

# § . 基础知识题 – 浮点数机内存存储格式(IEEE 754)理解



要求:

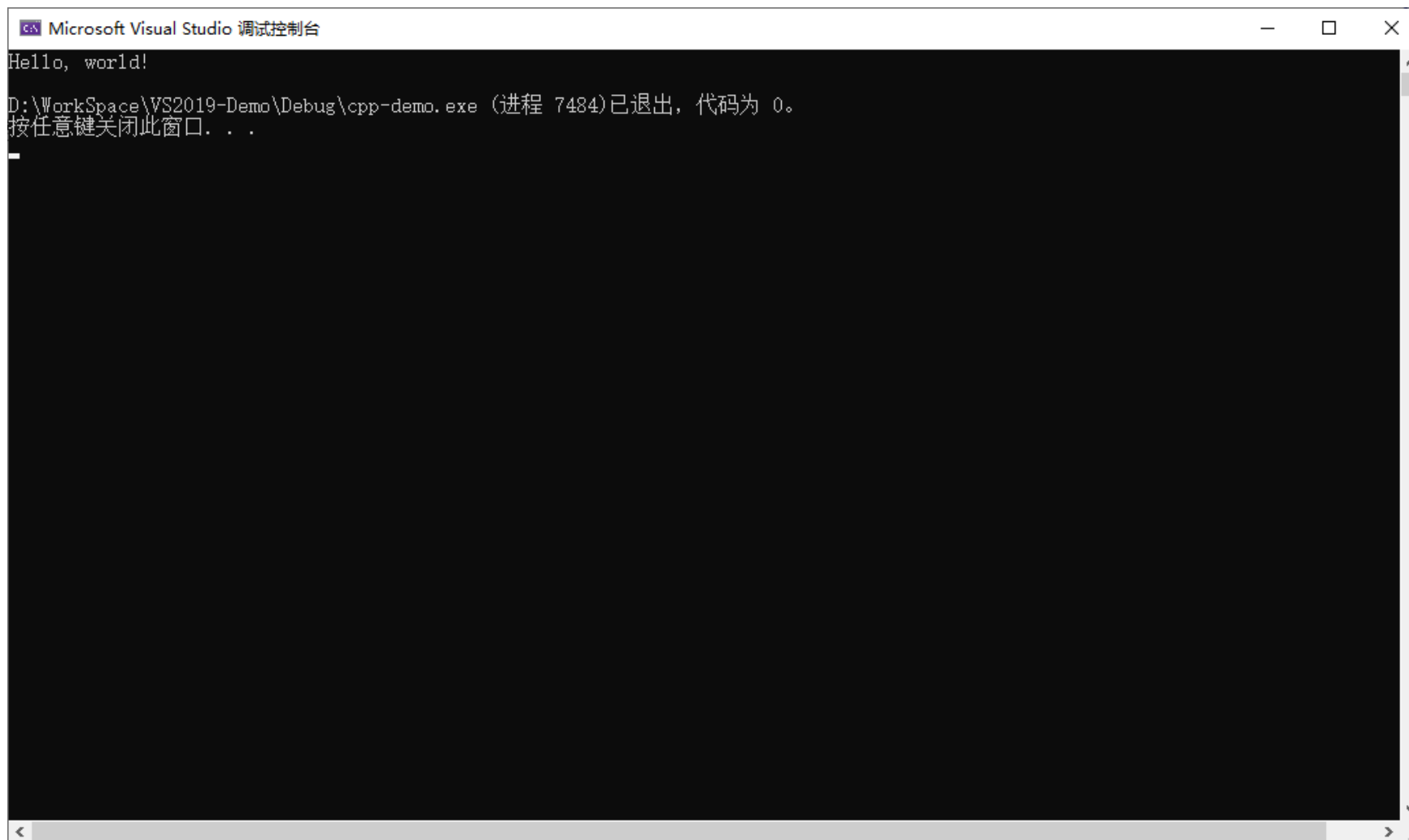
- 1、完成本文档中所有的题目并写出分析、运行结果
- 2、无特殊说明，均使用VS2022编译即可
- 3、直接在本文件上作答，**写出答案/截图（不允许手写、手写拍照截图）**即可；填写答案时，为适应所填内容或贴图，**允许调整**页面的字体大小、颜色、文本框的位置等
  - ★ 贴图要有效部分即可，不需要全部内容
  - ★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可
  - ★ **不允许**手写在纸上，再拍照贴图
  - ★ **允许**在各种软件工具上完成（不含手写），再截图贴图
- 4、转换为pdf后提交
- 5、**3月14日前**网上提交本次作业（在“文档作业”中提交）

