

§. 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

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 occupies most of the left side of the slide.

例：有效贴图

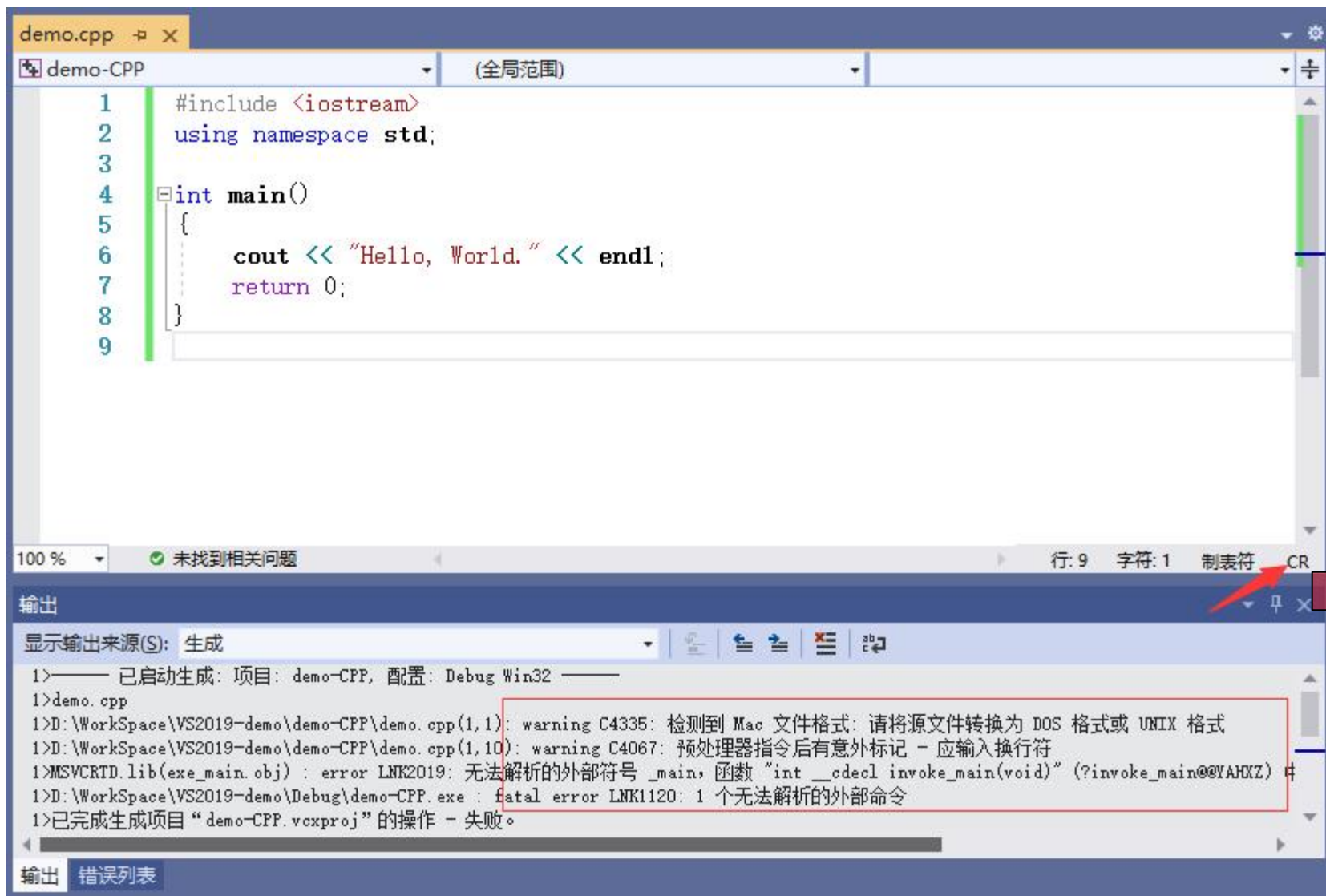
A screenshot of the Microsoft Visual Studio debug console window, showing only the "Hello, world!" text. The window is titled "Microsoft Visual Studio 调试控制台". This is an example of a valid screenshot.

