

哈尔滨工业大学

实验报告

实验（二）

题 目 DataLab 数据表示

专 业 计算机

学 号 1190300321

班 级 1936602

学 生 郑晟赫

指 导 教 师 刘宏伟

实 验 地 点 G712

实 验 日 期 2021.4.1

计算机科学与技术学院

目 录

第 1 章 实验基本信息	- 4 -
1.1 实验目的	- 4 -
1.2 实验环境与工具	- 4 -
1.2.1 硬件环境	- 4 -
1.2.2 软件环境	- 4 -
1.2.3 开发工具	- 4 -
1.3 实验预习	- 4 -
第 2 章 实验环境建立	- 5 -
2.1 UBUNTU 下 CODEBLOCKS 安装 (5 分)	- 5 -
2.2 64 位 UBUNTU 下 32 位运行环境建立 (5 分)	- 5 -
第 3 章 C 语言的数据类型与存储	- 7 -
3.1 类型本质 (1 分)	- 7 -
3.2 数据的位置-地址 (2 分)	- 7 -
3.3 MAIN 的参数分析 (2 分)	- 8 -
3.4 指针与字符串的区别 (2 分)	- 9 -
第 4 章 深入分析 UTF-8 编码	- 10 -
4.1 提交 UTF8LEN.C 子程序(1 分)	- 10 -
4.2 C 语言的 STRCMP 函数分析 (2 分)	- 10 -
4.3 讨论: 按照姓氏笔画排序的方法实现 (2 分)	- 10 -
第 5 章 数据变换与输入输出	- 11 -
5.1 提交 CS_ATOI.C	- 11 -
5.2 提交 CS_ATOF.C	- 11 -
5.3 提交 CS_ITOA.C	- 11 -
5.4 提交 CS_FTOA.C	- 11 -
5.5 讨论分析 OS 的函数对输入输出的数据有类型要求吗	- 11 -
第 6 章 整数表示与运算	- 12 -
6.1 提交 FIB_DG.C	- 12 -
6.2 提交 FIB_LOOP.C	- 12 -
6.3 FIB 溢出验证	- 12 -
6.4 除以 0 验证:	- 12 -
6.5 万年虫验证	- 12 -
6.6 2038 虫验证	- 12 -
第 7 章 浮点数据的表示与运算	- 14 -

7.1 手动 FLOAT 编码:	- 14 -
7.2 特殊 FLOAT 数据的处理.....	- 14 -
7.3 验证浮点运算的溢出	- 15 -
7.4 类型转换的坑.....	- 15 -
7.5 讨论 1: 有多少个 INT 可以用 FLOAT 精确表示	- 15 -
7.6 讨论 2: 怎么验证 FLOAT 采用的向偶数舍入呢	- 15 -
7.7 讨论 3: FLOAT 能精确表示几个 1 元内的钱呢	- 16 -
7.8 FLOAT 的微观与宏观世界	- 16 -
7.9 讨论: 浮点数的比较方法.....	- 16 -
第 8 章 舍尾平衡的讨论	- 17 -
8.1 描述可能出现的问题.....	- 17 -
8.2 给出完美的解决方案.....	- 17 -
第 9 章 总结	- 18 -
9.1 请总结本次实验的收获.....	- 18 -
9.2 请给出对本次实验内容的建议.....	- 18 -
参考文献	- 19 -

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; 512GHD Disk

1.2.2 软件环境

Windows10 64 位; Vmware 15 以上; Ubuntu 20.04 64 位

1.2.3 开发工具

Visual Studio 2019 64 位; 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。

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。

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-10.1 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 (5 分)

