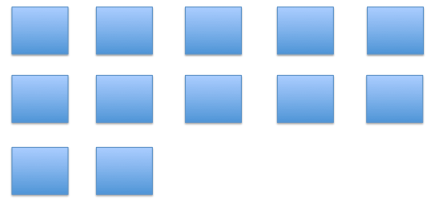
# 进制

## 什么是进制

* 是一种计数的方式，数值的表示形式

数一下方块的个数



汉字：十一 十进制：11 二进制：1011 八进制：13

* 多种进制：十进制、二进制、八进制、十六进制。也就是说，同一个整数，我们至少有4种表示方式
* 软件开发，肯定要了解这个

## 二进制

1. 特点：只有0和1，逢2进1
2. 书写格式：0b或者0b开头
3. 使用场合：二进制指令\二进制文件，变量在内存中就是二进制存储
4. 二进制和十进制的互相转换
5. n为二进制位所能表示的数据范围（不考虑负数）：0~2的n次方-1

## 八进制

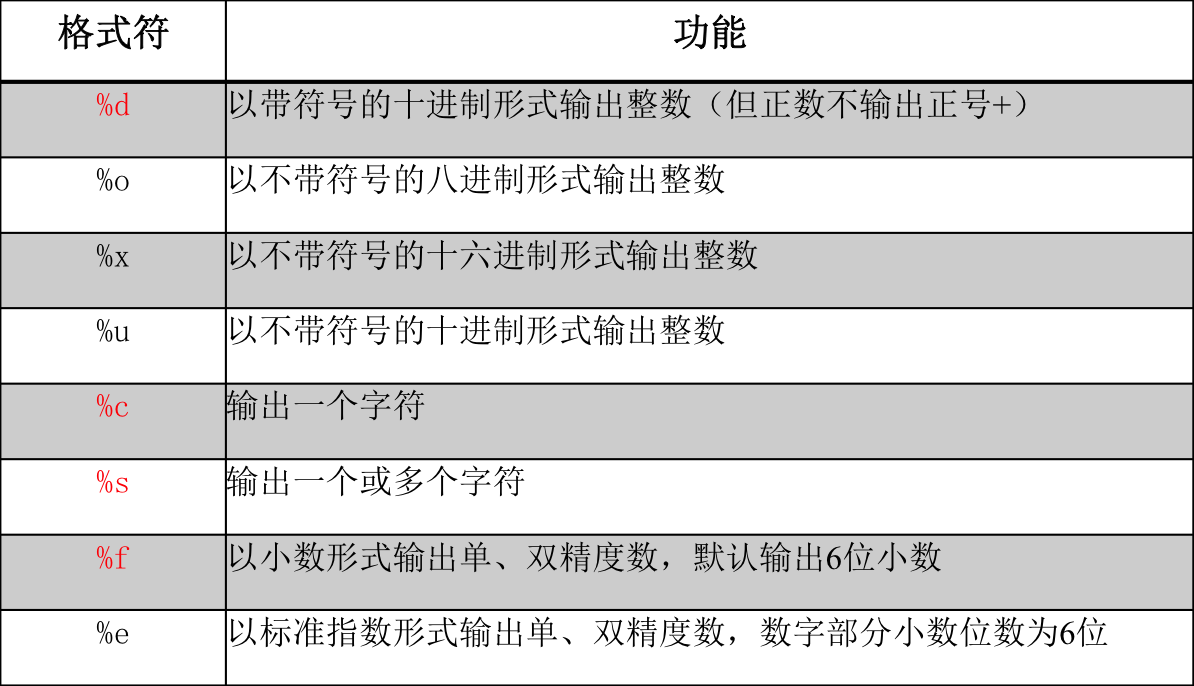
1. 特点：0~7，逢八进一
2. 书写格式：0开头
3. 八进制和二进制的互相转换

## 十六进制

1. 特点：0~F，逢十六进一
2. 书写格式：0x或者0X开头
3. 十六进制和二进制的互相转换

## 总结

1. Mac中计算器的使用
2. printf以不同进制形式进行输出



## 习题

1> 判断下列数字是否合理

00011 0x0011 0x7H4 10.98 0986 .089 -109

+178 0b325 0b0010 0xFFdc 96f 96.0f 96.0F

-.003 15.4e6 10e8.7 7.6e-6

2> 分别写出它们的十进制、八进制、十六进制

0b0011 1101 0b0111 1011

3> 写出它们的二进制

67 056 0x004f

# 变量的内存分析

研究变量在内存中的具体存储情况

## 字节和地址

为了更好地理解变量在内存中的存储细节，先来认识一下内存中的“字节”和“地址”。

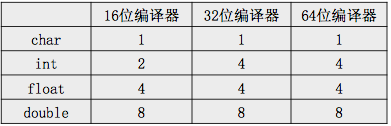
1. 内存以“字节为单位”



1. 不同类型占用的字节是不一样的

## 变量的存储

1. 所占用字节数跟类型有关，也跟编译器环境有关



1. 变量实例

int b = 10;

int a = 134;

* 内存由大到小寻址
* 只存储二进制形式
* 每个变量都有地址：第一个字节的地址就是变量的地址

1. 查看内存地址的两种方式：%x和%p
2. 查看整数的二进制形式

// 输出整数的二进制形式

void putBinary(int n)

{

int bits = sizeof(n) \* 8;

while (bits-->0) {

printf("%d", n>>bits&1);

if (bits%4==0) printf(" ");

}

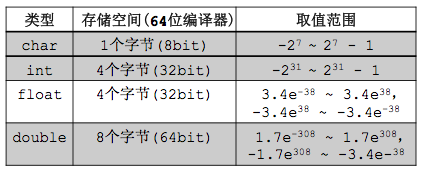
printf("\n");

