

**实验报告**

**实 验（二）**

题 目 DataLab 数据表示

专 业 计算机类

学　　 号 1190200523

班　　 级 1903002

学 生 石翔宇

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021.4.2

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 4 -](#_Toc69046952)

[1.1 实验目的 - 4 -](#_Toc69046953)

[1.2 实验环境与工具 - 4 -](#_Toc69046954)

[1.2.1 硬件环境 - 4 -](#_Toc69046955)

[1.2.2 软件环境 - 4 -](#_Toc69046956)

[1.2.3 开发工具 - 4 -](#_Toc69046957)

[1.3 实验预习 - 4 -](#_Toc69046958)

[第2章 实验环境建立 - 6 -](#_Toc69046959)

[2.1 Ubuntu下CodeBlocks安装 - 6 -](#_Toc69046960)

[2.2 64位Ubuntu下32位运行环境建立 - 6 -](#_Toc69046961)

[第3章 C语言的数据类型与存储 - 7 -](#_Toc69046962)

[3.1 类型本质 - 7 -](#_Toc69046963)

[3.2 数据的位置-地址 - 7 -](#_Toc69046964)

[3.3 main的参数分析 - 9 -](#_Toc69046965)

[3.4 指针与字符串的区别 - 9 -](#_Toc69046966)

[第4章 深入分析UTF-8编码 - 11 -](#_Toc69046967)

[4.1 提交utf8len.c子程序 - 11 -](#_Toc69046968)

[4.2 C语言的strcmp函数分析 - 11 -](#_Toc69046969)

[4.3讨论：按照姓氏笔画排序的方法实现（选做，不做要求） - 11 -](#_Toc69046970)

[第5章 数据变换与输入输出 - 12 -](#_Toc69046971)

[5.1 提交**cs\_atoi.c** - 12 -](#_Toc69046972)

[5.2 提交**cs\_atof.c** - 12 -](#_Toc69046973)

[5.3 提交**cs\_itoa.c** - 12 -](#_Toc69046974)

[5.4 提交**cs\_ftoa.c** - 12 -](#_Toc69046975)

[5.5 讨论分析OS的函数对输入输出的数据有类型要求吗 - 12 -](#_Toc69046976)

[第6章 整数表示与运算 - 13 -](#_Toc69046977)

[6.1 提交fib\_dg**.c** - 13 -](#_Toc69046978)

[6.2 提交**fib\_loop.c** - 13 -](#_Toc69046979)

[6.3 fib溢出验证 - 13 -](#_Toc69046980)

[6.4 除以0验证： - 13 -](#_Toc69046981)

[7.1 正数表示范围 - 14 -](#_Toc69046982)

[7.2浮点数的编码计算 - 14 -](#_Toc69046983)

[7.3特殊浮点数值的编码 - 14 -](#_Toc69046984)

[7.4浮点数除0 - 15 -](#_Toc69046985)

[7.5 Float的微观与宏观世界 - 15 -](#_Toc69046986)

[7.6 讨论：任意两个浮点数的大小比较 - 15 -](#_Toc69046987)

[第8章 舍尾平衡的讨论 - 16 -](#_Toc69046988)

[8.1 描述可能出现的问题 - 16 -](#_Toc69046989)

[8.2 给出完美的解决方案 - 16 -](#_Toc69046990)

[第9章 总结 - 17 -](#_Toc69046991)

[9.1 请总结本次实验的收获 - 17 -](#_Toc69046992)

[9.2 请给出对本次实验内容的建议 - 17 -](#_Toc69046993)

[参考文献 - 18 -](#_Toc69046994)

# 第1章 实验基本信息

## 1.1 实验目的

* 熟练掌握计算机系统的数据表示与数据运算
* 通过C程序深入理解计算机运算器的底层实现与优化
* 掌握VS/CB/GCC等工具的使用技巧与注意事项

## 1.2 实验环境与工具

### 1.2.1 硬件环境

* Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
* 16GB RAM
* 1TB HDD + 512G SSD

