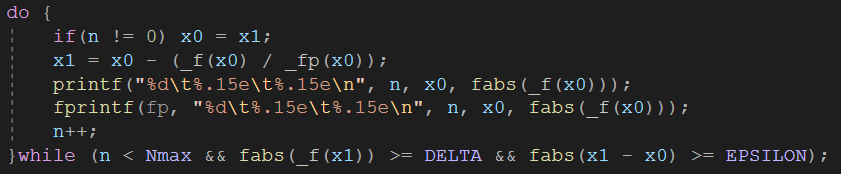
**고급소프트웨어 실습 4주차 보고서**

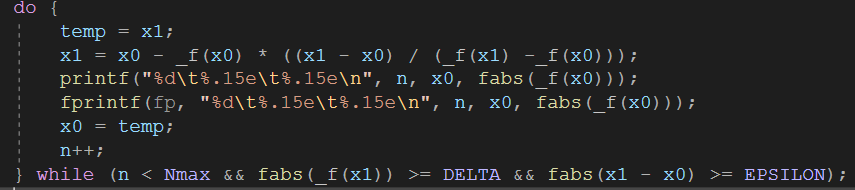
**20191657 최세은**

**실습 1-1**

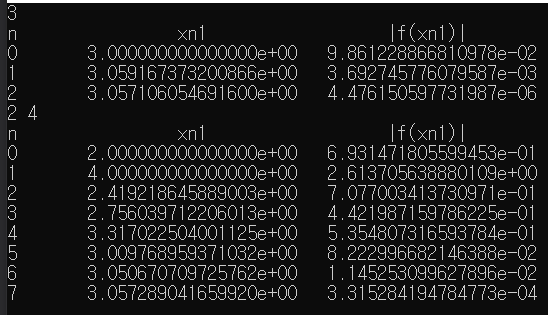
Newton-Raphson 방법과 Secant 방법을 구현했다.

둘 다 do-while 문으로 반복하며 세 가지 종료조건에 만족하지 않는 이상 계속 반복된다.





**2)**

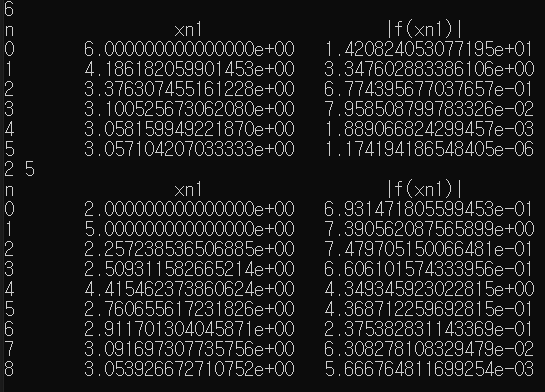


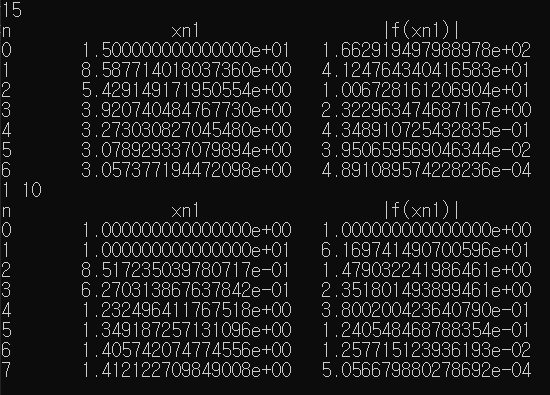
우선 다른 두 방법으로 계산했음에도 값이 거의 같게 나오고, f(xn1) 값도 매우 작으므로 근에 거의 수렴함을 알 수 있다.