# §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

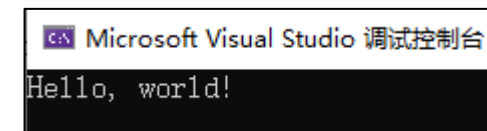


贴图要求：只需要截取输出窗口中的有效部分即可，如果全部截取/截取过大，则视为无效贴图

例：无效贴图

A screenshot of the Microsoft Visual Studio debug console window. The window is titled "Microsoft Visual Studio 调试控制台". It contains the text "Hello, world!" followed by a message indicating that the program "D:\Workspace\VS2019-Demo\Debug\cpp-demo.exe (进程 7484) 已退出, 代码为 0." and a prompt "按任意键关闭此窗口. . .". The window is large and shows the full context of the output.

例：有效贴图

A screenshot of the Microsoft Visual Studio debug console window, cropped to show only the "Hello, world!" output. The window title "Microsoft Visual Studio 调试控制台" is visible at the top.



## §. 基础知识题 - 浮点数机内存储格式(IEEE 754)理解

附：用WPS等其他第三方软件打开PPT，将代码复制到VS2022中后，如果出现类似下面的**编译报错**，则观察源程序编辑窗

的右下角是否为CR，如果是，单击CR，在弹出中选择CRLF，再次CTRL+F5运行即可

```
demo.cpp + x
demo-CPP (全局范围)
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     cout << "Hello, World." << endl;
7     return 0;
8 }
9
```

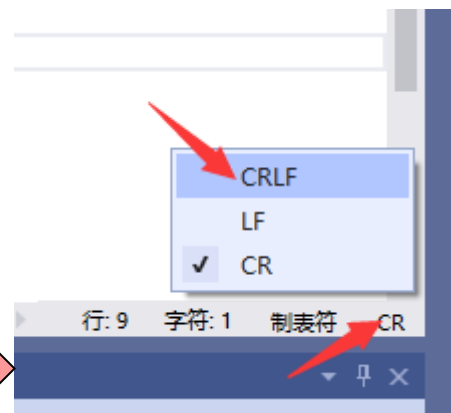
100 % 未找到相关问题 行: 9 字符: 1 制表符 CR

输出

显示输出来源(S): 生成

```
1>—— 已启动生成: 项目: demo-CPP, 配置: Debug Win32 ——
1>demo.cpp
1>D:\WorkSpace\VS2019-demo\demo-CPP\demo.cpp(1,1): warning C4335: 检测到 Mac 文件格式: 请将源文件转换为 DOS 格式或 UNIX 格式
1>D:\WorkSpace\VS2019-demo\demo-CPP\demo.cpp(1,10): warning C4067: 预处理器指令后有意外标记 - 应输入换行符
1>MSVCRTD.lib(exe_main.obj) : error LNK2019: 无法解析的外部符号 _main, 函数 "int __cdecl invoke_main(void)" (?invoke_main@YAHXZ) 中
1>D:\WorkSpace\VS2019-demo\Debug\demo-CPP.exe : fatal error LNK1120: 1 个无法解析的外部命令
1>已完成生成项目 "demo-CPP.vcxproj" 的操作 - 失败。
```

输出 错误列表



# §. 基础知识题 - 浮点数机内存储格式(IEEE 754)理解



基础知识：用于看懂float型数据的内部存储格式的程序如下：

**注意：**除了对黄底红字的具体值进行改动外，其余部分不要做改动，也暂时不需要弄懂为什么（需要第6章的知识才能弄懂）

```
#include <iostream>
using namespace std;
int main()
{
    float f = 123.456;
    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)*(p+1) << endl;
    cout << hex << (int)*(p+2) << endl;
    cout << hex << (int)*(p+3) << endl;
    return 0;
}
```