### 1.2.2 软件环境

* Windows 10 21H1
* Ubuntu 20.04 LTS

### 1.2.3 开发工具

* VSCode，CodeBlocks，gcc+gdb

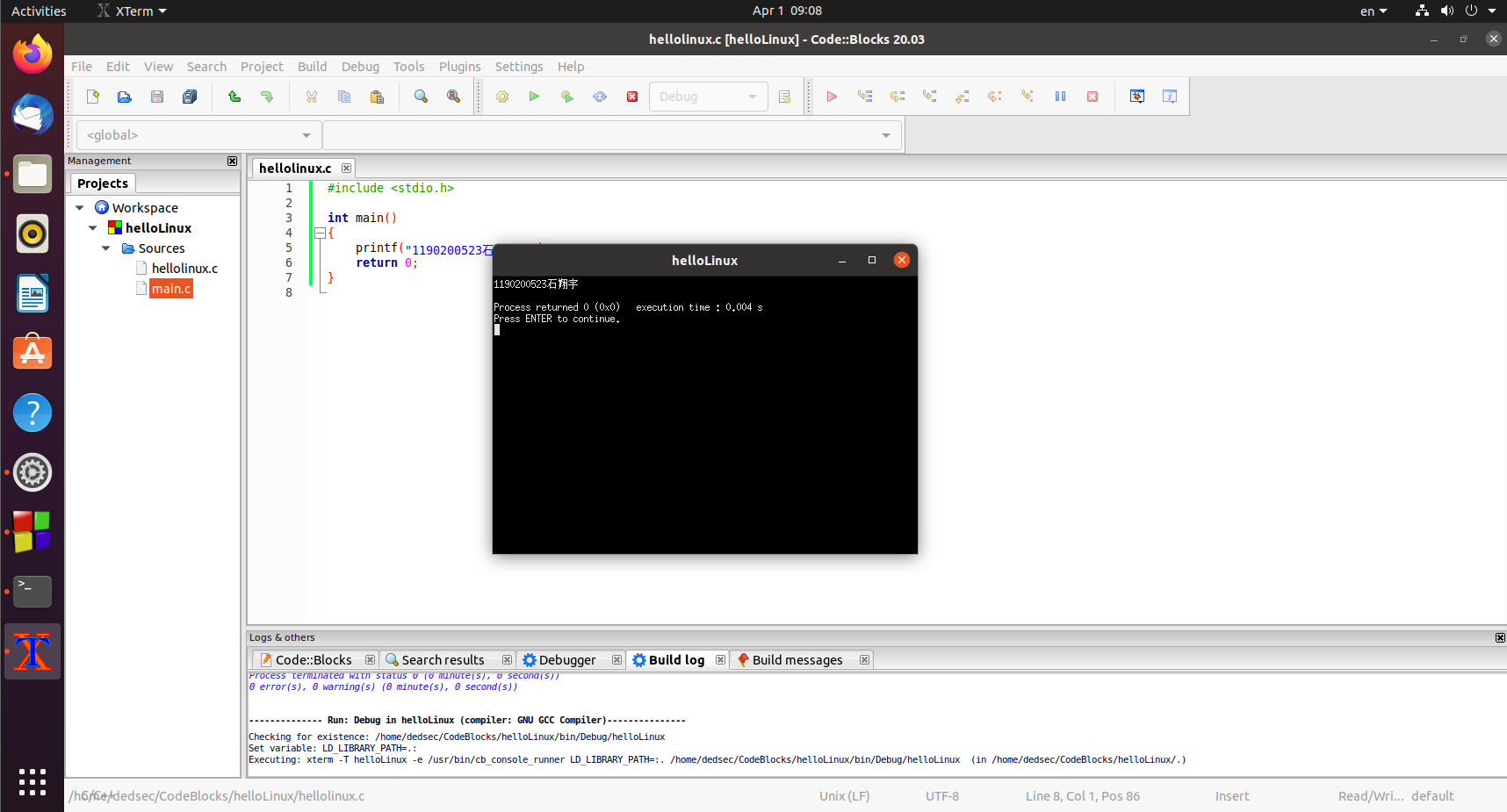
## 1.3 实验预习

* 上实验课前，必须认真预习实验指导书（PPT或PDF）
* 了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。
* 采用sizeof在Windows的VS/CB以及Linux的CB/GCC下获得C语言每一类型在32/64位模式下的空间大小
  + Char /short int/int/long/float/double/long long/long double/指针
* 编写C程序，计算斐波那契数列在int/long/unsigned int/unsigned long类型时，n为多少时会出错
  + 先用递归程序实现，会出现什么问题？
  + 再用循环方式实现。
* 写出float/double类型最小的正数、最大的正数（非无穷）
* 按步骤写出float数-1.1在内存从低到高地址的字节值-16进制
* 按照阶码区域写出float的最大密度区域范围及其密度，最小密度区域及其密度（区域长度/表示的浮点个数）