CodeBlocks 运行界面截图：编译、运行 hellolinux.c

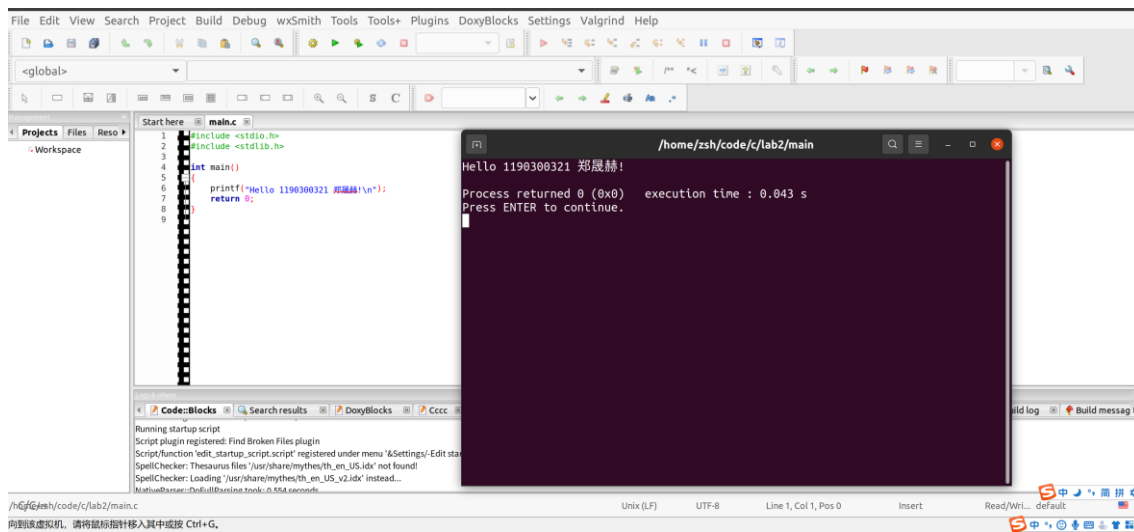


图 2-1 Ubuntu 下 CodeBlocks 截图

2.2 64 位 Ubuntu 下 32 位运行环境建立 (5 分)

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。



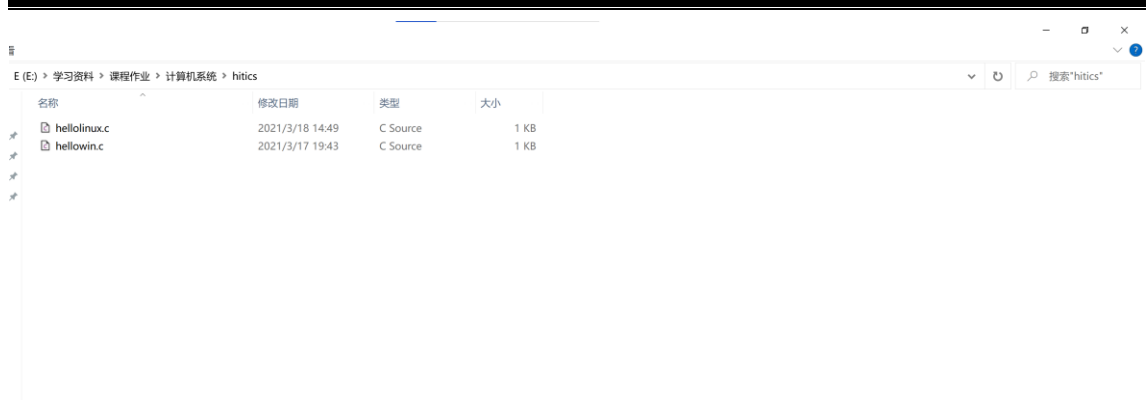


图 2-2 Ubuntu 与 Windows 共享目录截图

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/32	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：在编译阶段计算出结果，在运行时是一个常量

3.2 数据的位置-地址 (2 分)

打印 x、y、z 输出的值：

```
zsh@zsh-virtual-machine:~/code/c/lab2$ gcc lab2_address.c -o lab2_address
zsh@zsh-virtual-machine:~/code/c/lab2$ ./lab2_address
-1190300321
350182187020058624.000000
1190300321郑晟赫
```

