哈爾濱工業大學

实验报告

实 验（二）

题　　目 DataLab 数据表示

专　　业 计算机类

学　　号 1180300829

班　　级 1803008

学　　生 余涛

指 导 教 师 吴锐

实 验 地 点 G709

实 验 日 期 2019.9.28

计算机科学与技术学院

目 录

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。

  答：1.递归方式：f(n) = f(n-1) + f(n-2)，n>=3;

f(1) =f(2)=1;

其时间复杂度：O（2^n）

2.循环方式：

```
for(int i = 3; i < n; i++) {
    temp = res;
    res = pre + res;
    pre = temp;
    }
```

其时间复杂度： O(n)

int 为 47

long int 为 47

unsigned int 为 48

unsigned long 为 48

■ **写出 float/double 类型最小的正数、最大的正数（非无穷）**

答：float 最小值：1.4E-45

最大值：3.4028235E38

double 最小值：4.9E-324

最大值：1.7976931348623157E308

■ **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**

答：cd cc 8c bf

■ **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）**

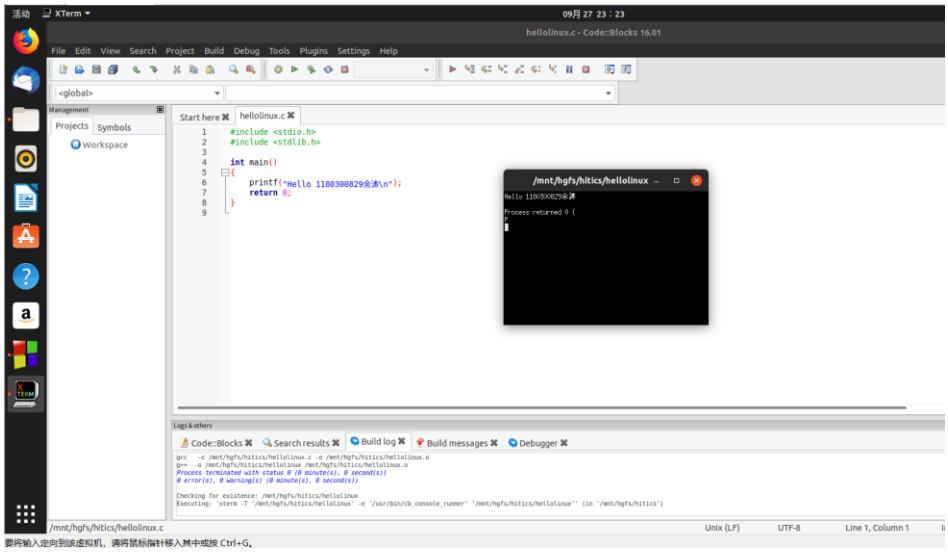
答：最大：-1.1111(23 个 1)×2^{-126}~1.1111(23 个 1)×2^{-126}(包含正负浮点数)；2^{25} 个浮点数 ；

