§.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

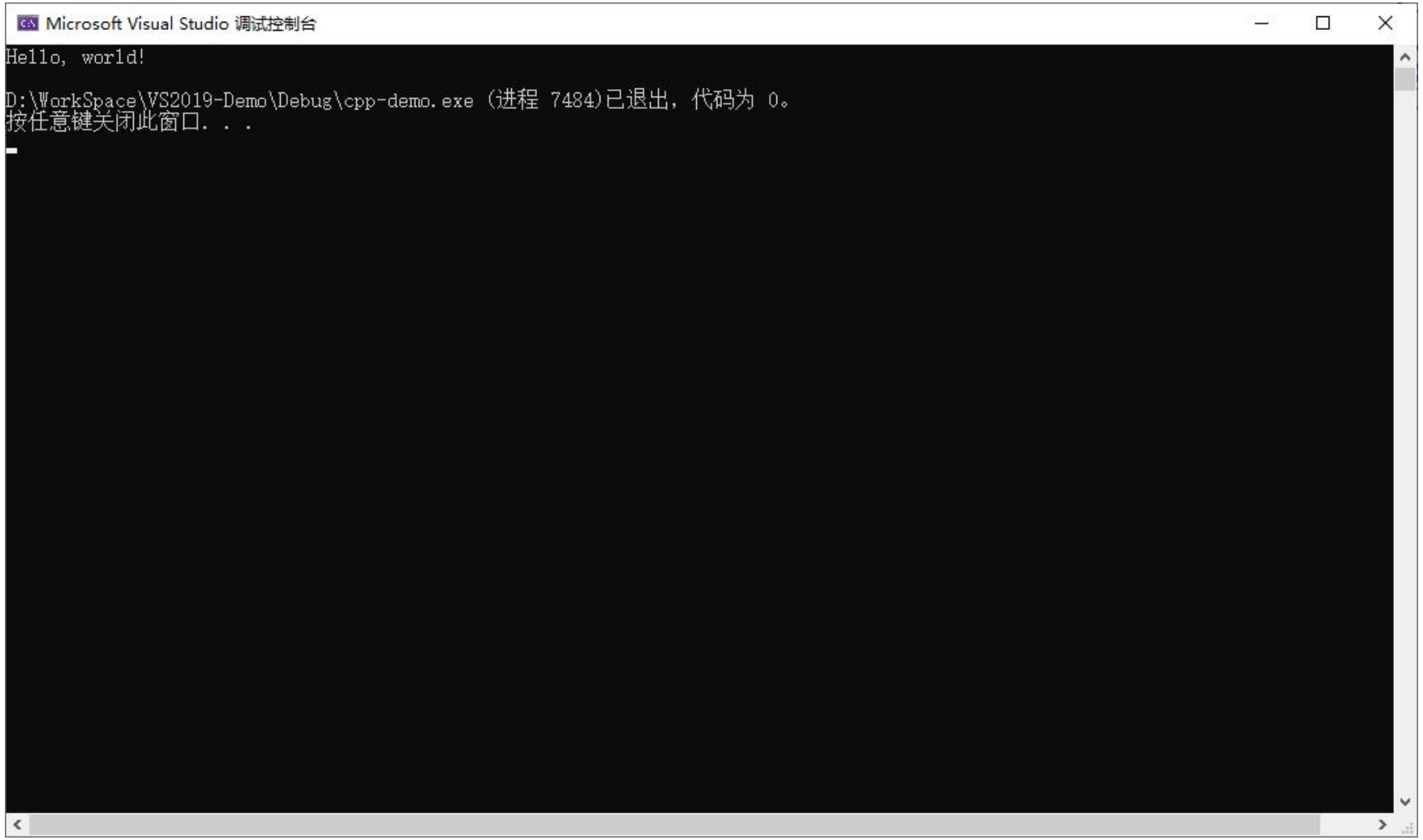
4、转换为pdf后提交

5、9月15日前网上提交本次作业（在“文档作业”中提交）

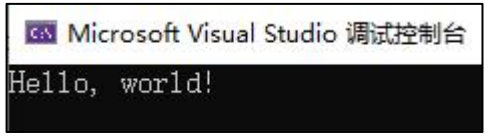
§.基础知识题 - 浮点数机内存储格式(IEEE 754)理解

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

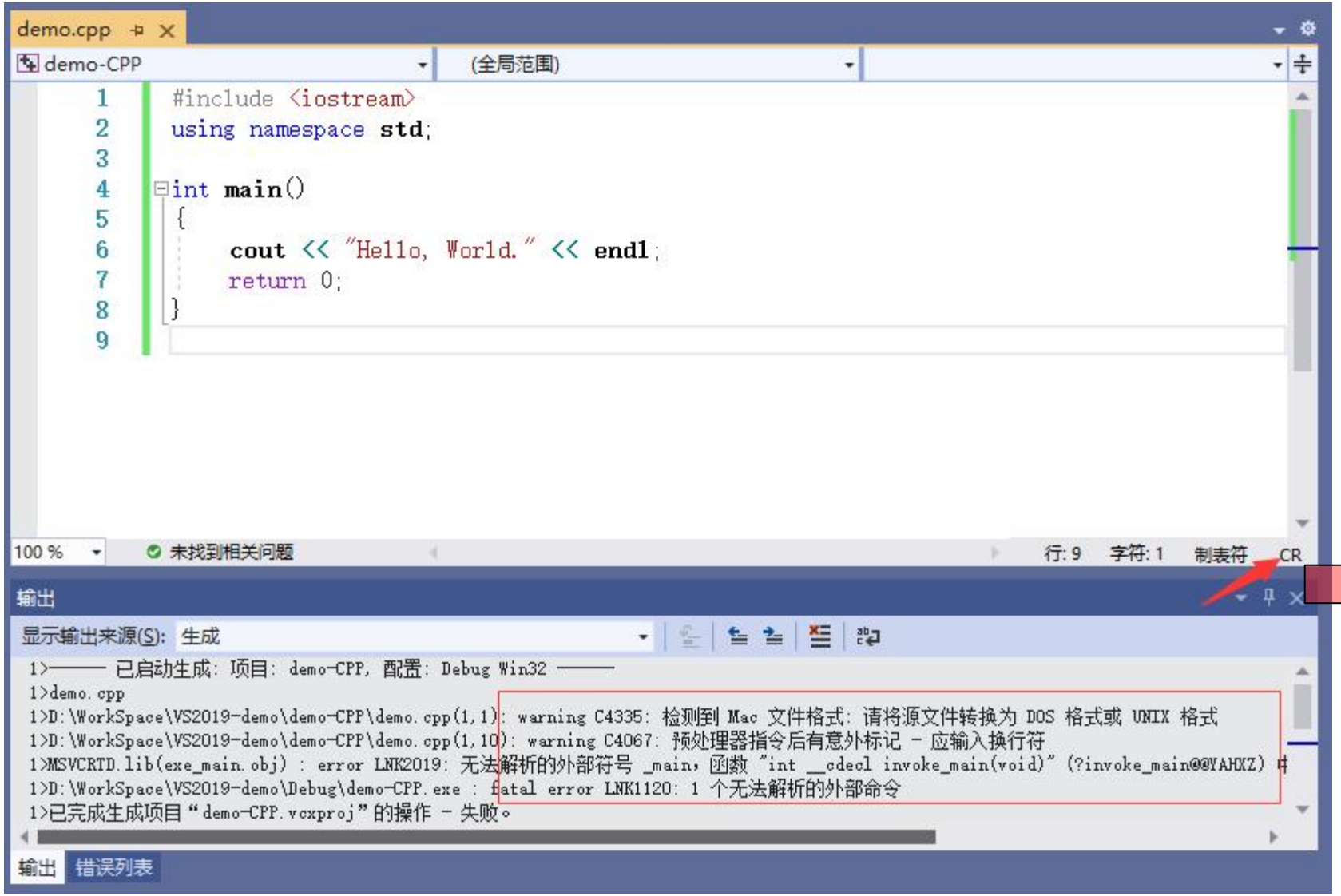


例：有效贴图



§.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

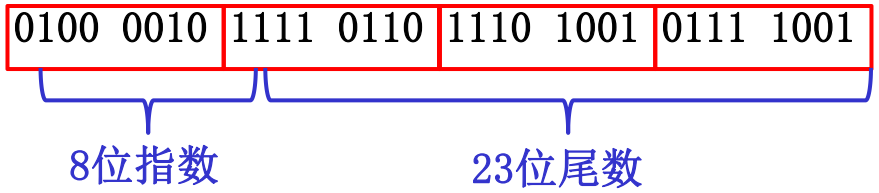
基础知识：用于看懂float型数据的内部存储格式的程序如下：

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 = 123.456f;
    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;
}
```

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：



§.基础知识题 - 浮点数机内存储格式(IEEE 754)理解

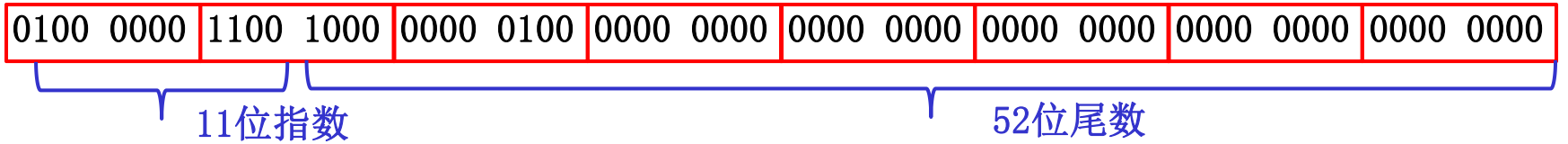
基础知识：用于看懂double型数据的内部存储格式的程序如下：

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：



§.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

§.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)

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2：1234567.7654321

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 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__0.17737555038452148 44___(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__1.17737555038452148 44___(加整数部分的1后)

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

= 0.17737555038452148 44 => 加1 => 1.17737555038452148 44

1.17737555038452148 44 * 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 x 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.1234567.7654321 （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：___0100 1010 0000 0011 0101 1010 1101 1011___

