浮点数

```
#include <stdio.h>
int main()
{
        int a=10;
        double b=10.12;//没有后缀默认为double
        //double代表着浮点数  double是双精度浮点的首单词
        int c=3;
        printf("%d/%d=%d\n",a,c,a/c);
        printf("%f/%d=%f\n",b,c,b / c);
    printf("double的大小为%d",sizeof(a));//输出为8，代表着其大小为8字节

        return 0;
}
```

double代表的是双浮点数，双浮点数类型占用8个字节（64位），相比于float（单精度浮点型）而言，double提供更加高的精度，通常能保证15到16位有效数字，减少误差并提供更为准确的结果

```
#include <stdio.h>
int main(){
    float a = 10.1f;//后缀为指定a这个数为float
    int b =2;
    //float 代表的是单浮点数
    printf("%f\n",&a);
    printf("%d\n",&b);
    printf("float的大小为%d",sizeof(a));//输出结果为4，代表着其占四个字节
    return 0;
}
```

float代表的是单精度浮点数，float占用4个字节（32位），精度相较于double低

```
#include <stdio.h>
int main(){
    long double a = 1.213124123124213L;//L后缀代表着指定long double为a的类型
    printf("long double 的大小为%d\n",sizeof(a));
    printf("%f",a);
```

```
    return 0;
}
```

long double是扩展精度，其精度为4倍精度，其大小与平台相关，可能为8字节，12字节，16字节，且精度在不同平台差别很大，在X86 Linux为80位扩展精度，arm通常与double相同吗，少数平台会有128位四倍精度。

```
#include <stdio.h>
int main(){
    //浮点数的输入
    double a;
    float b;
    long double c;
    scanf ("%f\n",b);//float用%f
    scanf ("%lf\n",a);//double必须使用%lf
    scanf ("%Lf\n",c);//long double用%Lf
    return 0;
}
```

浮点数的输入输出，输入而言，

float 用 %f,

double 用 %lf,

long double 用%Lf.

对于输出而言,

%f 代表的是输出十进制小数

%e 代表的是输出科学计数法

%g 代表的是自动选择%f 或 %e

%a 代表的是十六进制浮点数（C99）

%Lf 为long double的输出格式

float 和 double 在printf中可以都使用%f,因为float会被提升为double(float-->doubl)