最小：1.0000(23 个 0)×2^{127}~1.11111(23 个 1)×2^{127}；2^{23} 个浮点数

第 2 章 实验环境建立

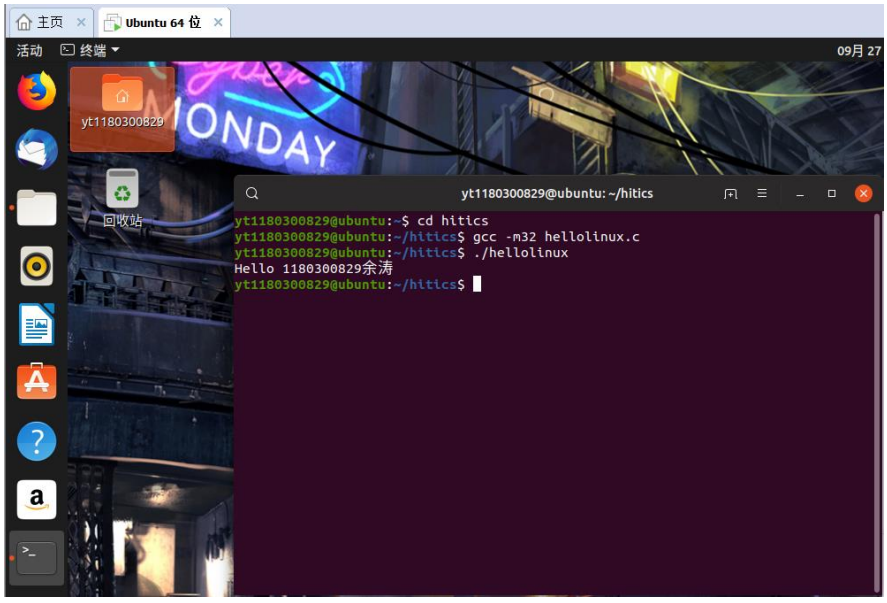
2.1 Ubuntu 下 CodeBlocks 安装

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

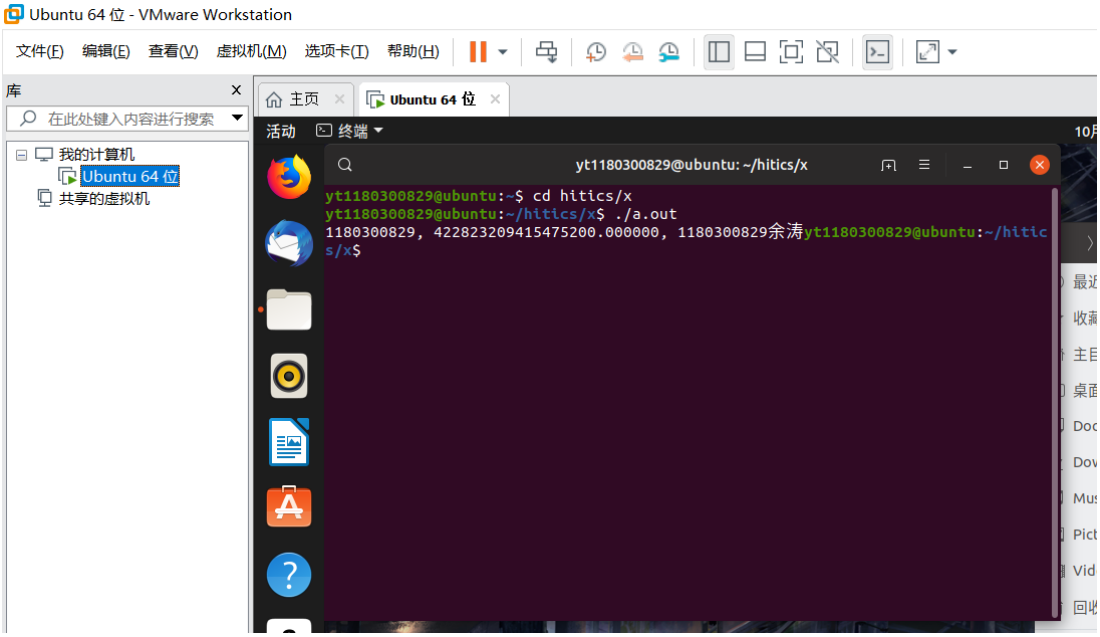
3.1 类型本质（1 分）

	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	8	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：由编译器来计算，编译阶段就计算出结果了，在运行时就是个常量。编译阶段可以确定数据类型，根据数据类型换算数据的长度

