

§. 基础知识题 – 浮点数机内存储格式(IEEE 754)理解



要求:

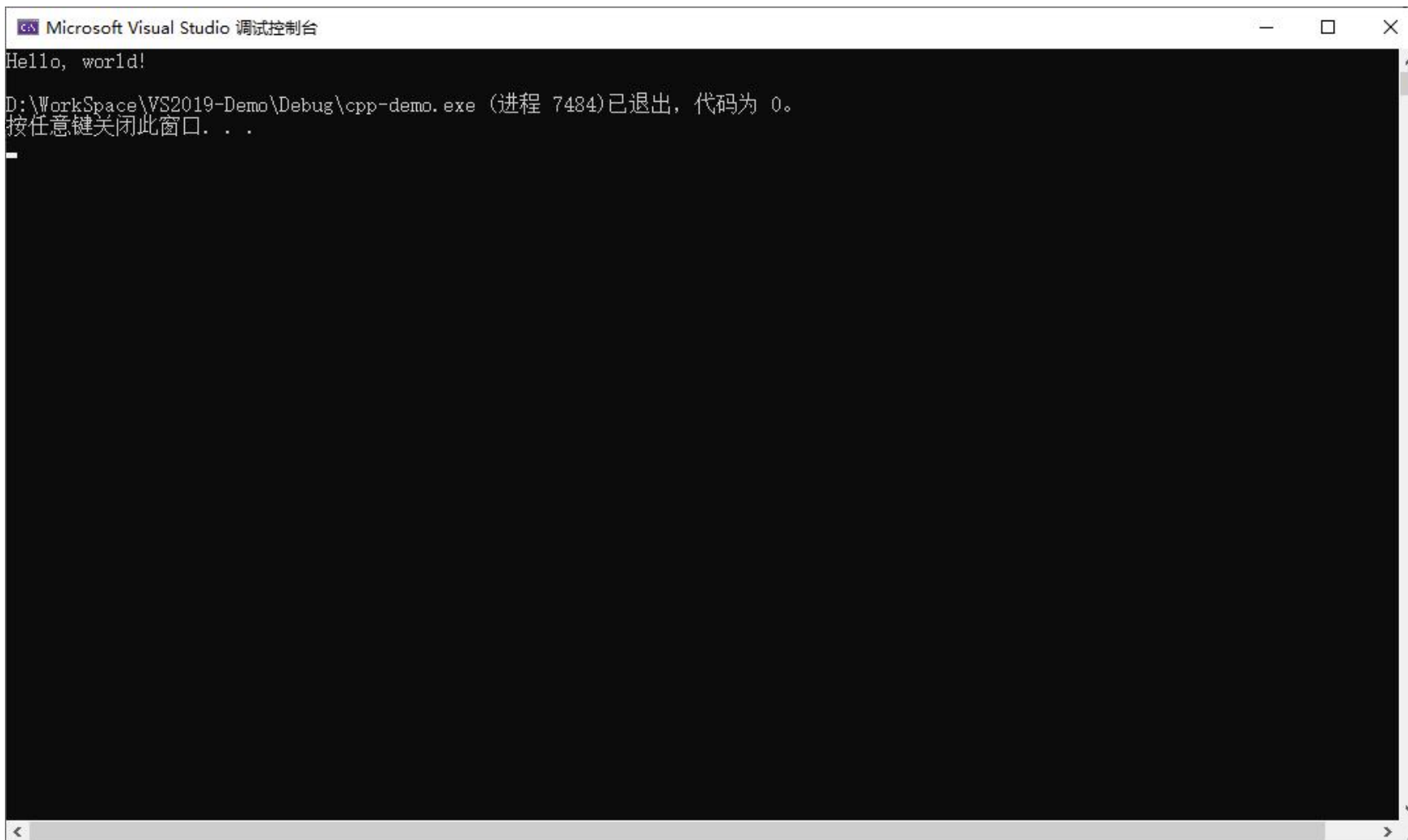
- 1、完成本文档中所有的题目并写出分析、运行结果
- 2、无特殊说明，均使用VS2022编译即可
- 3、直接在本文件上作答，**写出答案/截图（不允许手写、手写拍照截图）**即可；填写答案时，为适应所填内容或贴图，**允许调整**页面的字体大小、颜色、文本框的位置等
 - ★ 贴图要有效部分即可，不需要全部内容
 - ★ 在保证一页一题的前提下，具体页面布局可以自行发挥，简单易读即可
 - ★ **不允许**手写在纸上，再拍照贴图
 - ★ **允许**在各种软件工具上完成（不含手写），再截图贴图
- 4、转换为pdf后提交
- 5、**9月15日前**网上提交本次作业（在“文档作业”中提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

