▪ 수업활동일지: 토의·토론(개별 제출)

교과목명	응용수학	분반	001
수업일자	2021.10.06	교수자명	김형석
이름	이지호	학번	20163290

▣ 토의·토론 주제

방정식 관련 강의 마지막 부분에 있는 문제 해결하기

- 이분법, 오차조정법, 뉴턴방법, 할선법의 알고리즘을 작성하여 주어진 문제의 근을 구하시오.

(오차 한도는 0.001)

- 어떤 방법이 가장 빠른 시간에 근사근을 구하는지를 설명하시오.

▣ 토의·토론 내용정리

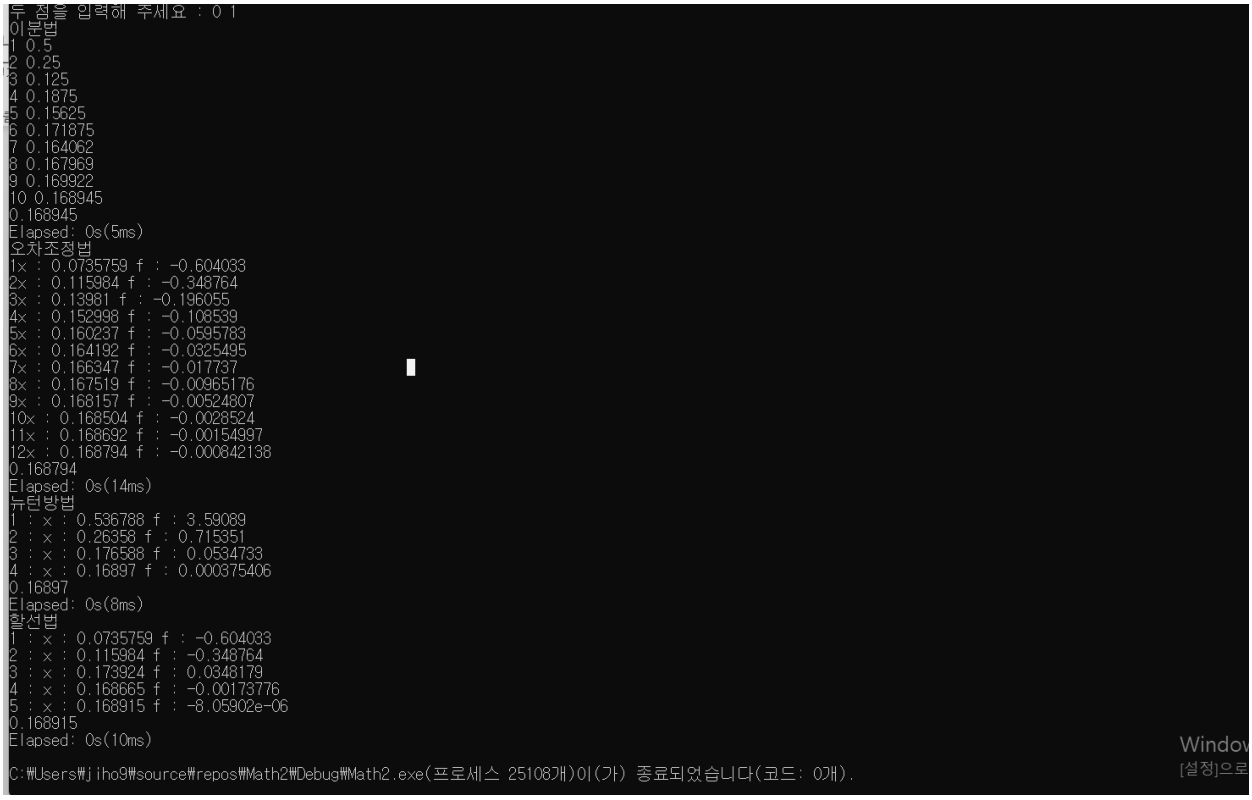
```
#include <iostream>
#include<iomanip>
#include <math.h>
#include <ctime>

using namespace std;

double F(double x) {
	return 5*x*exp(x) - 1;
}

double mF(double x) {
	return (5 * x + 5)*exp(x);
}

double Bisection(double a, double b) {

	double e1 = 0.001;
	double e2 = 0.001;
	int Num = 1000;

	double center;
	int i = 1;

	while (i <= Num) {
		center = a + (b - a) / 2;
		cout << i << " " << center << endl;

		if (fabs(F(center)) < e1 || ((b - a) / 2 < e2)) {
			cout << center << endl;
```

```
                        return center;
                }
                i++;
                if (F(b) * F(center) < 0) {
                        a = center;
                }
                else {
                        b = center;
                }
        }
        return 0;
}

double RegularFalsi(double a, double b) {

        double f0 = F(a), f1 = F(b);
        double x, f;
        double e = 0.001;
        int Num = 10000;

        if (f0 * f1 > 0.0) {
                cout << "Error" << endl;
        }
        int i = 1;
        while(i <Num){
                x = a - (a - b) * f0 / (f0 - f1);
                f = F(x);
                cout << i <<  "x : " << x << " " << "f : " << f << endl;

                if (fabs(f) < e) {
                        cout << x << endl;
                        break;
                }

                if (f0 * f < 0) {
                        b = x;
                        f1 = f;
                }
                else {
                        a = x;
                        f0 = f;
                }
                i++;
        }

        return 0;
}

int Newton(double b) {

        double e = 0.001;
        double x, x0 = b;
        int N = 100;

        int i = 0;
        while (i < N) {
                x = x0 - (F(x0) / mF(x0));
                if (fabs(x - x0) < e) {
```

```
                        cout << x0 << endl;
                        break;
                }
                i++;
                x0 = x;