3.2 数据的位置-地址（2 分）

打印 x、y、z 输出的值：截图 1

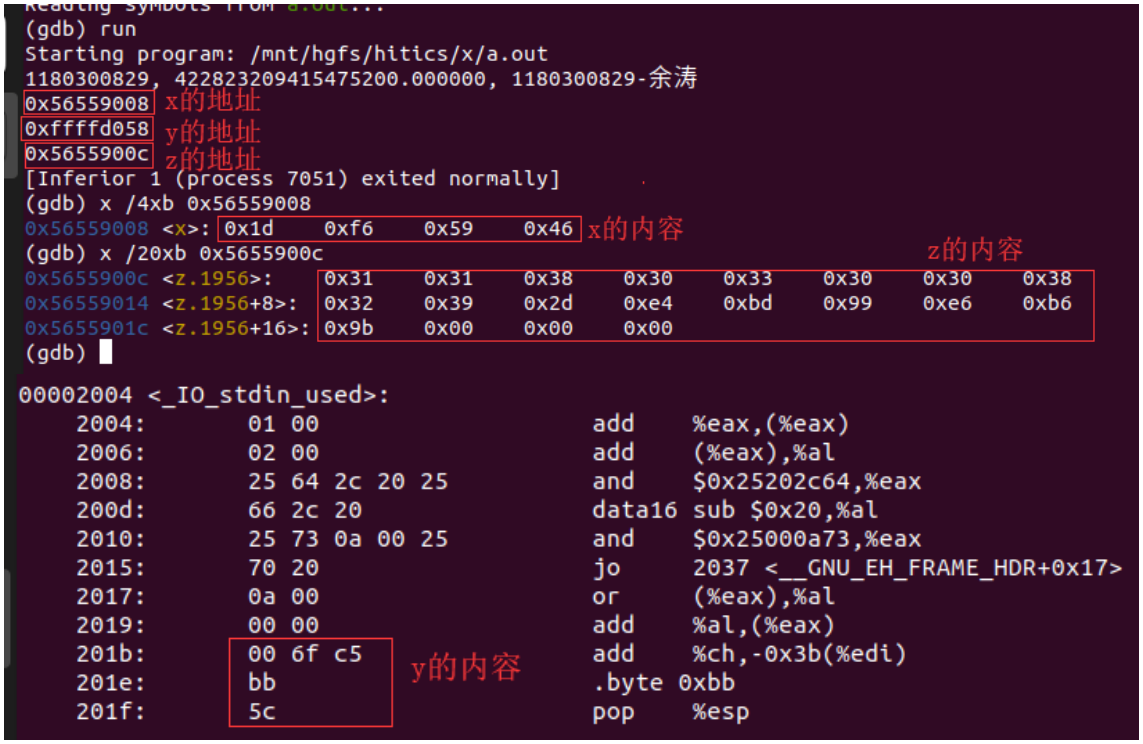


反汇编查看 x、y、z 的地址，每字节的内容：截图 2，标注说明

x 的地址：0x56559008；内容 1d f6 59 46

y 的地址：0xffffd058；内容 6f c5 bb 5c

z 的地址：0x5655900c；内容：31 31 38 30 30 38 32 39 2d e4 bd 99 e6 b6 9b



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 的代码段： sbb $0x314659f6,%eax

```
Disassembly of section .data:

00004000 <__data_start>:
    4000:       00 00                   add    %al,(%eax)
        ...

00004004 <__dso_handle>:
    4004:       04 40                   add    $0x40,%al
        ...

00004008 <x>:                                   x的代码段
    4008:       1d f6 59 46             sbb    $0x314659f6,%eax

0000400c <z.1956>:
    400c:       31 31                   xor    %esi,(%ecx)
    400e:       38 30                   cmp    %dh,(%eax)
    4010:       33 30                   xor    (%eax),%esi
    4012:       30 38                   xor    %bh,(%eax)
    4014:       32 39                   xor    (%ecx),%bh
    4016:       e4 bd                   in     $0xbd,%al
    4018:       99                      cltd
    4019:       e6 b6                   out    %al,$0xb6
    401b:       9b                      fwait
        ...
```

