 计算以下数的二进制值、八进制，十六进制的值
- 215

将十六进制与八进制数赋值给变量

```c
/*hex-oct-int*/
#include <stdio.h>

int main() {
    unsigned int i = 0xDEF;
    unsigned int j = 06757;
    printf("i in Decimal is %u\n",i);
    printf("j in Decimal is %u\n",j);
    return 0;
}
```

0x 开头表示是 十六进制的数值

0 开头表示是 八进制的数值

```
i in Decimal is 3567
j in Decimal is 3567
```

变量 i 能保存多大的数值，任意大小?

"-1"在内存中是一堆1

字节（Byte）和无符号数据类型

- **字节**（byte/unsigned char）
  - 8位二进制数
  - 表示范围 0 .. 255（2^8）
- **字**（word/ unsigned short）
  - 2个字节
  - 表示范围 0..65535（2^{16}）
- 无符号整数（unsigned integer）
- 无符号长整数（unsigned long）
  - 4个字节
  - 表示范围 0.. 4294967295（2^{32}）

```c
/*byte maxmum*/
#include <stdio.h>
#define UINT_MAX -1

typedef unsigned char byte;
typedef unsigned short word;

int main() {
    byte b = UINT_MAX;
    word w = UINT_MAX;
    unsigned int i = UINT_MAX;
    unsigned long l = UINT_MAX;
    printf("Maxmum byte is %X , %u\n",b,b);
    printf("Maxmum word is %X , %u\n",w,w);
    printf("Maxmum int is %X , %u\n",i,i);
    printf("Maxmum long is %X , %u\n",l,l);
    return 0;
}
```

替换

1. FF
2. FFFF

为什么可以用-1表示不同类型无符号整数的最大数值?

负数（Negtive）的表示

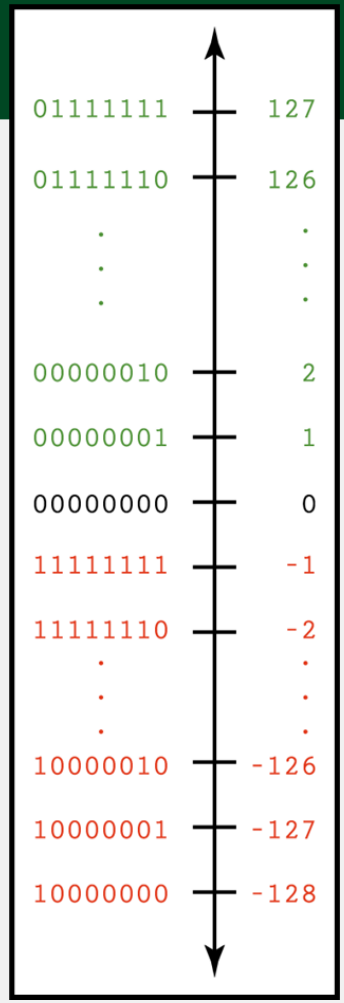
- 补（Complement ）
  - a+b = 常数，则称 a 是 b 的补，b 是 a 的补
- 二进制的补

$$Negtive(i) = 2^k - i$$

其中 ***k*** 是位数

- 例如： byte 的数
  - 负数 + 对应的正数 = 256
  - 负数第一位是1正数是0，所以它称为**符号位**
  - -1 即是从第k位借位减一，一定是全 111...111

