

# § . 基础知识题 – 浮点数机内存存储格式(IEEE 754)理解



要求:

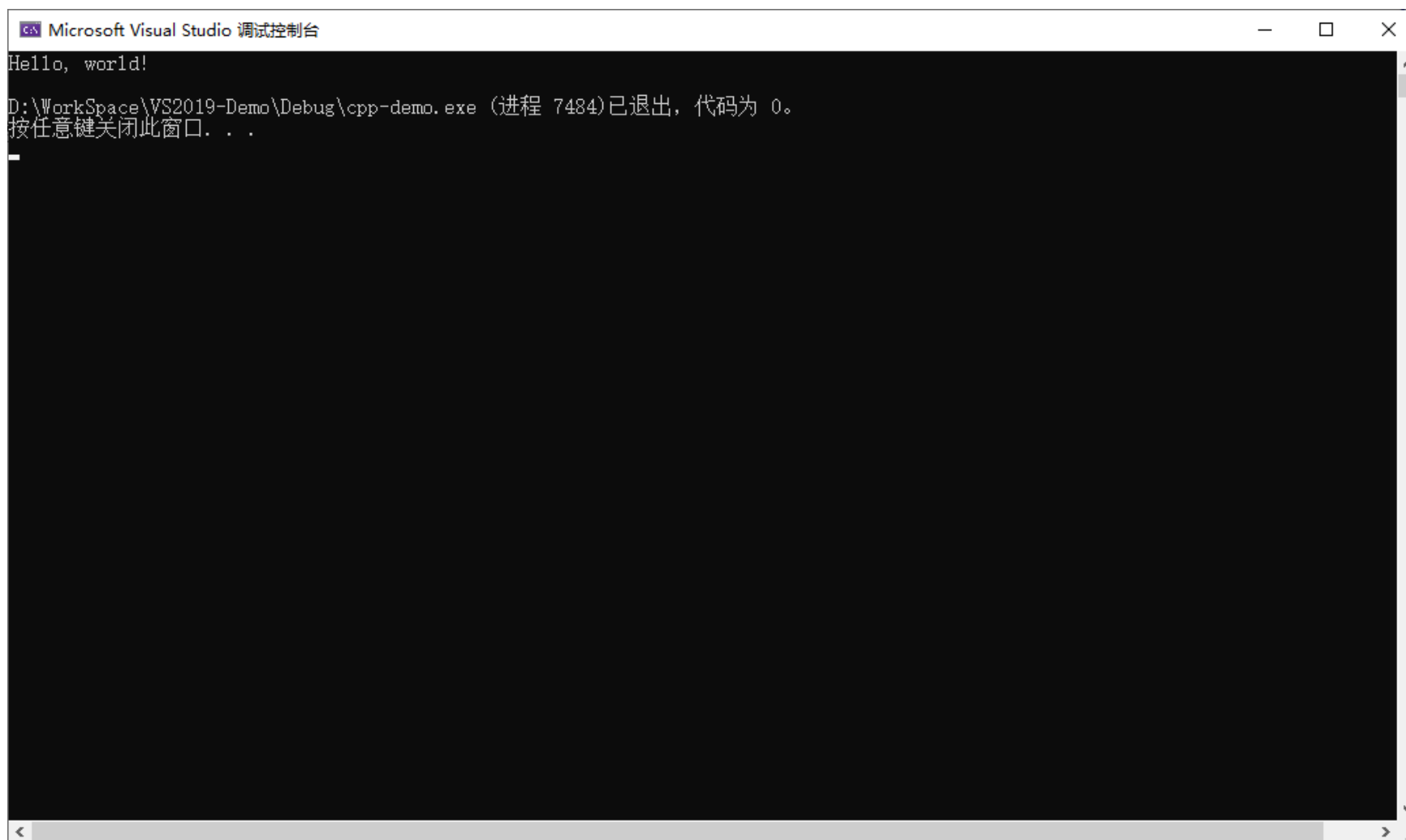
- 1、完成本文档中所有的题目并写出分析、运行结果
- 2、无特殊说明，均使用VS2022编译即可
- 3、直接在本文件上作答，**写出答案/截图（不允许手写、手写拍照截图）**即可；填写答案时，为适应所填内容或贴图，**允许调整**页面的字体大小、颜色、文本框的位置等
  - ★ 贴图要有效部分即可，不需要全部内容
  - ★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可
  - ★ **不允许**手写在纸上，再拍照贴图
  - ★ **允许**在各种软件工具上完成（不含手写），再截图贴图
- 4、转换为pdf后提交
- 5、**9月15日前**网上提交本次作业（在“文档作业”中提交）