# 第2章 实验环境建立

## 2.1 Ubuntu下CodeBlocks安装

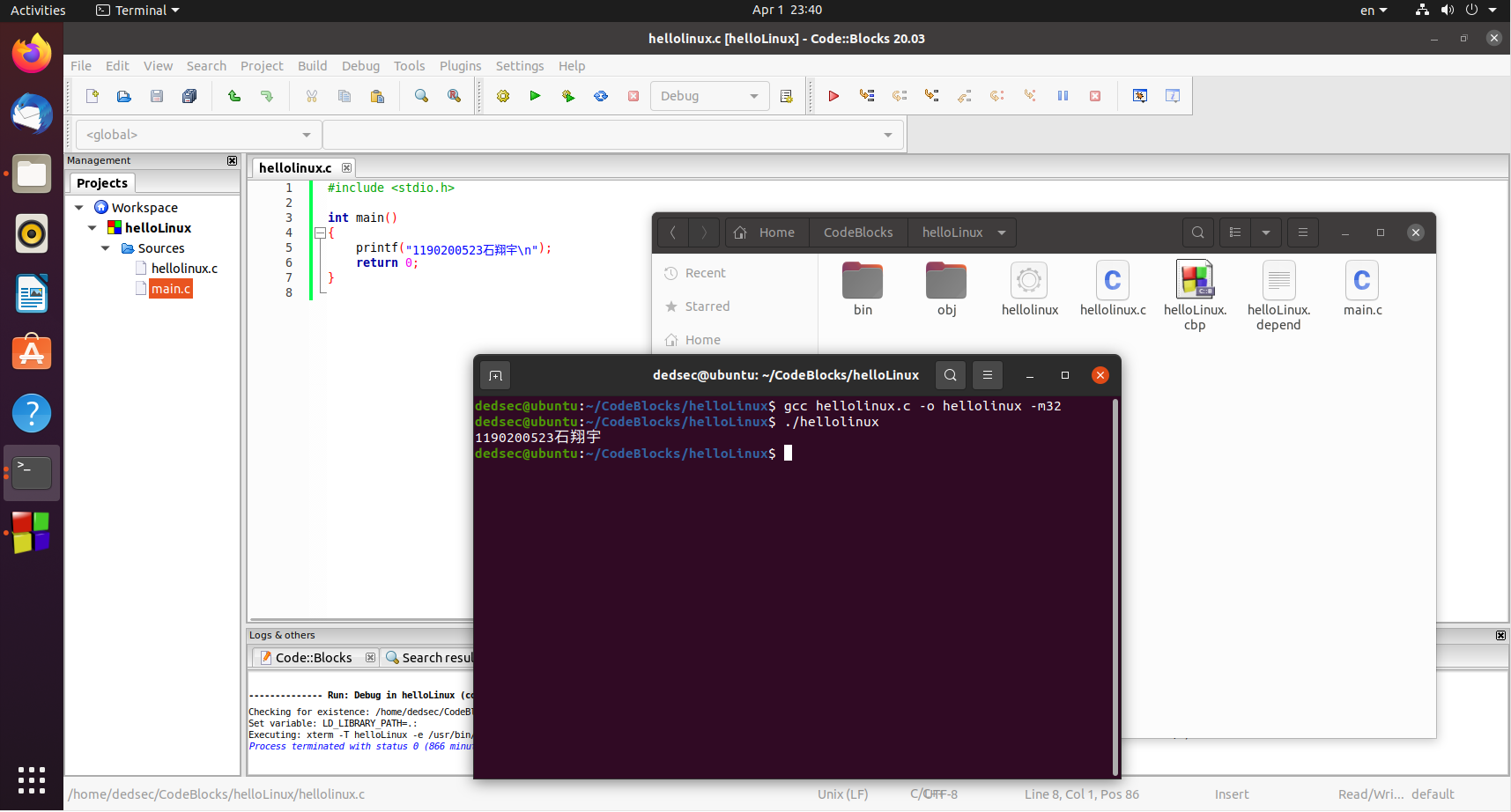
CodeBlocks运行界面截图：编译、运行hellolinux.c



## 2.2 64位Ubuntu下32位运行环境建立

在终端下，用gcc的32位模式编译生成hellolinux.c。执行此文件。

Linux及终端的截图。



# 第3章 C语言的数据类型与存储

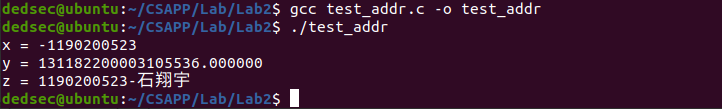
## 3.1 类型本质

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Win/VS/x86** | **Win/VS/x64** | **Win/CB/32** | **Win/CB/64** | **Linux/CB/32** | **Linux/CB/64** |
| char | 1 | 1 | 1 | 1 | 1 | 1 |
| short | 2 | 2 | 2 | 2 | 2 | 2 |
| int | 4 | 4 | 4 | 4 | 4 | 4 |
| long | 4 | 4 | 4 | 4 | 4 | 8 |
| long long | 8 | 8 | 8 | 8 | 8 | 8 |
| float | 4 | 4 | 4 | 4 | 4 | 4 |
| double | 8 | 8 | 8 | 8 | 8 | 8 |
| long double | 8 | 8 | 12 | 16 | 12 | 16 |
| 指针 | 4 | 8 | 4 | 8 | 4 | 8 |

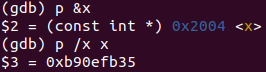