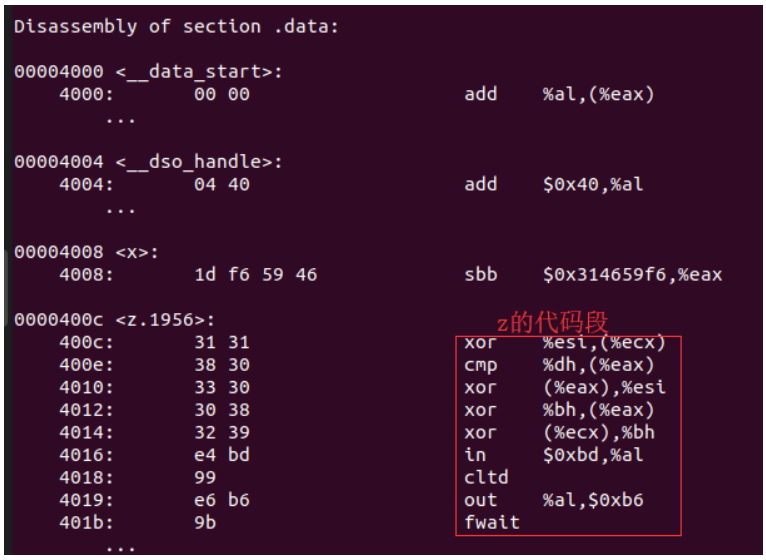
y 的代码段：

```
add     %ch,-0x3b(%edi)
.byte   0xbb
pop     %esp
```

```
00002004 <_IO_stdin_used>:
    2004:       01 00                   add    %eax,(%eax)
    2006:       02 00                   add    (%eax),%al
    2008:       25 64 2c 20 25          and    $0x25202c64,%eax
    200d:       66 2c 20                data16 sub $0x20,%al
    2010:       25 73 0a 00 25          and    $0x25000a73,%eax
    2015:       70 20                   jo     2037 <__GNU_EH_FRAME_HDR+0x17>
    2017:       0a 00                   or     (%eax),%al
    2019:       00 00                   add    %al,(%eax)      y的代码段
    201b:       00 6f c5    y的内容     add    %ch,-0x3b(%edi)
    201e:       bb                      .byte 0xbb
    201f:       5c                      pop    %esp
```

z 的代码段：

```
xor     %esi,(%ecx)
cmp      %dh,(%eax)
xor     (%eax),%esi
xor     %bh,(%eax)
xor     (%ecx),%bh
in      $0xbd,%al
cltd
out     %al,$0xb6
fwait
```



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：____数值型常量储存在常量区，而变量储存在动态区，程序运行期间其大小处于动态变化中。处于该区的变量也会时而被创建时而被销毁_____

字符串常量与变量在存储空间上的区别是：___字符串常量储存在常量区，只可读，而变量储存在静态全局初始化区，和全局变量都储存在数据段，只在定义它的文件中可用，而文件之外是不可以被看见的_____

常量表达式在计算机中处理方法是：_____在程序编译时就将值储存在内存中，不可改变，无法直接修改其内容_____

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

x 的地址：0x56559008；内容 1d f6 59 46

y 的地址：0xffffd044；内容 6f c5 bb 5c

z 的地址：0x5655900c；内容：31 31 38 30 30 38 32 39 2d e4 bd 99 e6 b6 9b

argc 的地址：0xffffd080;内容：01 00 00 00

argv 的地址：0xffffd03c;内容：90 4d 62 01

```
(gdb) run
Starting program: /mnt/hgfs/hitics/y/a.out
1180300829, 422823209415475200.000000, 1180300829-余涛
/mnt/hgfs/hitics/y/a.out
0x56559008 x的地址
0xffffd044 y的地址
0x5655900c z的地址
0xffffd080 argc的地址
0xffffd03c argv的地址
[Inferior 1 (process 7141) exited normally]
(gdb) x /4xb 0x56559008                    x的内容
0x56559008 <x>: 0x1d    0xf6    0x59    0x46
(gdb) x /20xb 0x5655900c                                    z的内容
0x5655900c <z.2475>:    0x31    0x31    0x38    0x30    0x33    0x30    0x30    0x38
0x56559014 <z.2475+8>:  0x32    0x39    0x2d    0xe4    0xbd    0x99    0xe6    0xb6
0x5655901c <z.2475+16>: 0x9b    0x00    0x00    0x00
```

内存 1

地址: 0x0133F938 argc的地址

```
0x0133F938  01 00 00 00 88 4d 62 01 78 93 62 01 01 00 00 00 88 4d 62 01 78 93 62 01 ac f9 33 01
0x0133F96E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 74 a5
0x0133F9A4  a8 20 6b a6 00 00 00 00 b4 f9 33 01 3d 1d bd 00 bc f9 33 01 d8 20 bd 00 cc f9 33 01
0x0133F9DA  eb 5d 00 00 00 00 00 00 00 00 00 00 f0 1b 01 00 00 00 00 00 00 00 00 00 00 00 00 00
0x0133FA10  d8 f9 33 01 00 00 00 00 30 fa 33 01 a0 9e 84 77 9b 44 55 2b 00 00 00 00 38 fa 33 01
```