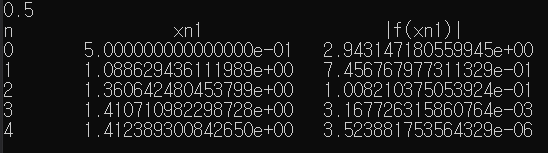
**3)**

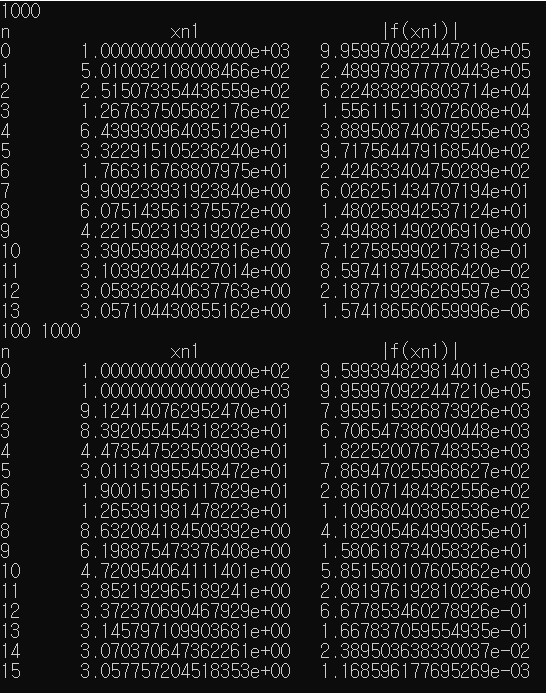
위에서 Newton-Raphson 방법일 땐 2회, Secant 방법일 땐 7회로 수행 속도가 측정되었다. 따라서 자료에 적혀있듯 Newton-Raphson 방법일 때 더 빠른 것이 확인되었다.

**4)**









여러 임의의 초기값들로 두 가지 방법을 이용해 근을 구해본 결과 항상 빠르게 수렴했다.

**실습 1-2**

실습 1-1에서 구현한 Newton-Raphson 방법으로 해당 함수 f3의 네가지 근을 추론했더니 약 1.1, 2.2, 3.3, 4.4로 나왔다.

|  |
| --- |
| n xn1 |f(xn1)|  0 1.250000000000000e+00 9.201989173889160e-01  1 1.042045765089152e+00 5.088188970821008e-01  2 1.095069668334355e+00 3.969382181926306e-02  3 1.099959434832845e+00 3.198038344152110e-04  n xn1 |f(xn1)|  0 2.160000000000000e+00 1.082853194433824e-01  1 2.199405505800810e+00 1.596032560058802e-03  n xn1 |f(xn1)|  0 3.420000000000000e+00 3.328822928075965e-01  1 3.303218105517611e+00 8.604752839687535e-03  2 3.299995028689731e+00 1.242246460719798e-05  n xn1 |f(xn1)|  0 4.220000000000000e+00 1.043716460385667e+00  1 4.493756232340251e+00 8.712083369516108e-01  2 4.412022603296798e+00 9.790573178332806e-02  3 4.400239358647029e+00 1.870339609240546e-03 |

**실습 1-4**

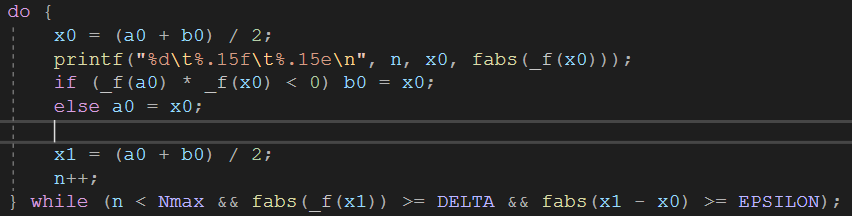
**2)**

|  |
| --- |
| n xn1 |f(xn1)|  0 3.000000000000000e+00 9.861228866810978e-02  1 2.704163133995671e+00 5.207513019400256e-03  2 2.718245098722536e+00 1.351220623102556e-05  n xn1 |f(xn1)|  0 3.000000000000000e+00 9.861230850219727e-02  1 2.704163074493408e+00 5.207538604736328e-03  2 2.718245029449463e+00 1.353025436401367e-05 |

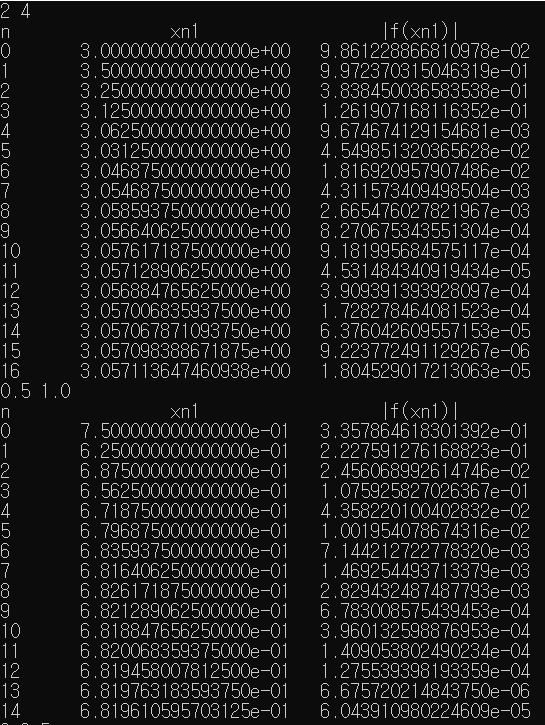
거의 같게 나왔다. 그러나 e = 2.71828182846.. 와 비교하면 double 형으로 계산했을 때 미세하게 더 정확함을 알 수 있다.