§. 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，四个字节的价值依次为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;
}
```

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     double d = 2.53e7;
6     unsigned char* p = (unsigned char*)&d;
7     cout << hex << (int)(*p) << endl;
8     cout << hex << (int)*(p + 1) << endl;
9     cout << hex << (int)*(p + 2) << endl;
10    cout << hex << (int)*(p + 3) << endl;
11    cout << hex << (int)*(p + 4) << endl;
12    cout << hex << (int)*(p + 5) << endl;
13    cout << hex << (int)*(p + 6) << endl;
14    cout << hex << (int)*(p + 7) << endl;
15    return 0;
16 }
```

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

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

（小端向）

§ . 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×tamp=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 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 0.2000000476837158203125 (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 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. 2353726.6273532 (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 1010 1000 1111 1011 (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)

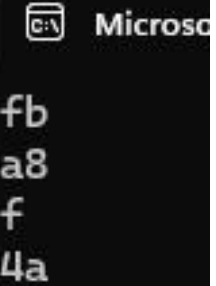
指数表示的十进制形式是21 (32bit中的原始形式按IEEE754的规则转换)

尾数是000 1111 1010 1000 1111 1011 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.12234437465667725 (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 1.12234437465667725 (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：1100-1010-1011-1111-0111-0011-1111-1000（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0111-1110-1110-0111-1111-000（填32bit中的原始形式）

尾数转换为十进制小数形式是0.4957265853881836（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1.4957265853881836（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：0011-1011-0001-1010-0100-0000-1111-1000 (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0011-0100-1000-0001-1111-000 (填32bit中的原始形式)

尾数转换为十进制小数形式是0.2051076889038086 (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1.2051076889038086 (加整数部分的1)

Microsoft

f8

40

1a

3b



§. 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：1011-1011-1100-1101-1001-0010-0011-0011（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1001-1011-0010-0100-0110-011（填32bit中的原始形式）

尾数转换为十进制小数形式是 0.6060241460800171（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 1.6060241460800171（加整数部分的1）

Microsoft

33
92
cd
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

(1) 得到的64bit的机内表示是： 0100 0001 0100 0001 1111 0101 0001 1111 0101 0000
0100 1101 0001 1100 0001 0011 (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 0001 (填64bit中的原始形式)

指数转换为十进制形式是16705 (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15682 (64bit中的原始形式按IEEE754的规则转换)

尾数是0001 1111 0101 0001 1111 0101 0000 0100 1101 0001 1100 0001 0011 (填64bit中的原始形式)

尾数转换为十进制小数形式是0. 12234431617412578 (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. 12234431617412578 (加整数部分的1)



Micro

13

1c

4d

50

1f

f5

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

(1) 得到的64bit的机内表示是：1100 0001 0101 0111 1110 1110 0111 1111
0000 1111 0001 0000 0101 1000 0011 1101 (不是手算，用P.5方式打印)

(2) 其中：符号位是1

指数是**1000 0010 101**(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**0111 1110 1110 0111 1111 0000 1111 0001 0000 0101 1000 0011 1101**(填64bit中的原始形式)

尾数转换为十进制小数形式是0.4957266415053845 (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1.4957266415053845 (加整数部分的1)

Microsoft Visual Stu

3d
58
10
f
7f
ee
57
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. 0.001234567 (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 1000 0001 1111
0000 0110 0011 0011 0010 1000 0111 1111 (不是手算，用P.5方式打印)

(2) 其中：符号位是0

指数是0111 1110 110 (填64bit中的原始形式)

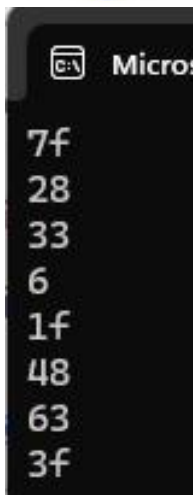
指数转换为十进制形式是1014 (64bit中的原始形式按二进制原码形式转换)

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 1000 0001 1111 0000 0110 0011 0011 0010 1000 0111 1111 (填64bit中的原始形式)

尾数转换为十进制小数形式是0.205107712 (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1.205107712 (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 0111 1001 1011 0010 0100 0110
0110 1100 0101 0011 1001 1000 1110 0010 (不是手算，用P.5方式打印)

(2) 其中：符号位是1

指数是**01111110111**(填64bit中的原始形式)

