§.基础知识题 - 浮点数机内存储格式(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

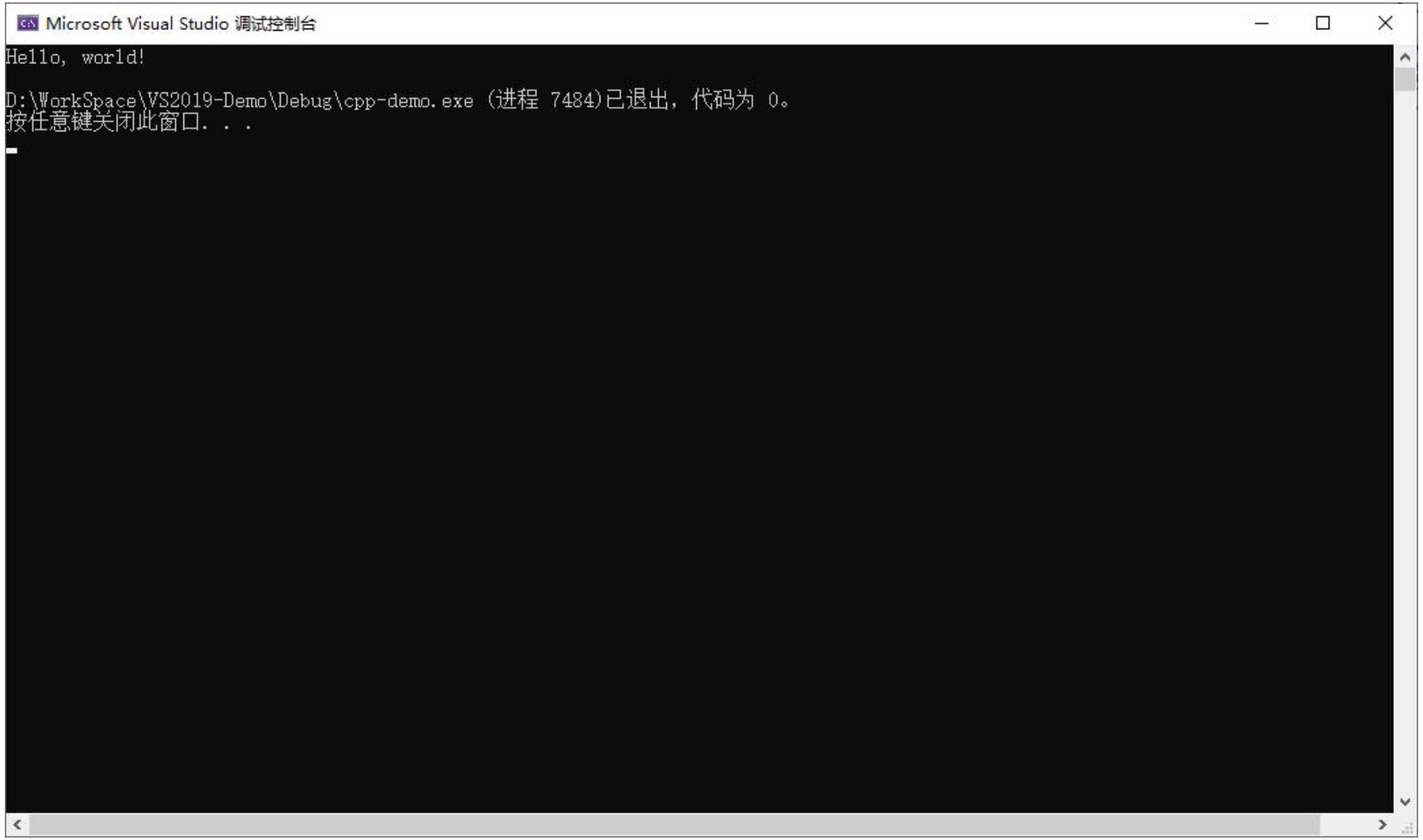
4、转换为pdf后提交

5、3月14日前网上提交本次作业（在“文档作业”中提交）

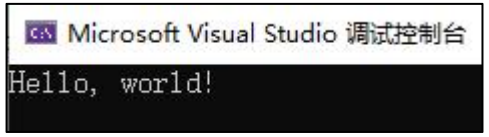
§.基础知识题 - 浮点数机内存储格式(IEEE 754)理解

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

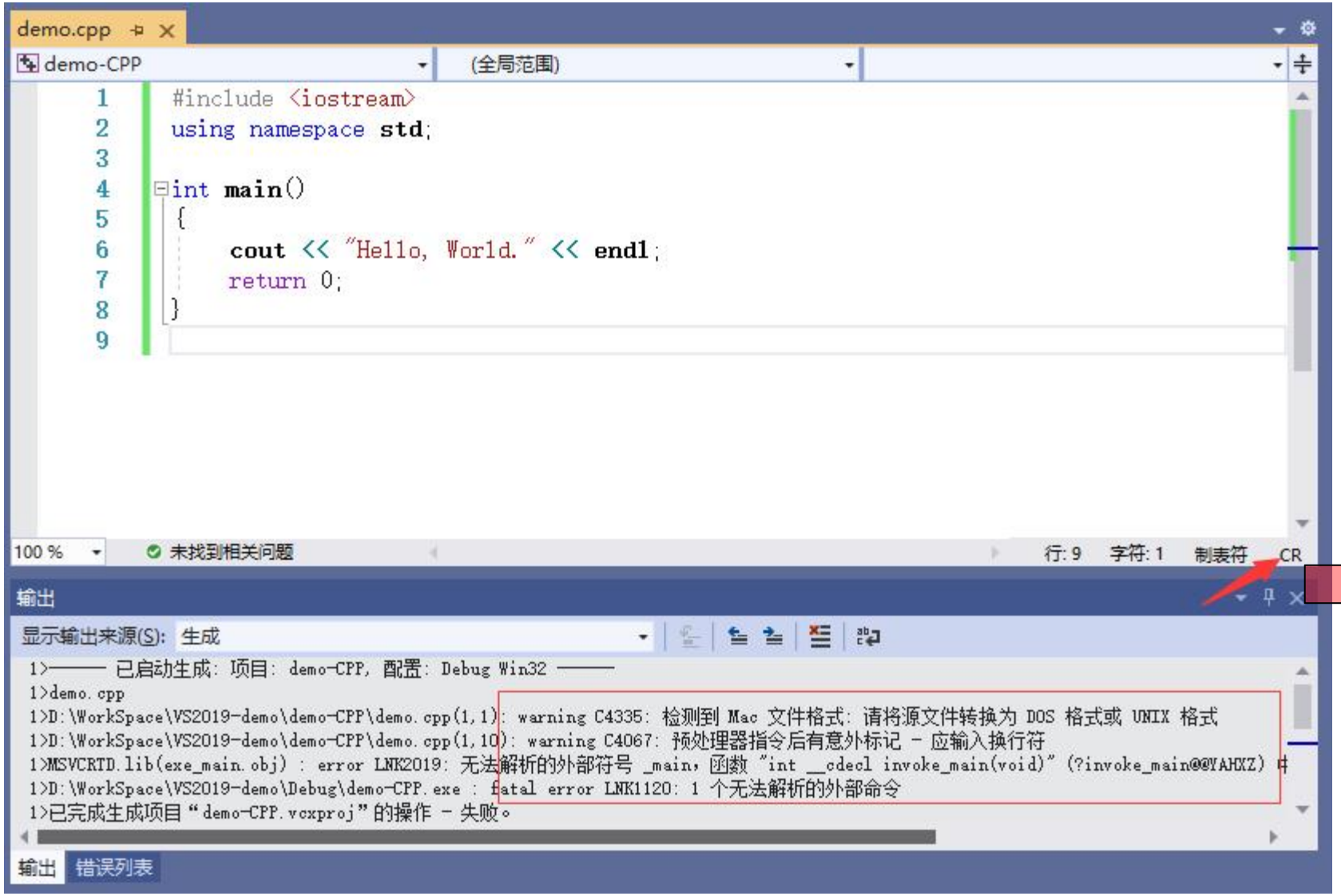


例：有效贴图



§.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
```

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;
}
```

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/BV1iW411d7hd?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=AuouME0

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的方法：

(1) 得到的32bit的机内表示是：__0011 1111 1001 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 1001 1010____(填32bit中的原始形式)

尾数转换为十进制小数形式是__0.20000004768371582031 25___(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__1.20000004768371582031 25___(加整数部分的1后)

001 1001 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 + ... + 0.00000023841857910156 25(详见右侧蓝色) = 0.20000004768371582031 25

=> 加1 = 1.20000004768371582031 25 (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 +
0.0000009536743164062 5 +
0.0000002384185791015625

0.20000004768371582031 25

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

