Hydrostatic Table
계산프로그램 사양서

SECTION I. 프로그램 개요

본 프로그램은 사용자 입력값(LBP, Offset Talbe)를 이용하여 여러 흘수에 대한 대상 선박의 몇 가지 유체정역학적 특성값을 나타 낼 수 있도록 하였다.
Hydrostatic Talbe 를 출력하기 위해 유체정역학 계수를 읽어오고 출력 할 수 있는 입출력을 담당하는 File_IO 클래스를 구현하였으며 File_IO 로부터 몇 가지 변수(LBP, Offset Value, number of Waterline, number of station)를 넘겨받아 유체 정역학적 계수값을 연산하는 Program 클래스를 구성하였다. 또한 몇 가지 부가기능을 추가하여 사용자 입력에 대한 확장성을 제공 할 수 있도록 노력하였다.

▣ 주요 특징

- LBP 및 Offset Table 의 순서에 상관없이 값을 읽어 올 수 있도록 구현하였다.
- LBP, lbp, LPP, Length Between Perpendiculars 등 다양한 표현으로 명시하여도 값을 읽어 올 수 있도록 하였다.
- Offset Table 의 가로축과 세로축의 스테이션과 워터라인을 서로 바꾸어도 자동으로 인식하여 계산 될 수 있도록 하였다. (조건 : st 및 wl 을 명시 할 것)
- 기존 Offset.csv 에서는 워터라인 개수와 스테이션 개수를 지정해 주었으나 본 프로그램에서는 Offset Table 로부터 가로열과 세로열의 개수를 카운트하여 그 값을 가져오도록 구현하였다.
- system 의 내장 method 인 RegularExpressions 를 이용하여 사용자 입력값에 콤마(,)가 있을 시 이를 필터링 하도록 하였다. 이를 위하여 다음의 라이브러리를 추가하였다. <using System.Text.RegularExpressions>

SECTION II. 클래스 구현

1. File_IO 클래스

▣ 개요

File_IO 클래스에서는 Offset.csv 의 파일을 읽고 워터라인 개수와 스테이션개수를 Offset Table 에서 직접 카운트한다. 또한 switch 문을 이용한 분기제어를 이용하여 LBP 와 Offset Table 의 입력 순서에 상관없이 변수를 받아 올 수 있도록 하여 입력 형식에 대한 유연성을 높일 수 있도록 하였다. 그리고 Offset Value 를 읽는 과정에서 사용자측의 입력실수에 의한 콤마값을 필터링 할 수 있는 사용자 정의 함수(Split_CSV(string input))를 구현하였다.

Hydrostatic.csv 출력을 위해 main 메서드에서 계산한 값들을 담아 놓은 2 차원 배열 HydrostaticTable 를 넘겨받아 지정된 워터라인 별로 출력할 수 있도록 하였다.

▣ 변수 선언

- **public static double LBP : LBP 값을 저장하는 변수**
- **public static int NumberOfWaterLine : 워터라인 개수를 저장하는 변수**
- **public static int NumberOfStation : 스테이션 개수를 저장하는 변수**
- **Public static double[,] Offset : Offset 테이블을 2 차원 배열로 저장**
- **static string Buff : 불러온 파일을 한 줄씩 읽어 저장하는 변수**
- **static string[] tokens : Buff 에 저장된 배열을 콤마로 나누어 담는 배열**
- **static bool Row_WL : Offset table 가로열이 워터라인 인지 전달하는 변수**
- **public static List<double> Row : Offset 배열의 가로열 값을 저장하는 변수**

- File_IO 클래스에서 csv 파일 Reading 이 끝나고 할당된 일부 변수의 경우 Program 클래스에서 계산하기 위해 값을 넘겨주어야 하므로 접근 한정자로 Public 을 지정하였다.
- 값이 할당되고 각종 배열의 크기 지정이나 반복 횟수조건 지정 등 반복해서 쓰이는 변수의 경우 static 을 이용하여 정적으로 선언하여 메모리상에 상주하도록 하였다.
- **Row_WL 의 경우 Buff 를 읽고 wl 이 포함되어 있으면 1 그렇지 않다면 0 의 값을 가지도록 하여 추후 가로열이 스테이션일 경우 정형화된 계산을 위한 행렬 스왑의 판단 기준이 되는 변수이다. 디폴트 값은 true 이다.**
- List Row 의 경우 offset 의 첫 번째 가로열 값을 읽어서 워터라인 혹은 스테이션 간격 값을 저장해 놓는 리스트를 생성하였다. 추후 워터라인 및 스테이션 간격 값이 필요할 때 Row[1]의 값을 참조하도록 하였다.(단, 모든 입력은 등 간격으로 하여야 함) 이때 아래의 예시로 든 WL 의 문자를 필터링 할 수 있도록 정규표현식을 이용하여 Empty 값으로 Replace 시키도록 하였다.
  ex) Row = [B.L.WL , 1WL , 2WL, 3WL, ...]