**//注：忽略本题出现的warning**

Microsoft  
79  
e9  
f6  
42

上例解读：单精度浮点数123.456，在内存中占四个字节，四个字节的值依次为0x42 0xf6 0xe9 0x79（按打印顺序逆向取）

转换为32bit则为：0100 0010 1111 0110 1110 1001 0111 1001

符号位

8位指数

23位尾数

# §. 基础知识题 - 浮点数机内存储格式(IEEE 754)理解



基础知识：用于看懂double型数据的内部存储格式的程序如下：

**注意：**除了对黄底红字的具体值进行改动外，其余部分不要做改动，也暂时不需要弄懂为什么（需要第6章的知识才能弄懂）

```
#include <iostream>
using namespace std;
int main()
{
    double d = 1.23e4;
    unsigned char* p = (unsigned char*)&d;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)*(p+1) << endl;
    cout << hex << (int)*(p+2) << endl;
    cout << hex << (int)*(p+3) << endl;
    cout << hex << (int)*(p+4) << endl;
    cout << hex << (int)*(p+5) << endl;
    cout << hex << (int)*(p+6) << endl;
    cout << hex << (int)*(p+7) << endl;
    return 0;
}
```

Microsoft  
0  
0  
0  
0  
6  
c8  
40

上例解读：双精度浮点数1.23e4，在内存中占八个字节，八个字节的值依次为0x40 0xc8 0x06 0x00 0x00 0x00 0x00 0x00(逆向)

转换为64bit则为：0100 0000 1100 1000 0000 0100 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

符号位

11位指数

52位尾数

# § . 基础知识题 – 浮点数机内存存储格式(IEEE 754)理解



自学内容：自行以“IEEE754” / “浮点数存储格式” / “浮点数存储原理” / “浮点数存储方式”等关键字，在网上搜索相关文档，读懂并了解浮点数的内部存储机制

学长们推荐的网址：

<https://baike.baidu.com/item/IEEE%20754/3869922?fr=aladdin>

<https://zhuanlan.zhihu.com/p/343033661>

[https://www.bilibili.com/video/BV1iW41ld7hd?is\\_story\\_h5=false&p=4&share\\_from=ugc&share\\_medium=android&share\\_plat=android&share\\_session\\_id=e12b54be-6ffa-4381-9582-9d5b53c50fb3&share\\_source=QQ&share\\_tag=s\\_i&timestamp=1662273598&unique\\_k=AuouMEO](https://www.bilibili.com/video/BV1iW41ld7hd?is_story_h5=false&p=4&share_from=ugc&share_medium=android&share_plat=android&share_session_id=e12b54be-6ffa-4381-9582-9d5b53c50fb3&share_source=QQ&share_tag=s_i&timestamp=1662273598&unique_k=AuouMEO)

[https://blog.csdn.net/gao\\_zhennan/article/details/120717424](https://blog.csdn.net/gao_zhennan/article/details/120717424)

<https://www.h-schmidt.net/FloatConverter/IEEE754.html>



# §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

例: float型数的机内表示

格式要求: 多字节时, 每8bit中间加一个空格或- (例: "11010100 00110001" 或 "11010100-00110001")

例1: 100.25

下面是float机内存储手工转十进制的方法:

(1) 得到的32bit的机内表示是: 0100 0010 1100 1000 1000 0000 0000 0000 (42 c8 80 00)

(2) 其中: 符号位是 0

指数是 1000 0101 (填32bit中的原始形式)

指数转换为十进制形式是 133 (32bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 6 (32bit中的原始形式按IEEE754的规则转换)

1000 0101

- 0111 1111

= 0000 0110 (0x06 = 6)

尾数是 100 1000 1000 0000 0000 0000 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.56640625 (32bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.56640625 (加整数部分的1后)

100 1000 1000 0000 0000 0000 =  $2^0 + 2^{-1} + 2^{-4} + 2^{-8}$

= 0.5 + 0.0625 + 0.00390625 = 0.56640625 => 加1 => 1.56640625

1.56640625 x  $2^6$  = 100.25 (此处未体现出误差)

