

§ . 基础知识题 – 浮点数机内存存储格式(IEEE 754)理解



要求:

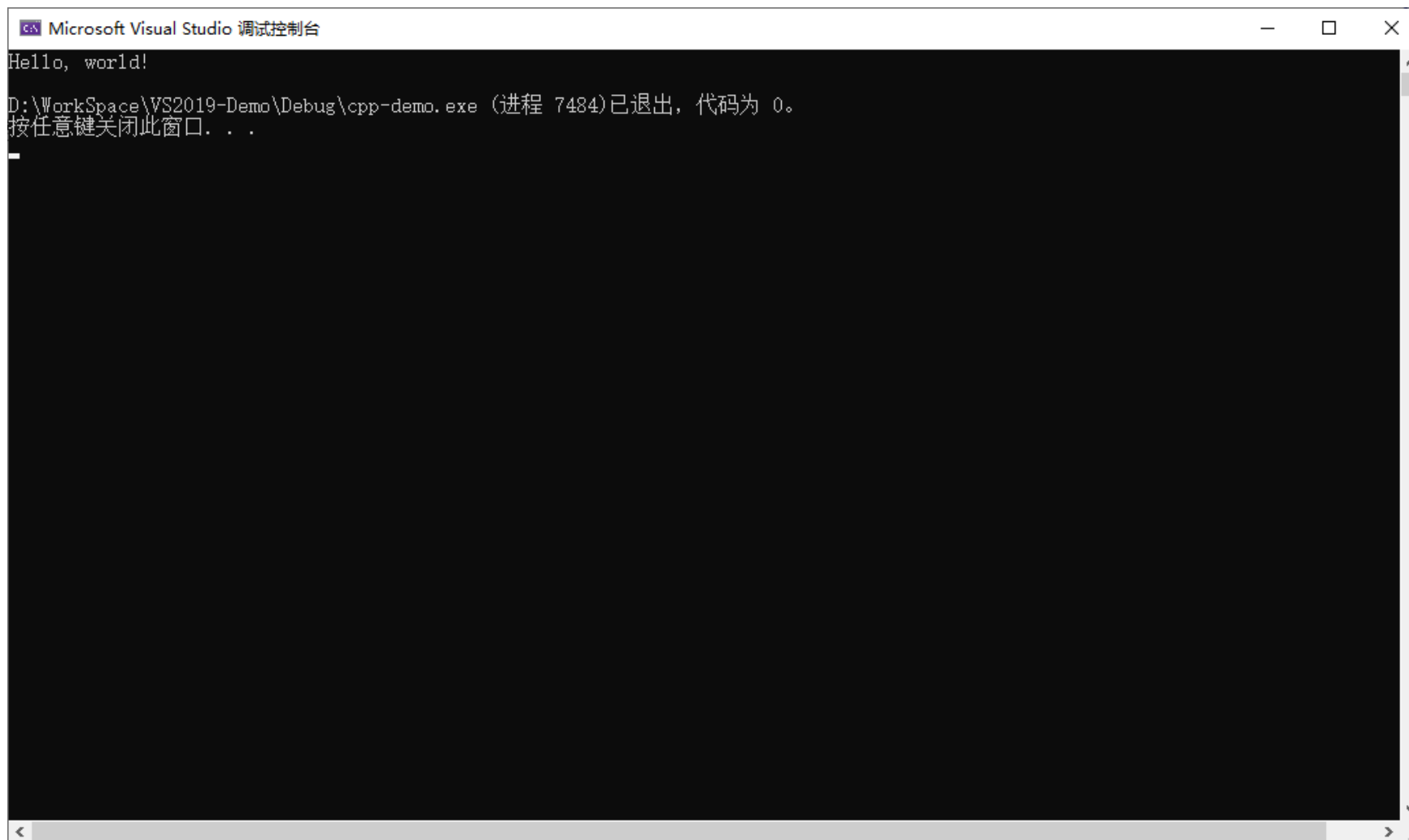
- 1、完成本文档中所有的题目并写出分析、运行结果
- 2、无特殊说明，均使用VS2022编译即可
- 3、直接在本文件上作答，**写出答案/截图（不允许手写、手写拍照截图）**即可；填写答案时，为适应所填内容或贴图，**允许调整**页面的字体大小、颜色、文本框的位置等
 - ★ 贴图要有效部分即可，不需要全部内容
 - ★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可
 - ★ **不允许**手写在纸上，再拍照贴图
 - ★ **允许**在各种软件工具上完成（不含手写），再截图贴图
- 4、转换为pdf后提交
- 5、**3月14日前**网上提交本次作业（在“文档作业”中提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

§. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

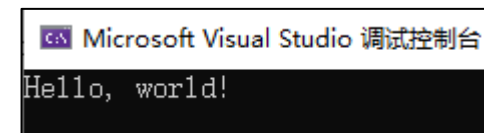


贴图要求：只需要截取输出窗口中的有效部分即可，如果全部截取/截取过大，则视为无效贴图

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 "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 and shows a scroll bar on the right side.

例：有效贴图

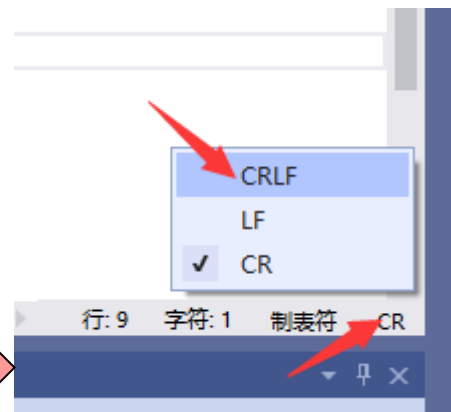
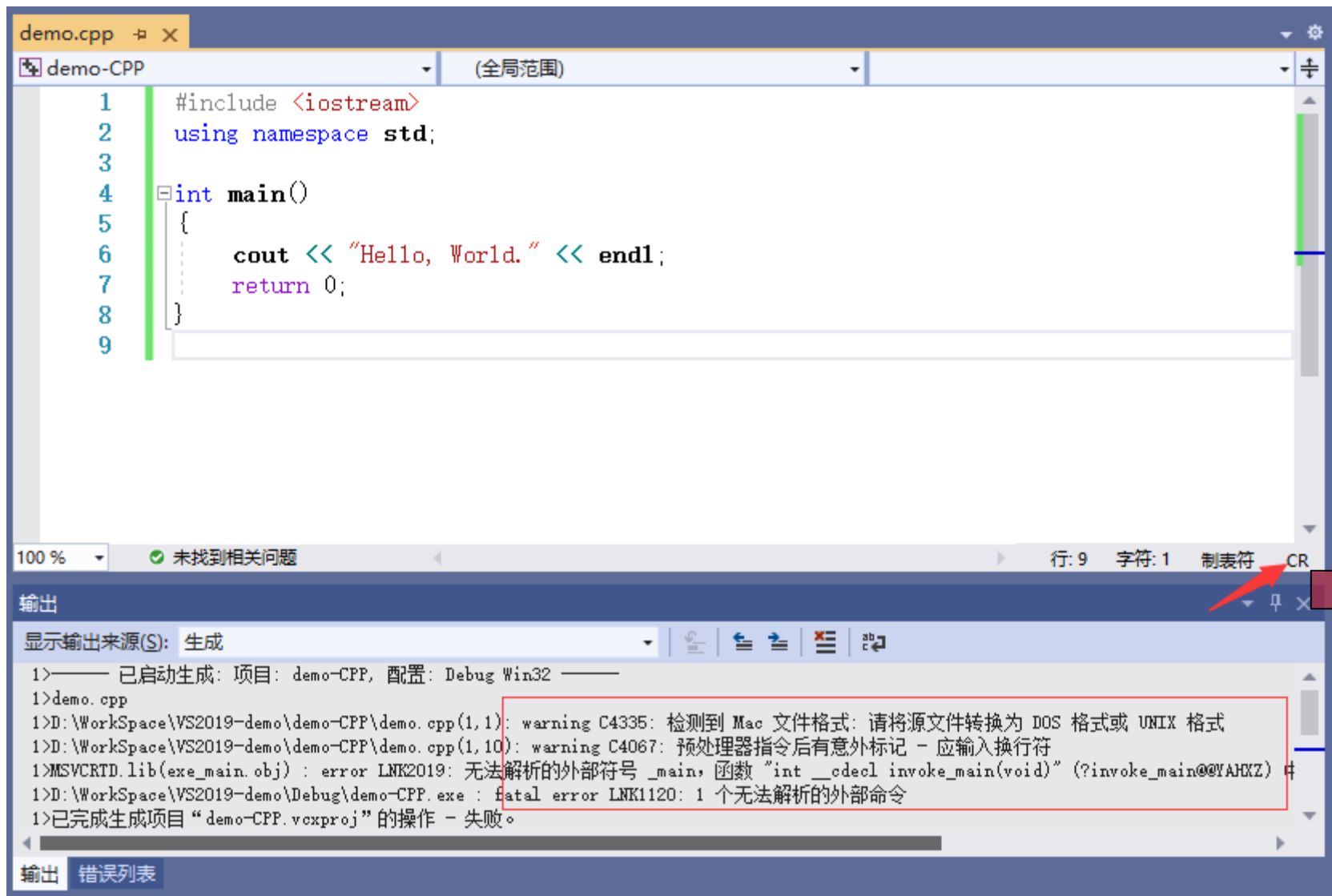
A screenshot of the Microsoft Visual Studio debug console window, showing only the first line of output: "Hello, world!". The window is titled "Microsoft Visual Studio 调试控制台".