▣ 함수 선언

public static string[] Split_CSV(string input)

- string 값을 입력받아 사용자측 실수로 입력된 콤마를 필터링하여 콤마로 나누는 메서드
- 주로 **Buff** 문자열 변수를 입력받아 **tokens** 배열에 할당할 때 사용됨
  ex) tokens = Split_CSV(Buff);

- 워크시트를 이용하여 사용자 측 실수로 콤마 값이 입력되었을 경우 워크시트는 콤마를 포함한 해당 값을 인용부호(“”)로 감싸고 저장하므로 정규표현식(Regular Expressions)을 이용하여 해당 패턴을 검색 후 인용부호로 감싸진 값의 콤마값을 Point 로 Replace 시키고 인용부호를 삭제 하는 과정을 구현하였다.
- 해당 문자열에서 일치하는 패턴을 검색하여 split 하여 분해 한 뒤 합치는 과정을 거치므로 일반화컬렉션 중 List<double>를 이용하여 형식 매개변수를 double 로 제한하여 처리속도 향상을 도모하였다.
- 합치는 과정에서 List 를 이용함으로써 크기를 미지 지정해 주어야 쓸 수 있는 배열의 단점을 회피하여 때에 따라 추가가 간편하도록 하였다.
- Notepad 에서 편집하고 저장 할 경우에 콤마를 입력한 실수에 대해서는 본 함수의 적용이 불가능하다.

```
public static string[] Split_CSV(string input)
        {
            Regex csvSplit = new Regex("(?:^|,)(\"(?:[^\"]+|\"\")*\"|[^,]*)");
            List<string> list = new List<string>();
            string curr = null;
            foreach (Match match in csvSplit.Matches(input))
            {
                curr = match.Value;
                if (0 == curr.Length)
                {
                    list.Add("");
                }
                list.Add(curr.TrimStart(','));
            }
            for (int i = 0; i < list.Count(); i++)
            {
                list[i] = Regex.Replace(list[i], "[,]+\"|\"[,]+", ""); //따옴표 및 콤마 제거
                list[i] = Regex.Replace(list[i], "\"", ""); //콤마만 제거
                list[i] = Regex.Replace(list[i], ",", "."); //오타 난 콤마 치환
                list[i] = Regex.Replace(list[i], @"\.\.", "");
            }
            return list.ToArray<string>();
        }
```

▲ Split_CSV 함수 정의

public static void ReadFile(string _fileName)

- **_fileNmae** 으로 지정된 csv 파일의 Stream 값을 읽어와 ReadLine() 메서드를 이용하여 선박계산에 필요한 각종 값들을 읽어오는 메서드
- sr 을 한줄 씩 읽으면서도 사용자 입력형식의 유연성을 주기위하여 Switch 구문을 이용한 분기제어를 사용하였다.
- 분기 제어의 경우 LBP 읽기 분기와 Offset 읽기 분기로 나누어지며 무반반목분(While(true))을 이용하여 다 읽을 때까지 반복 동작하도록 하여(파일의 끝까지 다 읽으면 ReadLine() 시 null 값이 반환됨) 이 과정을 통해 LBP 와 Offset Table 의 순서에 구애받지 않도록 설계하였다.
- Offset Table 의 크기를 측정하기 위해 가로열을 **Buff** 에 담아 Split_CSV 메서드를 이용하여 나누어진 tokens 의 개수를 가져오도록 구현하였다. 세로열의 경우 한줄 씩 읽으면서 **Buff** 양식이 Offset 배열의 양식과 안 맞을 때까지의 카운트 넘버를 넘겨받도록 하였다. 이 과정에서 정규표현식을 이용한 패턴검색을 통해 첫 번째 줄의 **Buff** 내용 중 WL 이 있다면 **Row_WL 의 값을 1 로 할당하고 그렇지 않다면 0 으로 할당하여 가로열과 세로열을 구분할 수 있도록 하였다.**

```
switch (tokens[0])  //분기제어를 이용한 사용자 입력값 가져오기.
                {
                    case "LBP":
                    case "Lbp":
                    case "lbp":
                    case "LPP":
                    case "Lpp":
                    case "lpp":
                    case "Length Between Perpendiculars":
                    case "length between perpendiculars":
                        LBP = double.Parse(tokens[1]);
                        Console.WriteLine("LBP = {0}", LBP);;
                        break;
                    case "Offset":
                    case "offset":
                        tokens = Split_CSV(Buff);

                        ~~ 중략 ~~
                        break;
                    default:
  Console.WriteLine("{0} is an invalid Offset value. Check the manual.", tokens[0]);
                        break;
                }
```