下面是十进制手工转float机内存储的方法:

100 = 0110 0100 (整数部分转二进制为7位, 最前面的0只是为了8位对齐, 可不要)

0.25 = 01 (小数部分转二进制为2位)

100.25 = 0110 0100.01 = 1.1001 0001 x  $2^6$  (确保整数部分为1, 移6位)

符号位: 0

阶码: 6 + 127 = 133 = 1000 0101

尾数(舍1): 1001 0001 => 1001 0001 0000 0000 0000 000 (补齐23位, 后面补14个蓝色的0)

100 1000 1000 0000 0000 0000 (从低位开始四位一组, 共23位)

注意:

- 1、作业中绿底/黄底文字/截图可不填
- 2、计算结果可借助第三方工具完成, 没必要完全手算

本页不用作答





# §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

例: float型数的机内表示

格式要求: 多字节时, 每8bit中间加一个空格或- (例: "11010100 00110001" 或 "11010100-00110001")

例2: 1.2

下面是float机内存储手工转十进制的方法:

(1) 得到的32bit的机内表示是: 0011 1111 1001 1001 1001 1001 1010 (3f 99 99 9a)

(2) 其中: 符号位是 0

指数是 0111 1111 (填32bit中的原始形式)

指数转换为十进制形式是 127 (32bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 0 (32bit中的原始形式按IEEE754的规则转换)

0111 1111

- 0111 1111

= 0000 0000 (0x0 = 0)

尾数是 001 1001 1001 1001 1010 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.2000000476837158203125 (32bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.2000000476837158203125 (加整数部分的1后)

001 1001 1001 1001 1010 =  $2^{-3} + 2^{-4} + 2^{-7} + 2^{-8} + 2^{-11} + 2^{-12} + 2^{-15} + 2^{-16} + 2^{-19} + 2^{-20} + 2^{-22}$

= 0.125 + ... + 0.0000002384185791015625 (详见右侧蓝色) = 0.2000000476837158203125

=> 加1 = 1.2000000476837158203125 (此处已体现出误差)

下面是十进制手工转float机内存储的方法:

1 = 1 (整数部分转二进制为1位)

0.2 = 0011 0011 0011 0011 0011 0011 (小数部分无限循环, 转为二进制的24位)

=> 0011 0011 0011 0011 0011 010 (四舍五入为23位, 此处体现出误差)

1.2 = 1.0011 0011 0011 0011 0011 010 = 1.0011 0011 0011 0011 0011 010 x  $2^0$  (确保整数部分为1, 移0位)

符号位: 0

阶码: 0 + 127 = 127 = 0111 1111

尾数(舍1): 0011 0011 0011 0011 0011 010 (共23位)

001 1001 1001 1001 1001 1010 (从低位开始四位一组, 共23位)

注意:

- 1、作业中绿底/黄底文字/截图可不填
- 2、计算结果可借助第三方工具完成, 没必要完全手算

0.125 +  
0.0625 +  
0.0078125 +  
0.00390625 +  
0.00048828125 +  
0.000244140625 +  
0.000030517578125 +  
0.0000152587890625 +  
0.0000019073486328125 +  
0.00000095367431640625 +  
0.0000002384185791015625

0.2000000476837158203125

本页不用作答





# §. 基础知识题 - 浮点数机内存储格式(IEEE 754)理解

## 1、float型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或- (例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”)

A. 2352495. 5942532 (此处设学号是1234567，需换成本人学号，小数为学号逆序，非本人学号0分，下同!!!)

注：尾数为正、指数为正

(1) 得到的32bit的机内表示是： 0100 1010 0000 1111 1001 0101 1011 1110 (4a 0f 95 be)

(2) 其中：符号位是 0

指数是 1001 0100 (填32bit中的原始形式)

指数转换为十进制形式是 148 (32bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 21 (32bit中的原始形式按IEEE754的规则转换)

尾数是 000 1111 1001 0101 1011 1110 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.1217572689056396484375 (32bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.1217572689056396484375 (加整数部分的1)

注：转换为十进制小数用附加的工具去做，自己去网上找工具也行，但要满足精度要求 (下同!!!)



