**西安交通大学实验报告**

计算机试验班001 王嘉禾2193211079

**课程名称：**计算机科学技术导论

**实验内容：**输入一个数，输出该数据在计算机中的存储方式

**实验日期：**2021年5月30日

**实验环境：**windows10/visual studio 2019/Dev c++ 5.1.1

**总体设计**

整形数在计算机中以补码存储，浮点数有单精度和双精度两种存储方式，首先判断输入的数据是整形数还是浮点数。

*bool isint(double f) //判断整形数or浮点数*

*{*

*int x=f;*

*double temp=f-x;*

*if (temp>-(1e-6) && temp<1e-6) return 1;*

*//由于浮点数并不是完全精确存储，只判断到小数点后6位*

*return 0;*

*}*

对于整形数，若为正数，则原码反码补码均相同，只需作一个简单的进制转换，若为负数，则反码为对应绝对值的原码每一位依次取反，补码为反码加一（二进制）。

*void tranint(int n)*

*{*

*int f=0,k=31;*

*if (n<0)*

*{*

*n=-n-1; //负数*

*f=1;*

*}*

*memset(a,0,sizeof(a));*

*while(n>0) //进制转换部分*

*{*

*a[k]=(n%2);*

*n/=2;*

*k--;*

*}*

*if (f) for(int i=0;i<=31;++i) a[i]=1-a[i]; //负数的补码规则不同*

*for(int i=0;i<=31;++i)*

*{*

*cout << a[i];*

*if ((i+1)%8==0) cout << ' ';*

*}*

*cout << endl;*

*return;*

*}*

对于浮点数，依次处理符号位，指数部分和尾数部分（尾数部分只保留数据二进制形式中的有效数字且去除首位的1，指数部分即为实现此操作需要移动小数点的位数的二进制码）。对于指数部分，若整数部分不为零，则整数部分的长度减一就是小数点移动的位数，若整数部分为零，则对小数进行“模拟进制转换操作”，并将计数器不断累加，当出现1时计数器的值为小数点右移位数，对于两种精度，双精度的存储精度较高，考虑到高精度的存储方式可以兼容低精度的数据，因此优先处理双精度，在判断其精度是否可以用单精度存储，若可以则也输出单精度存储的结果。