# §. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

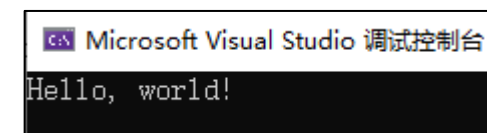


贴图要求：只需要截取输出窗口中的有效部分即可，如果全部截取/截取过大，则视为无效贴图

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". The output text is: "Hello, world!", "D:\Workspace\VS2019-Demo\Debug\cpp-demo.exe (进程 7484)已退出, 代码为 0.", and "按任意键关闭此窗口. . .". The window is large, showing the full context of the output.

例：有效贴图

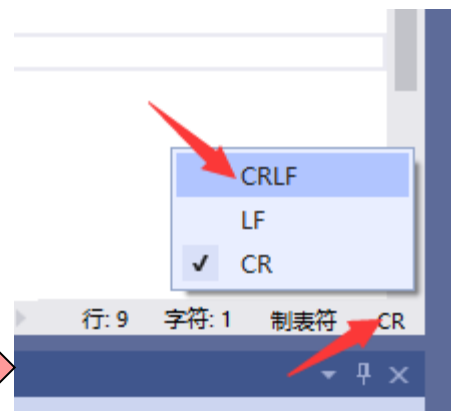
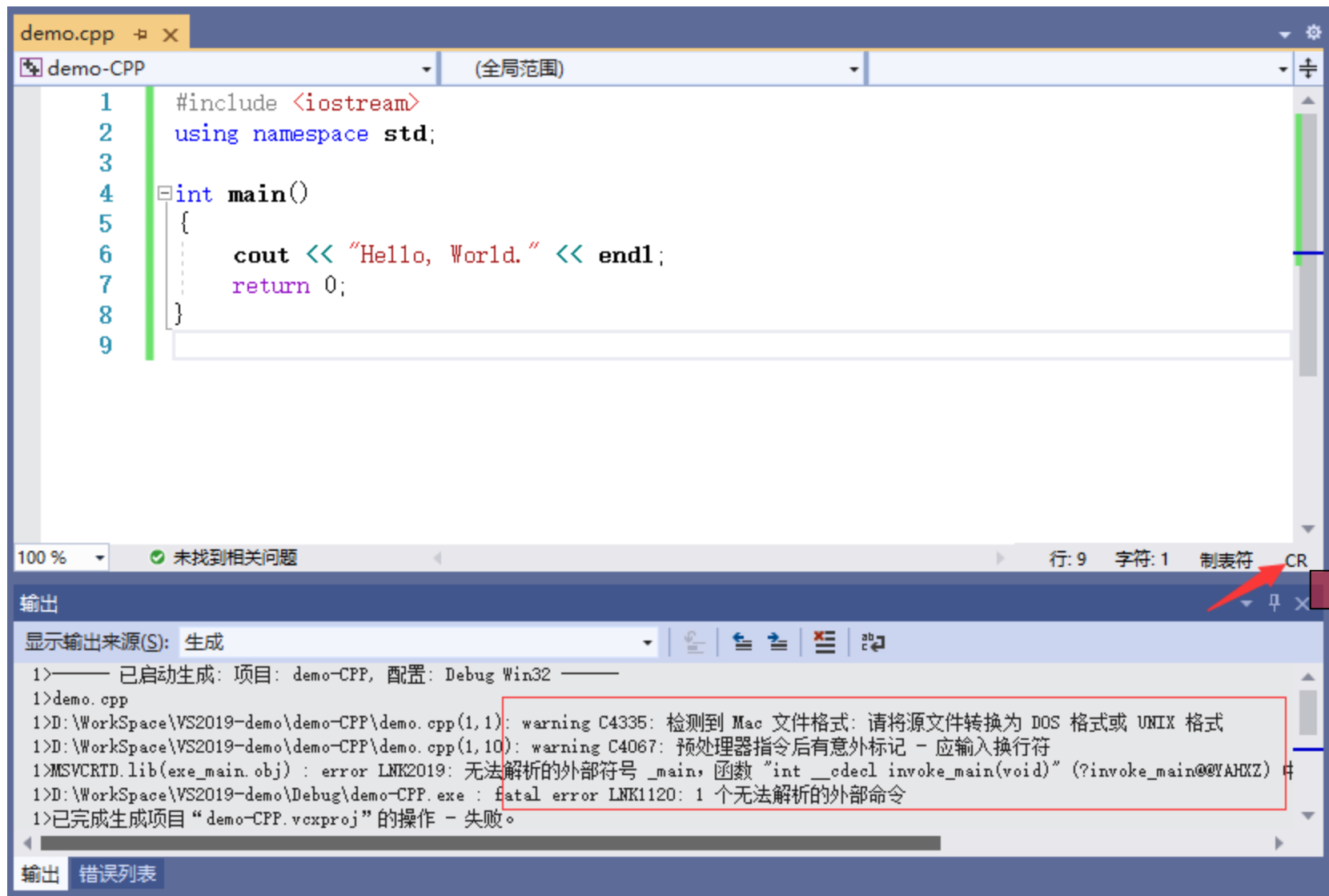
A screenshot of the Microsoft Visual Studio debug console window, cropped to show only the first line of output: "Hello, world!". The window title is "Microsoft Visual Studio 调试控制台".