## §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

### 1、float型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或- (例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”)

B. -5942532.2352495 (设学号为1234567，按规则更换为学号和学号逆序)

注：尾数为负、指数为正

(1) 得到的32bit的机内表示是： 1100 1010 1011 0101 0101 1010 0000 1000 (ca b5 5a 08)

(2) 其中：符号位是 1

指数是 1001 0101 (填32bit中的原始形式)

指数转换为十进制形式是 149 (32bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 22 (32bit中的原始形式按IEEE754的规则转换)

尾数是 011 0101 0101 1010 0000 1000 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.41681003570556640625 (32bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.41681003570556640625 (加整数部分的1)



# §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

## 1、float型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或- (例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”)

C. 0.002352495 (设学号为1234567，按规则更换为学号和学号逆序)

注：尾数为正、指数为负

(1) 得到的32bit的机内表示是： 0011 1011 0001 1010 0010 1100 0101 0001 (3b 1a 2c 51)

(2) 其中：符号位是 0

指数是 0111 0110 (填32bit中的原始形式)

指数转换为十进制形式是 118 (32bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 -9 (32bit中的原始形式按IEEE754的规则转换)

尾数是 001 1010 0010 1100 0101 0001 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.20447742938995361328125 (32bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.20447742938995361328125 (加整数部分的1)



# §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

## 1、float型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或- (例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”)

D. -0.005942532 (设学号为1234567，按规则更换为学号和学号逆序)

注：尾数为负、指数为负

(1) 得到的32bit的机内表示是： 1011 1011 1100 0010 1011 1001 1001 0010 (bb c2 b9 92)

(2) 其中：符号位是 1

指数是 0111 0111 (填32bit中的原始形式)

指数转换为十进制形式是 119 (32bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 -8 (32bit中的原始形式按IEEE754的规则转换)

尾数是 100 0010 1011 1001 1001 0010 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.5212881565093994140625 (32bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.5212881565093994140625 (加整数部分的1)



## §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

### 2、double型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或- (例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”)

A. 2352495.5942532 (设学号为1234567，按规则更换为学号和学号逆序)

注：尾数为正、指数为正

(1) 得到的64bit的机内表示是：0100 0001 0100 0001 1111 0010 1011 0111 1100 1100 0001 0000 0111 1100 0000 0000 (41 41 f2 b7 cc 10 7d 26)

(2) 其中：符号位是0

指数是100 0001 0100 (填64bit中的原始形式)

指数转换为十进制形式是1044 (64bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是21 (64bit中的原始形式按IEEE754的规则转换)

尾数是0001 1111 0010 1011 0111 1100 1100 0001 0000 0111 1100 0000 0000 (填64bit中的原始形式)

尾数转换为十进制小数形式是0.121757313849002457573078572750091552734375 (64bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是1.121757313849002457573078572750091552734375 (加整数部分的1)



## §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

### 2、double型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或- (例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”)

B. -5942532.2352495 (设学号为1234567，按规则更换为学号和学号逆序)

注：尾数为负、指数为正

(1) 得到的64bit的机内表示是： 1100 0001 0101 0110 1010 1011 0100 0001 0000 1111 0000 1110 0101 0000 0000 0000 (c1 56 ab 41 0f 0e 53 eb)

(2) 其中：符号位是 1

指数是 100 0001 0101 (填64bit中的原始形式)

指数转换为十进制形式是 1045 (64bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 22 (64bit中的原始形式按IEEE754的规则转换)

尾数是 0110 1010 1011 0100 0001 0000 1111 0000 1110 0101 0000 0000 0000 (填64bit中的原始形式)

尾数转换为十进制小数形式是 0.4168100917931951698847115039825439453125 (64bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.4168100917931951698847115039825439453125 (加整数部分的1)



## §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

### 2、double型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或- (例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”)

C. 0.002352495 (设学号为1234567，按规则更换为学号和学号逆序)

注：尾数为正、指数为负

(1) 得到的64bit的机内表示是： 0011 1111 0110 0011 0100 0101 1000 1010 0010 0010 1101 1001 0001 1110 0000 0000 (3f 63 45 8a 22 d9 1d e9)