C编译器对sizeof的实现方式：sizeof是一个操作符，由编译器来计算，编译阶段计算出结果，在运行时是个常量。

## 3.2 数据的位置-地址

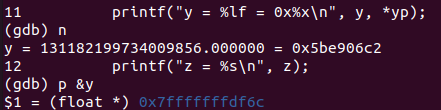
打印x、y、z输出的值：



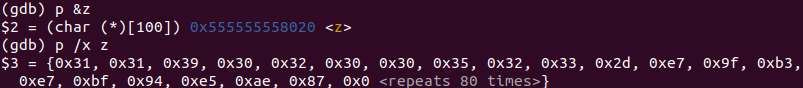
反汇编查看x、y、z的地址，每字节的内容：



x的地址为0x2004，内容为0xb90efb35

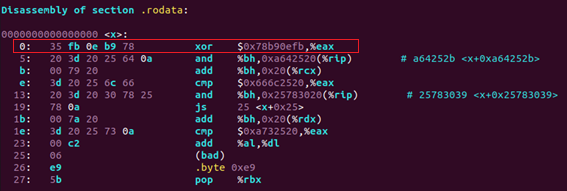


y的地址为0x7fffffffdf6c，内容为0x5be906c2

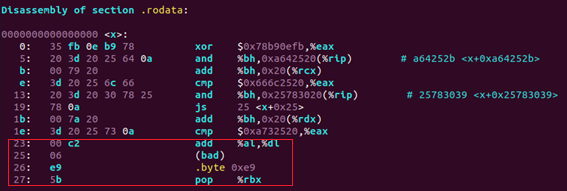


z的地址为0x555555558020，内容为{0x31, 0x31, 0x39, 0x30, 0x32, 0x30, 0x30, 0x35, 0x32, 0x33, 0x2d, 0xe7, 0x9f, 0xb3, 0xe7, 0xbf, 0x94, 0xe5, 0xae, 0x87, 0x0 <repeats 80 times>}

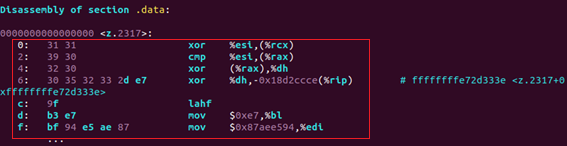
反汇编查看x、y、z在代码段的表示形式。



x在代码段中表示如图红框所示



y在代码段中表示如图红框所示



z在代码段中表示如图红框所示

x与y在 编译 阶段转换成补码与ieee754编码。

数值型常量与变量在存储空间上的区别是：数值型常量一般存储在常量存储区（.rodata段），而变量则存储在.data段

字符串常量与变量在存储空间上的区别是：字符串变量的名字及其所需的存储空间是显式定义的，并通过名字来引用相应的字符串变量，存储在.data段。而字符串常量所需的存储空间是隐式定义的，并且没有名字，存储在.rodata段。