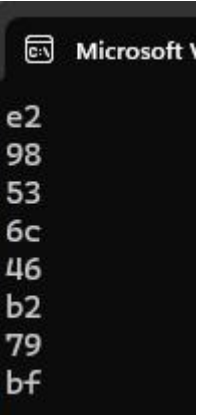
指数转换为十进制形式是1015(64bit中的原始形式按二进制原码形式转换)

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**1001 1011 0010 0100 0110 0110 1100 0101 0011 1001 1000 1110 0010**(填64bit中的原始形式)

尾数转换为十进制小数形式是0.606024192 (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1.606024192 (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. 31位：符号位，表示尾数正负；30-23位：偏移后指数位，化为十进制后-127正负即为符号，大小表示指数大小；22-0位：尾数位

(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 3.4×10^{38} ？

因为单精度型的float是：内存会分配4个字节的长度空间。2的3次方的4次方，也就是2的12次方 ($3.4 \times 10^{-38} \sim 3.4 \times 10^{38}$)

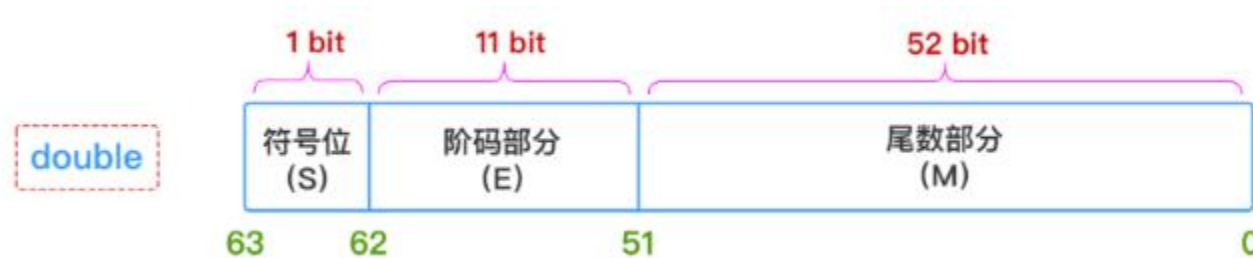
单精度数的尾数用23位存储，加上默认的小数点前的1位1， $2^{(23+1)} = 16777216$ 。因为 $10^7 < 16777216 < 10^8$ ，所以说单精度浮点数的有效位数是7位

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？

我们说有效位是 6 至 7 位，其实是在说，它有六七位是“可信的”

16777217, 16777216有效数是7位； 3.123456 有效数是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？

以上问题解释同本页 (1)



- (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} ？
- Double(双精度浮点数)使用 64 位 (8字节) 来储存一个浮点数. 它可以表示十进制的15或16位有效数字, 负值取值范围为 $-1.79769313486231570E+308$ 到 $-4.94065645841246544E-324$, 正值取值范围为 $4.94065645841246544E-324$ 到 $1.79769313486231570E+308$ 一个字节两位, 8个字节就是16位。
- 它的64位可以划分为 符号位 指数位 和 尾数位它的64位可以划分为 符号位 指数位 和 尾数位它的64位可以划分为 符号位 指数位 和 尾数位, 308只能作为指数位存储, 因此上限只能是 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?

1. 截断误差的例子

如果我们尝试将一个非常小的数存储在double类型中, 它可能会遇到截断误差, 这可能导致我们失去一些有效位数。例如:

```
double a = 0.0000001234567890123456789;  
std::cout << std::setprecision(20) << a << std::endl;
```

输出的是:

```
1.2345678901234568e-07
```

在这种情况下, 我们只能存储15位有效数字, 因为最后一位数字被截断了。

2. 舍入误差的例子

如果我们尝试在double类型中存储一个非常大的数, 它可能会遇到舍入误差, 这可能导致我们失去一些精度。例如:

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

输出结果:

```
123456789012345.671875
```

在这种情况下, 我们只能存储16位有效数字, 但我们失去了一些精度, 因为一些数字被舍入了。

§. 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。偏移法。

(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;
}
```

double转float时尾数指数分别转换, 指数转换不改变, 而小数转换左侧无法转换为二进制有限小数, 会截断; 右侧可以转化为有限小数, 不会截断。