(2) 其中：符号位是 0

指数是 011 1111 0110 (填64bit中的原始形式)

指数转换为十进制形式是 1014 (64bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 -9 (64bit中的原始形式按IEEE754的规则转换)

尾数是 0011 0100 0101 1000 1010 0010 0010 1101 1001 0001 1110 0000 0000 (填64bit中的原始形式)

尾数转换为十进制小数形式是 0.2044774400000051173265092074871063232421875 (64bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.2044774400000051173265092074871063232421875 (加整数部分的1)





## §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

### 2、double型数的机内表示

格式要求：多字节时，每4bit中间加一个空格或- (例：“1101 0100 0011 0001” 或 “1101-0100-0011-0001”)

D. -0.005942532 (设学号为1234567，按规则更换为学号和学号逆序)

注：尾数为负、指数为负

(1) 得到的64bit的机内表示是： 1011 1111 0111 1000 0101 0111 0011 0010 0100 1001 1000 0110 1110  
1000 0000 0000 (bf 78 57 32 49 86 e5 40)

(2) 其中：符号位是 1

指数是 011 1111 0111 (填64bit中的原始形式)

指数转换为十进制形式是 1015 (64bit中的原始形式按二进制原码形式转换)

指数表示的十进制形式是 -8 (64bit中的原始形式按IEEE754的规则转换)

尾数是 1000 0101 0111 0011 0010 0100 1001 1000 0110 1110 1000 0000 0000 (填64bit中的原始形式)

尾数转换为十进制小数形式是 0.52128819200015641399659216403961181640625 (64bit中的原始形式按二进制原码形式转换)

尾数表示的十进制小数形式是 1.52128819200015641399659216403961181640625 (加整数部分的1)



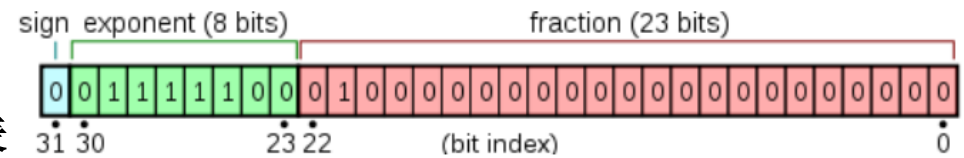
# §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

## 3、总结

(1) float型数据的32bit是如何分段来表示一个单精度的浮点数的？给出bit位的分段解释

尾数的正负如何表示？尾数如何表示？指数的正负如何表示？指数如何表示？

分为符号位、偏移后的指数位和尾数位。符号位用于表示浮点数的正负；偏移后的指数位用于表示浮点数化为二进制后的指数，但该值的十进制为原指数+偏移量，在32bit中偏移量为127；尾数用于表示浮点数化为二进制后的尾数，此处要求该二进制数首位非0，做尾数时需隐藏高位1并在低位补0。指数的正负处理在加偏移量的过程中，保证了偏移后的结果均为正数；尾数的正负即体现在整体符号位上。



(2) 为什么float型数据只有7位十进制有效数字？为什么最大只能是 $3.4 \times 10^{38}$ ？

有些资料上说有效位数是6~7位，能找出6位/7位不同的例子吗？

将32位浮点数可表示的数视为表盘上的蓝点的话，由于经过一定次数变换最终相当于变换了指数位，而指数位的变换导致数字值非线性变化，点的间隔实际上是在不断增大的，而计算机实际只能储存表盘上存在的蓝点对应的值，不在位置的会被校准到最近的蓝点值，所以在不同的范围内计算有效数字的方法是不一样的，有时候结果也不一样，但综合各个范围，可以贮存到7位的十进制有效数字；因为IEEE 754对指数位做出了特殊规定，全0或全1用于表示非规格数或是特殊数，这导致实际的规格数的范围缩小了，其指数部分只能取到 $[-126, 127]$ ；出于舍入规则导致有时候精度不足7位，如1024.001会舍入到1024.000，精度似乎不足7位，而1024.0011会舍入到1024.001，精度又达到7位了。