## §. 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



# §. 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 = 123.456f;
    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;
}
```

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

8位指数

23位尾数

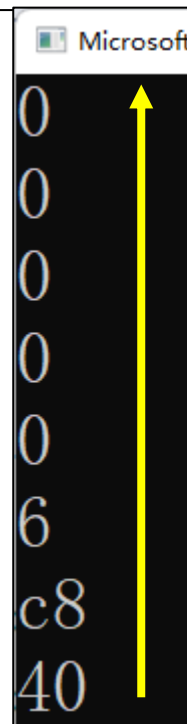
# §. 基础知识题 - 浮点数机内存储格式(IEEE 754)理解



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;
}
```



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

11位指数

52位尾数

# § . 基础知识题 – 浮点数机内存储格式(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)



# §. 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)

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 0000 (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: 1234567.7654321

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 1001 1001 0110 1011 0100 0011 1110 (49 96 b4 3e)

(2) 其中: 符号位是 0

指数是 1001 0011 (填32bit中的原始形式)

指数转换为十进制形式是 147 (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 20 (32bit中的原始形式按IEEE754的规则转换)

1001 0011

- 0111 1111

= 0001 0100 (0x14 = 20)

尾数是 001 0110 1011 0100 0011 1110 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.1773755503845214844 (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 1.1773755503845214844 (加整数部分的1后)

001 0110 1011 0100 0011 1110 =  $2^{-3} + \dots + 2^{-22}$

= 0.1773755503845214844 => 加1 => 1.1773755503845214844

$1.1773755503845214844 * 2^{20} = 1234567.75$  (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:

1234567 = 0001 0010 1101 0110 1000 0111 (整数部分转二进制为21位)

0.7654321 = 11000... (小数部分转二进制, 再要3位就够了)

1234567.7654321 = 0001 0010 1101 0110 1000 0111 110 =  $1.0010 1101 0110 1000 0111 110 \times 2^{20}$  (移20位)

符号位: 0

阶 码:  $20 + 127 = 147 = 1001 0011$

尾 数: 0010 1101 0110 1000 0111 110 (23位)

001 0110 1011 0100 0011 1110 (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

### 1、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”)

A. 2053932.2393502 (此处假设学号是1234567，各人换成自己的学号，按1234567做的0分!!!)

注：尾数为正、指数为正

(1) 得到的32bit的机内表示是：01001001 11111010 10111001 01100010

(2) 其中：符号位是0

指数是10010011 (填32bit中的原始形式)