(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 0011 0101 1010 1101 1011__(填32bit中的原始形式)
尾数转换为十进制小数形式是__0.0262101888656__(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__1.0262101888656___(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.-7654321.1234567 （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：__1100 1010 1111 0111 1001 0011 0000 0000_____

(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__111 0111 1001 0011 0000 0000___(填32bit中的原始形式)
尾数转换为十进制小数形式是_0.9341735839843__(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_1.9341735839843__(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.0.001234567 （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：___0011 1011 0000 1101 0000 1010 1000 1100____

(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__000 1101 0000 1010 1000 1100___(填32bit中的原始形式)

尾数转换为十进制小数形式是__0.1018843650817___(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__1.1018843650817___(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.-0.007654321 （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：__1011 1100 0000 0100 1110 1010 0101 0111__

(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___-7___(32bit中的原始形式按IEEE754的规则转换)

尾数是___000 0100 1110 1010 0101 0111__(填32bit中的原始形式)

尾数转换为十进制小数形式是___0.0384014844894__(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___1.0384014844894___(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.1234567.7654321 （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：0100 0001 0100 0000 0110 1011 0101 1011 0110 0111 1101 0111 0001 0100 0100 1110

(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 0000 0110 1011 0101 1011 0110 0111 1101 0111 0001 0100 0100 1110 (填64bit中的原始形式)

尾数转换为十进制小数形式是 0.0262102180725 (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 1.0262102180725 (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.-7654321.1234567 （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：1100 0001 0101 1110 1111 0010 0110 0000 0000 1101 1100 0110 0000 0111 1011 1111

(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 1110 1111 0010 0110 0000 0000 1101 1100 0110 0000 0111 1011 1111 (填64bit中的原始形式)

尾数转换为十进制小数形式是 0.9341736352948 (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 1.9341736352948 (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.0.001234567 （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：0011 1111 0110 0001 1010 0001 0101 0001 1000 1101 1010 1011 0001 0010 1010 1101

(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 0001 1010 0001 0101 0001 1000 1101 1010 1011 0001 0010 1010 1101 (填64bit中的原始形式)

尾数转换为十进制小数形式是 0.101884416 (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 1.101884416 (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.-0.007654321 （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：1011 1111 1000 0000 1001 1101 0100 1010 1110 1101 1101 0011 1100 0111 0101 0111

(2) 其中：符号位是 1