在10 相反数 在2进制中相加恒为常数



二进制补的计算方法

- 二进制的补

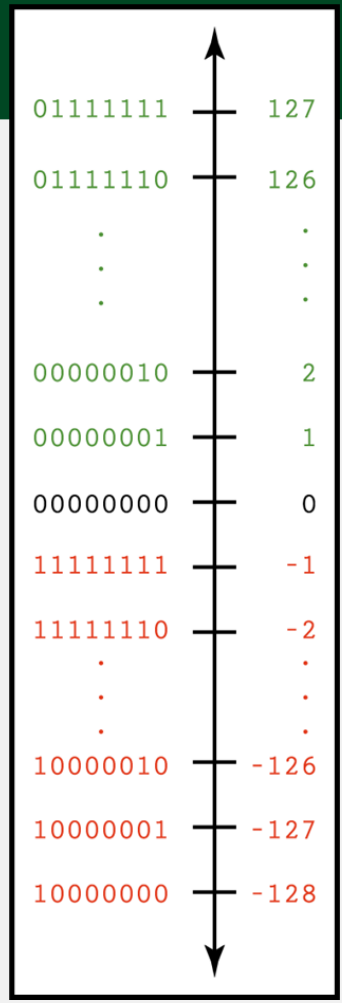
$$Negtive(i) = \left(2^k - 1\right) - i + 1$$

- 那么 22 = （00010110）$_2$的负数的表示是?

```
   1111,1111    -- 2^8 - 1
 - 0001,0110    -- i求二进制
   1110,1001    -- 按位求1的补
 +         1    -- 加一
   1110,1010    -- 补数-i
```

二进制数的负数（补数），总是等于按位求反，再加一。

在CPU中，只有加法电路，没有减法电路。 因为 a – b = a + (-b)



整数类型的特征

- 字符（char）
  - 1个字节
  - 表示范围 -128 .. 127
- 短整数（short）
  - 2个字节
  - 表示范围 -32768..32767
- 整数（integer）
- 长整数（long）
  - 4个字节
  - 表示范围 -2147483648..2147483647

```c
/*short-max-min*/
#include <stdio.h>
#define UINT_MAX -1
#define SHORT_MAX (unsigned short) UINT_MAX >> 1
#define SHORT_MIN ~SHORT_MAX

int main() {
    short i,j;
    i = 0x8000; // SHORT_MIN;
    j = 0x7FFF; // SHORT_MAX;
    printf("Minmum is %hX , %d\n",i,i);
    printf("Maxmum is %hX , %d\n",j,j);
    return 0;
}
```

参数大小的*长度修饰符*，*h*用于short，*l*用于long；

转换格式指定符，*d*用于整数，*u*用于无符号整数；

整数前缀

0x 十六进制 0 八进制 0b 二进制

练习

- 计算以下数的 char 类型的十六进制值
- -7

- 分别写出各个变量的八进制值
- char x=3; char y=-6; char z=x-y;

溢出（Overflow）（1）

```c
/*overflow*/
#include <stdio.h>

int main() {
    char k = 127;  //0x7F;
    char m = k + 10;
    char n = k + k + 10;
    printf("%d, %d, %d\n",k+10, m, n);
}
```

表达式是*运算符*及其*操作数*的序列，它指定一个运算。
https://zh.cppreference.com/w/c/language/expressions

输出是:

```
137, -119, 8
```

这都是什么？无数草泥马在风中狂奔!

这些表达式的结果是整数，当赋值给一个位数少的变量时，直接**抛弃高位**。例如:

- k+10 = 0x89，作为32位整数 137，作为8位整数-119
- k+k+10 =0x108, 赋值n时，仅保留0x08

溢出（Overflow） (2)

```c
/*overflow constant*/
#include <stdio.h>

int main() {
    char k = 255+1;
    printf("%X",k);
}
```

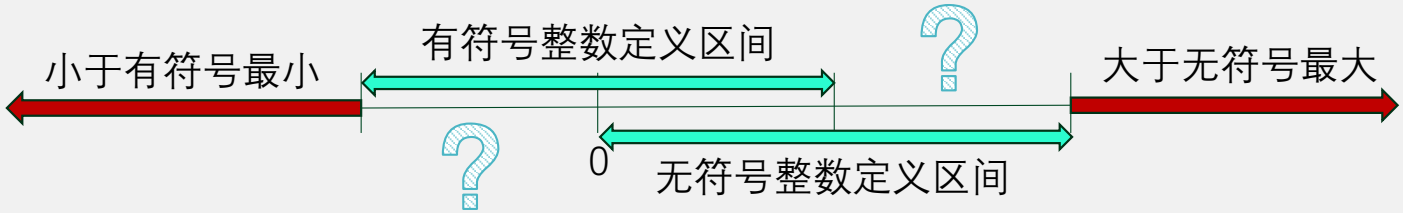
输出是:

```
100-test.c: In function 'main':
100-test.c:4:14: warning: overflow in
conversion from 'int' to 'char' changes value
from '256' to '0' [-Woverflow]
     char k = 255+1;
              ^~~
```