常量表达式在计算机中处理方法是：尽量用左右移代替乘除法，编译阶段就计算出了常量表达式的值，储存在内存中

**3.2提示：**

**①在linux下生成可执行程序，假设是a.out。然后用objdump -dx a.out > a-dump.s 生成反汇编文件a-dump.s，查看a-dump.s**

**②gdb ./a.out ↙ layout asm ↙ b main ↙ r ↙ disp argc ↙….**

## 3.3 main的参数分析

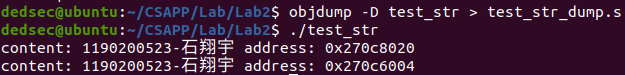
反汇编查看x、y、z的地址截图4；

命令行传递参数，反汇编观察argc、argv的地址与内容，截图4。



## 3.4 指针与字符串的区别

cstr的地址与内容截图，pstr的内容与截图，截图5



pstr修改内容会出现什么问题：访问无效内存，出现段错误



# 第4章 深入分析UTF-8编码

## 4.1 提交utf8len.c子程序

## 4.2 C语言的strcmp函数分析

分析论述：strcmp到底按照什么顺序对汉字排序

答：strcmp按照汉字的编码（例UTF-8）的二进制从小到大对汉字排序。

## 4.3讨论：按照姓氏笔画排序的方法实现（选做，不做要求）

分析论述：应该怎么实现呢？

答：提前预处理出一个按照姓氏笔画排序的汉字列表，按照该列表得出先后顺序，即可根据姓氏笔画排序。

# 第5章 数据变换与输入输出

## 5.1 提交**cs\_atoi.c**

## 5.2 提交**cs\_atof.c**

## 5.3 提交**cs\_itoa.c**

## 5.4 提交**cs\_ftoa.c**

## 5.5 讨论分析OS的函数对输入输出的数据有类型要求吗

论述如下：有。

OS的函数将输入输出的数据都看成字符串来处理。对于输入，OS先接收输入的字符串，再进行进一步处理；对于输出，将数据转换成字符串再输出。

例如：

sprintf函数：输出int类型，输入要求为字符串地址。

atoi函数：输出int类型，输入字符串首地址。

atof函数：输出double类型，输入字符串首地址。

itoa函数：返回转换后字符串首地址，输入待转换int数以及转换进制。

ftoa函数：返回转换后字符串首地址，输入float和基数。

# 第6章 整数表示与运算

## 6.1 提交fib\_dg**.c**

## 6.2 提交**fib\_loop.c**

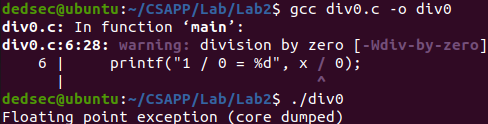
## 6.3 fib溢出验证

int 时从n= 47时溢出，long时n= 93 时溢出。

unsigned int 时从n= 48 时溢出，unsigned long时n= 94 时溢出。

## 6.4 除以0验证：

除以0：截图1



除以极小浮点数，截图：

**第7章 浮点数据的表示与运算**

## 7.1 正数表示范围

写出float/double类型最小的正数、最大的正数（非无穷）

Float:

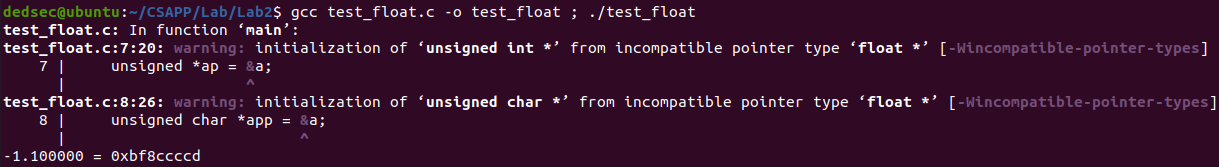
Double:

