Simple code `sum.f90`

```
program summary
  real :: sum, sum1
  integer :: n

  sum = 1.
  sum1 = 0.
  n = 1

  do while ( sum > sum1 )
     sum1 = sum
     n = n + 1.
     sum = sum + 1./n
  enddo
  print*,'End after ',n,'-steps,  sum = ', sum

end program summary
```

see also `https://msekce.karlin.mff.cuni.cz/~dolejsi/Vyuka/NS_source/Fortran/index.html` link

translation of the program `sum.f90` from the command line:

- `gfortran sum.f90 -o sum` – single precision
- `gfortran -fPIC -fdefault-real-8 sum.f90 -o sum` – double precision

1. Write a simple code showing that $\sum_{n=1}^{\infty} \frac{1}{n}$ is finite in the finite precision arithemtic. Try the single and double precision arithemtics.

2. Find experimentally the approximate values of OFL, UFL and ϵ_{mach}. Try the single and double precision arithemtics. Compare the obtained valued with the theoretical ones.

3. Try and explain the behaviour of the following codes

```
   eps = 1.
10 eps = eps/2.
   write(*,'(es18.10)') eps
   eps1 = eps + 1
   if(eps1 > 1.) goto 10
```

   and

```
   eps = 1.
10 eps = eps/2.
   write(*,'(es18.10)') eps
   if(eps > 0.) goto 10
```

   Explain the differences?

4. The number $e = 2.7182817459106445\ldots$ can be defined as $e = \lim_{n\to\infty}(1 + 1/n)^n$. This suggests an algorithm for calculating e: choose n large and evaluate $e^* = (1 + 1/n)^n$. Write a simple code and explain the results, Explain this effect, i.e, why the approximation e^* of the Euler number e is first increasing for increasing n and then it decrease until complete information is lost.

5. Write a code for the solution of the quadratic equation $ax^2 + bx + c = 0$, which is robust with respect the overflow, underflow and the cancellation. Test the following data:

   - a =6, $b = 5$, $c = -4$

- a =6E+30, b = 5E+30, c = −4E+30
- $a = 1$, $b = -1E + 6$, $c = -1$

6. The Taylor series for the error function is

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{k!(2k+1)}.$$

   This series converges for all $x \in \mathbb{R}$. Programme it and try $x = 0.5$, $x = 1.0$, $x = 5$ and $x = 10$. Explain the results.

7. Numerical differentiating of a function f is based on the formula:

$$f'(\bar{x}) \approx \frac{f(\bar{x} + h) - f(\bar{x})}{h} =: Df(\bar{x}; h).$$

   - Determine the dependence of discretization and rounding errors on h.
   - For which h the formula is the most accurate (in finite precision arithmetic).
   - Write a simple code for $f(x) = x^2$ at $\bar{x} = 1.5$ and test several values h.
   - Try to find an algorithm, which gives the optimal size of h.