阅读编译错误提示。
如果忘记写该语句结束的分号，提示是什么?
如果你是copy-paste来的程序，包含汉字空格或分号，提示是什么?

溢出（Overflow）的定义与检出

- 整数溢出（Overflow）
  - 赋予变量的值超出变量类型定义的范围。例如 char i = -129
- C语言语法层面可对**常数赋值**进行溢出检查
  - 例如 i = -1；如果 i 是无符号数则是最大值



- 表达式赋值则按转换规则，而不会做任何检查
- C语言程序员有义务保证不发生溢出！

历史上最著名的缺陷（BUG）与溢出

- 1996年6月4日，对于Ariane 5火箭的初次航行来说，这样一个错误产生了灾难性的后果。发射后仅仅37秒，火箭偏离它的飞行路径，解体并爆炸了。6亿美元付之一炬。
- 错误分析：
  - during execution of a **data conversion** from **64-bit floating point** to **16-bit signed integer value**. The floating point number which was converted had a value greater than what could be represented by a 16-bit signed integer. This resulted in an Operand Error.

参考阅读：C语言的整型溢出问题 | 酷 壳 - CoolShell

整型常量（Constants）的书写形式

- 允许整数类型的值直接用于表达式。
- 语法
  - 十进制常量 整数后缀(可选)
  - 八进制常量 整数后缀(可选)
  - 十六进制常量 整数后缀(可选)
- 整数后缀
  - 无后缀
  - u 或 U （无符号数）
  - l 或 L （长整型）
  - lu 或 LU （无符号长整型）
  - ll 或 LL （长长整型）

```
int d = 42;
int o = 052;
int x = 0x2a;
int X = 0X2A;
long l = 420L;
long long ll = 052LU;
```

详细参考：
整数常量 - cppreference.com

前缀：0b 二进制
0 八进制
0x 十六进制

浮点数及其表示（选讲）

Floatings and their representation

浮点数（floating number）与格式化

```c
/*floating format*/
#include <stdio.h>

int main() {
    double x,y;
    x = -2.263;
    y = 13.367;
    double z = x + y;
    // 详细说明见:
https://zh.cppreference.com/w/c/io/fprintf
    printf("%-8.2f add %08.2f equal %.4e\n",x,y,z);
    return 0;
}
```

