myNP.h

```
#include "../../include/myNP.h"
```

자신의 폴더에 위치에 있는 myNP.h라는 헤더파일 가져오는 방법

```
#define     PI      3.14159265358979323846264338327950288419716939937510582
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

PI 값 정의 실시

기본 library를 include 시키기

Differentiation

printVec()

입력한 벡터를 출력한다. 출력시 소수점 2자리까지 표현을 실시한다.

```
void printVec(double* _vec, int _row);
```

Parameters

> _vec: 입력할 벡터를 모아 배열로 만든 것
>
> _row: 벡터들을 모은 것의 전체 행들

Example code

```
#include "../../include/myNP.h"

int main(){
    double y[3] = { 1.00, 2.00, 3.00 };
    int m = 3;
    printVec(y, m);
    return 0;
}
```

output

```
Vector[0]=1.00
Vector[1]=2.00
Vector[2]=3.00
```

Warning

- y는 double, m은 int의 형식이어야 한다.
- m은 y의 배열의 크기이다.

gradient1D()

주어진 벡터를 1차 미분을 실시한다.

x좌표와 y좌표의 개수는 각각 m개씩 존재한다.

각 x좌표마다 미분값을 구한다.

1. 첫번째 x좌표의 미분값은 3-point forward를 이용한다.
2. 마지막 x좌표의 미분값은 3-point backward를 이용한다.
3. 나머지 구간의 x좌표의 미분값은 2-point central을 이용한다.

```
void gradient1D(double _x[], double _y[], double dydx[], int m);
```

Parameters

> _x[]: x좌표의 값을 모아 놓은 배열을 의미한다
>
> _y[]: y좌표의 값을 모아 놓은 배열을 의미한다
>
> _dydx[]: 각 x좌표의 미분값을 모아 놓은 것이다
>
> m: 배열의 크기를 의미한다

Example code

```
#include "../../include/myNP.h"

int main(){
    int m = 5;
    double x[5] = { 0 };
    for (int i = 0; i < m; i++) x[i] = 0.2 * i;

    double y[] = { -5.87, -4.23, -2.55, -0.89, 0.67 };
    double  dydx[5] = { 0 };

    gradient1D(x, y, dydx, m);
    printVec(dydx, m);
    return 0;
}
```

output

```
Vector[0]=8.10
Vector[1]=8.30
Vector[2]=8.35
Vector[3]=8.05
Vector[4]=7.55
```

Warning

- x, y, dydx는 double, m은 int의 형식이어야 한다.
- m은 x, y, dydx의 배열의 크기이다.
- x, y, dydx의 배열의 크기는 같아야 한다.

Error Handling

- m은 3이상이어야 한다.

grdientFunc()

주어진 함수를 1번 미분을 한다.

1. 첫번째 x좌표의 미분값은 3-point forward를 이용한다.
2. 마지막 x좌표의 미분값은 3-point backward를 이용한다.
3. 나머지 구간의 x좌표의 미분값은 2-point central을 이용한다.

```
void gradientFunc(double func(const double x), double x[], double dydx[], int
m);
```

Parameters

> func(const doube x): 미분을 할 함수를 의미한다
>
> _x[]: x좌표의 값을 모아 놓은 배열을 의미한다
>
> _dydx[]: 각 x좌표의 미분값을 모아 놓은 것이다
>
> m: 배열의 크기를 의미한다

Example code

```
#include "../../include/myNP.h"
double myFunc(const double x) {
    return  x * x;
}

int main(){
    int m = 5;
    double x[5] = { 0 };
    for (int i = 0; i < m; i++) x[i] = 0.2 * i;

    double dydx[5];
    gradientFunc(myFunc, x, dydx, m);
    printVec(dydx, m);
    return 0;
```

```
}
```

output

```
Vector[0]=0.00
Vector[1]=0.40
Vector[2]=0.80
Vector[3]=1.20
Vector[4]=1.60
```

Warning

- 원하는 함수도 같이 입력해주어야 한다.
- x, dydx는 double, m은 int의 형식이어야 한다.
- m은 x, dydx의 배열의 크기이다.
- x, dydx의 배열의 크기는 같아야 한다.

acceleration()

주어진 함수를 2번 미분을 한다.

y좌표는 주어진 함수에 x좌표에 값을 넣어 구한다.

1. 첫번째 x좌표의 2번 미분값은 4-point forward를 이용한다.
2. 마지막 x좌표의 2번 미분값은 4-point backward를 이용한다.
3. 나머지 구간의 x좌표의 2번 미분값은 3-point central을 이용한다.

```
void acceleration(double x[], double y[], double dy2dx2[], int m);
```

Parameters

> x[]: x좌표의 값을 모아 놓은 배열을 의미한다
>
> y[]: y좌표의 값을 모아 놓은 배열을 의미한다
>
> dy2dx2[]: 각 x좌표의 2번 미분값을 모아 놓은 것이다
>
> m: 배열의 크기를 의미한다

Example code

```
#include "../../include/myNP.h"
double myFunc(const double x) {
    return  x * x*x;
}

int main(){

    int m = 5;
    double x[5] = { 0 };
    for (int i = 0; i < m; i++) x[i] = 0.2 * i;
    double dy2dx2[5];
    double y[5] = { 0 };
    for (int i = 0; i < m; i++) {
```

```
        y[i] = myFunc(x[i]);
    }
    acceleration(x, y, dy2dx2, m);
    printVec(dy2dx2, m);
    return 0;
}
```

output

```
Vector[0]=-0.00
Vector[1]=1.28
Vector[2]=2.40
Vector[3]=3.60
Vector[4]=4.80
```

Warning

- x, y, dy2dx2는 double, m은 int의 형식이어야 한다.
- m은 x, y, dy2dx2의 배열의 크기이다.
- x, y, dy2dx2의 배열의 크기는 같아야 한다.

Error Handling

- m은 4이상이어야 한다