

§ . 基础知识题 – 浮点数机内存存储格式(IEEE 754)理解



要求:

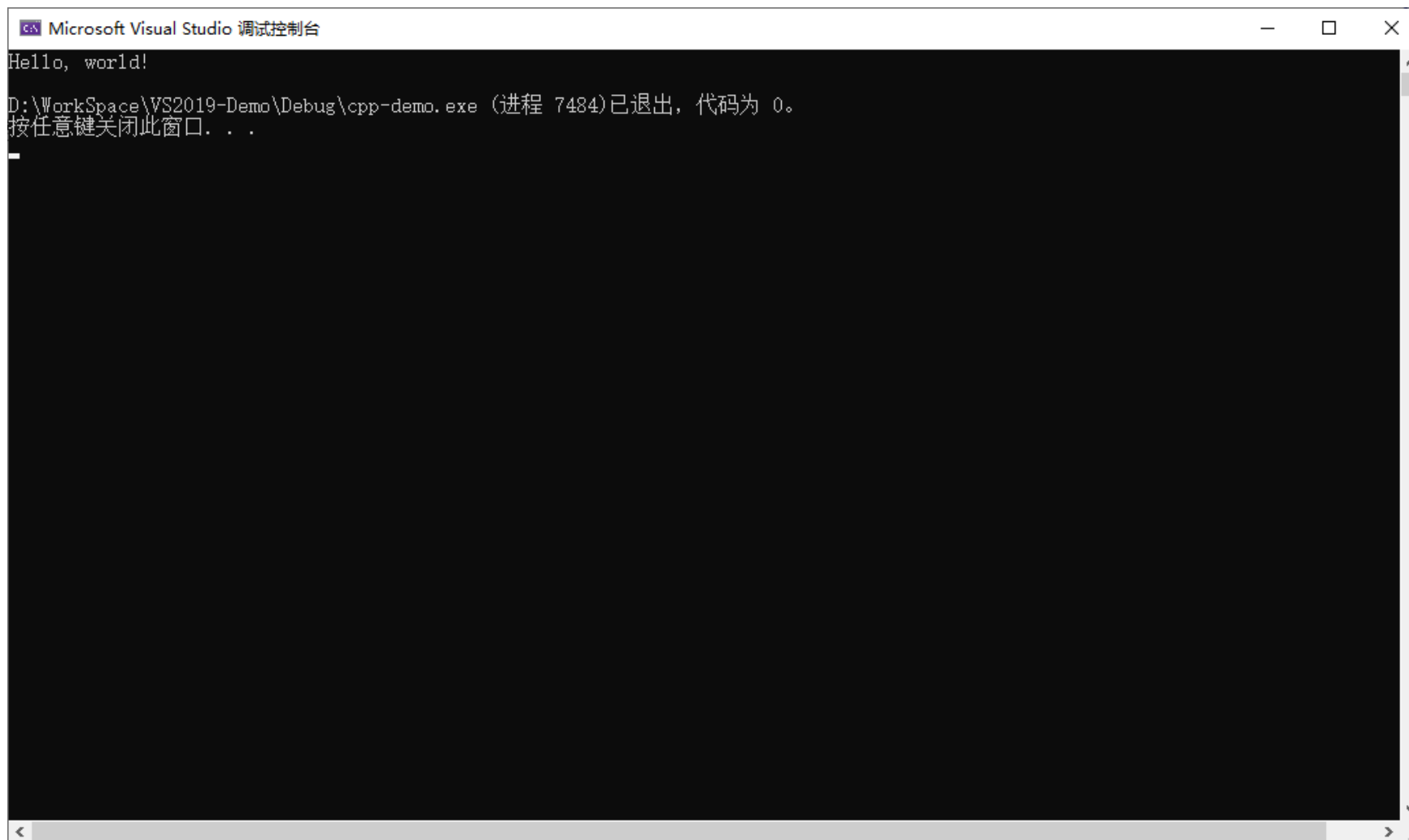
- 1、完成本文档中所有的题目并写出分析、运行结果
- 2、无特殊说明，均使用VS2022编译即可
- 3、直接在本文件上作答，**写出答案/截图（不允许手写、手写拍照截图）**即可；填写答案时，为适应所填内容或贴图，**允许调整**页面的字体大小、颜色、文本框的位置等
 - ★ 贴图要有效部分即可，不需要全部内容
 - ★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可
 - ★ **不允许**手写在纸上，再拍照贴图
 - ★ **允许**在各种软件工具上完成（不含手写），再截图贴图
- 4、转换为pdf后提交
- 5、**3月7日前**网上提交本次作业（在“文档作业”中提交）

§. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

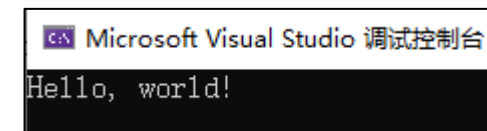


贴图要求：只需要截取输出窗口中的有效部分即可，如果全部截取/截取过大，则视为无效贴图

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, showing only the "Hello, world!" output. The window is titled "Microsoft Visual Studio 调试控制台". This is an example of a valid screenshot as it captures the relevant output without unnecessary context.



