请阅读以下7个程序，分别运行程序，分析结果，写成报告提交

**要求：**

**1.每道程序运行结果截屏；**

**2.结合已学知识点，对程序运行结果进行分析。**

**3.每人提交一份PBL报告，报告命名方式为：浮点数表示方法报告-学号-姓名.doc/PDF**

**1.阅读并运行以下C语言程序源代码，打印结果，分析程序功能，讨论无法表示的整数问题。**

#include<stdio.h>

#include<stdlib.h>

int main()

{

float f1 = 16777216.0;

float f2 = 16777217.0;

float f3 = 16777218.0;

printf("16,777,216:%f\n",f1);

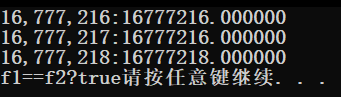
printf("16,777,217:%f\n",f2);

printf("16,777,218:%f\n",f3);

printf("f1==f2?%s",f1==f2?"true":"false");

return 0;

}



浮点数精度有限，当精度不够时，存储为其接近的值

**2.阅读并运行以下C语言程序源代码，打印结果，分析程序功能，讨论浮点数除零问题。**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

int main()

{

float a = 0.0;

float b;

a = a/0;

b = -sqrt(-1);

printf("a=%f b=%f\n",a,b);

return 0;

}



1.#INF00表示这是一个无穷大的正数，这个数超出了计算机可以表示的范围了，同 inf，同理

-1.INF00表示这是一个无穷大的负数，这个数的绝对值超出了计算机的表示范围,同 -inf。

1.#IND00或者 NaN表示这个结果不是个数

QNAN是指Quiet Not a Number

**阅读并运行以下C语言程序源代码，打印结果，分析程序功能，讨论浮点数双零问题。**

#include<stdio.h>

#include<stdlib.h>

union

{

char c[4];

float f;

int i;

}t1,t2;

int main()

{

t1.i = 0x80000000;

t2.i = 0x00000000;

if(t1.f == t2.f)

{

printf("float data is equal\n");

}

if(t1.i != t2.i)

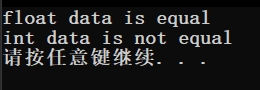
{

printf("int data is not equal\n");

}

return 0;

}



union在结构中都具有单独的内存位置，联合成员则共享同一个内存位置。

Float的第一位为符号位,尾数以原码形式存储，t1,t2分别表示-0和0,所以相等

int以补码形式存储，所以t1=-2147483648,t2=0不相等

**4.阅读并运行以下C语言程序源代码，打印结果，分析程序功能，讨论浮点数的精度问题。**

#include<stdio.h>

#include<stdlib.h>

int main()

{

double a = 0.1;

double b = 0.2;

double c = 0.3;

double d = a+b;

printf("a(0.1):%.30g\n",a);

printf("b(0.2):%.30g\n",b);

printf("c(0.3):%.30g\n",c);

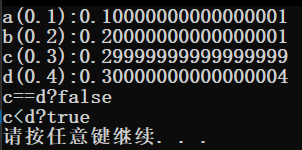
printf("d(0.4):%.30g\n",d);

printf("c==d?%s\n",c==d?"true":"false");

printf("c<d?%s\n",c<d?"true":"false");

return 0;

}



因为浮点数的基数是2，所以无法刚好表示出0.1,0.2,0.3等值，只能取接近的值进行存储，所以计算时产生了误差

**5.阅读并运行以下C语言程序源代码，打印结果，分析程序功能，讨论浮点数运算误差问题。**

#include<stdio.h>

#include<stdlib.h>

voidtest\_float\_cal()

{

float a,b,c;

int d;

b = 3.3;

c = 1.1;

a = b/c;

d = b/c;

printf("a=%f,d=%d",a,d);

if(a == 3.0)

{

printf("\nFloat a=3.3/1.1==3.0");

}

}

voidtest\_double\_cal()

{

double a,b,c;

int d;

b = 3.3;

c = 1.1;

a = b/c;

d = b/c;

printf("\n\na=%f,d=%d",a,d);

if(a != 3.0)

{

printf("\nDouble a=3.3/1.1!=3.0");

}

}

int main()

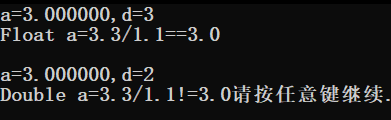
{

test\_float\_cal();

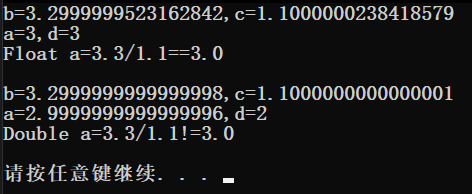
test\_double\_cal();

return 0;

}



这个程序很难看出来啊，所以我改了代码把详细的数据打印了出来



因为二进制数无法恰好表示3.3,1.1等值，所以出现误差

double比float的精度高，因此出现了不一致的结果

**6.阅读并运行以下C语言程序源代码，打印结果，分析程序功能，讨论浮点数结合律问题。**

#include<stdio.h>

#include<stdlib.h>

int main()

{

float a = 3.14;

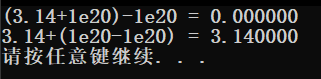
float b = 1e20;

printf("(3.14+1e20)-1e20 = %f\n",(a+b)-b);

