

§ . 基础知识题 – 浮点数机内存存储格式(IEEE 754)理解



要求:

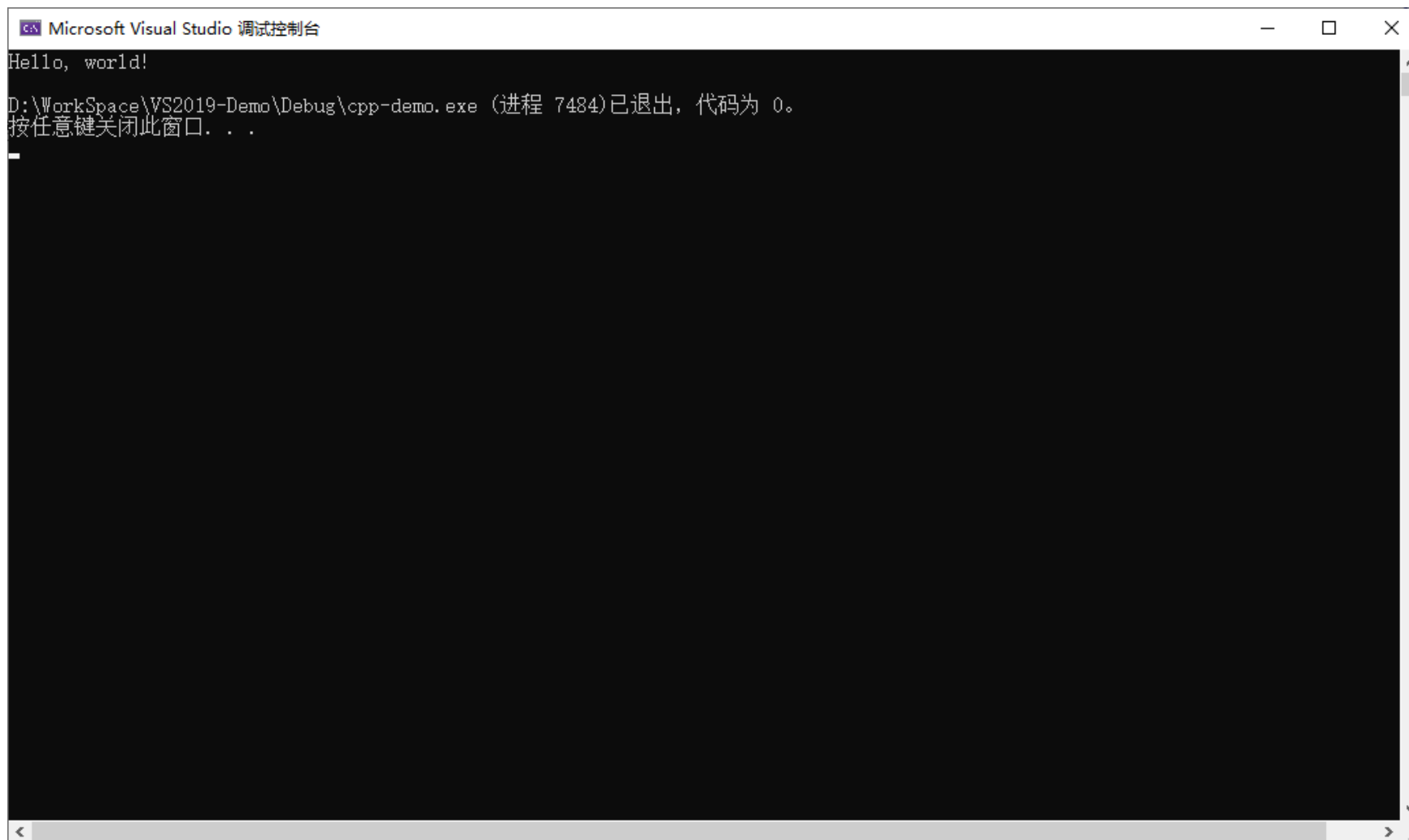
- 1、完成本文档中所有的题目并写出分析、运行结果
- 2、无特殊说明，均使用VS2022编译即可
- 3、直接在本文件上作答，**写出答案/截图（不允许手写、手写拍照截图）**即可；填写答案时，为适应所填内容或贴图，**允许调整**页面的字体大小、颜色、文本框的位置等
 - ★ 贴图要有效部分即可，不需要全部内容
 - ★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可
 - ★ **不允许**手写在纸上，再拍照贴图
 - ★ **允许**在各种软件工具上完成（不含手写），再截图贴图
- 4、转换为pdf后提交
- 5、**9月15日前**网上提交本次作业（在“文档作业”中提交）

§. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解

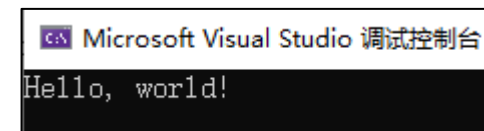


贴图要求：只需要截取输出窗口中的有效部分即可，如果全部截取/截取过大，则视为无效贴图

例：无效贴图

A screenshot of the Microsoft Visual Studio debug console window. The window title is "Microsoft Visual Studio 调试控制台". The output text is: "Hello, world!", "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 entire window, including its title bar and scrollbars, is captured, which is considered an invalid screenshot according to the requirements.