}

## 负数在内存中的存储

1. 一个字节的取值范围
2. 负数的表示形式
3. 原码、反码、补码

## 取值范围



## 习题

写出下列变量在内存中的存储情况

int a = 134;

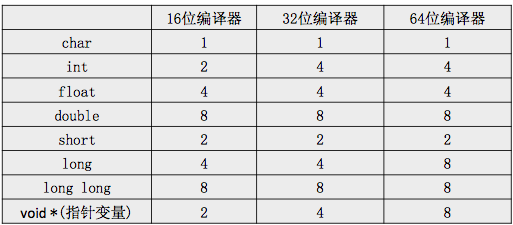
int b = 0;

int c = -10;

# 类型说明符

## short和long

* 100l和100ll和100的区别
* long和long long的输出
* 不同类型所占用的存储空间



1> short和long可以提供不同长度的整型数，也就是可以改变整型数的取值范围。在64bit编译器环境下，int占用4个字节（32bit），取值范围是-231~231-1；short占用2个字节（16bit），取值范围是-215~215-1；long占用8个字节（64bit），取值范围是-263~263-1

2> 总结一下：在64位编译器环境下，short占2个字节(16位)，int占4个字节(32位)，long占8个字节(64位)。因此，如果使用的整数不是很大的话，可以使用short代替int，这样的话，更节省内存开销。

3> 世界上的编译器林林总总，不同编译器环境下，int、short、long的取值范围和占用的长度又是不一样的。比如在16bit编译器环境下，long只占用4个字节。不过幸运的是，ANSI \ ISO制定了以下规则：

* short跟int至少为16位(2字节)
* long至少为32位(4字节)
* short的长度不能大于int，int的长度不能大于long
* char一定为为8位(1字节)，毕竟char是我们编程能用的最小数据类型

4> 可以连续使用2个long，也就是long long。一般来说，long long的范围是不小于long的，比如在32bit编译器环境下，long long占用8个字节，long占用4个字节。不过在64bit编译器环境下，long long跟long是一样的，都占用8个字节。

5> 还有一点要明确的是：short int等价于short，long int等价于long，long long int等价于long long

## signed和unsigned

1> 首先要明确的：signed int等价于signed，unsigned int等价于unsigned

2> signed和unsigned的区别就是它们的最高位是否要当做符号位，并不会像short和long那样改变数据的长度，即所占的字节数。

* signed：表示有符号，也就是说最高位要当做符号位，所以包括正数、负数和0。其实int的最高位本来就是符号位，已经包括了正负数和0了，因此signed和int是一样的，signed等价于signed int，也等价于int。signed的取值范围是-231 ~ 231 - 1
* unsigned：表示无符号，也就是说最高位并不当做符号位，所 以不包括负数。在64bit编译器环境下面，int占用4个字节（32bit），因此unsigned的取值范围是：0000 0000 0000 0000 0000 0000 0000 0000 ~ 1111 1111 1111 1111 1111 1111 1111 1111，也就是0 ~ 232 - 1

# 位运算

## & 按位与

1. 功能

只有对应的两个二进位均为1时，结果位才为1，否则为0。

1. 举例: 比如9&5，其实就是1001&101=1，因此9&5=1
2. 规律

* 二进制中，与1相&就保持原位，与0相&就为0

## | 按位或

1. 功能

只要对应的二个二进位有一个为1时，结果位就为1，否则为0。

1. 举例: 比如9|5，其实就是1001|101=1101，因此9|5=13

## ^ 按位异或

1> 功能

当对应的二进位相异（不相同）时，结果为1，否则为0。

2> 举例: 比如9^5，其实就是1001^101=1100，因此9^5=12

3> 规律

* 相同整数相^的结果是0。比如5^5=0
* 多个整数相^的结果跟顺序无关。比如5^6^7=5^7^6
* 因此得出结论：a^b^a = b

## ~ 取反

对整数a的各二进位进行取反，符号位也取反（0变1，1变0）

## << 左移

* 把整数a的各二进位全部左移n位，高位丢弃，低位补0。左移n位其实就是乘以2的n次方
* 由于左移是丢弃最高位，0补最低位，所以符号位也会被丢弃，左移出来的结果值可能会改变正负性

## >> 右移

* 把整数a的各二进位全部右移n位，保持符号位不变。右移n位其实就是除以2的n次方
* 为正数时， 符号位为0，最高位补0
* 为负数时，符号位为1，最高位是补0或是补1 取决于编译系统的规定

## 习题

1. 在不用引入其他变量的情况下，使用位异或^运算符实现两个变量值的互换
2. 使用位与&运算符变量的奇偶性
3. 编写一个函数，用来输出整数在内存中的二进制形式

# char类型

## 存储细节

ASCII单字节表（双字节GBK\GB2312\GB18030\Unicode）

## 常见错误

char c = A;

char c = "A";

char c = 'ABCD';

char c = '男';

## 当做整型使用

在-128~127范围内，可以当做整数来用

## %c和%d\%i的使用

printf(“%d”, ‘A’);

printf(“%c”, 68);

## 转义字符

|  |  |  |
| --- | --- | --- |
| **转义字符** | **意义** | **ASCII码值** |
| \n | 将当前位置移到下一行开头（回车换行） | 10 |
| \t | 跳到下一个TAB位置 | 9 |
| \\ | 代表一个反斜线字符 | 92 |
| \' | 代表一个单引号字符 | 39 |
| \" | 代表一个双引号字符 | 34 |
| \0 | 空字符 | 0 |

## 习题

1. 编写一个函数，将小写字母转为大写
2. 说出程序的输出结果

int main()

{

int i = 67 + '4';

char c = 'c' - 10;

printf("%d - %c\n", i, i);

printf("%d - %c\n", c, c);

return 0;

}