反汇编查看 x、y、z 的地址，每字节的内容：

```
(gdb) p&x
$1 = (int *) 0x555555558014 <x>
```

X 的地址是 0x555555558014

```
(gdb) p&y
$2 = (float *) 0x555555558018 <y>
```

Y 的地址是 0x555555558018

```
(gdb) p&z
$3 = (char (*)[20]) 0x7fffffffde70
```

Z 的地址是 0x7fffffffde70

反汇编查看 x、y、z 在代码段的表示形式。

```

93: float y=350182200109133824;
94: 118b: f3 0f 10 05 7d 0e 00 movss 0xe7d(%rip),%xmm0 # 2010 <_IO_stdin_used+0x10>
95: 1192: 00
96: 1193: f3 0f 11 45 dc movss %xmm0,-0x24(%rbp)
97: char z[]="1190300321-郑晟赫";
98: 1198: 48 b8 31 31 39 30 33 movabs $0x3330303330393131,%rax
99: 119f: 30 30 33
100: 11a2: 48 ba 32 31 2d e9 83 movabs $0x99e69183e92d3132,%rdx
101: 11a9: 91 e6 99
102: 11ac: 48 89 45 e0 mov %rax,-0x20(%rbp)
103: 11b0: 48 89 55 e8 mov %rdx,-0x18(%rbp)
104: 11b4: c7 45 f0 9f e8 b5 ab movl $0xabb5e89f,-0x10(%rbp)
105: 11bb: c6 45 f4 00 movb $0x0,-0xc(%rbp)
106: printf("%d",x);
107: 11bf: 8b 05 4b 2e 00 00 mov 0x2e4b(%rip),%eax # 4010 <x>
108: 11c5: 89 c6 mov %eax,%esi
109: 11c7: 48 8d 3d 36 0e 00 00 lea 0xe36(%rip),%rdi # 2004 <_IO_stdin_used+0x4>
110: 11ce: b8 00 00 00 00 mov $0x0,%eax
111: 11d3: e8 98 fe ff ff callq 1070 <printf@plt>
112: printf("%lf",y);
113: 11d8: 66 0f ef c9 pxor %xmm1,%xmm1
114: 11dc: f3 0f 5a 4d dc cvtss2sd -0x24(%rbp),%xmm1
115: 11e1: 66 48 0f 7e c8 movq %xmm1,%rax
116: 11e6: 66 48 0f 6e c0 movq %rax,%xmm0
117: 11eb: 48 8d 3d 15 0e 00 00 lea 0xe15(%rip),%rdi # 2007 <_IO_stdin_used+0x7>
118: 11f2: b8 01 00 00 00 mov $0x1,%eax
119: 11f7: e8 74 fe ff ff callq 1070 <printf@plt>

```

x 与 y 在__编译__阶段转换成补码与 iee754 编码。

数值型常量与变量在存储空间上的区别是：变量占据存储空间，常量不占据存储空间

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（一维字符数组名）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

常量表达式在计算机中处理方法是：常量表达式在编译时就已经被计算好了，然后直接储存到内存中

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

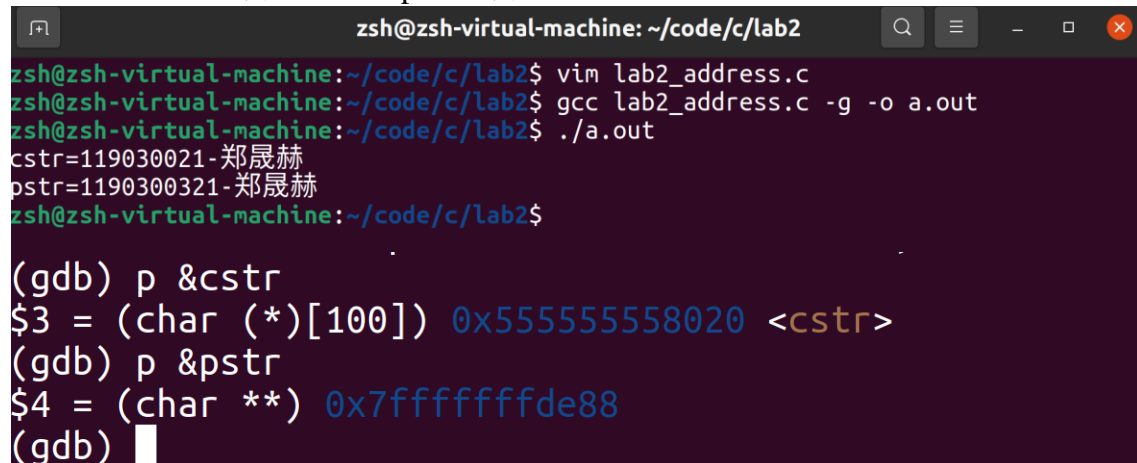
```

(gdb) p *(argv+1)
$1 = 0x7fffffffde2ef "/home/zsh/code/c/lab2/a.out"
(gdb) p*(argv+2)
$2 = 0x7fffffffde30b "x"
(gdb) p *(argv+3)
$3 = 0x7fffffffde30d "y"
(gdb) p *(argv+4)
$4 = 0x7fffffffde30f "z"
(gdb) p &argc
$5 = (int *) 0x7fffffffde6c