例：有效贴图

A screenshot of the Microsoft Visual Studio debug console window, but only the output text is captured. The text is: "Hello, world!". This is considered a valid screenshot as it only contains the effective output part.



§. 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 x
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
```

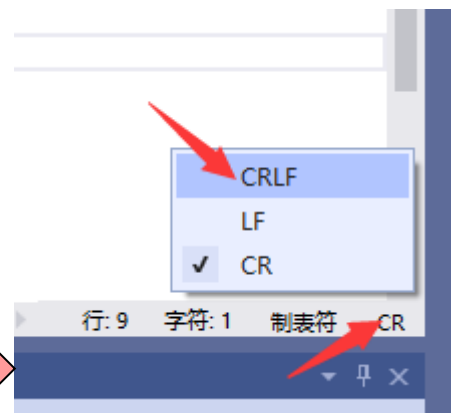
100 % 未找到相关问题 行: 9 字符: 1 制表符 CR

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 = 123.456f;
    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;
}
```

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

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

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/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



§. 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)

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 0.1773755503845214844 (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 1.1773755503845214844 (加整数部分的1后)

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

= 0.1773755503845214844 => 加1 => 1.1773755503845214844

$1.1773755503845214844 * 2^{20} = 1234567.75$ (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. 2152988.8892512 (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： 01001010000000110110100001110100

(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 0110 1000 0111 0100 (填32bit中的原始形式)

尾数转换为十进制小数形式是 0.026625156402588 (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 1.026625156402588 (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. -8892512. 2152988 (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：_____11001011000001111011000001100000_____

(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 0111 1011 0000 0110 0000_____ (填32bit中的原始形式)

尾数转换为十进制小数形式是_____0. 0600700378417969_____ (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. 0600700378417969_____ (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. 0.002152988 (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：_____ 00111011000011010001100100100101 _____

(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 0001 1001 0010 0101___ (填32bit中的原始形式)

尾数转换为十进制小数形式是_0.1023298501968384_ (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_1.1023298501968384_ ((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. -0.008892512 (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：_____10111100000100011011000111100110_____

(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____001 0001 1011 0001 1110 0110_____(填32bit中的原始形式)

尾数转换为十进制小数形式是__0.1382415294647217____(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__1.1382415294647217____(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. 2152988.8892512 (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：

_____0100000101000000011011010000111001110001110100101111101110111011_____

(2) 其中：符号位是__0__

指数是_____10000010100_____ (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_____0000011011010000111001110001110100101111101110111011_____ (填64bit中的原始形式)

尾数转换为十进制小数形式是_____ 0.026625103593444832 _____ (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_____ 1.026625103593444832 _____ (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 . -8892512. 2152988 (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：

_____1100000101100000111101100000110000000110111000111011101001001111_____

(2) 其中：符号位是_____1_____

指数是_____10000010110_____ (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_____0000111101100000110000000110111000111011101001001111_____ (填64bit中的原始形式)

尾数转换为十进制小数形式是_____ 0. 0600700635074138 _____ (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. 0600700635074138 _____ (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. 0.002152988 (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：

_____0011111101100001101000110010010010100001100011101100101001001101_____

(2) 其中：符号位是___0___

指数是___01111110110___ (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___0001101000110010010010100001100011101100101001001101___ (填64bit中的原始形式)

尾数转换为十进制小数形式是___0.102329856___ (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___1.102329856___ (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 . -0.008892512 (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：

_____1011111110000010001101100011110011000001110000010001100111010101_____

(2) 其中：符号位是__1_____

指数是__01111111000_____ (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)

形式) 尾数是_____0010001101100011110011000001110000010001100111010101_____ (填64bit中的原始形式)

转换) 尾数转换为十进制小数形式是__ 0.138241536 _____ (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__ 1.138241536 _____ (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