(1) 得到的32bit的机内表示是：__0100 1010 0000 1111 1001 0001 0111 0000__(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 1001 0001 0111 0000_____(填32bit中的原始形式)
尾数转换为十进制小数形式是____0.12162590026855469_____(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____1.12162590026855469_____(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

(1) 得到的32bit的机内表示是：_1100 1011 0000 1011 0011 0010 1110 0100_(不是手算，用P.4方式打印)

(2) 其中：符号位是___1_____

指数是___1001 0110_____(填32bit中的原始形式)

指数转换为十进制形式是___150_____(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___23_____(32bit中的原始形式按IEEE754的规则转换)

尾数是___000 1011 0011 0010 1110 0100_____(填32bit中的原始形式)

尾数转换为十进制小数形式是___0.08749055862426758_____(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____1.08749055862426758_____(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

(1) 得到的32bit的机内表示是：_0011 1011 0001 1010 0010 0111 1011 0000___(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_____1_____(32bit中的原始形式按IEEE754的规则转换)

尾数是____001 1010 0010 0111 1011 0000_____(填32bit中的原始形式)
尾数转换为十进制小数形式是___0.20433616638183594____(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_____1.20433616638183594_____(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

(1) 得到的32bit的机内表示是：_1011 1100 0001 0101 0111 0110 1010 1100___(不是手算，用P.4方式打印)