```


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



```
zsh@zsh-virtual-machine: ~/code/c/lab2
zsh@zsh-virtual-machine:~/code/c/lab2$ vim lab2_address.c
zsh@zsh-virtual-machine:~/code/c/lab2$ gcc lab2_address.c -g -o a.out
zsh@zsh-virtual-machine:~/code/c/lab2$ ./a.out
cstr=119030021-郑晟赫
pstr=1190300321-郑晟赫
zsh@zsh-virtual-machine:~/code/c/lab2$

(gdb) p &cstr
$3 = (char (*)[100]) 0x555555558020 <cstr>
(gdb) p &pstr
$4 = (char **) 0x7fffffffde88
(gdb)
```

pstr 修改内容会出现什么问题：修改指针的时候会报错，因为修改后的内容超过了原来指针所指向内容的空间，从而发生了内存分配的异常，从而会报错。

第 4 章 深入分析 UTF-8 编码

4.1 提交 utf8len.c 子程序 (1 分)

4.2 C 语言的 strcmp 函数分析 (2 分)

分析论述：strcmp 到底按照什么顺序对汉字排序

C 语言中，在 Window 系统下汉字使用的编码一般为是两个字节的 GB2312 编码（Linux 下一般为 utf-8，Unicode 编码汉字一般为三个字节），strcmp 通过比较通过比较编码的大小来决定某一个字符的大小，因此在对汉字进行排序的时候，针对不同的编码方式，比较的时候都是根据编码从高位到低位进行比较，当比较到某一位比较出大小后就将这一大小认为是两个汉字的大小比较出大小。

4.3 讨论：按照姓氏笔画排序的方法实现 (2 分)

分析论述：应该怎么实现呢？

构建一个数组或一个数据库，存储每个汉字关于笔画顺序的编码，在内存充足的情况下可以考虑使用哈希存储的方式加快使用时的查找速度，每次排序时，在存储的信息中查找对应的汉字的笔画，得到所对应的笔画数后对笔画数进行排序。

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

`os` 模块是一个用于访问操作系统的模块，包含普遍的操作系统功能，如复制、创建、修改、删除文件及文件夹。`os` 模块提供了一个可移植的方法来使用操作系统的功能，使得程序能够跨平台使用，即它允许一个程序在编写后不需要任何改动，就可以在 `Linux` 和 `Windows` 等操作系统下都能运行，便于编写跨平台的应用。`OS` 的函数将输入输出的数据都看成字符串来处理。通过输入输出流来实现，例如在 `Linux` 下 `stdin` 为标准输入流，标准的输入设备默认键盘；`stdout` 为标准输出流，标准的输出设备默认屏幕；`stderr` 为标准错误流，只有程序出错时才会执行的流程。`OS` 先接收输入的字符串，再进行进一步处理；对于输出，也就将要输出的数据转换成字符串输出的。例如：

`sprintf()`: C 库函数 `int sprintf(char *str, const char *format, ...)` 发送格式化输出到 `str` 所指向的字符串。

`atoi()`: C 库函数 `int atoi(const char *str)` 把参数 `str` 所指向的字符串转换为一个整数（类型为 `int` 型）。

`itoa()`: C 库函数 `char *itoa (int value, char *str, int base)` 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()`: C 库函数 `char *ftoa(float f, int *status)` 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:

```
int b = 0;
printf("%d\n", 132/b);
```

Microsoft Visual Studio 调试控制台

E:\Code\cpp\lab2\x64\Debug\lab2.exe (进程 3592) 已退出, 代码为 -1073741676。
按任意键关闭此窗口. . .

除以极小浮点数, 截图:

```
printf("%d\n", 1/1e-100);
```

e:\Code\cpp\ics-lab2\div.exe

630506365
请按任意键继续. . .

