

**实验报告**

**实 验（二）**

题 目 DataLab 数据表示

专 业 计算机

学　　 号 1180300308

班　　 级 11803003

学 生 刘义

指 导 教 师 史先俊

实 验 地 点 管理楼G712

实 验 日 期 9.25

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 4 -](#_Toc20308905)

[1.1 实验目的 - 4 -](#_Toc20308906)

[1.2 实验环境与工具 - 4 -](#_Toc20308907)

[1.2.1 硬件环境 - 4 -](#_Toc20308908)

[1.2.2 软件环境 - 4 -](#_Toc20308909)

[1.2.3 开发工具 - 4 -](#_Toc20308910)

[1.3 实验预习 - 4 -](#_Toc20308911)

[第2章 实验环境建立 - 5 -](#_Toc20308912)

[2.1 Ubuntu下CodeBlocks安装（5分） - 5 -](#_Toc20308913)

[2.2 64位Ubuntu下32位运行环境建立（5分） - 5 -](#_Toc20308914)

[第3章 C语言的数据类型与存储 - 6 -](#_Toc20308915)

[3.1 类型本质（1分） - 6 -](#_Toc20308916)

[3.2 数据的位置-地址（2分） - 6 -](#_Toc20308917)

[3.3 main的参数分析（2分） - 6 -](#_Toc20308918)

[3.4 指针与字符串的区别（2分） - 6 -](#_Toc20308919)

[第4章 深入分析UTF-8编码 - 7 -](#_Toc20308920)

[4.1 提交utf8len.c子程序(1分) - 7 -](#_Toc20308921)

[4.2 C语言的strcmp函数分析（2分） - 7 -](#_Toc20308922)

[4.3讨论：按照姓氏笔画排序的方法实现（2分） - 7 -](#_Toc20308923)

[第5章 数据变换与输入输出 - 8 -](#_Toc20308924)

[5.1 提交**cs\_atoi.c** - 8 -](#_Toc20308925)

[5.2 提交**cs\_atof.c** - 8 -](#_Toc20308926)

[5.3 提交**cs\_itoa.c** - 8 -](#_Toc20308927)

[5.4 提交**cs\_ftoa.c** - 8 -](#_Toc20308928)

[5.5 讨论分析OS的函数对输入输出的数据有类型要求吗 - 8 -](#_Toc20308929)

[第6章 整数表示与运算 - 9 -](#_Toc20308930)

[6.1 提交fib\_dg**.c** - 9 -](#_Toc20308931)

[6.2 提交**fib\_loop.c** - 9 -](#_Toc20308932)

[6.3 fib溢出验证 - 9 -](#_Toc20308933)

[6.4 除以0验证： - 9 -](#_Toc20308934)

[6.5 万年虫验证 - 9 -](#_Toc20308935)

[6.6 2038虫验证 - 9 -](#_Toc20308936)

[第7章 浮点数据的表示与运算 - 10 -](#_Toc20308937)

[7.1 写出float/double类型最小的正数、最大的正数（非无穷） - 10 -](#_Toc20308938)

[7.2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。 - 10 -](#_Toc20308939)

[编写程序在内存验证，截图。 - 10 -](#_Toc20308940)

[7.3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。 - 10 -](#_Toc20308941)

[7.4编写C程序，验证C语言中float除以0/极小浮点数后果，截图 - 10 -](#_Toc20308942)

[7.5 Float的微观与宏观世界 - 10 -](#_Toc20308943)

[第8章 舍尾平衡的讨论 - 11 -](#_Toc20308944)

[10.1 描述可能出现的问题 - 11 -](#_Toc20308945)

[10.2 给出完美的解决方案 - 11 -](#_Toc20308946)

[第9章 总结 - 12 -](#_Toc20308947)

[9.1 请总结本次实验的收获 - 12 -](#_Toc20308948)

[9.2 请给出对本次实验内容的建议 - 12 -](#_Toc20308949)

[参考文献 - 13 -](#_Toc20308950)