(2) 其中：符号位是____1_____

指数是____0111 1000_____(填32bit中的原始形式)
指数转换为十进制形式是_____120_____(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_____3_____(32bit中的原始形式按IEEE754的规则转换)

尾数是__001 0101 0111 0110 1010 1100_____(填32bit中的原始形式)
尾数转换为十进制小数形式是__0.16768407821655273__(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___1.16768407821655273_____(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.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

(1) 得到的64bit的机内表示是：0100 0001 0100 0001 1111 0010 0010 1101
1111 0100 1100 0100 1011 0110 0111 1110(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 0010 0010 1101 1111 0100 1100 0100 1011 0110 0111 1110___(填64bit中的原始形式)
尾数转换为十进制小数形式是__0.12162585842761997__(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__1.12162585842761997___(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.-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

(1) 得到的64bit的机内表示是：_1100 0001 0110 0001 0110 0110 0101 1100
1000 0111 1000 0110 1111 0000 0001 0100__(不是手算，用P.5方式打印)

(2) 其中：符号位是_____1_____

指数是____100 0001 0110 0001_(填64bit中的原始形式)
指数转换为十进制形式是__1046 (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_____23_____(64bit中的原始形式按IEEE754的规则转换)

尾数是__0001 0110 0110 0101 1100 1000 0111 1000 0110 1111 0000 0001 0100____(填64bit中的原始形式)
尾数转换为十进制小数形式是_____0.08749058666490317_____(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_____1.08749058666490317_____(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

(1) 得到的64bit的机内表示是：_0011 1111 0110 0011 0100 0100 1111 0101
1111 0101 1011 0010 0110 1011 1001 1010_(不是手算，用P.5方式打印)

(2) 其中：符号位是____0_____

指数是___011 1111 0110 0011__(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 1111 0101 1111 0101 1011 0010 0110 1011 1001 1010_(填64bit中的原始形式)
尾数转换为十进制小数形式是___0.204336128_(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___1.204336128_(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.-0.007654321 （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

(1) 得到的64bit的机内表示是：_1011 1111 1000 0010 1010 1110 1101 0101
1000 0100 1100 0110 0001 0001 1011 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 1000 0010__(填64bit中的原始形式)
指数转换为十进制形式是___1016_____(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____-7_____(64bit中的原始形式按IEEE754的规则转换)

尾数是___1010 1110 1101 0101
1000 0100 1100 0110 0001 0001 1011 0110_(填64bit中的原始形式)
尾数转换为十进制小数形式是__0.1676840959999999___(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__1.1676840959999999_(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？

从右往左数，index从0开始，0到22为尾数位，23到30为指数位，31为符号位。

尾数的正负由符号位表示，0为正，1为负。尾数去掉1.之后的部分由尾数位用二进制表示，指数恒为正，通过将指数+偏移量127得到的正数用二进制表示，记录在指数位。

(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$3.4x10^{38}$ ？

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？

因为指数位的最小值为0000 0001，而尾数位值的最小值是000 0000 0000 0000 0000 0001，所以能表示的最小数值的间隔为0.0000001192092896，精度为七位，随着指数的增大，尾数的最小间隔也会变大，导致表示的小数位数也减少，综合来看，float的十进制精度为6.最大为01111111 11111111111111111111111,约等于$3.4x10^{38}$。

而有些小数因为正好能被23位尾数表示，所以精度为7，但并不是所有的精度都为7，当有的刚好能被23位尾数表示的时候，精度就为7，比如0.00011111011110000101000111101011100001010001111010111000010 10=0.1234375，而0.1234567的第七位不能被精确表示，所以精度为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？

从右往左数，index从0开始，0-51为尾数位，52-62为指数位，63位为符号位。

尾数的正负由符号位表示，0为正，1为负。尾数去掉1.之后的部分由尾数位用二进制表示，指数恒为正，通过将指数+偏移量127得到的正数用二进制表示，记录在指数位。

(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1.7x10308 ？
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？

当整数在1-2之间的时候，尾数的最小间隔为10^-15，随着指数位的增大，尾数所能表示的最小间隔将增大，使得小数的位数不断减小，综合来看，double只有15为二进制有效数字。最大的值为011111111111 1111111111111111111111111111111111111111111111111111=1.7x10^308

如0.123456789012345，它可以被精确地表示为一个double。但如果我们尝试表示0.1234567890123456，最后一位数字6将无法被精确表示，因为double的精度限制在15位数左右。

对于某些特殊的数字，double可以精确表示16位十进制数字。如数字0.1234567890123125可以被精确表示为double。这是因为0.1234567890123125的二进制表示为0.0001111110011010110111100011011101010100010110001000000100000000所以它的精度为16位。

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。称为指数位的规范化处理，将实际指数加上偏移量之后再转换为二进制，确保恒为正数。

(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;
}
```

因为1.2无法被二进制准确表示，而c++中不加F默认为double型double的精度比float高，向下转型会有精度损失，所以会报错，但是100.25能被精确表示，并且在float的精度范围之内，所以不会出现精度损失。