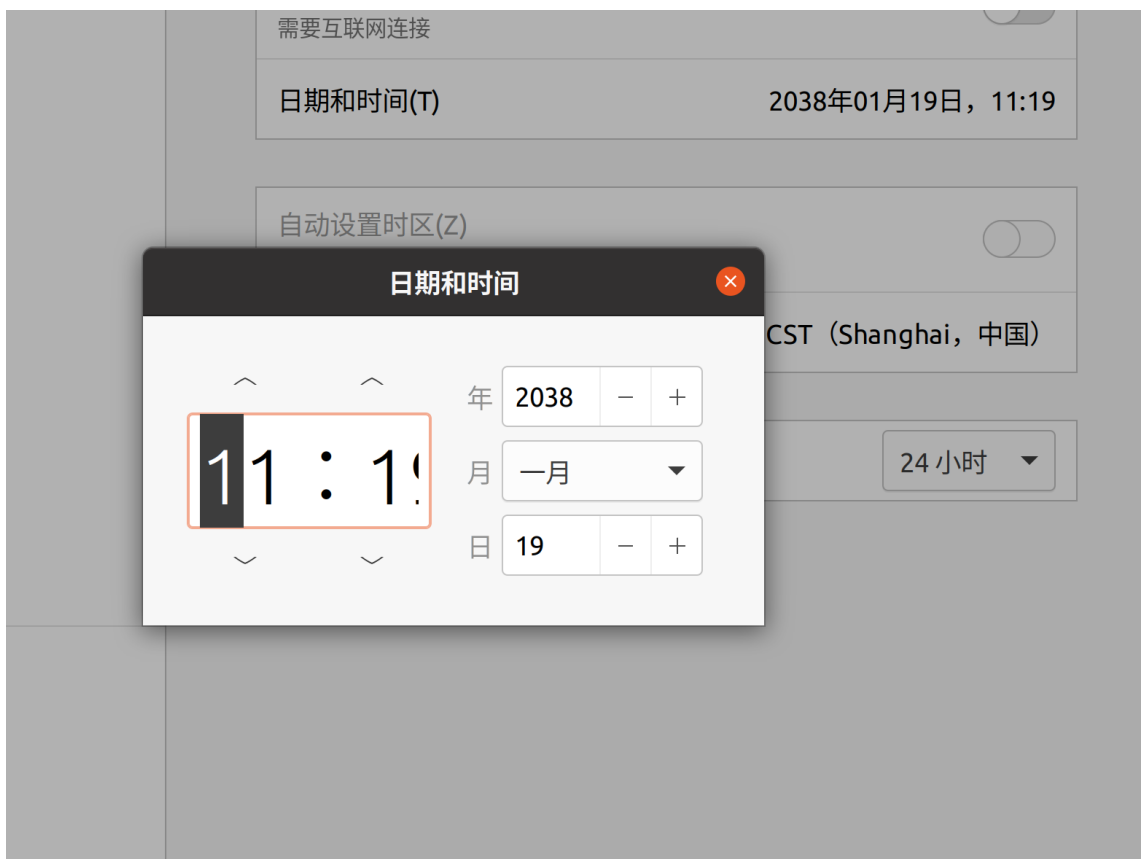
6.5 万年虫验证

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）。

10-2 转换：-1010.000110011001100.....

科学计数：-1.0100 001 1001 1001 1001 1001*2³


尾数规格化与舍入：0100001 1001 1001 1001 1001

阶码表示：1000 0100

754 格式：c1 21 99 9a (1100 0001 0010 0001 1001 1001 1001 1010)

小端存储格式：9a 99 21 c1(1001 1010 1001 1001 0010 0001 1100 0001)

编写程序在内存验证手动编码的正确性，截图。

 e:\Code\cpp\ics-lab2\test.exe

c121999a

请按任意键继续. . .

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。

```
e:\Code\cpp\ics-lab2\floatx.exe
10进制输出为:
+0:0.000000E+000
-0:-0.000000E+000
最小浮点正数:1.175494E-038
最大浮点正数:3.402823E+038
最小正规格化数:1.175494E-038
正无穷大:1.#INF00E
Nan:1.#QNAN0.64

16进制输出为:
+0:00000000
-0:00000000
最小浮点正数:00008000
最大浮点正数:ffff7fff
最小正规格化数:00008000
正无穷大:0000807f
Nan:0000c07f
请按任意键继续. . .
```

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

e:\Code\cpp\ics-lab2\float0.exe

```
浮点数除以0: 1.#INF00
浮点数除以极小的浮点数: 8507059173023461600000000000000000000000.000000
请按任意键继续. . .
```

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=_____1190300321_____，现 x=_____1190300288_____

7.5 讨论 1：有多少个 int 可以用 float 精确表示

有_____150994944_____个 int 数据可以用 float 精确表示。

是哪些数据呢？绝对值小于等于 24 次方或者大于 2 的 24 次方但除最高的 24 位外的位都是 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！

```
19 int main()
20 {
21     float a=150994920;//1000 1111 1111 1111 1111 1110 1000
22     float b=150994936;//1000 1111 1111 1111 1111 1111 1000
23     printf("a:");
24     e:\Code\cpp\ics-lab2\test2.exe
```

```
a: feff0f4d
b: 0000104d
请按任意键继续. . .
```

