第一章作业

问题叙述：分别以单精度和双精度数据类型采用以下公式分别计算 ln2 的近似值

(1)

$$\ln(1+x)=\sum_{n=1}^{\infty}(-1)^{n+1}\frac{x^n}{n},\ (-1<x\le 1)$$

(2)

$$\ln\frac{1+x}{1-x}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+\cdots+\frac{x^{2n+1}}{(2n+1)}+\cdots\right),\ (-1<x<1)$$

- 要求结果具有 4 位有效数字；
- 尝试得到更高精度的结果；
- 对结果进行分析、讨论。

问题分析：首先，观察两种方法，第一种方法有正负变换，而第二种没有，因此猜测第二种算法更加精确。其次，问题中要求了结果具有 4 位有效数字，因此可以采用迭代方法估计有效数字。预先设定容限 epsilon_s，再逐次计算 epsilon_a，直至小于。要想得到更高精度的结果，只需要进一步缩小容限的值，就能够使结果更加精确。

Python 程序：见另附程序文件。

Python 程序结果：

计算得出结果，均符合 4 位有效数字的约束。

```
ln2的真值是:  0.6931471805599453
公式1:
单精度: y=0.6931222081184387, n=28853.0, epsilon_s=0.00005, epsilon_a=0.0000499628
双精度: y=0.6931645082791072, n=28856.0, epsilon_s=0.00005, epsilon_a=0.0000499968
公式2:
单精度: y=0.6931460499763489, n=5.0, epsilon_s=0.00005, epsilon_a=0.0000162524
双精度: y=0.6931460473908271, n=5.0, epsilon_s=0.00005, epsilon_a=0.0000162881

进程已结束,退出代码0
```

改变 epsilon_s 使其变小，就可以得到精确度更高的结果，如下图所示。可见使用公式 1 计算时，要达到 6 位有效数字的精度，需要进行三百万次的迭代。使用 time 库检测程序运行时间，达到了惊人的 13.3 秒。

```
ln2的真值是:  0.6931471805599453
公式1:
单精度: y=0.6931377649307251, n=3050404.0, epsilon_s=0.00000050, epsilon_a=0.0000004300
双精度: y=0.6931473538467536, n=2885392.0, epsilon_s=0.00000050, epsilon_a=0.0000005000
公式2:
单精度: y=0.6931471824645996, n=7.0, epsilon_s=0.00000050, epsilon_a=0.0000001720
双精度: y=0.6931471702560119, n=7.0, epsilon_s=0.00000050, epsilon_a=0.0000001392
运行耗时 13.317945

进程已结束,退出代码0
```

对计算结果的进一步分析：观察分析公式 1 与公式 2 的结果，为了达到 4 位有效数字的精度，使用公式 2 无论单精度还是双精度都只需要进行 5 次迭代（n=5），然而使用公式 1 时，却需要进行约 3 万次的运算（n=28853）。从误差分析的角度看，这是因为使用公式 1 时会出现正负交替相加的情况，而这就会造成减性抵消的舍入误差，因此需要进行更多次的运算。为了达到更高的精度时，使用公式 2 也更有效率。如上图，为达到 6 位精度，公式 2 需要运算 7 次，而公式 1 需要三百万次。这样看来，公式 2 更加精确，更加适合计算机运算。

观察 ε_a，可以发现公式 2 的 ε_a 更小，也能证明公式 2 效率更高：进行了更少次数的运算而相对误差更小。

再观察单双精度的差别，可以发现无论公式 1 还是公式 2，单双精度对于迭代的次数（n）并没有太大的影响。为了节约计算机的空间利用，针对本问题可以只采用单精度的数据类型来实现。