§. 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
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
```

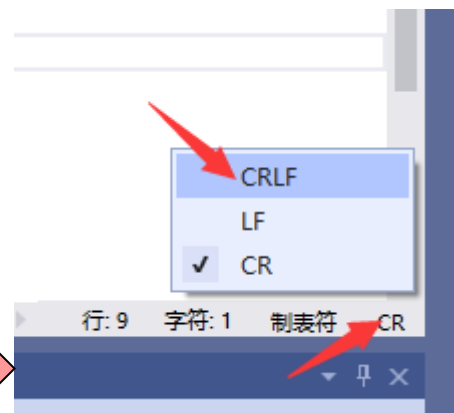
100 % 未找到相关问题 行: 9 字符: 1 制表符 CR

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

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. 2253893. 3983522 (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 1001 1001 0001 0001 0110_ (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 1001 1001 0001 0001 0110__ (填32bit中的原始形式)

尾数转换为十进制小数形式是_0. 074739933013916_ (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. 074739933013916 _ (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”)

B. -3983522. 2253893 (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 0111 0011 0010 0010 1000 1001_ (不是手算，用P. 4方式打印)

(2) 其中：符号位是__1__

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_111 0011 0010 0010 1000 1001_ (填32bit中的原始形式)

尾数转换为十进制小数形式是_0. 899491429328918_ (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. 899491429328918_ (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. 0.002253893 (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 0011 1011 0110 0000 1101_ (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 0011 1011 0110 0000 1101_ (填32bit中的原始形式)

尾数转换为十进制小数形式是__0.153993248939514__ (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__1.153993248939514__ (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. -0.003983522 (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 1000 0010 1000 1000 0011 0100_ (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_000 0010 1000 1000 0011 0100_ (填32bit中的原始形式)

尾数转换为十进制小数形式是_0.019781589508056640625_ (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_1.019781589508056640625_ (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. 2253893. 3983522 (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 0011 0010 0010 0010 1011 0010 1111 1101 0011 0100 0111 0100_ (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)

74
34
fd
b2
22
32
41
41

尾数是_0001 0011 0010 0010 0010 1011 0010 1111 1101 0011 0100 0111 0100_ (填64bit中的原始形式)

尾数转换为十进制小数形式是_0. 07474012296304710645244995248504_ (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. 07474012296304710645244995248504_ (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. -3983522. 2253893 (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 0100 1110 0110 0100 0101 0001 0001 1100 1101 1001 1000 1110 0111 1100_ (不是手算，用P. 5方式打印)

(2) 其中：符号位是____1____

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_1110 0110 0100 0101 0001 0001 1100 1101 1001 1000 1110 0111 1100_ (填64bit中的原始形式)

尾数转换为十进制小数形式是_0. 90046798009362216674844603403471_ (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. 90046798009362216674844603403471_ (加整数部分的1)

7c
8e
d9
1c
51
64
4e
c1



§. 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. 0.002253893 (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 0010 0111 0110 1100 0001 1001 0111 0010 1000 0110 1001 1111 0000_ (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__ (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_0010 0111 0110 1100 0001 1001 0111 0010 1000 0110 1001 1111 0000_ (填64bit中的原始形式)

尾数转换为十进制小数形式是_0.15399321599999993281926435884088_ (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_1.15399321599999993281926435884088_ (加整数部分的1)

f0
69
28
97
c1
76
62
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. -0.003983522 (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 0000 0101 0001 0000 0110 1000 1011 0110 1000 0000 0110 0010 1100_ (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_0000 0101 0001 0000 0110 1000 1011 0110 1000 0000 0110 0010 1100_ (填64bit中的原始形式)

尾数转换为十进制小数形式是__0.0197816319999999379319888248574__ (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_1.0197816319999999379319888248574_ (加整数部分的1)

2c
6
68
8b
6
51
70
bf

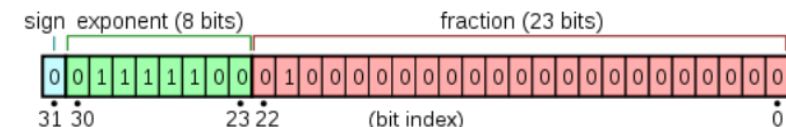
§. 基础知识题 - 浮点数机内存储格式(IEEE 754)理解



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？



答：如图，这32个二进制位的内存编号从高到低（从31到0），共包含如下几个部分：符号位，偏移后的指数位，尾数位。符号位：占据最高位（第31位）这一位，用于表示这个浮点数是正数还是负数，为0表示正数，为1表示负数。指数位占据第30位到第23位这8位，用于表示以2为底的指数，IEEE754规定，指数位用于表示 $[-127, 128]$ 范围内的指数，不过为了表示起来更方便，浮点型的指数位都有一个固定的偏移量(bias)，用于使 指数 + 这个偏移量 = 一个非负整数，规定：在32位单精度类型中，这个偏移量是127。尾数位：占据剩余的22位到0位这23位。用于存储尾数。在以二进制格式存储十进制浮点数时，首先要把十进制浮点数表示为二进制格式，拿十进制数20.5举例：十进制浮点数20.5 = 二进制10100.1然后，需要把这个二进制数转换为以2为底的指数形式：二进制10100.1 = 1.01001×2^4 注意转换时，对于乘号左边，加粗的那个二进制数1.01001，需要把小数点放在左起第一位和第二位之间。且第一位需要是个非0数。这样表示好之后，其中的1.01001就是尾数。

(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 3.4×10^{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，加上默认的小数点前的1位1， $2^{(23+1)} = 16777216$ 。因为 $10^7 < 16777216 < 10^8$ ，所以说单精度浮点数的有效位数是7位。但因为可能会有四舍五入，所以float型最少有6位有效数字。又因为指数位用于表示 $[-127, 128]$ 范围内的指数， 2^{127} 约为 3.4×10^{38} ，所以最大只能是 3.4×10^{38} 。

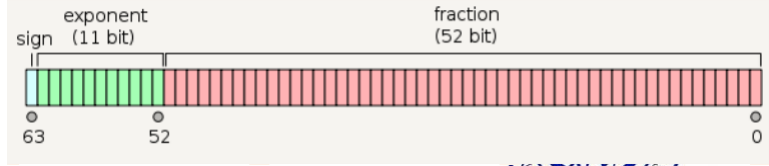
```
#include <stdio.h>
int main()
{
    float i=48965.943;
    printf("%f",i);
    return 0;
}
```

48965.941406
Process returned 0 (0x0) execution time : 0.340 s
Press any key to continue.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     float i = 1234567.123;
6     cout << i << endl;
7     return 0;
8 }
9
```