                cout << i << " : " << "x : " << x << " " << "f : " << F(x) << endl;
        }
        return 0;
}

double secant(double a, double b) {

        double e = 0.001;
        double xn, x0 = a, x1 = b;
        int i = 1;
        int Num = 100;

        while (i < Num) {
                xn = x1 - F(x1) * ((x1 - x0) / (F(x1) - F(x0)));
                cout << i << " : " << "x : " << xn << " " << "f : " << F(xn) << endl;
                if (i == Num || fabs(F(xn)) < e) {
                        cout << xn << endl;
                        break;
                }
                x0 = x1;
                x1 = xn;
                i++;
        }
        return 0;
}

int main() {

        double a, b;
        cout << "두 점을 입력해 주세요 : ";
        cin >> a >> b;

        cout << "이분법₩n";
        clock_t startTime1 = clock();

        Bisection(a, b);

        clock_t endTime1 = clock();
        clock_t elapsed1 = endTime1 - startTime1;
        double timeInSecond1 = (double)(elapsed1 / CLOCKS_PER_SEC);
        cout << "Elapsed: " << timeInSecond1 << "s(" << elapsed1 << "ms)" << "₩n";

        cout << "오차조정법₩n";
        clock_t startTime2 = clock();

        RegularFalsi(a, b);

        clock_t endTime2 = clock();
        clock_t elapsed2 = endTime2 - startTime2;
        double timeInSecond2 = (double)(elapsed2 / CLOCKS_PER_SEC);
        cout << "Elapsed: " << timeInSecond2 << "s(" << elapsed2 << "ms)" << "₩n";

        cout << "뉴턴방법₩n";
```

```
        clock_t startTime3 = clock();

        Newton(b);

        clock_t endTime3 = clock();
        clock_t elapsed3 = endTime3 - startTime3;
        double timeInSecond3 = (double)(elapsed3 / CLOCKS_PER_SEC);
        cout << "Elapsed: " << timeInSecond3 << "s(" << elapsed3 << "ms)" << "₩n";

        cout << "할선법₩n";
        clock_t startTime4 = clock();

        secant(a, b);

        clock_t endTime4 = clock();
        clock_t elapsed4 = endTime4 - startTime4;
        double timeInSecond4 = (double)(elapsed4 / CLOCKS_PER_SEC);
        cout << "Elapsed: " << timeInSecond4 << "s(" << elapsed4 << "ms)" << "₩n";
}
```

실행결과



시간은 왜인지 모르겠지만 이분법이 느리게 나왔습니다.

하지만 코드의 구성에 따라서 실행속도가 달라질 수 있기 때문에

코드 지연시간이 아닌 수행횟수에 따라 구분해보면 뉴턴법이 가장 빠르다는 것을 알 수 있었습니다.

▣ 수업 성찰(배운점·느낀점)
다양한 방법의 근사근을 구하는 프로그램을 구현 해볼 수 있는 기회가 되어서 좋았습니다.