▲ Switch 구문을 통한 분기제어 부분

public static double[,] Swap(object[,] _Arr)
public static double[,] Swap(double[,] _Arr)

- 본 함수는 double 타입과 object 타입의 데이터 형식을 가진 2 차원 배열의 행과 열을 교환하기 위해 오버라이딩을 이용하여 정의하였다.
- System 내장 메서드 GetLength(0), GetLength(1)를 이용하여 가로열의 크기와 세로열의 크기를 반환받아 가로 세로의 크기가 Reverse 된 2 차원 배열을 생성하고 반복문을 통해 요소들이 할당되도록 하였다.
- double 형을 가진 Array 를 인자로 받는 Swap 메서드는 주로 **Row_WL 의 값이 0 일 경우 호출되며 Swap 된 Offset Value Array 를 반환하도록 구성되었다.**
- object 형을 Array 인자로 받는 Swap 메서드는 주로 모든 계산과정이 끝난 후 결과를 출력하기 위해 각각의 유체정역학적 값들을 가로방향으로 담아두고 출력을 위해 세로방향으로 바꾸기 호출된다. 이 때 유체 정역학적 값(double) 뿐만 아니라 TPC(m), KM(m), LCF(m)등의 문자도 포함하므로 object 형으로 하였다.(string 형으로 할 경우 Offset Value 들의 타입캐스팅이 필요하다)

```
public static double[,] Swap(double[,] _Arr) //offset 스왑용
        {
            double[,] Reverse = new double[_Arr.GetLength(1), _Arr.GetLength(0)];
            for (int j = 0; j<_Arr.GetLength(1); j++)
            {
                for (int i = 0; i < _Arr.GetLength(0); i++)
                {
                    Reverse[j, i] = _Arr[i, j];
                }
            }
            return Reverse;
        }
```

▲ Swap 함수 정의

public static void FileOut(string _Filename)

- **_fileNmae** 으로 지정된 csv 파일을 출력하는 메서드
- 본 프로그램에서는 출력될 내용을 Program 클래스의 HydrostaticTable 라는 2 차원 배열에 할당하였기 때문에 반복문을 이용하여 해당 배열의 모든 요소를 출력할 수 있도록 하였다. 이로써 추후 Data 가 추가 되어도 HydrostaticTable 에 할당하면 되므로 출력 함수의 범용성 높일 수 있도록 하였다.
- 이 때 지정된 흘수에서의 유체정역학적 값만 출력해야 하므로 반복문 사용 시 특정 count 에서는 출력을 skip 할 수 있는 로직을 추가하였다.

```
public static void FileOut(string _Filename)
        {
            StreamWriter sw =
            new StreamWriter(new FileStream(_Filename, FileMode.Create));

            for (int i = 0; i < 13; i++)
            {
                if (i == 1 || i == 2) { continue; } //wl=2~3 스킵

                if (i > 0) { Program.HydrostaticTable[0, i] = i + 1; }

                for (int text = 0; text < Program.
                HydrostaticTable.GetLength(0); text++)
                {
                    sw.Write("{0},", Program.HydrostaticTable[text,i]);
                }
                sw.Write("\n");
            }
            sw.Close();
        }//File Writing 끝
```

▲ 파일출력 함수 정의

2. Program 클래스

▣ 개요

Program 클래스에서는 File_IO 클래스에서 추출된 LBP, Offset Value 및 스테이션 개수, 워터라인 개수를 넘겨받아 각종 선박계산을 수행하도록 하였다. 이 과정에서 다양한 유체정역학적 계수 값들을 워터라인 개수만큼의 크기의 배열을 생성하고 각각 워터라인 값을 인자로 가지는 유체정역학 계수들을 계산하는 메서드들을 구현하였다. 그리고 계산이 완료된 후 결과값을 File_IO 에 간단하게 넘겨 줄 수 있도록 HydrostaticTable 라는 이름을 가진 2 차원 배열을 생성하였다.