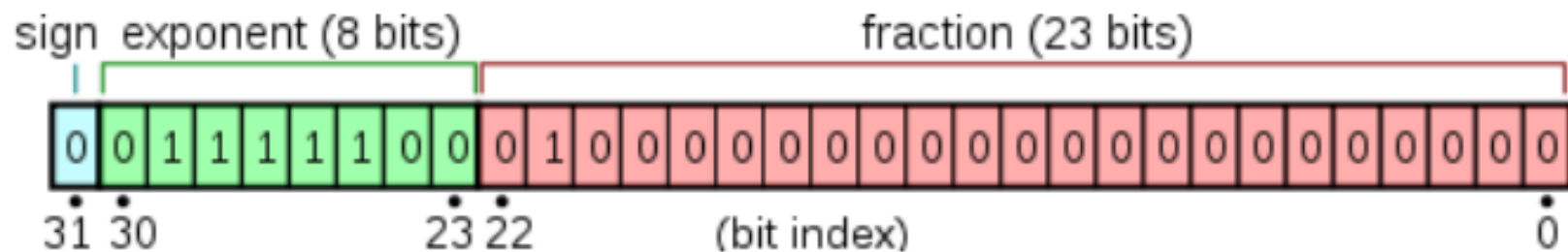
- (1) float型数据的32bit是如何分段来表示一个单精度的浮点数的？给出bit位的分段解释
尾数的正负如何表示？尾数如何表示？指数的正负如何表示？指数如何表示？
- (2) 为什么float型数据只有7位十进制有效数字？为什么最大只能是 3.4×10^{38} ？
有些资料上说有效位数是6~7位，能找出6位/7位不同的例子吗？
- (3) double型数据的64bit是如何分段来表示一个双精度的浮点数的？给出bit位的分段解释
尾数的正负如何表示？尾数如何表示？指数的正负如何表示？指数如何表示？
- (4) 为什么double型数据只有15位十进制有效数字？为什么最大只能是 1.7×10^{308} ？
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？

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



(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？



32 位浮点数内存占用示意图，共使用了 32 个小格子

1. 符号位（蓝色）

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。（尾数正负）

2. 偏移后的指数位（绿色）

指数位占据第 30 位到第 23 位这 8 位。如上图的绿色部分。

用于表示以 2 为底的指数。8 位二进制可以表示 256 种状态，IEEE754 规定，指数位用于表示 $[-127, 128]$ 范围内的指数。

浮点型的指数位都有一个固定的偏移量(bias)，用于使 指数 + 偏移量 = 一个非负整数。

在 32 位单精度类型中，这个偏移量是 127。

3. 尾数位（红色）

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，首先需要把十进制浮点数表示为二进制格式

然后，需要把这个二进制数转换为以 2 为底的指数形式：

尾数部分的最高位始终为 1.，所以可以隐藏高位 1. 尾数其实只的是是隐藏了整数部分 1. 之后，剩下的小数部分



(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？

一、大体来说, 32 位浮点数的精度是 7 位有效数。由于计算机内数据的存储是离散的，所以最小的数据单元之间是有间隔的，这就导致了有效数字的产生（过小的数据无法准确表示）。由浮点数的存储形式，我们可以推知：间隔 = 指数间差值 / 移动次数 = (终点对应的值 - 起点对应的值) / 2^{23} 。

根据右图 wiki 整理好的间隔数据，我们可以知道：

1、1024 ~ 2048 范围中的 间隔约为 0.000122070，只能精确到小数点后三位（无法存储 0.0005），加上四位整数（1024-2048），是七位有效数字

2、8388608 ~ 16777215 范围中的间隔为 1, 所以精度为 7-8 位整数。

3、1 ~ 2 范围中的间隔为 1.19209×10^{-7} ，只能精确到小数点后 6 位（无法存储 0.0000012），加上一位整数 (1-2)，是七位有效数字。（其他范围内同理）

真实指数	偏移后的指数	最小值	最大值	间隔
Actual Exponent (unbiased)	Exp (biased)	Minimum	Maximum	Gap
-1	126	0.5	≈ 0.999999940395	$\approx 5.96046 \times 10^{-8}$
0	127	1	≈ 1.999999880791	$\approx 1.19209 \times 10^{-7}$
1	128	2	≈ 3.999999761581	$\approx 2.38419 \times 10^{-7}$
2	129	4	≈ 7.999999523163	$\approx 4.76837 \times 10^{-7}$
10	137	1024	≈ 2047.999877930	$\approx 1.22070 \times 10^{-4}$
11	138	2048	≈ 4095.999755859	$\approx 2.44141 \times 10^{-4}$
23	150	8388608	16777215	1
24	151	16777216	33554430	2
127	254	$\approx 1.70141 \times 10^{38}$	$\approx 3.40282 \times 10^{38}$	$\approx 2.02824 \times 10^{31}$