指数是 011 1111 1000 (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 0000 1001 1101 0100 1010 1110 1101 1101 0011 1100 0111 0101 0111 (填64bit中的原始形式)

尾数转换为十进制小数形式是 0.038401536 (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 1.038401536 (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？

（1）float型的数据的32bit中，其中第一位是符号位，之后的8位是指数位，最后的23位是尾数位。

（2）尾数的正负是通过符号位来进行表示的，正数为0，负数为1.

（3）尾数是采用类似于十进制的科学计数法，将十进制的数化为整数部分为1的二进制小数，然后省略整数，剩下照抄，并补齐或者删减到23位即可。

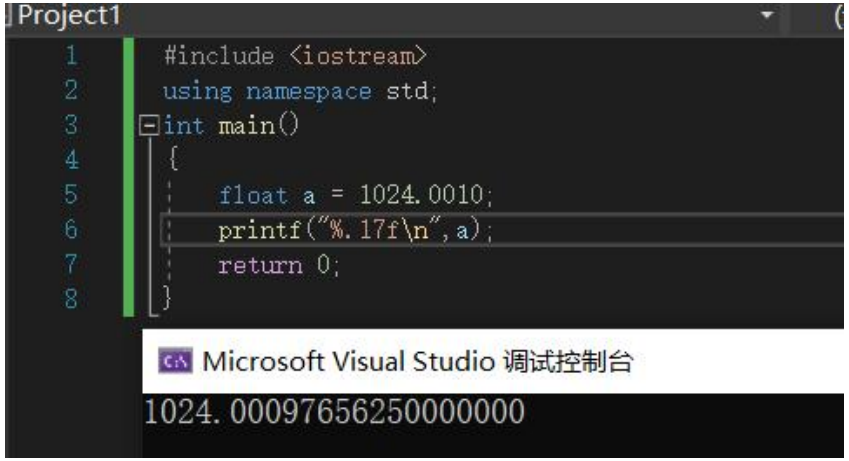
（4）指数部分并不是采用二进制补码的形式，为了方便比较，采用固定偏移值的方式，即将二进制转化为十进制后减去127就得到了真实的十进制指数。

(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？

（1）因为尾数的$2^{23}=8388608$是一个七位数，即为$10^6<2^{23}<10^7$，所以精度是7位，并且真实指数的十进制最大值是127，尾数的最大值是极其接近2，所以最大值是2^{128}，就是$3.4x10^{38}$.

（2）因为$\log 2^{23}=6.923$介于6-7之间，所以一定能够有6位有效数字的精度，不一定有7位有效数字的精度。



(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？

（1）double型的数据的64bit中，其中第一位是符号位，之后的11位是指数位，最后的52位是尾数位。

（2）尾数的正负是通过符号位来进行表示的，正数为0，负数为1.

（3）尾数是采用类似于十进制的科学计数法，将十进制的数化为整数部分为1的二进制小数，然后省略整数，剩下照抄，并补齐或者删减到52位即可。

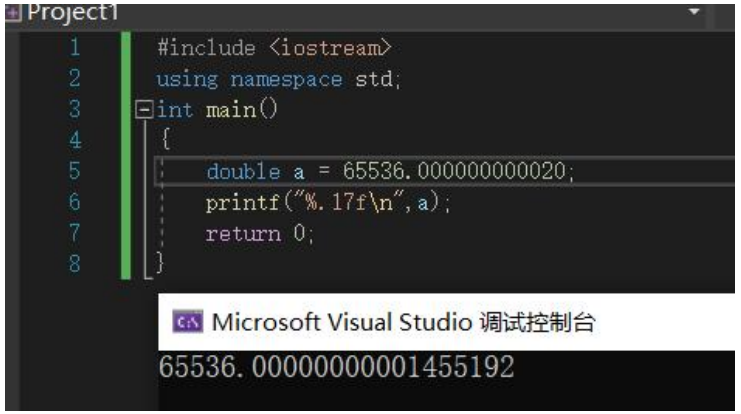
（4）指数部分并不是采用二进制补码的形式，为了方便比较，采用固定偏移值的方式，即将二进制转化为十进制后减去1023就得到了真实的十进制指数。

(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$1.7x10^{308}$ ？

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？

（1）因为尾数的$2^{23}=8388608$是一个七位数，即为$10^{15}<2^{52}<10^{16}$，所以精度是15位，并且真实指数的十进制最大值是1023，尾数的最大值是极其接近2，所以最大值是2^{1024}，就是$1.7x10^{308}$.

（2）因为$\log 2^{52}=15.6535$介于15-16之间，所以一定能够有15位有效数字的精度，不一定有16位有效数字的精度。



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