# 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

X64 CPU；

2GHz；

2G RAM；

256GHD Disk 以上

### 1.2.2 软件环境

Windows7 64位以上；

VirtualBox/Vmware 11以上；

Ubuntu 16.04 LTS 64位/优麒麟 64位

### 1.2.3 开发工具

Visual Studio 2010 64位以上；

CodeBlocks；

vi/vim/gpedit+gcc

## 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

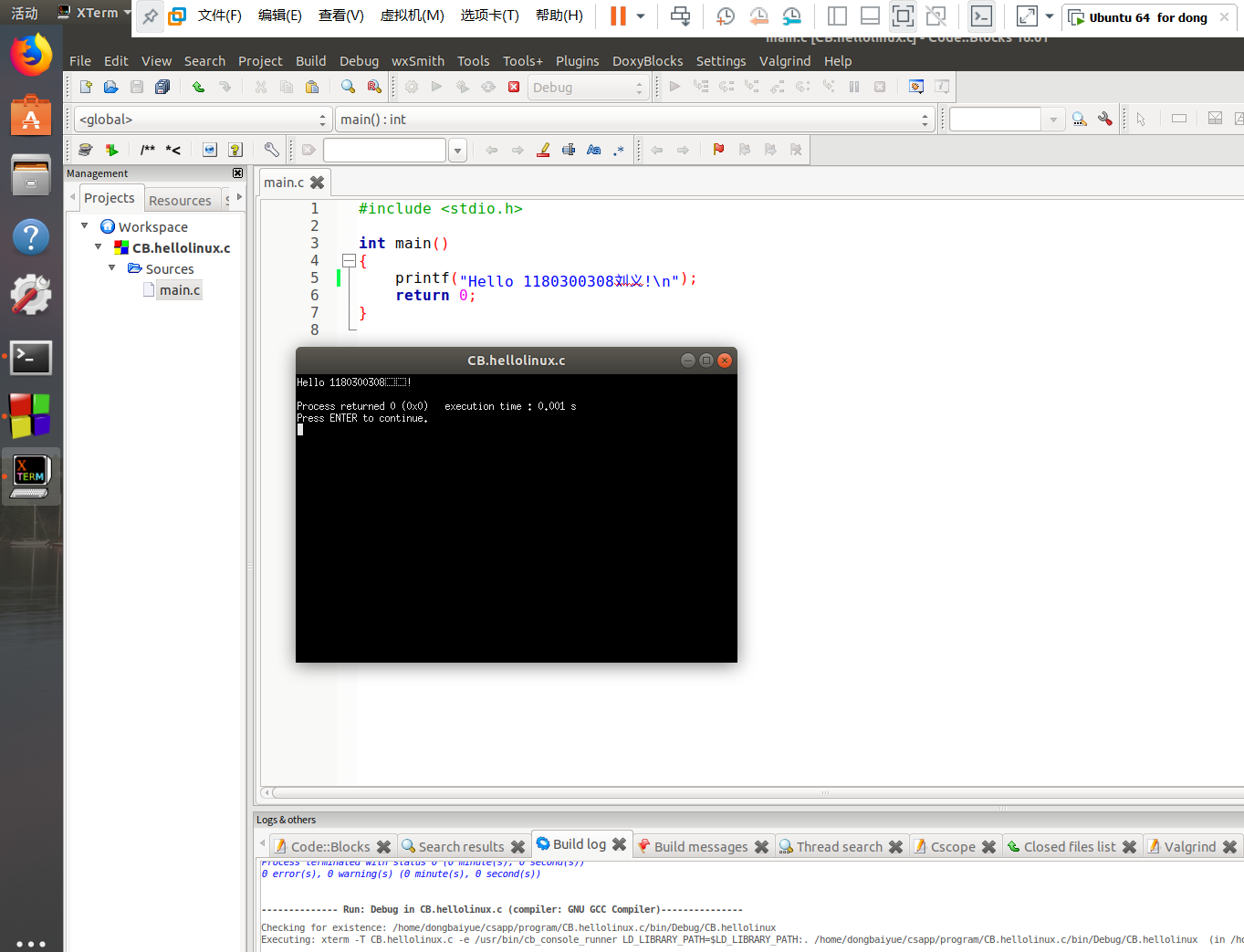


图2-1 Ubuntu下CodeBlocks截图

## 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。

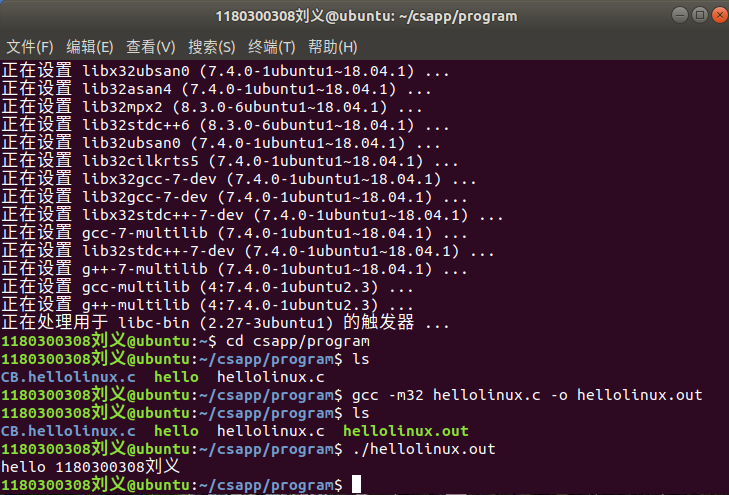


图2-2 Ubuntu与Windows共享目录截图

# 第3章 C语言的数据类型与存储

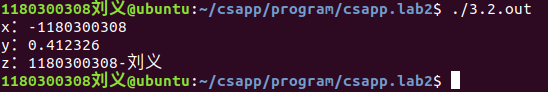