内存 1 内存 2

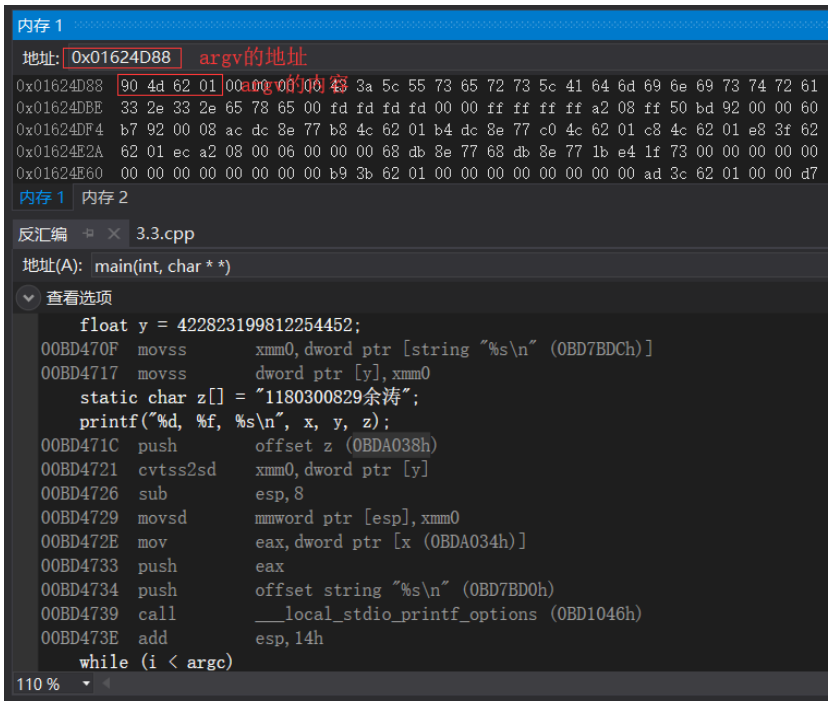
反汇编 3.3.cpp

地址(A): main(int, char * *)

查看选项

```
    float y = 422823199812254452;
00BD470F  movss       xmm0,dword ptr [string "%s\n" (0BD7BDCh)]
00BD4717  movss       dword ptr [y],xmm0
    static char z[] = "1180300829余涛";
    printf("%d, %f, %s\n", x, y, z);
00BD471C  push        offset z (0BDA038h)
00BD4721  cvtss2sd    xmm0,dword ptr [y]
00BD4726  sub         esp,8
00BD4729  movsd       mmword ptr [esp],xmm0
00BD472E  mov         eax,dword ptr [x (0BDA034h)]
00BD4733  push        eax
00BD4734  push        offset string "%s\n" (0BD7BD0h)
00BD4739  call        ___local_stdio_printf_options (0BD1046h)
00BD473E  add         esp,14h
    while (i < argc)
```