# § . 基础知识题 - 浮点数机内存储格式(IEEE 754) 理解

## 3、总结

(3) double型数据的64bit是如何分段来表示一个双精度的浮点数的？给出bit位的分段解释

尾数的正负如何表示？尾数如何表示？指数的正负如何表示？指数如何表示？

与32位类似的，分为符号位(63)、偏移后的指数位(52-62)和尾数位(0-51)。符号位用于表示浮点数的正负；偏移后的指数位用于表示浮点数化为二进制后的指数，但该值的十进制为原指数+偏移量，在64bit中偏移量为1023；尾数用于表示浮点数化为二进制后的尾数，此处要求该二进制数首位非0，做尾数时需隐藏高位1并在低位补0。指数的正负处理在加偏移量的过程中，保证了偏移后的结果均为正数；尾数的正负即体现在整体符号位上。

(4) 为什么double型数据只有15位十进制有效数字？为什么最大只能是 $1.7 \times 10^{308}$ ？

有些资料上说有效位数是15~16位，能找出15位/16位不同的例子吗？

与float型数据32bit表示类似，double采用的这种储存方式也导致蓝点为非线性分布，如果只考虑计算机的精确能力，可以达到15位十进制的有效数字，但同样由于舍入规则导致实际精度可能有一些变化，导致在面对具体数的时候可能是15-16位的，根据在查得的表格可举例9007199254740993.2会舍入到9007199254740994，似乎实际精度只有15位，而4503599627370497.2又可以达到16位精确度。

Actual Exponent (unbiased) 实际指数 (无偏)	Exp (biased) Exp (偏置)	Minimum 最低	Maximum 最大	Gap 差距
-1	1022	0.5	$\approx 0.9999999999999988978$	$\approx 1.11022e-16$
0	1023	1	$\approx 1.9999999999999777955$	$\approx 2.22045e-16$
1	1024	2	$\approx 3.999999999999955911$	$\approx 4.44089e-16$
2	1025	4	$\approx 7.99999999999911822$	$\approx 8.88178e-16$
10	1033	1024	$\approx 2047.9999999999772626$	$\approx 2.27374e-13$
11	1034	2048	$\approx 4095.9999999999545253$	$\approx 4.54747e-13$
52	1075	4503599627370496	9007199254740991	1
53	1076	9007199254740992	18014398509481982	2
1023	2046	$\approx 8.98847e307$	$\approx 1.79769e308$	$\approx 1.99584e292$

注：

- 文档用自己的语言组织
- 篇幅不够允许加页
- 如果用到某些小测试程序进行说明，可以贴上小测试程序的源码及运行结果
- 为了使文档更清晰，允许将网上的部分图示资料截图后贴入
- 不允许在答案处直接贴某网址，再附上“见\*\*”（或类似行为），否则文档作业部分直接总分-50



# §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

## 4、思考

(1) 8/11bit的指数的表示形式是2进制补码吗？如果不是，一般称为什么方式表示？

指数位由于是偏移后的指数，全部为正数，故无需采用二进制补码的形式表示，原码形式即可。

(2) double赋值给float时，下面两个程序，double型常量不加F的情况下，左侧有warning，右侧无warning，为什么？

总结一下规律

```
#include <iostream>
using namespace std;
int main()
{
    float f = 1.2;
    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)*(p+1) << endl;
    cout << hex << (int)*(p+2) << endl;
    cout << hex << (int)*(p+3) << endl;
    return 0;
}
```

warning C4305: “初始化”: 从“double”到“float”截断

```
#include <iostream>
using namespace std;
int main()
{
    float f = 100.25;
    unsigned char* p = (unsigned char*)&f;
    cout << hex << (int)(*p) << endl;
    cout << hex << (int)*(p+1) << endl;
    cout << hex << (int)*(p+2) << endl;
    cout << hex << (int)*(p+3) << endl;
    return 0;
}
```

这是由于两个数机内储存的区别，1.2机内储存为0011 1111 1111 0011 0011 0011 0011 0011 0011 0011 0011 0011 0011 0100 0000 0000，而100.25机内储存为0100 0000 0101 1001 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000；在double赋值给float时会出现丢弃，这时0不会体现出差别，而1就会导致变化。