IEEE 754浮点数十六进制相互转换(32位,四字节,单精度)

10进制 150994936

16进制 4D 0F FF FF

将 float 型变量 a 赋值为 150994920, b 赋值为 150994936 可以看出 a 二进制表示为 1000 1111 1111 1111 1111 1110 1000, b 的二进制表示为 1000 1111 1111 1111 1111 1111 1000, 由于浮点数尾数只有 23 位, 因此需要进行舍入, 由于 a 第 24 位为

0,25 位为 0, 由于发生向偶数舍入, 则 24 位保持为 0, 舍入后 a 为 4D 0F FF FE, 而 b 24 位则变为 0, b 为 4D 10 00 00, 因此 b 比原来大 16, 16 进制表示为 1。

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?

有 3 个: 0.25, 0.50, 0.75。

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 (区域长度/表示的浮点个数): $-(1-2^{-23}) \times 2^{-126} \sim (1-2^{-23}) \times 2^{-126}$ 、 2^{-149} 、 $2^{127} \sim (2-2^{-23}) \times 2^{127}$ 、 2^{104}

微观世界: 能够区别最小的变化 2^{-149} , 其 10 进制科学记数法为 1.40×10^{-45}

宏观世界: 不能区别最大的变化 2^{104} , 其 10 进制科学记数法为 2.028241×10^{31}

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。

比较方法: 1. 符号不同的浮点数可以直接比较大小; 2. 若符号相同, 由于浮点数存储的方式, 不可以直接使用 == 比较, 比如对于两个浮点数 a, b, 如果要比较大小, 那么常常会设置一个精度, 如果 $\text{fabs}(a-b) \leq 1e-6$, 那么就是相等了。类似的, 判断大于的时候, 在 c 语言中使用语句就是 $\text{if}(a > b \ \&\& \ \text{fabs}(a-b) > 1e-6)$ 。判断小于的时候, 使用语句 $\text{if}(a < b \ \&\& \ \text{fabs}(a-b) > 1e-6)$ 。

对于以及转化为 IEEE 型的浮点数: 首先判断是否为 NAN, 若为两个数都为 NAN, vb 中默认相等, cb 中不可比较, 再判断两个数符号位, 如果一个为正数, 一个负数, 则正数大于负数, 若为 0, 则正 0 和 0 相等。若符号位相等, 余下部分直接比较即可。

第 8 章 舍尾平衡的讨论

8.1 描述可能出现的问题

在数据统计中，常常会根据精度呈现或者单位换算等要求，需要对数据执行四舍五入的操作，这种操作称为舍位处理。简单直接的舍位处理有可能会带来隐患，原本平衡的数据关系可能会被打破。

例如， $13,451.00 \text{ 元} + 45,150.00 \text{ 元} + 2,250.00 \text{ 元} - 5,649.00 \text{ 元} = 55,202.00 \text{ 元}$ 现在单位变成单位万元，仍然保留两位小数，根据 4 舍 5 入的原则： $1.35 \text{ 万元} + 4.52 \text{ 万元} + 0.23 \text{ 万元} - 0.56 \text{ 万元} = 5.54 \text{ 万元}$ ，出现 0.02 万的误差，与真实数据不相等。

8.2 给出完美的解决方案

首先：提高数据精度，float 换 double \Rightarrow long double，要求提高上报数据精度，原来整数，现要求小数后 2 位，原来 2 位，现要求 5 位等。同时，除了上报统计数据，还要上报明细数据。这也是信息系统建设要求从区县级集中到市级集中，再到省级集中，再到全国大集中。其次，可以在计算机中建一张表，储存的信息为经过统计后最适合的舍入情况，在使用的时候，输入的是需要舍入的位后的几位数字，经过查表之后可以得到一个舍入情况，可以根据这个舍入情况决定是进位或者是将尾部舍去。

第9章 总结

9.1 请总结本次实验的收获

- 1.熟悉了浮点数 float 型变量的编码表示，计算处理，精确度，宏观微观的精确范围和浮点数的舍入规则。
- 2.对于数据表示以及数据的溢出有了更加深刻的理解。
- 3.学会了使用反汇编查看代码中变量的内容与地址

9.2 请给出对本次实验内容的建议

希望 ppt 上能多一点实验进行中的指导。

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] 湛颖. 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>.