§. 基础知识题 - 浮点数机内存储格式(IEEE 754)理解

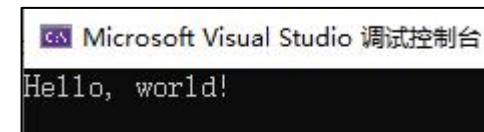


贴图要求：只需要截取输出窗口中的有效部分即可，如果全部截取/截取过大，则视为无效贴图

例：无效贴图

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 newline, and then "D:\Workspace\VS2019-Demo\Debug\cpp-demo.exe (进程 7484)已退出, 代码为 0." followed by another newline and "按任意键关闭此窗口. . .". The window is large, showing a significant portion of the screen.

例：有效贴图

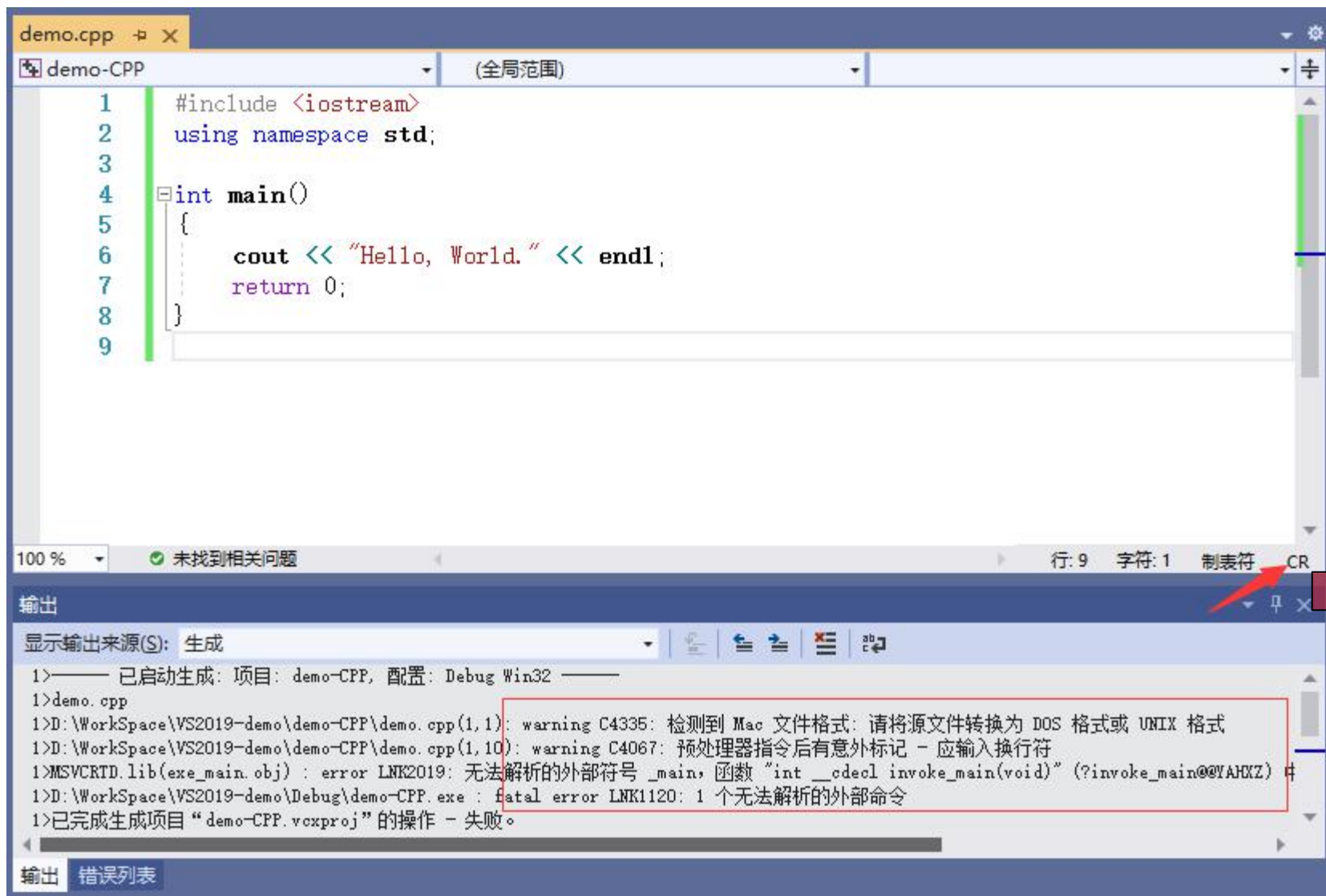
A screenshot of the Microsoft Visual Studio debug console window, showing only the first line of output: "Hello, world!". The window is titled "Microsoft Visual Studio 调试控制台". This is an example of a "valid" screenshot as it captures only the relevant output.