## 7.2浮点数的编码计算

（1）按步骤写出float数-1.1的浮点编码计算过程，写出该编码在内存中从低地址字节到高地址字节的16进制数值

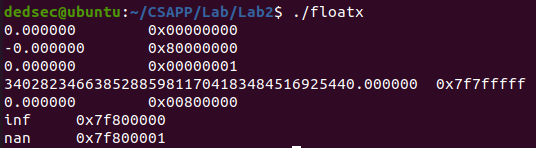
1. 十进制转二进制：
2. 变成：
3. 即：
4. 变成二进制：1 01111111 0001100110011001100110011
5. 向偶数舍入：1 01111111 00011001100110011001101
6. 分位：1011 1111 1000 1100 1100 1100 1100 1101
7. 转成16进制：bf8ccccd

（2）验证：编写程序，输出值为-1.1的浮点变量其各内存单元的数值，截图。



## 7.3特殊浮点数值的编码

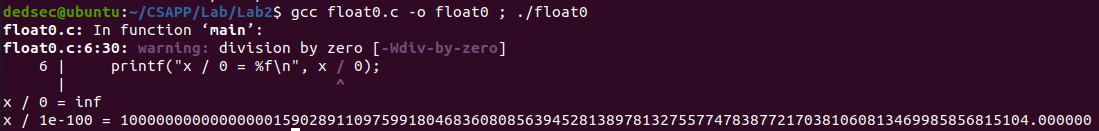
（1）构造多float变量，分别存储+0-0，最小浮点正数，最大浮点正数、最小正的规格化浮点数、正无穷大、Nan,并打印最可能的精确结果输出（十进制/16进制）。截图。



（2）提交子程序floatx.c

## 7.4浮点数除0

（1）编写C程序，验证C语言中float除以0/极小浮点数后果，截图



（2）提交子程序float0.c

## 7.5 Float的微观与宏观世界

按照阶码的数值区域，float编码最密集区域的阶码编码是：00000000 和 00000001, 该区域中相邻浮点数编码的数值的间距是：；float编码最稀疏区域的阶码编码是：11111110, 该区域中相邻浮点数编码的数值的间距是：。

最小正数变成十进制科学记数法，最可能能精确到多少1.401298464324817e-45

最大正数变成十进制科学记数法，最可能能精确到多少3.4028234663852886e+38

## 7.6 讨论：任意两个浮点数的大小比较

论述比较方法以及原因。

方法：将两个浮点数相减得到差（），将取绝对值得到。取一极小数与相比较，若小于，则与相等。若大于，则判断是否大于（c的符号位是0还是1）。若大于，则；若小于，则。

原因：浮点数的存储有精度限制，有些数无法用浮点数精确表示，因此无法直接对浮点数进行大小比较。

# 第8章 舍尾平衡的讨论

## 8.1 描述可能出现的问题

在对数据进行统计的过程中，常常由于精度等问题对数据进行舍入。若舍入方法不合理，在大量的数据积累后，可能会产生很大的误差。

## 8.2 给出完美的解决方案

1. 单向舍位平衡：如果在数据统计时，每个数据只用于一次合计，那么在处理舍位平衡时，只需要根据合计值的误差，调整使用的各项数据就可以了。
2. 双向舍位平衡：如果数据在行向和列向两个方向同时需要计算合计值，同时还需要计算所有数据的总计值，这种情况下处理舍位平衡时就复杂得多了。此时处理舍位平衡时，不仅要求最终的总计值准确，同时行向和列向计算的合计值也要与对应行、列的数据平衡，这种情况下的舍位平衡称为双向舍位平衡。

# 第9章 总结

## 9.1 请总结本次实验的收获

* 对数据的存储有了更深的理解。
* 学习了UTF-8编码的编码方式。
* 深入理解了浮点数的表示方法，也对浮点数的处理有了更清楚的认识。

## 9.2 请给出对本次实验内容的建议

* 希望实验有关的材料排版能够更加严谨易读，去除冗杂部分，强调重点部分，精简实验内容。
* 部分材料模糊不清，有歧义，希望可以修改。

注：本章为酌情加分项。

# 参考文献

[1] 大卫 R.奥哈拉伦，兰德尔 E。布莱恩特. 深入理解计算机系统[M]. 机械工业

出版社.2017.7