§. 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 = 123.456;
    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×tamp=1662273598&unique_k=AuouMEO

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. 1234567. 7654321 (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

2352017. 7102532

(1) 得到的32bit的机内表示是：0100 1010 0000 1111 1000 1110 0100 0111 (4a f 8e 47) (不是手算，用P. 4方式打印)

(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 1000 1110 0100 0111__ (填32bit中的原始形式)

尾数转换为十进制小数形式是_0. 12152945995330810546875_ (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. 12152945995330810546875_ (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. -7654321.1234567 (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

-7102532.2352017

(1) 得到的32bit的机内表示是： 1100 1010 1101 1000 1100 0000 1000 1000 不是手算，用P.4方式打印)

(2) 其中：符号位是__1__

指数是__1001 0101__ (填32bit中的原始形式)

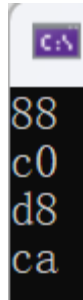
指数转换为十进制形式是__149__ (32bit中的原始形式按二进制原码形式转换)

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__101 1000 1100 0000 1000 1000__ (填32bit中的原始形式)

尾数转换为十进制小数形式是__-0.69337558746337890625__ (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__-1.69337558746337890625__ (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. 0.001234567 (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

0.002352017

(1) 得到的32bit的机内表示是：0011 1011 0001 1010 0010 0100 0100 1100 (3b 1a 24 4c) (不是手算，用P. 4方式打印)

(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 0100 0100 1100__ (填32bit中的原始形式)

尾数转换为十进制小数形式是_0.204232692718505859375_ (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_1.204232692718505859375_ (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. -0.007654321 (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

-0.007102532

(1) 得到的32bit的机内表示是： 1011 1011 1110 1000 1011 1100 0101 1011 (不是手算，用P. 4方式打印)

(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__110 1000 1011 1100 0101 1011__ (填32bit中的原始形式)

尾数转换为十进制小数形式是_-0.81824815273284912109375_ (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_-1.81824815273284912109375_ (加整数部分的1)

5b
bc
e8
bb



§. 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. 1234567. 7654321 （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

2352017. 7102532

(1) 得到的64bit的机内表示是：

0100 0001 0100 0001 1111 0001 1100 1000 1101 1010 1110 1001 1001 0011 1110 1001 （不是手算，用P.5方式打印）

(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 0001 1100 1000 1101 1010 1110 1001 1001 0011 1110 1001（填64bit中的原始形式）

尾数转换为十进制小数形式是

0. 1215294410005702818722284064278937876224517822265625（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. 1215294410005702818722284064278937876224517822265625（加整数部分的1）