§. 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;
}
```

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;
}
```

Microsoft
0
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
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------

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=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 = $2^0 + 2^{-1} + 2^{-4} + 2^{-8}$

= $0.5 + 0.0625 + 0.00390625 = 0.56640625 \Rightarrow$ 加1 $\Rightarrow 1.56640625$

$1.56640625 \times 2^6 = 100.25$ (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 = $1.1001 0001 \times 2^6$ (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 \Rightarrow 1001 0001 0000 0000 0000 0000 (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. 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

2152402.2042512

(1) 得到的32bit的机内表示是：_____0100 1010 0000 0011 0101 1111 0100 1000_____

(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 1111 0100 1000_____ (填32bit中的原始形式)

尾数转换为十进制小数形式是_____0.026345252990722656_____ (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_____1.026345252990722656_____ (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

-2042512.2152402

(1) 得到的32bit的机内表示是：____1100 1001 1111 1001 0101 0100 1000 0001____

(2) 其中：符号位是____1____

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)

尾数是____111 1001 0101 0100 1000 0001____ (填32bit中的原始形式)

尾数转换为十进制小数形式是____0.947891354560852____ (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____1.947891354560852____ (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

0.002152402

(1) 得到的32bit的机内表示是：___0011 1011 0000 1101 0000 1111 0101 0000___

(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 1111 0101 0000___ (填32bit中的原始形式)

尾数转换为十进制小数形式是___0.10202980041503906___ (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___1.10202980041503906___ (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

-0.002042512

(1) 得到的32bit的机内表示是：_____1011 1011 0000 0101 1101 1011 1010 1010_____

(2) 其中：符号位是_____1_____

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 0101 1101 1011 1010 1010_____ (填32bit中的原始形式)

尾数转换为十进制小数形式是__0.04576611518859863__ (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__1.04576611518859863_ (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

2152402.2042512

(1) 得到的64bit的机内表示是：

___01000001 01000000 01101011 11101001 00011010 00100100 11100111 01000000___

(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 1110 1001 0001 1010 0010 0100 1110 0111 0100 0000_ (填64bit中的原始形式)

尾数转换为十进制小数形式是__0.026345350385284405_ (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__1.026345350385284405__ (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

-2042512. 2152402

(1) 得到的64bit的机内表示是：

__11000001 00111111 00101010 10010000 00110111 00011001 11111011 01010011__

(2) 其中：符号位是__1__

指数是__100 0001 0011__ (填64bit中的原始形式)