## 3.1 类型本质（1分）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 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：\_\_编译器将sizeof()替换为相应常量,\_具体编译器实现使用case函数\_\_\_\_

## 3.2 数据的位置-地址（2分）

打印x、y、z输出的值：截图1



反汇编查看x、y、z的地址，每字节的内容：截图2，标注说明

x

地址：0x56557008

各字节内容：11101100 00001011 10100110 10111001

y

地址：0xffffcfb8

各字节内容：01110101 00111000 00010101 01101110 10001101 01100011 11011010 00111111

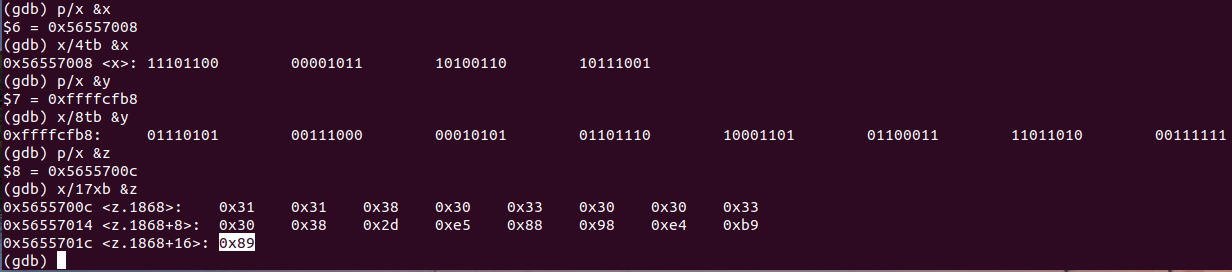
z

地址：0x5655700c

各字节内容：0x31 0x31 0x38 0x30 0x33 0x30 0x30 0x33

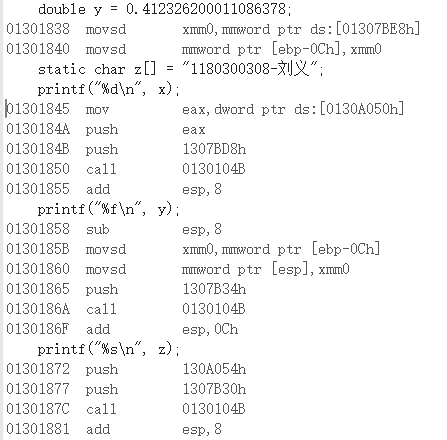
0x30 0x38 0x2d 0xe5 0x88 0x98 0xe4 0xb9

0x89



截图2

反汇编查看x、y、z在代码段的表示形式。截图3，标注说明



x与y在\_\_\_\_汇编\_\_\_\_阶段转换成补码与ieee754编码。

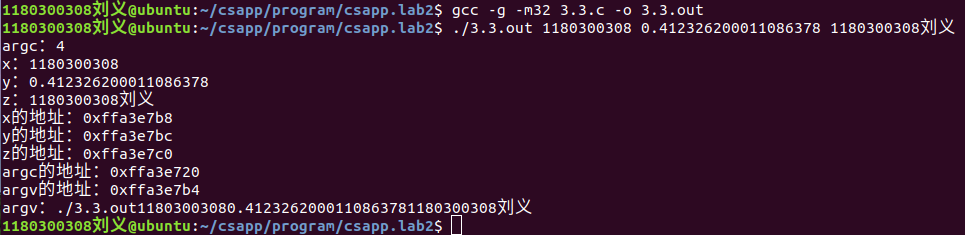
数值型常量与变量在存储空间上的区别是：\_\_变量在栈内存储，数值型常量在栈外常量存储区存储\_\_