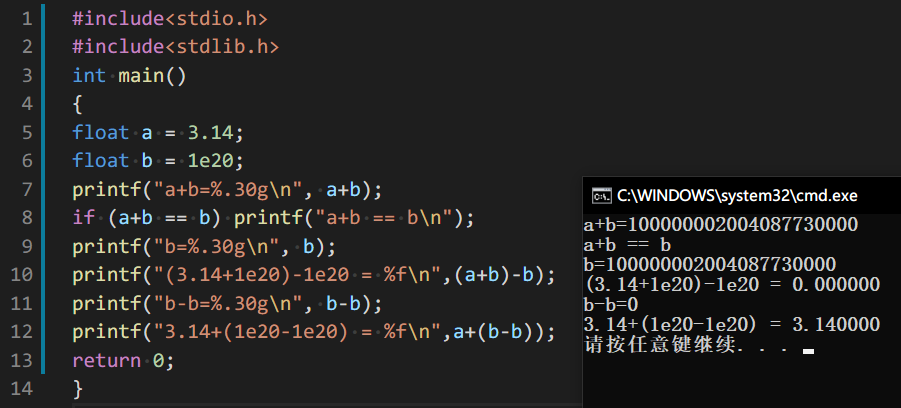
printf("3.14+(1e20-1e20) = %f\n",a+(b-b));

return 0;

}



同样改写代码输出详细数据得到



先计算a+b时因为float精度不够结果与b的值相同，所以(a+b)-b结果为0

先计算b-b等价于a+0所以结果为a,3.14

**7.阅读并运行以下C语言程序源代码，打印结果，分析程序功能,，讨论浮点数溢出问题。**

#include<stdio.h>

#include<stdlib.h>

union

{

char c[4];

float f;

int i;

}t1,t2,t3,t4;

// 输出字符的十六进制编码

voidchar\_hex\_out(char a)

{

const char HEX[] = "0123456789ABCDEF";

int index = a & 0x0F;

printf("%c%c",HEX[(a&0xF0)>>4],HEX[a&0x0F]);

}

// 输出4个字节数据的十六进制编码，可用于输出4字节变量的机器码

voidfour\_byte\_out(char \* addr)

{

//输出指针变量的值，指针本质上是内存地址，是一个无符号数

//假设采用小端方式存储

char\_hex\_out(\* (addr+3));

char\_hex\_out(\* (addr+2));

char\_hex\_out(\* (addr+1));

char\_hex\_out(\* (addr+0));

printf("\n");

}

voidfloat\_add\_overflow()

{

t1.i = 0x7F000000;

t2.i = 0x7F000000;

t3.f = t1.f+t2.f;

four\_byte\_out((char\*)(&t1.f));

four\_byte\_out((char\*)(&t2.f));

four\_byte\_out((char\*)(&t3.f));

printf("t1=%.60f\n",t1.f);

printf("t2=%.60f\n",t2.f);

printf("t1+t2=%.60f\n",t3.f);

}

voidfloat\_sub\_overflow()

{

t1.i = 0x00C00000;

t2.i = 0x00800000;

t3.f = t1.f-t2.f;

four\_byte\_out((char\*)(&t1.f));

four\_byte\_out((char\*)(&t2.f));

four\_byte\_out((char\*)(&t3.f));

printf("t1=%.61f\n",t1.f);

printf("t2=%.61f\n",t2.f);

printf("t1-t2=%.61f\n",t3.f);

}

int main()

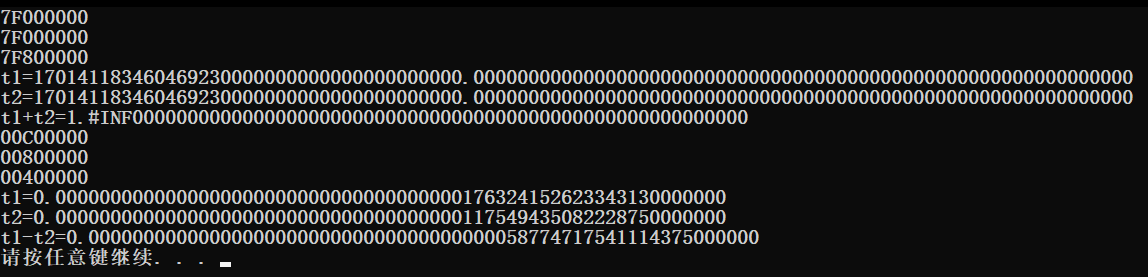
{

float\_add\_overflow();

float\_sub\_overflow();

return 0;

}



Float\_add\_overflow

T1,t2阶码为1111 1110尾数为000 0000 0000 0000 0000 0000 表示2^254

相加发生上溢

T3正无穷INF 正负号0 指数域1111 1111 尾数域000 0000 0000 0000 0000 0000

Float\_sub\_overflow

T1 阶码为0000 0001尾数为100 0000 0000 0000 0000 0000 表示 1.5\*2^-126

T2 最小的规约数 指数域0000 0001尾数域000 0000 0000 0000 0000 0000 数值±2≈ ±1.18×10

T3中间大小的非规约数 指数域0000 0000尾数域100 0000 0000 0000 0000 0000 数值±2× 2= ±2≈ ±5.88×10