指数转换为十进制形式是__1043__ (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__20__ (64bit中的原始形式按IEEE754的规则转换)

尾数是_1111 0010 1010 1001 0000 0011 0111 0001 1001 1111 1011 0101 0011_ (填64bit中的原始形式)

尾数转换为十进制小数形式是__0. 9478914406206129__ (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. 9478914406206129_ (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

0.002152402

(1) 得到的64bit的机内表示是：

00111111 01100001 10100001 11101010 00000110 01010100 10111111 00000101

(2) 其中：符号位是____0____

指数是__11 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 1110 1010 0000 0110 0101 0100 1011 1111 0000 0101_ (填64bit中的原始形式)

尾数转换为十进制小数形式是__0.1020298239999994__ (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__1.1020298239999994_ (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

-0.002042512

(1) 得到的64bit的机内表示是：

__10111111 01100000 10111011 01110101 01000111 10111011 11100111 10001000__

(2) 其中：符号位是__1__

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_0000 1011 1011 0111 0101 0100 0111 1011 1011 1110 0111 1000 1000_ (填64bit中的原始形式)

尾数转换为十进制小数形式是_0.0457661439999999_ (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_1.0457661439999999_ (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：

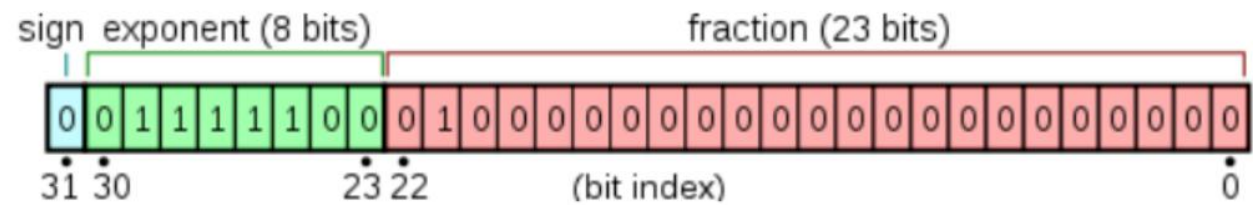
- 文档用自己的语言组织
- 篇幅不够允许加页
- 如果用到某些小测试程序进行说明，可以贴上小测试程序的源码及运行结果
- 为了使文档更清晰，允许将网上的部分图示资料截图后贴入
- 不允许在答案处直接贴某网址，再附上“见**”（或类似行为），否则文档作业部分直接总分-50



(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？

① 如何分段/分段解释？

float型数据的32bit是分成三段来表示一个单精度的浮点数的，分别是：1符号位，8指数位，23尾数位。



32位浮点数内存占用示意图, 共使用了32个小格子

② 尾数的正负如何表示？

通过符号位来表示：符号位为0表示正数，符号位为1表示负数。

③ 尾数如何表示？

首先将十进制浮点数表示为二进制格式，然后将该二进制数转换成以2为底的指数形式，得到尾数、指数（尾数的小数点放在第一位和第二位之间，然后保证第一位数非0）。

只取尾数中小数点后的部分，若不够23位，则在末尾补零，得到尾数在32bit中的表示。

④ 指数如何表示？指数的正负如何表示？

对于32bit浮点数，指数位用于表示[-127, 128]范围内的指数。

对指数进行偏移（+127）后再存入指数位（8位，若不够8位则在高位补0）中。

若指数位存储的数小于127，则指数为负；若大于127，则指数为正。



(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？

① 为什么float型数据只有7位十进制有效数字？

float型数据的尾数位有23位，也就是说最小能表示到 $2^{-23} = 1.19 \times 10^{-7}$ 。

除去最后一位，再加上小数点前的一位，即只有7位十进制有效数字。

② 为什么最大只能是 3.4×10^{38} ？

float型数据的指数为有8位，而 $(2^8 - 1) - 127 = 128$ ，也就是说最大能表示到 $2^{128} = 3.4 \times 10^{38}$ 。