字符串常量与变量在存储空间上的区别是：\_\_变量在栈内存储，字符型常量在栈外常量存储区存储\_\_

常量表达式在计算机中处理方法是：\_编译时计算、替换为相应的结果（值）\_

## 3.3 main的参数分析（2分）

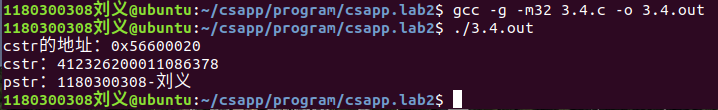
反汇编查看x、y、z的地址，argc的地址，argv的地址与内容，截图4



## 3.4 指针与字符串的区别（2分）

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\_\_\_指针pstr指向常量区的字符串，只有读取权限，没有写入权限。若修改，运行出错。\_\_\_\_



# 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

分析：strcmp按照汉字的编码（GB2312或Unicode）进行排序，GB2312里一级汉字是按拼音排序的，Unicode里的汉字按笔划顺序排序

## 4.3讨论：按照姓氏笔画排序的方法实现

分析论述：应该怎么实现呢？

分析：将姓名编码为Unicode格式，使用strcmp函数比较排序

# 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： OS所有的输入输出都被当作对相应文件的读和写来执行，文件按字节读写，对数据无类型要求

# 第6章 整数表示与运算

## 6.1 提交fib\_dg**.c**

## 6.2 提交**fib\_loop.c**

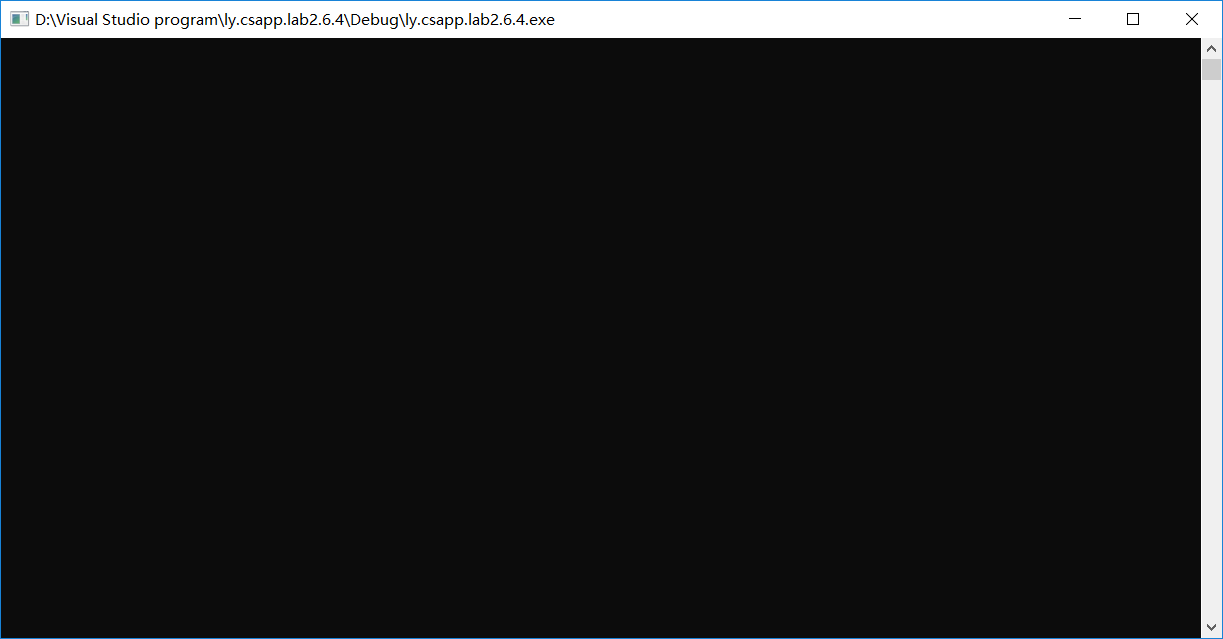
## 6.3 fib溢出验证

int 时从n=\_\_\_\_\_47\_\_\_\_时溢出，long时n=\_\_\_\_\_\_47\_\_\_\_\_\_时溢出。

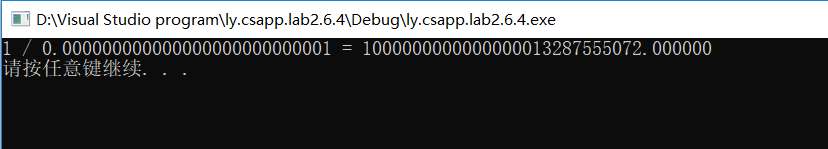
unsigned int 时从n=\_\_\_\_48\_\_\_\_\_时溢出，unsigned long时n=\_\_\_\_48\_\_\_\_时溢出。