double 表示浮点类型，这里定义了变量 x 和 y

变量赋值

精度4位，科学计算法输出

用0补齐，宽度8，精度2位，浮点数输出

左对齐，宽度8，精度2位，浮点数输出

输出是:

```
-2.26    add 00013.37 equal 1.1104e+001
```

浮点数数据类型与常量表示

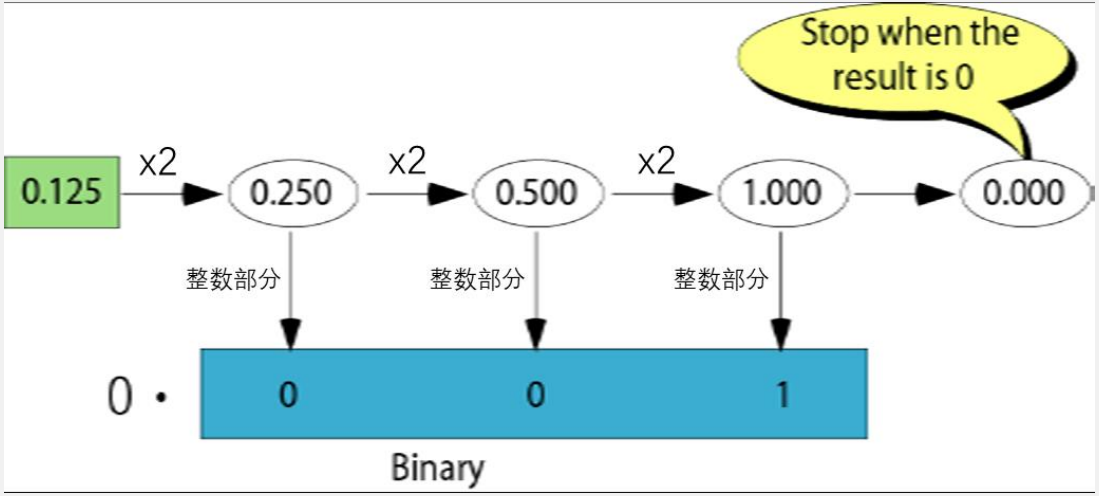
- 双精度数（double）
  - 8个字节（64位）
  - 有效数字 15
  - 指数部分 10^{-307}~10^{308}
- 单精度数（float）
  - 4个字节（32位）
  - 有效数字 6
  - 指数部分 10^{-37}~10^{38}

```
double d = 42.0;
double o = 4.2e1;
float x = 42.0f;
float y = 4.2e1f;
```

详细参考：
浮点常量 - cppreference.com
算术类型 - cppreference.com

小数（Fraction part）转二进制

基本原理与方法：



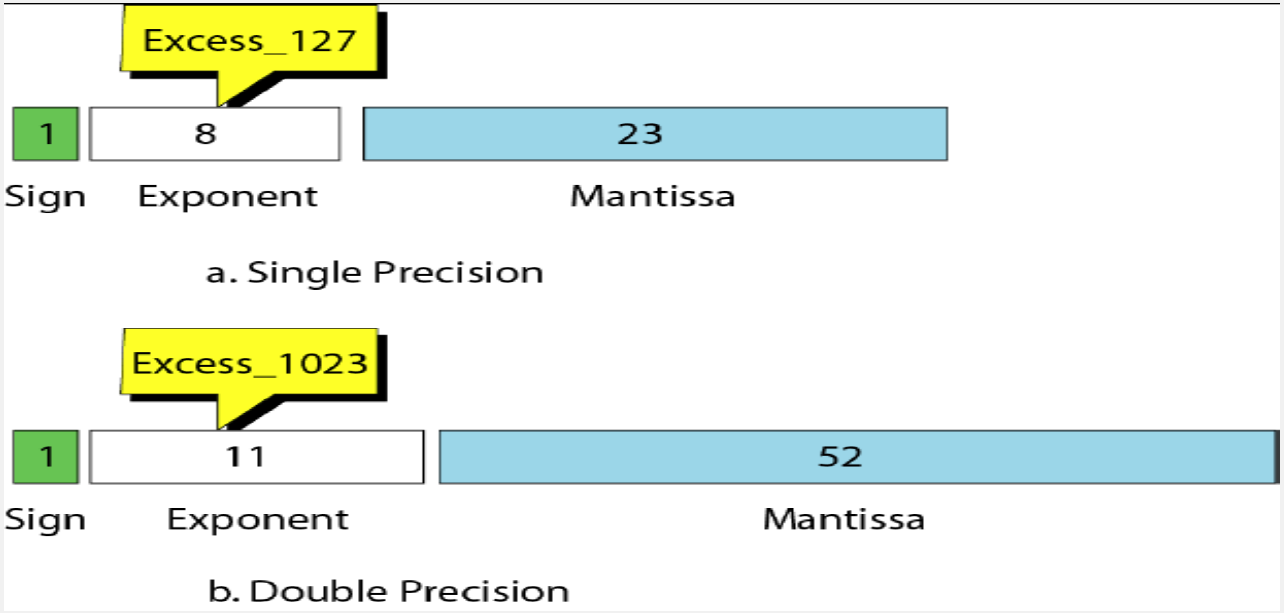

```
   0.15
x2 0.30 -- 0
x2 0.60 -- 0
x2 1.20 -- 1 *
x2 0.40 -- 0 *
x2 0.80 -- 0 *
x2 1.60 -- 1 *
x2 1.20 -- 1 …
```