③ 有效位数6位/7位的例子？

```
float x0 = 123456.0;
float x1 = 123456.1;
float x2 = 123456.2;
float x3 = 123456.3;
float x4 = 123456.4;
float x5 = 123456.5;
float x6 = 123456.6;
float x7 = 123456.7;
float x8 = 123456.8;
float x9 = 123456.9;
```

```
printf("%.5f\n", x0);
printf("%.5f\n", x1);
printf("%.5f\n", x2);
printf("%.5f\n", x3);
printf("%.5f\n", x4);
printf("%.5f\n", x5);
printf("%.5f\n", x6);
printf("%.5f\n", x7);
printf("%.5f\n", x8);
printf("%.5f\n", x9);
```

如左图所示：

1. 对于浮点数 123456.0 123456.1 123456.2 123456.5 123456.6 123456.7 而言，有效位数7位；

2. 对于浮点数 123456.3 123456.4 123456.8 123456.9 而言，有效位数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？

① 如何分段？分段解释？

double型数据的64bit是分成三段来表示一个单精度的浮点数的，分别是：1符号位，11指数位，52尾数位。

② 尾数的正负如何表示？

通过符号位来表示：符号位为0表示正数，符号位为1表示负数。

③ 尾数如何表示？

首先将十进制浮点数表示为二进制格式，然后将该二进制数转换成以2为底的指数形式，得到尾数、指数（尾数的小数点放在第一位和第二位之间，然后保证第一位数非0）。

只取尾数中小数点后的部分，若不够52位，则在末尾补零，得到尾数在64bit中的表示。

④ 指数如何表示？指数的正负如何表示？

对于64bit浮点数，指数位用于表示 $[-1023, 1024]$ 范围内的指数。

对指数进行偏移（+1023）后再存入指数位（11位，若不够11位则在高位补0）中。

若指数位存储的数小于1023，则指数为负；若大于1023，则指数为正。



(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？

① 为什么double型数据只有15位十进制有效数字？

double型数据的尾数位有52位，也就是说最小能表示到 $2^{-52} = 2.2 * 10^{-16}$ 。
除去最后一位，再加上小数点前的一位，即只有16位十进制有效数字。

② 为什么最大只能是 1.7×10^{308} ？

double型数据的指数为有11位，而 $(2^{11} - 1) - 1023 = 1024$ ，也就是说最大能表示到 $2^{1024} = 1.7 * 10^{308}$ 。

③ 有效位数位15位/16位的例子？

```
double x0 = 11111111111111.0;
double x1 = 11111111111111.1;
double x2 = 11111111111111.2;
double x3 = 11111111111111.3;
double x4 = 11111111111111.4;
double x5 = 11111111111111.5;
double x6 = 11111111111111.6;
double x7 = 11111111111111.7;
double x8 = 11111111111111.8;
double x9 = 11111111111111.9;

printf("%.5lf\n", x0);
printf("%.5lf\n", x1);
printf("%.5lf\n", x2);
printf("%.5lf\n", x3);
printf("%.5lf\n", x4);
printf("%.5lf\n", x5);
printf("%.5lf\n", x6);
printf("%.5lf\n", x7);
printf("%.5lf\n", x8);
printf("%.5lf\n", x9);
```

```
选择 Microsoft Visual Studio 调试控制台
11111111111111. 00000
11111111111111. 09375
11111111111111. 20312
11111111111111. 29688
11111111111111. 40625
11111111111111. 50000
11111111111111. 59375
11111111111111. 70312
11111111111111. 79688
11111111111111. 90625
```

如左图所示：

1 . 对于浮点数 11111111111111.0 11111111111111.2
11111111111111.4 11111111111111.5
11111111111111.7 11111111111111.9,

有效位数为16位。

2 . 对于浮点数 11111111111111.1 11111111111111.3
11111111111111.6 11111111111111.8,

有效位数为15位。