## 6.4 除以0验证：

除以0：截图1

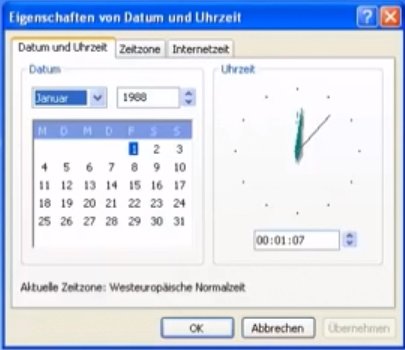


除以极小浮点数，截图：



## 6.5 万年虫验证

你的机器到9999年12月31日23:59:59后，时钟怎么显示的？Windows/Linux下分别截图：





## 6.6 2038虫验证

2038年1月19日中午11:14:07后你的计算机时间是多少，Windows/Linux下分别截图





# 第7章 浮点数据的表示与运算

## 7.1手动float编码：

按步骤写出float数-10.1在内存从低到高地址的字节值（16进制）。

-10.1(10) = -1010.0001100110011001100…(2)

= -1. 01000011001100110011001100… \*

符号位S：1

阶码exp：10000010

frac：01000011001100110011001100…

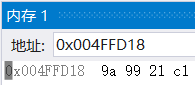
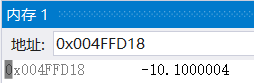
保留23位，向偶数舍入，即01000011001100110011010

2进制编码为：1100 0001 0010 0001 1001 1001 1001 1010

16进制编码为：c1 21 99 9a

小段存储，内存各字节为：9a 99 21 c1

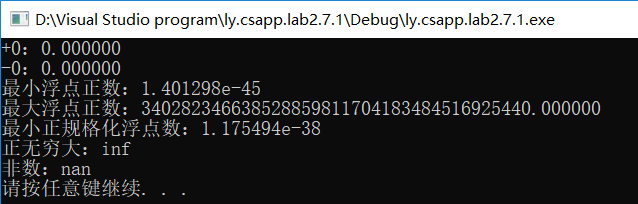
编写程序在内存验证手动编码的正确性，截图。

## 7.2特殊float数据的处理

提交子程序floatx.c，要求：

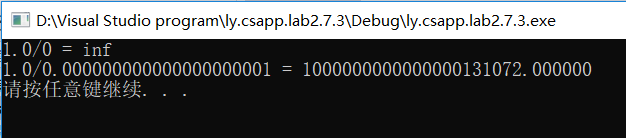
构造多float变量，分别存储+0-0，最小浮点正数，最大浮点正数、最小正的规格化浮点数、正无穷大、Nan,并打印最可能的精确结果输出（十进制/16进制）。截图。



## 7.3验证浮点运算的溢出

提交子程序float0.c

编写C程序，验证C语言中float除以0/极小浮点数后果，截图



## 7.4 类型转换的坑

实验指导PPT第5步骤的x变量，执行 x=(int)(float)x 后结果为多少？

原x=\_\_\_-1180300308\_\_\_\_，现x=\_\_\_\_-1180300288\_\_\_\_

## 7.5 讨论1：有多少个int可以用float精确表示

有\_\_\_\_2^23\*18\_\_个int数据可以用float精确表示。

是哪些数据呢？\_\_\_

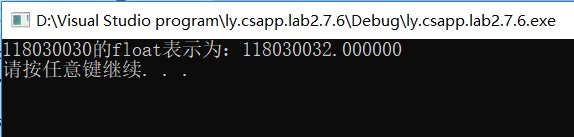
1. 2进制表示不超过24位的；

2.（若2进制表示超过24位）2进制表示除高24位外其余各位均为0\_\_\_\_

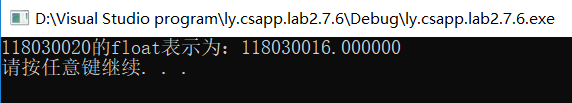
## 7.6 讨论2：怎么验证float采用的向偶数舍入呢

基于上个讨论，开发程序或举几个特例用C验证即可！

截图与标注说明！



118030030的二进制表示为：111000010001111111011001110(2)，第24位为1，第25位为1；float精度保留高24位，其余各位均为0，向偶数舍入，即111000010001111111011010000(2) = 118030032。