CAN M

ad
93
e9
da
c8
f1
41
41



§. 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. -7654321. 1234567 (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

-7102532. 2352017

(1) 得到的64bit的机内表示是：

1100 0001 0101 1011 0001 1000 0001 0001 0000 1111 0000 1101 1000 1011 0110 1110 (不是手算，用P.5方式打印)

(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1011 0001 1000 0001 0001 0000 1111 0000 1101 1000 1011 0110 1110 (填64bit中的原始形式)

尾数转换为十进制小数形式是

-0.693375643539833941275674078497104346752166748046875 (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

-1.693375643539833941275674078497104346752166748046875 (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. 0.001234567 (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

0.002352017

(1) 得到的64bit的机内表示是：

0011 1111 0110 0011 0100 0100 1000 1001 1000 0011 0000 0111 0100 0010 0010 0101 (不是手算，用P.5方式)

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

尾数转换为十进制小数形式是

0.2042327040000000426545057052862830460071563720703125 (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

1.2042327040000000426545057052862830460071563720703125 (加整数部分的1)

25
42
7
83
89
44
63
3f



§. 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. -0.007654321 (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

-0.007102532

(1) 得到的64bit的机内表示是：

1011 1111 0111 1101 0001 0111 1000 1011 0110 1010 1000 1010 0110 1011 0000 0110 (不是手算，用P.5方式打印)

(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)

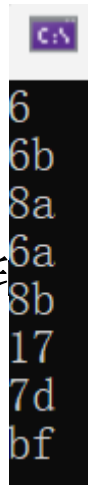
尾数是1101 0001 0111 1000 1011 0110 1010 1000 1010 0110 1011 0000 0110 (填64bit中的原始形式)

尾数转换为十进制小数形式是

-0.818248191999999985313252182095311582088470458984375 (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

-1.818248191999999985313252182095311582088470458984375 (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？

第1位是符号位，第2-9位是指数位，后23位为尾数位。

尾数正负由符号位决定，尾数表示由尾数的原始形式按二进制原码形式转换，

指数正负由指数二进制原码减去二进制下0111 1111决定，指数表示时由指数在32bit中的原始形式按IEEE754的规则转换

(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 3.4×10^{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？

单精度数的尾数用23位存储，加上小数点前的1位共24位， $10^6 < 2^{24} < 10^7$ ，所以为七位十进制有效数字。尾数最大为111

1111 1111 1111 1111 1111，指数最大为1111 1111，表示的最大数为1.111 1111 1111 1111 1111 1111 $\times 2^{128}$ ，为 3.4×10^{38}

```
float n = 3.14159;  
float r = 0.123456;
```

```
3.141590118408203125  
0.12345600128173828125
```

(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？

第1位是符号位，第2-12位是指数位，后为尾数位。

尾数正负由符号位决定，尾数表示由尾数的原始形式按二进制原码形式转换，

指数正负由指数二进制原码减去二进制下0111 1111 1111决定，指数表示时由指数在32bit中的原始形式按IEEE754的规则转换



(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 1.7×10^{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？

单精度数的尾数用52位存储，加上小数点前的1位共53位， $10^{15} < 2^{53} < 10^{16}$ ，所以为15位十进制有效数字。尾数最大为1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111，指数最大为111 1111 1111，表示的最大数为 $1.1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111 \times 2^{1024}$ ，为 1.7×10^{308}

```
double n = 3.14159265358979;  
double r = 0.1234567890123456;
```

```
3.1415926535897900074  
0.1234567890123455941
```

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？

不是

阶码

(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;
}
```

如果不指定数据类型，那么小数常量会被系统默认认为是 **double** 类型的。而**double**类型不能被系统转化为**float**类型，所以需要加上**f**，**100.25**的小数部分可以精确表示，从**double**型转到**float**型精度不变，而**1.2**的小数部分用二进制无法精确表示，从**double**型到**float**型会造成精度缺失而报**warning**错。

规律：若小数部分能够精确表示，则可以不加**f**，若不能精确表示，则不加**f**会导致转化时数据精度丢失。