二、对于规格数，由浮点数的存储形式，最大值 $< 2 \times 2^{127} \approx 3.4 \times 10^{38}$

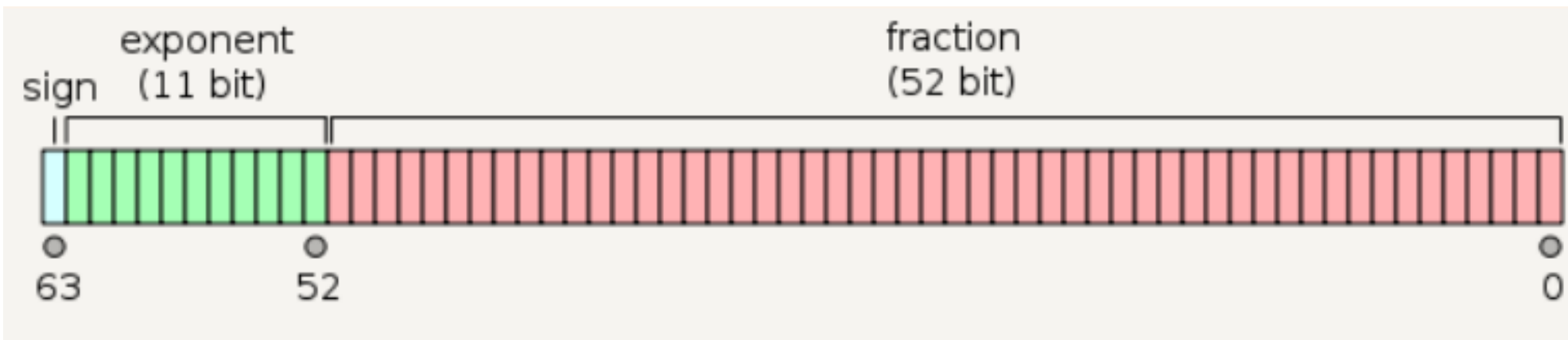
三、对于精度只有 6 位，是因为存储的值不是一个蓝点值时，会发生舍入，自动舍入到离它最近的一个蓝点值。

例如 1024.001，会舍入到离它最近的蓝点 1024.000976...，体现的好像精度不足 7 位。

而 1024.0011，就会舍入到离它最近的蓝点 1024.00109...，体现的好像精度又足 7 位了。



(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？



64 位浮点数内存占用示意图，共使用了 64 个小格子

1. 符号位（蓝色）

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。（尾数正负）

2. 偏移后的指数位（绿色）

指数位占据第 62 位到第 52 位这 11 位。如上图的绿色部分。

用于表示以 2 为底的指数。11 位二进制可以表示 2048 种状态，IEEE754 规定，指数位用于表示 $[-1023, 1024]$ 范围内的指数。

浮点型的指数位都有一个固定的偏移量(bias)，用于使 指数 + 偏移量 = 一个非负整数。

在 64 位单精度类型中，这个偏移量是 1023。

3. 尾数位（红色）

尾数位：占据剩余的 51 位到 0 位这 52 位。用于存储尾数。

在以二进制格式存储十进制浮点数时，首先需要把十进制浮点数表示为二进制格式

然后，需要把这个二进制数转换为以 2 为底的指数形式：

尾数部分的最高位始终为 1.，所以可以隐藏高位 1. 尾数其实只的是是隐藏了整数部分 1. 之后剩下的小数部分



(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？

一、由于计算机内数据的存储是离散的，所以最小的数据单元之间是有间隔的，这就导致了有效数字的产生（过小的数据无法准确表示）。由浮点数的存储形式，我们可以推知：间隔 = 指数间差值 / 移动次数 = (终点对应的值 - 起点对应的值) / 2^{52} 。因为 $2^{52} = 4503599627370496$ ，这意味着最多能有16位有效数字，但是能绝对能保证的为15位，精度为 15~16 位。

二、对于规格数，由浮点数的存储形式，最大值 $< 2 \times 2^{1023} \approx 1.7 \times 10^{308}$

三、

1、 $1024 \sim 2048$ 范围中的 间隔 约为 2.27×10^{-13} ，只能精确到小数点后12位，加上四位整数，是16位有效数字

2、 $8388608 \sim 16777215$ 范围中的间隔为 0.0000000018 ，只能精确到小数点后8位，加上7-8位整数，精度为15-16位整数。

3、 $1 \sim 2$ 范围中的间隔为 2.22×10^{-16} ，只能精确到小数点后15位，加上一位整数（1-2），是16位有效数字。（其他范围内同理）