指数转换为十进制形式是147 (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20 (32bit中的原始形式按IEEE754的规则转换)

尾数是1111010 10111001 01100010 (填32bit中的原始形式)

尾数转换为十进制小数形式是0.958782434463501 (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1.958782434463501 (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8bit中间加一个空格或- (例：“11010100 00110001” 或 “11010100-00110001”)

B. -2393502. 2053932 (此处假设学号是1234567，各人换成自己的学号，按1234567做的0分!!!)

注：尾数为负、指数为正

(1) 得到的32bit的机内表示是： 11001010 00010010 00010110 01111001

(2) 其中：符号位是 1

指数是 10010100 (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 0010010 00010110 01111001 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.14131081104278564 (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 1.14131081104278564 (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8bit中间加一个空格或- (例：“11010100 00110001” 或 “11010100-00110001”)

C. 0.002053932 (此处假设学号是1234567，各人换成自己的学号，按1234567做的0分!!!)

注：尾数为正、指数为负

(1) 得到的32bit的机内表示是：0111011 00000110 10011011 01000011

(2) 其中：符号位是0

指数是1110110 (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0000110 10011011 01000011 (填32bit中的原始形式)

尾数转换为十进制小数形式是0.0516132116317749 (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1.0516132116317749 (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8bit中间加一个空格或- (例：“11010100 00110001” 或 “11010100-00110001”)

D. -0.002393502 (此处假设学号是1234567，各人换成自己的学号，按1234567做的0分!!!)

注：尾数为负、指数为负

(1) 得到的32bit的机内表示是： 1111011 00011100 11011100 01001101

(2) 其中：符号位是 1

指数是 1110110 (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 0011100 11011100 01001101 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.2254730463027954 (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 1.2254730463027954 (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8bit中间加一个空格或- (例：“11010100 00110001” 或 “11010100-00110001”)

A. 2053932. 2393502 (此处假设学号是1234567，各人换成自己的学号，按1234567做的0分!!!)

注：尾数为正、指数为正

(1) 得到的64bit的机内表示是： 01000001 00111111 01010111 00101100 00111101 01000110 00001110 00000001

(2) 其中：符号位是 0

指数是 10000010011 (填64bit中的原始形式)

指数转换为十进制形式是 1043 (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 20 (64bit中的原始形式按IEEE754的规则转换)

尾数是 1111 01010111 00101100 00111101 01000110 00001110 00000001 (填64bit中的原始形式)

尾数转换为十进制小数形式是 0.9587824243070602 (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 1.9587824243070602 (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8bit中间加一个空格或- (例：“11010100 00110001” 或 “11010100-00110001”)

B. -2393502. 2053932 (此处假设学号是1234567，各人换成自己的学号，按1234567做的0分!!!)

注：尾数为负、指数为正

(1) 得到的64bit的机内表示是： 11000001 01000010 01000010 11001111 00011010 01001010 01010011 00001010

(2) 其中：符号位是 1

指数是 10000010100 (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 0010 01000010 11001111 00011010 01001010 01010011 00001010 填64bit中的原始形式)

尾数转换为十进制小数形式是 0.1413107897726058 (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 1.1413107897726058 (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8bit中间加一个空格或- (例：“11010100 00110001” 或 “11010100-00110001”)

C. 0.002053932 (此处假设学号是1234567，各人换成自己的学号，按1234567做的0分!!!)

注：尾数为正、指数为负

(1) 得到的64bit的机内表示是：00111111 01100000 11010011 01101000 01011000 10010101 00101000 10101101

(2) 其中：符号位是\_\_0\_\_

指数是\_\_01111110110\_\_ (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0000 11010011 01101000 01011000 10010101 00101000 10101101 (填64bit中的原始形式)

尾数转换为十进制小数形式是\_\_0.051613184000000034\_\_ (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\_\_1.051613184000000034\_\_ (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8bit中间加一个空格或- (例：“11010100 00110001” 或 “11010100-00110001”)