*void trandouble(double n)*

*{*

*memset(a,0,sizeof(a));*

*if (n<0)*

*{*

*a[0]=1;*

*n=-n;*

*}*

*m=(int)(n);*

*n-=m;*

*int k=0,k1,i; //k最终为移动小数点的位数*

*while(m>0) //整数部分进制转换*

*{*

*t[k]=m%2;*

*m/=2;*

*k++;*

*}*

*k--;*

*for(i=0;i<=(k-1)/2;++i) swap(t[i],t[k-i]);*

*i=k+1;*

*if (!i)*

*{*

*while(n\*(2.0)<1) //计算要使整数部分仅保留1需要移动小数点多少位（左移为正）*

*{*

*n\*=(2.0);*

*k--;*

*}*

*}*

*while(n>0 && i<=52) //小数部分进制转换*

*{*

*n\*=(2.0);*

*t[i]=(int)(n);*

*if (n>=(1.0)) n-=(1.0);*

*i++;*

*}*

*k1=k;*

*k+=1023; //指数位的特殊存储规则，双精度时1023代表0*

*if (!(k>=0) && k<=2047)*

*{*

*cout << "out of boundary!\n";*

*return;*

*}*

*i=11;*

*while(k>0 && i>0) //指数位转换为二进制*

*{*

*a[i]=k%2;*

*k/=2;*

*i--;*

*}*

*cout << "double: " << a[1] << " ";*

*for(int i=1;i<=11;++i) cout << a[i];*

*cout << " ";*

*for(int i=1;i<=52;++i) cout << t[i];*

*cout << endl;*

*k=k1+127; //尝试处理单精度存储，出位数不同外，规则与双精度完全相同*

*if (k>=0 && k<=255)*

*{*

*i=8;*

*while(k>0 && i>0)*

*{*

*a[i]=k%2;*

*k/=2;*

*i--;*

*}*

*cout << "float: " << a[1] << " ";*

*for(int i=1;i<=8;++i) cout << a[i];*

*cout << " ";*

*for(int i=1;i<=23;++i) cout << t[i];*

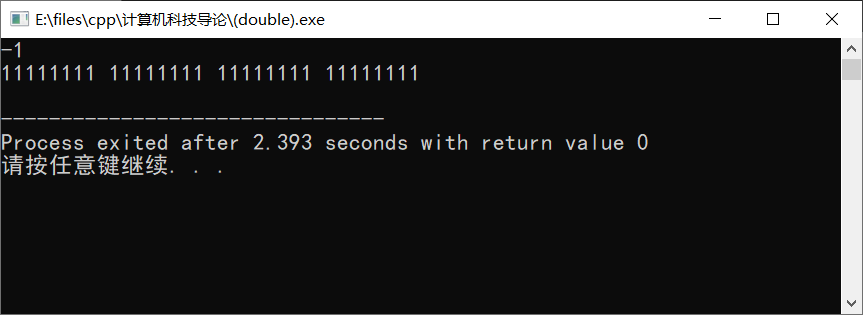
*cout << endl;*

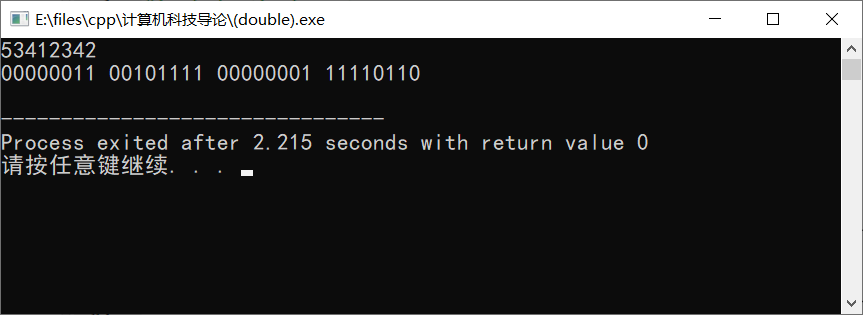
*}*

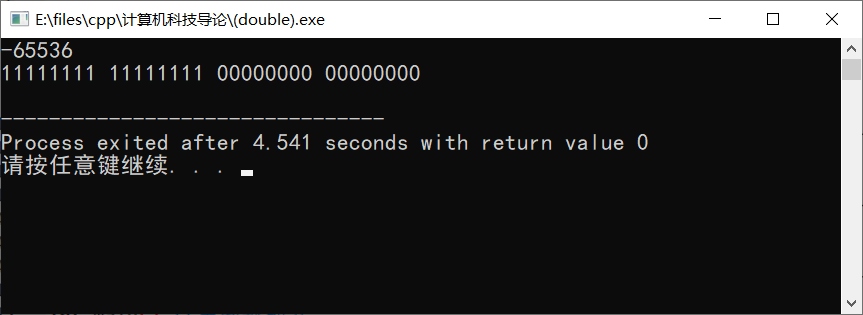
*return;*

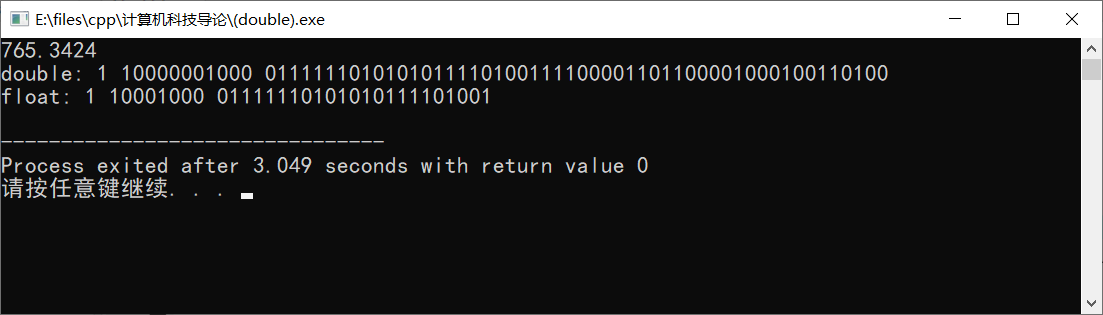
*}*

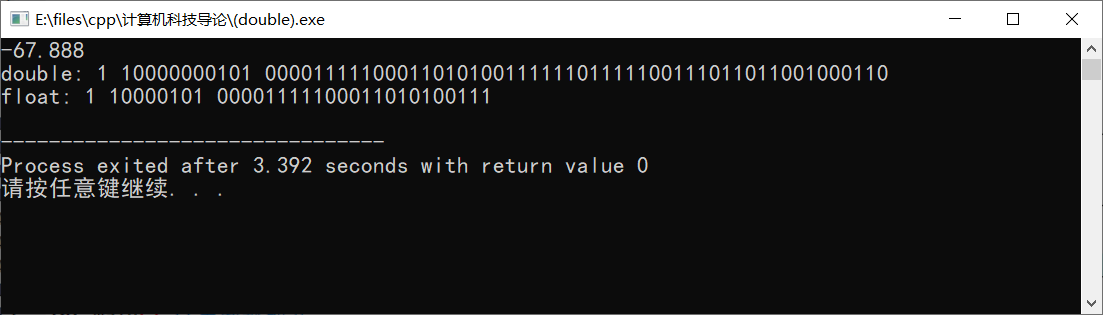
**实验结果**











**实验总结&问题**

对于整形数的处理较为简单，但是在运行程序时，发现输入-2147483648时程序可以成功运行，但在判断语句中加入（n>=-2147483648）时却无论输入什么数据都判断失败，原因暂时不详。对于浮点数的处理，如果模拟手工转换的过程，先将进制转换完毕再移动小数点，则将浪费大量时间复杂度，因为无法预先知道移动位数，因此，需要先计算小数点移动位数再处理小数。