Microsoft Visual Studio 调试 ×
1.23457e+06

(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？



答：如图，这64个二进制位的内存编号从高到低（从63到0），共包含如下几个部分：符号位，偏移后的指数位，尾数位。
符号位：占据最高位(第63位)这一位，用于表示这个浮点数是正数还是负数，为0表示正数，为1表示负数。指数位占据第62位到第52位这11位，用于表示以2为底的指数，IEEE754规定，指数位用于表示[-1023, 1024]范围内的指数，不过为了表示起来更方便，浮点型的指数位都有一个固定的偏移量(bias)，用于使 指数 + 这个偏移量 = 一个非负整数，规定：在32位单精度类型中，这个偏移量是1023。尾数位：占据剩余的51位到0位这52位。用于存储尾数。

(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1.7x10³⁰⁸ ？

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？
双精度的尾数用52位存储， $2^{(52+1)} = 9007199254740992$ ， $10^{16} < 9007199254740992 < 10^{17}$ ，所以双精度的有效位数是16位。但因为可能会有四舍五入，所以double型最少有15位有效数字。又因为指数位用于表示[-1023, 1024]范围内的指数， 2^{1024} 约为 1.7×10^{308} ，所以最大只能是 1.7×10^{308} 。

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

```
double a = 0.0000001234567890123456789;  
std::cout << std::setprecision(20) << a << std::endl;  
  
1.2345678901234568e-07
```

```
double b = 123456789012345.6789;  
std::cout << std::setprecision(20) << b << std::endl;  
  
123456789012345.671875
```



§. 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，是指数部分+偏移量得到的非负整数的2进制原码。

(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;
}
```

左侧程序1.2转化成二进制，小数部分0.2无限循环，double中用52位存储，将double赋值给float时，用23位存储，丢失了32位，精度下降，数据截断。

右侧程序100.25转化成二进制，小数部分0.25有限循环，double中只需要2位，用0补齐到52位，将double赋值给float时，用23位存储，截掉的是32位0，不影响结果。

规律：double赋值给float时，double型常量不加F的情况下，小数部分若为5的倍数，则不会发生截断。