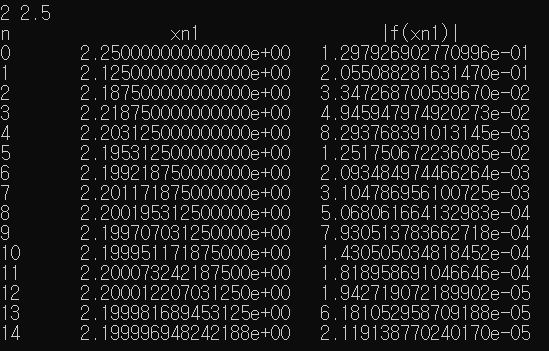
**과제 1**

현재 구간이 [a0, b0] 이고 그 둘의 중간 값이 x0일 때 f(a0)f(x0) < 0 이면 다음 구간은 [a0, x0], 아니면 다음 구간은 [x0, b0]이 된다.



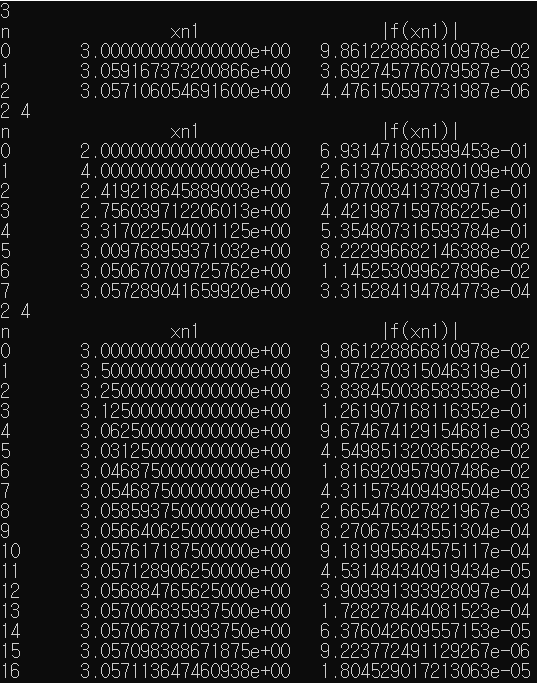
**1-3)**





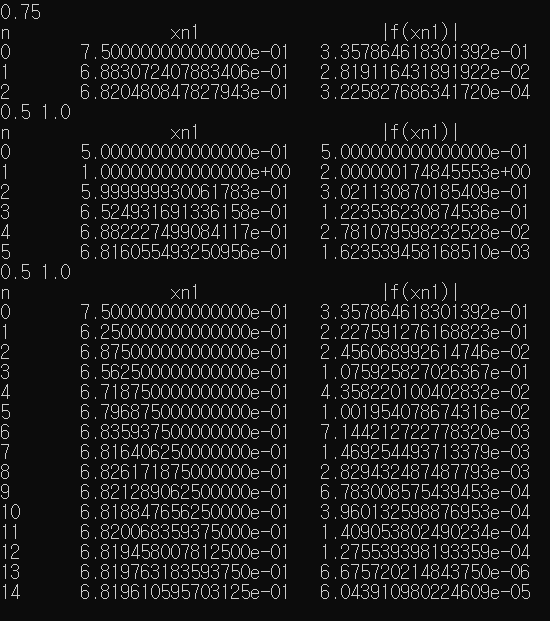
실습 때 사용한 함수 f1, f2, f3를 이용해 bisection 방법으로 근을 구한 결과이다. f1, f2는 secant 방법을 사용할 때와 같은 초기값 두 개를, f3은 네 개의 근 중 2.2를 찾기 위해 새로운 초기값 2, 2.5를 주었더니 모두 올바르게 근에 수렴했다.

**1-4)**



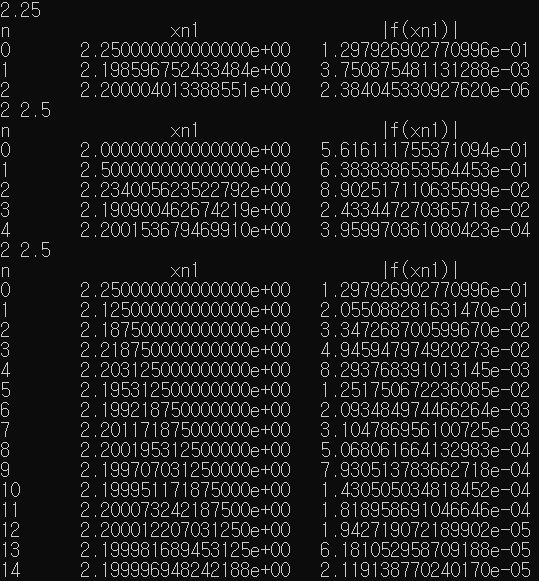
[함수 f1]