118030020的二进制表示为：111000010001111111011000100 (2)，第24位为0，第25位为1；float精度保留高24位，其余各位均为0，向偶数舍入，即111000010001111111011001000 (2) = 118030016。

## 7.7 讨论3：float能精确表示几个1元内的钱呢

人民币0.01-0.99元之间的十进制数，有多少个可用float精确表示？

是哪些呢？

（1.28 \* ）-（1.98 \* ）之间

Float = \*M\*

当E=-1时，M<1.98 = 1.111110101110000101000111101…(2)

Frac<=11111010111000010100011(2)，有8220836个float；

当E=-2、-3、-4、-5、-6时，frac取任意值，float共5\*2^23=41943040个；

当E=-7时，M>1.28 =1.010001111010111000010100011…(2)

Frac>=01000111101011100001011(2)，有7651189个float；

总计共有 57815065个十进制数可被float精确表示

**2** 若题意所说的0.01-0.99元之间的十进制数，是限制在小数点后2位的数字，则是0.01-0.99之间99个数中二进制表示的有效位数在24位以内的。

## 7.8 Float的微观与宏观世界

按照阶码区域写出float的最大密度区域的范围及其密度，最小密度区域及其密度（表示的浮点个数/区域长度）：\_\_\_\_\_\_、\_\_\_\_、\_\_\_\_、\_\_\_\_

微观世界：能够区别最小的变化\_\_\*== \_\_，其10进制科学记数法为\_\_1.40129\*\_\_

宏观世界：不能区别最大的变化\_\_(\*=\_\_\_\_\_\_，其10进制科学记数法为\_\_2.028241\*\_\_

## 7.9 讨论：浮点数的比较方法

从键盘输入或运算后得到的任意两个浮点数，论述其比较方法以及理由。

对两个浮点数作比较：

1. 先排除非数nan，返回错误，无法比较

2. 比较符号位。若符号位不同，符号位为0者较大；否则比较阶码大小

3. 比较阶码大小。若阶码不同，阶码大者较大；否则比较frac

4. 比较frac大小。若frac不同，frac大者较大；否则相等。

# 第8章 舍尾平衡的讨论

## 8.1 描述可能出现的问题

在报表统计中，经常因为精度变化的原因，对数据进行舍尾处理（即舍去末尾若干位）。简单的4舍5入操作可能会打破原本数据的平衡关系，而出现许多问题，如：

1. 平均值\*n 与 累加值不相等。

例：保留一位小数的原始的数据是：4.5 4.5。平均值是4.5，n = 2，累加值是9.0；

若是数据精度改为整数，则平均值为5，n = 2，累加值是9，。

2. 所有明细数据累加值，与各分项(如地市、行业等)累加值不同

同上例：若是数据精度改为整数，。

## 8.2 给出完美的解决方案

1. 提高数据精度，float=〉double=〉long double，提高上报数据精度，原来整数，现要求小数后2位，原来2位，现要求5位等。

2. 除了上报统计数据，还要上报明细数据。

3. 对4舍5入的一点修正：当保留n位有效数字，若第n+1位数字=5且后面数字为0时，则第n位数字若为偶数时就舍掉后面的数字，若第n位数字为奇数时加1（即向偶数舍入）。

4. 舍位平衡：在保持合计（累加）值正确的条件下，调整非合计数据舍位后的结果，使得数据关系重新平衡。

对一维的情况，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，调整各项非合计数据即可，这属于比较简单的情况。比较合理的方式是根据各项非合计数据绝对值的大小按比例分担平衡差（也就是合计值的误差），当然还需要考虑到“最小调整值”，所谓最小调整值，就是舍位后最小精度的单位值，例如在取整时，最小精度就是个位，最小调整值就是1或者-1。

对8.1中的例子而言，，平衡差为-1，最小调整值为1或者-1，又因为两个数据相等（5=5），将其中任一个调整为4，另一个不变，5+4=9，舍位平衡。

# 第9章 总结

## 9.1 请总结本次实验的收获

学习了vim、gdb的使用

初步了解了汇编语言

对多种数据类型编码以及计算机程序执行有了更加深入的理解

## 9.2 请给出对本次实验内容的建议

实验指导PPT有些乱，建议对每条指令进行较详细的说明

注：本章为酌情加分项。

# 参考文献

**为完成本次实验你翻阅的书籍与网站等**

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/ collection/anatmorp.