110 %

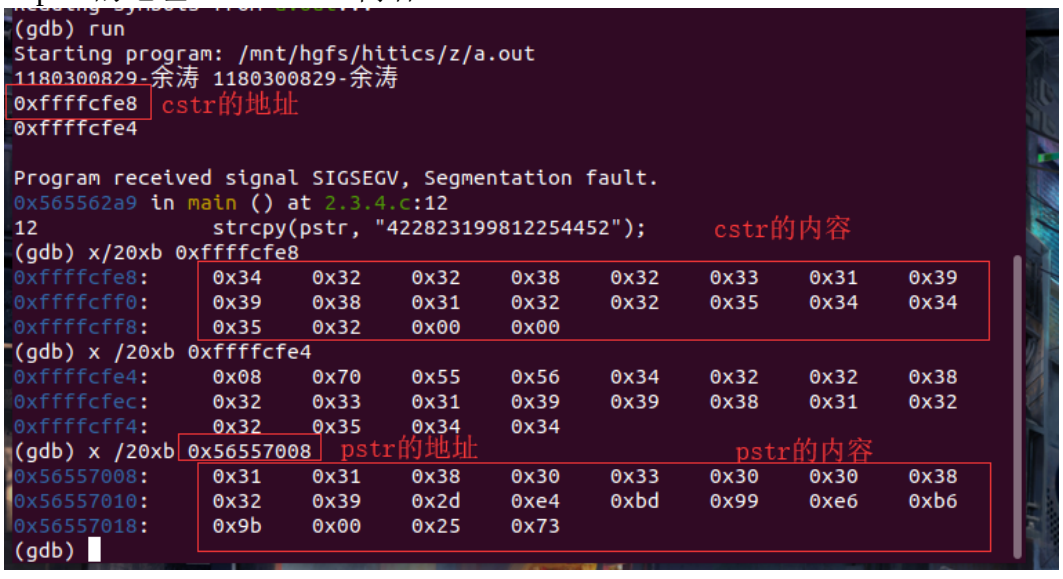


3.4 指针与字符串的区别（2 分）

cstr 的地址与内容截图，pstr 的内容与截图，截图 5

cstr 的地址：0xffffcfe8 内容：34 32 32 38 32 33 31 39 39 38 31 32 32 35 34 34 35 32

pstr 的地址：0xffffcfe4 内容：31 31 38 30 30 38 32 39 2d e4 bd 99 e6 b6 9b



pstr 修改内容会出现什么问题_____无法修改，显示段错误（核心已转储）_____

第 4 章 深入分析 UTF-8 编码

4.1 提交 utf8len.c 子程序

见附件

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

每个汉字都有其对应的 unicode 码， strcmp 按照汉字对应的 unicode 编码大小对汉字进行排序。所以用 strcmp 比较姓名的大小时，首先比较姓的 unicode 编码大小，若一样，则继续比较下一位名的 unicode 编码大小。

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？

先用一个数据库或文件，储存每个汉字关于笔画数量的编码，每次排序时，调用此数据库或文件中汉字对应的编码按照笔画数量进行排序，若笔画数量相同的姓氏，则按照汉字对应的 unicode 码进行排序，从而实现按照姓氏笔画排序的方法。

第 5 章 数据变换与输入输出

5.1 提交 cs_atoi.c 见附件

5.2 提交 cs_atof.c 见附件

5.3 提交 cs_itoa.c 见附件

5.4 提交 cs_ftoa.c 见附件

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。

应用程序是通过分别调用 read 和 write 函数来执行输入和输出的，其中两个函数为：ssize_t read(int fd,void *buf,size_t n)

ssize_t write(int fd,const void *buf,size_t n)

read 函数有一个 size_t 的输入参数和一个 ssize_t 的返回值。在 x86-64 系统中，size_t 被定义为 unsigned long,而 ssize_t(有符号的大小)被定义为 long。read 函数返回一个有符号的大小，而不是一个无符号大小，这是因为出错时它必须返回-1。

