Chapter 14: Standard Math

Section 14.1: Power functions - pow(), powf(), powl()

The following example code computes the sum of *1+4(3+3^2+3^3+3^4+...+3^N)* series using pow() family of standard math library.

```
#include <stdio.h>
#include <math.h>
#include <errno.h>
#include <fenv.h>

int main()
{
        double pwr, sum=0;
        int i, n;

        printf("\n1+4(3+3^2+3^3+3^4+...+3^N)=?\nEnter N:");
        scanf("%d",&n);
        if (n<=0) {
                printf("Invalid power N=%d", n);
                return -1;
        }

        for (i=0; i<n+1; i++) {
                errno = 0;
                feclearexcept(FE_ALL_EXCEPT);
                pwr = powl(3,i);
                if (fetestexcept(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW |
                        FE_UNDERFLOW)) {
                        perror("Math Error");
                }
                sum += i ? pwr : 0;
                printf("N= %d\tS= %g\n", i, 1+4*sum);
        }

        return 0;
}
```

Example Output:

```
1+4(3+3^2+3^3+3^4+...+3^N)=?
Enter N:10
N= 0    S= 1
N= 1    S= 13
N= 2    S= 49
N= 3    S= 157
N= 4    S= 481
N= 5    S= 1453
N= 6    S= 4369
N= 7    S= 13117
N= 8    S= 39361
N= 9    S= 118093
N= 10    S= 354289
```

Section 14.2: Double precision floating-point remainder: fmod()

This function returns the floating-point remainder of the division of `x/y`. The returned value has the same sign as x.

```
#include <math.h> /* for fmod() */
#include <stdio.h> /* for printf() */

int main(void)
{
    double x = 10.0;
    double y = 5.1;

    double modulus = fmod(x, y);

    printf("%lf\n", modulus); /* f is the same as lf. */

    return 0;
}
```

Output:

```
4.90000
```

Important: Use this function with care, as it can return unexpected values due to the operation of floating point values.

```
#include <math.h>
#include <stdio.h>

int main(void)
{
    printf("%f\n", fmod(1, 0.1));
    printf("%19.17f\n", fmod(1, 0.1));
    return 0;
}
```

Output:

```
0.1
0.09999999999999995
```

Section 14.3: Single precision and long double precision floating-point remainder: fmodf(), fmodl()

Version ≥ C99

These functions returns the floating-point remainder of the division of `x/y`. The returned value has the same sign as x.

Single Precision:

```
#include <math.h> /* for fmodf() */
#include <stdio.h> /* for printf() */
```

```c
int main(void)
{
    float x = 10.0;
    float y = 5.1;

    float modulus = fmodf(x, y);

    printf("%f\n", modulus); /* lf would do as well as modulus gets promoted to double. */
}
```

Output:

```
4.90000
```

Double Double Precision:

```c
#include <math.h> /* for fmodl() */
#include <stdio.h> /* for printf() */

int main(void)
{
    long double x = 10.0;
    long double y = 5.1;

    long double modulus = fmodl(x, y);

    printf("%Lf\n", modulus); /* Lf is for long double. */
}
```

Output:

```
4.90000
```