차례대로 Newton-Raphson, Secant, Bisection 방법으로 근을 구했더니 수행 속도가 2n, 7n, 16n으로 나왔다. 이론대로 Newton-Raphson 방법일 때 가장 빠르게 구해졌으며 Bisection 방법일 때 가장 느리게 구해졌다.



[함수 f2]

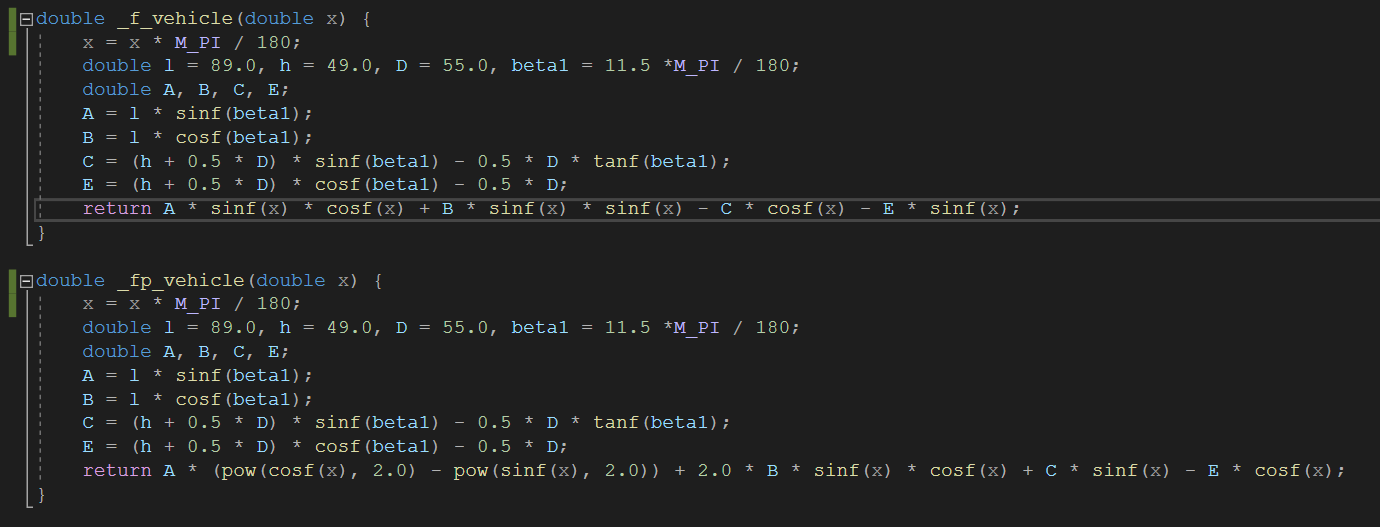
차례대로 Newton-Raphson, Secant, Bisection 방법으로 근을 구했더니 수행 속도가 2n, 5n, 14n으로 나왔다. 이론대로 Newton-Raphson 방법일 때 가장 빠르게 구해졌으며 Bisection 방법일 때 가장 느리게 구해졌다.



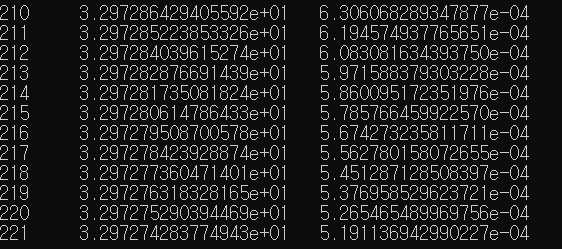
[함수 f3]

차례대로 Newton-Raphson, Secant, Bisection 방법으로 근을 구했더니 수행 속도가 2n, 4n, 14n으로 나왔다. 이론대로 Newton-Raphson 방법일 때 가장 빠르게 구해졌으며 Bisection 방법일 때 가장 느리게 구해졌다.

**과제 2**



자료에 나와있는 수치를 대입해 공식을 구현한 후 33 근처에서 근이 나온다고 하고 있으므로 초기값을 33으로 하여 Newton-Raphson 방법을 사용했다.



Newton-Raphson 방법은 저번 실습 1-1 때 구현했던 코드를 그대로 이용했다. 따라서 반복문 아래는 아래 식이 들어간다.