第 6 章 整数表示与运算

6.1 提交 fib_dg.c 见附件

6.2 提交 fib_loop.c 见附件

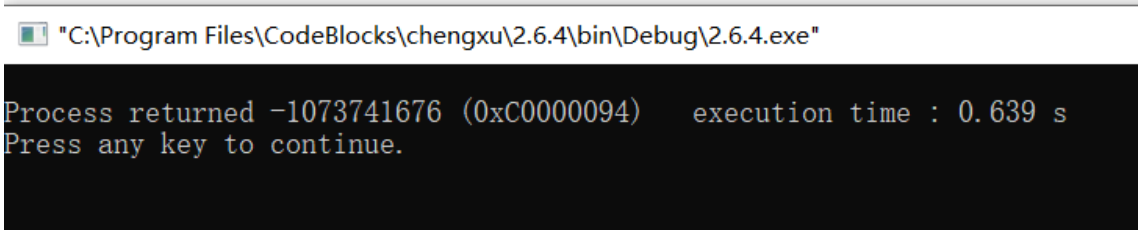
6.3 fib 溢出验证

int 时从 n=____47_____时溢出，long 时 n=____47_____时溢出。

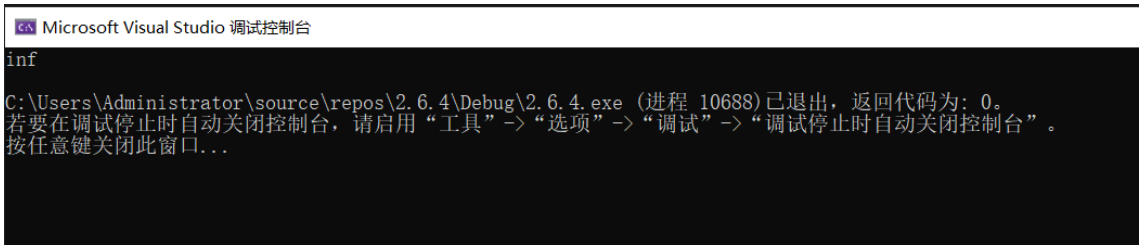
unsigned int 时从 n=____48_____时溢出，unsigned long 时 n=___48_____时溢出。

6.4 除以 0 验证：

除以 0：截图 1



除以极小浮点数，截图：



第 7 章 浮点数据的表示与运算

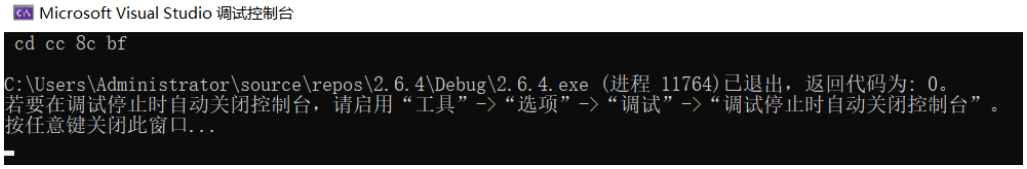
7.1 正数表示范围

写出 float/double 类型最小的正数、最大的正数（非无穷）
1.4E-45
4.9E-324

7.2 浮点数的编码计算

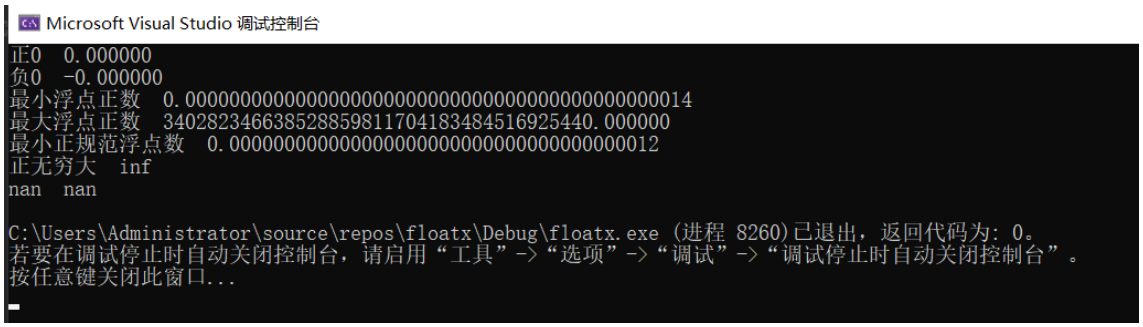
（1）按步骤写出 float 数-1.1 的浮点编码计算过程，写出该编码在内存中从低地址字节到高地址字节的 16 进制数值
解：S=1；
数值：float:-1.1
$-1.1_{(10)}=-1.00011001100110011001101_{(2)}\times 2^{0}$
尾数：$M=-1.00011001100110011001101_{(2)}$
$frac=00011001100110011001101_{(2)}$
阶码：E=0
Bias=127
Exp=127=01111111
编码结果：10111111100011001100110011001101
十六进制：BF8CCCCD
从低地址到高地址数值：CDCC8CBF

