```python
import os, sys
sys.path.append(os.path.join(os.path.dirname(__file__), '../ch02'))

from program2_1 import Dvector
from program7_1 import func1, func2

def main():
    N = 6
    eps = 10.0 ** -10.0

    print(f"2.0/(x*x) を [1,2] で積分します．最大反復回数は{N} です")
    print("結果は{:20.15f} です".format(romberg(1.0, 2.0, N, eps, func1)))

    print(f"4.0/(1+x*x) を [0,1] で積分します．最大反復回数は{N} です")
    print("結果は{:20.15f} です".format(romberg(0.0, 1.0, N, eps, func2)))


# ロンバーグ法
def romberg(a: float, b: float, N: int, eps: float, f) -> float:
    t = Dvector(0,N)
    h = b - a
    f0 = f(a)
    f1 = f(b)
    t[0] = h*( f0 + f1 ) / 2.0

    # ロンバーグ法
    for n in range(1, N+1):
        h = h / 2.0
        S = 0.0
        for j in range(1, int(2.0**n - 1.0)+1):
            S += f(a + j*h)
        t[n] = h*( f0 + 2.0*S + f1 ) / 2.0
        if abs(t[n] - t[n-1]) < eps:
            return t[n]
        k = n
        for m in range(1, n+1):
            k = k - 1
            t[k] = ( 4.0**m * t[k+1] - t[k] ) / ( 4.0**m - 1.0 )
            if k >= 1 and abs(t[k] - t[k-1]) < eps:
                return t[k]

    return t[N] # 収束しなければ t[N] を積分値とする


if __name__ == "__main__":
    main()
```