D. -0.002393502 (此处假设学号是1234567，各人换成自己的学号，按1234567做的0分!!!)

注：尾数为负、指数为负

(1) 得到的64bit的机内表示是： 10111111 01100011 10011011 10001001 10011010 00000011 01011101 00010011

(2) 其中：符号位是\_\_1\_\_

指数是\_\_01111110110\_\_ (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 10011011 10001001 10011010 00000011 01011101 00010011 (填64bit中的原始形式)

尾数转换为十进制小数形式是\_\_0.22547302400000002\_\_ (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\_\_1.22547302400000002\_\_ (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？
- (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？
- (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？
- (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？

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



(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？

32bit分为三部分，从高位到低位依次分为符号位（1位）、指数位（8位）和尾数位（23位）

$$\text{float} = (-1)^S * 1.F * 2^{E-127}$$

符号位：1bit，用来表示浮点数的正负，用0表示正，1表示负，记作S；

指数位：8bit，用来表示二进制的指数部分，反映了浮点数的表示范围，取值范围为0~255，记作E；

尾数位：23bit，用来表示二进制的小数部分，反映了浮点数的精度，记作F。



IEEE 754规定float型数据：尾数用原码表示；指数用移码表示，偏移量为127

尾码不存在正负之说

指数所对应的十进制小于127时为负，大于127时为正



(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？

首先，这里的有效数字包括了整数部分！

Float型数据的精度取决于尾数部分的位数，根据IEEE 754规定，float型数据的尾数部分占23bit  
 $2^{23} = 8388608$  即23位2进制对应7位十进制，所以说float型数据只有7位十进制有效数字

根据IEEE 754规则，除去指数位为全0和全1的两种特殊情况，指数真值(E-127)的取值范围在-126~127  
所以浮点数能储存的最大数为  $(1.111111111111111111111111) \times 2^{127} = 2^{128} = 3.4 \times 10^{38}$

```
int main()
{
    float f1 = 1024.0010f;
    float f2 = 1024.0001f;

    cout.setf( ios_base::fixed, ios_base::floatfield); // 定点数表示
    cout << "打印1024.0010----" << f1 << endl;
    cout << "打印1024.0010----" << f2 << endl;

    return 0;
}
```

```
打印1024.0010----1024.000977
打印1024.0010----1024.000122
```

其中1024.0001的有效位数有7位，  
而1024.0010的有效位数只有6位



(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？

64bit分为三部分，从高位到低位依次分为符号位（1位）、指数位11位）和尾数位（52位）

$$\text{float} = (-1)^S * 1.F * 2^{E-1023}$$

符号位：1bit，用来表示浮点数的正负，用0表示正，1表示负，记作S；

指数位：11bit，用来表示二进制的指数部分，反映了浮点数的表示范围，取值范围为0~2048，记作E；

尾数位：52bit，用来表示二进制的小数部分，反映了浮点数的精度，记作F。

IEEE 754规定double型数据：尾数用原码表示；指数用移码表示，偏移量为1023

尾码不存在正负之说

指数所对应的十进制小于1023时为负，大于1023时为正



(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？

首先，这里的有效数字包括了整数部分！

double型数据的精度取决于尾数部分的位数，根据IEEE 754规定，double型数据的尾数部分占52bit  
 $2^{52} = 4503599627370496$  即52位2进制对应16位十进制，所以说double型数据只有16位十进制有效数字

根据IEEE 754规则，除去指数位为全0和全1的两种特殊情况，指数真值(E-1023)的取值范围在-1022~1023  
所以浮点数能储存的最大数为  $(1.111111111111111111111111) \times 2^{1023} = 2^{1024} = 1.7 \times 10^{308}$

```
int main()
{
    double d1 = 1024000000000.0010;
    double d2 = 1024000000000.0001;

    cout.setf( fmtfl: ios_base::fixed, mask: ios_base::floatfield); // 定点数表示
    cout << "打印1024000000000.0010----" << d1 << endl;
    cout << "打印1024000000000.0001----" << d2 << endl;
    return 0;
}
```

```
打印1024000000000.0010----1024000000000.000977
打印1024000000000.0001----1024000000000.000122
```

其中1024000000000.0001的有效位数有16位，  
而1024000000000.0010的有效位数只有15位