（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。

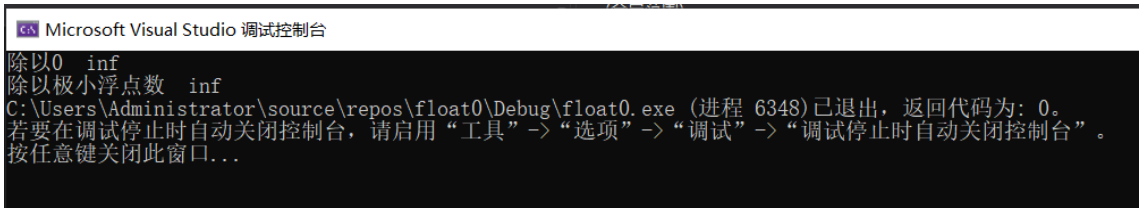


（2）提交子程序 floatx.c

见附件

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

见附件

7.5 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）：____-1.1111(23 个 1)×2^{-126}~1.1111(23 个 1)×2^{-126}(包含正负浮点数)_____、_____2^{25} 个浮点数_____、____1.0000(23 个 0)×2^{127}~1.11111(23 个 1)×2^{127}_____、____2^{23} 个浮点数____

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

_____3.4028234663852886e37_____

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。

比较方法：先定义一个很小的接近 0 的宏常量 EPS（一般定义为 1e-7），然后可以判断两个浮点数的差值是否小于 ESP。若小于，则可以认为这两个浮点数相等。若大于，则认为前数大于后数。

在对两个浮点数进行大小比较时，由于浮点数并非真正意义上的实数，只是其在某个范围内的近似，因此在比较时，不能简单地使用大于小于号进行比较，应该判断连个浮点数差值的绝对值是否近似为 0。

第 8 章 舍位平衡的讨论

8.1 描述可能出现的问题

舍位处理往往会采取四舍五入计算，这时就会产生误差，而如果报表中有这些数据的合计数值，那么舍位时产生的误差就会积累，有可能导致舍位过的数据与其合计值无法匹配。例如，保留一位小数的原始的数据是 4.5+4.5=9.0，而四舍五入只保留整数部分后，平衡关系就变为 5+5=9 了，看上去明显是荒谬的。
举例说明:
13,451.00 元 + 45,150.00 元 + 2,250.00 元 - 5,649.00 元 = 55,202.00 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 万元 + 4.52 万元 + 0.23 万元 - 0.56 万元 = 5.54 万元 ,出现 0.02 万的误差，平衡被打破。

8.2 给出完美的解决方案

将平衡差按照“最小调整值”，对绝对值比较大的数据进行分担调整：

设平衡差=求和后舍位值-舍位后求和值；“最小调整值”为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。如果舍位后合计值变小，则需要将数据调大，那么最小调整值就是 1；如果舍位后合计值变大，则需要将数据调小，最小调整值就是-1。而调整只针对绝对值比较大的数据，这样它们的相对偏差就会比较小。用平衡差的绝对值除以最小调整值得到需要调整数据的个数 n，然后对原始数据的绝对值进行排序，从大到小的 n 个数加上最小调整值或减去最小调整值（取决于舍位后求和值是变大还是变小），最后将调整后的值和未调整的值重新求和。

举例说明：假设有四个数 1.4，7.3 , 3.2 , 8.8。取整到个位，求和后舍位值=21，舍位后求和值=20，则平衡差为 1。对四个数舍位后绝对值从大到小进行排序，为 9 , 7 , 3 , 1，然后对第一个数加上最小调整值 1，变为 10 , 7 , 3 , 1，，相加后的值为 21，达到了舍位平衡。

第 9 章 总结

9.1 请总结本次实验的收获

本次实验我学习到了各种变量在内存中的地址，了解了各种变量变量在不同操作系统和不同位数机器的大小，学会了怎么用反汇编的方式查看变量在内存中的地址和内容，明白了内存由数据段、代码段、堆栈段等段组成，知道了反汇编的操作步骤，学会了 gcc 编译和 objdump/gdb 反汇编，知道了字符的 utf-8 编码方式，学会了了用 c 语言实现字符和浮点数、整数之间的转换，学会用 IEEE 754 编码构造浮点数的各种边界范围值，从而表达出浮点数的全部范围，理解了舍位平衡的概念。

9.2 请给出对本次实验内容的建议

希望实验的 ppt 的问题表述方式能够更改一下，很多句子模棱两可，容易无法理解或产生歧义。

希望能够减少重复实验内容，比如 3.2 和 3.3 中都要求打印 x，y，z 的地址和内容。

希望能够减少.c 文件内容和实验文字内容的重复考察，精简实验内容。

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.