$0.15 = (0.00100110011001\ldots)_2$

以此，只要用二进制表示小数部分，绝大多数十进制数转二进制必然产生舍入误差（精度误差）

IEEE 745-2008 标准与浮点数内存格式（了解）

单精度数与双精度数在内存中的存储格式：



实数转为二进制表示：

1. 把实数的整数和小数部分分别转为二进制。
2. 表示成二进制科学计数法。
3. 按IEEE规范，计算符号位，指数部分，小数部分。

例如：0.15

= 1.0011 0011 0011 0011 0011
  001[1] * 2^{-3}

注意：四舍五入

```
float x = 0.15f;
```

按IEEE 745规范：

```
Sign  0
Exp  127-3  即 0x7D = 0111 1100
Manti 0011,0011,0011,0011,0011,010
```

结果：

```
0011 1110 0001 1001
1001 1001 1001 1010
x 的二进制存储是 0x3E19999A
```

print-float.c

溢出（Overflow）和精度误差（accuracy error）

```c
/*floating accuracy*/
#include <stdio.h>

int main(void) {
    double x = 2.0e+307;
    x = x * 10;
    // 超出 double 范围的常量。
    printf("+2.0e+308 --> %e\n",  x);

    float y = 123456789.0f;
    printf("123456789f --> %.8e\n",  y);
    y = y + 1;
    printf("123456790f --> %.8e\n",  y);
    y = y + 1;
    //此处省去100句 y = y + 1;
    y = y + 1;
    printf("1234568XXf --> %.8e\n",  y);
}
```

```
+2.0e+308 --> 1.#INF00e+000
123456789f --> 1.23456792e+008
123456790f --> 1.23456792e+008
1234568XXf --> 1.23456792e+008
```

认识用有限位数表示数据产生的问题，
才能编出正确可靠的程序!

中山大學
SUN YAT-SEN UNIVERSITY

变量在计算机内部

内存（Memory）及其组织

- 内存是计算机存储信息的电路
- **位（bit，比特）**：
  - 存储1位二进制数
- **字节（Byte）**
  - 最基本的存储单元
  - 包含 8 个 bit
- **地址（Address）**
  - 存储单元的编码
- **地址空间（ Address Space）**
  - 0 .. 2^n；例如：n=32 （4G）

Address	Contents
00000000	11100011
00000001	10101001
⋮	⋮
11111100	00000000
11111101	11111111
11111110	10101010
11111111	00110011

内存容量

- 内存是地址用二进制表示。
- 每个内存存储单元使用一个地址
- 内存容量的单位

Unit	*Number of bytes*	*Approximation*
Kilobyte (K)	2^{10} bytes	10^{3} bytes
Megabyte (M)	2^{20} bytes	10^{6} bytes
Gigabyte (G)	2^{30} bytes	10^{9} bytes
Terabyte (T)	2^{40} bytes	10^{12} bytes

- 例如：8G内存，表示有2^{33}个地址或存储单元或Bytes

Var-Address-Value

- 每个变量都有类型、地址、值等属性。

```
/*vars-address-value*/
#include<stdio.h>

int main() {
        int i = 5;
        float x = 5.0f;

        printf("Addr:%p bin value:%#010x\n",&i,i);
        printf("Addr:%p bin value:%#010x\n",&x,*(int *)&x);

        return 0;
}
```

把整数和浮点变量的值改为负数，其二进制内容是…

地址*	值	
62FE18	00	x
19	00	
1A	A0	
1B	40	
62FE1C	05	i
1D	00	
1E	00	
1F	00	

*地址随运行环境浮动

中山大學
SUN YAT-SEN UNIVERSITY

谢 谢

中山大学计算机学院

中山大学MOOC课程组