▣ 변수 선언

- **static double LBP : LBP 값을 저장하는 변수**
- **static int NumberOfWaterLine : 워터라인 개수를 저장하는 변수**
- **static int NumberOfStation : 스테이션 개수를 저장하는 변수**
- **static double[,] Offset : Offset 테이블을 2 차원 배열로 저장**
- **static double[] WaterPlaneArea : 워터라인 별 수선면적을 저장하는 배열**
- **static double[] StationArea : 워터라인 별 스테이션 면적합을 저장하는 배열**
- **static double[] Volume : 워터라인 별 배수용적을 저장하는 배열**
- **static double[] VCB : 워터라인 별 높이방향 부력중심을 저장하는 배열**
- **static double[] LCB : 워터라인 별 길이방향 부력중심을 저장하는 배열**
- **static double[] TPC : 워터라인 별 1cm 침하 톤수를 저장하는 배열**
- **static double[] LCF : 워터라인 별 부면심의 길이방향 위치를 저장하는 배열**
- **static double[] KM : 워터라인 별 기선-횡메타센터까지의 높이를 저장하는 배열**
- **pulbic static objective[,] HydrostaticTable : 유체정역학적 계수 배열을 한꺼번에 출력하기 위해 정의한 2 차원 배열**

- 값이 할당되고 각종 배열의 크기 지정이나 반복 횟수조건 지정 등 반복해서 쓰이는 변수의 경우 static 을 이용하여 정적으로 선언하여 메모리상에 상주하도록 하였다.
- **HydrostaticTable 배열의 경우 출력 기능이 구현된 File_IO 클래스에서 접근 가능해야 하므로** 접근 한정자로 Public 을 지정하였다.

▣ 함수 선언

static double[] GetLineFromOffset(double[,] _Offset, int _Number, int _01)

- 2 차원 _Offset 배열을 인자로 입력받아 _Number 번째 (행/열)의 값을 1 차원 배열로 반환하는 매서드. 이 때 행/열 의 결정은 _01 인자로부터 결정되며 0 이면 가로, 1 이면 세로로 읽게 된다.

```
static double[] GetLineFromOffset(double[,] _Offset, int _Number, int _01)
{
    double[] Line = new double[_Offset.GetLength(_01)];
    for (int i = 0; i < _Offset.GetLength(_01); i++)
    {
        if(_01 == 1)
        {
            Line[i] = _Offset[_Number, i];
        }
        else if (_01 == 0)
        {
            Line[i] = _Offset[i, _Number];
        }
    }
    return Line;
}
```

▲ GetLineFromOffset 함수 정의

static double[] Simpson_Rule(double[,] _Arr, double _Interval, int _Num_of_Point)
static double[] Simpson_Rule(double[,] _Arr, double _Interval)

- 1 차원 _Arr 배열과 적분간격 인자 _Interval 및 적분 지점 개수 인자 _Num_of_Point 를 입력받아 심슨 적분을 수행하여 적분합을 반환하는 메서드
- 사용상의 편의를 제공하고자 마지막 인자 _Num_of_Point 를 생략하여 사용하게 되면 _Arr 배열의 모든 요소에 대하여 적분합이 되도록 오버라이딩 하였다.

static double[] Calc_Lever(double _startingValue, double _Gap, int _Num)

- 특정 1 차원 배열과의 모멘트 값을 구하기 위하여 모멘트 팔의 배열을 생성시키기 위해 모멘트 시작점 인자 _startingValue 와 간격 _Gap, 그리고 지점 개수 _Num 인자를 받아 등차수열의 일반항 공식을 이용하여 모멘트 팔의 배열을 생성하는 메서드
- 사용상의 편의를 제공하고자 마지막 인자 _startingValue 가 0 이 아니게 되면 특정 지점(ex. 중심선)으로부터의 모멘트 팔을 구할 수 있도록 구현하여 본 과제에서 요구하는 횡방향 KM 뿐만 아니라 길이방향 KM 도 구할 수 있도록 발판을 마련해 놓았다.

```
static double[] Calc_Lever(double _startingValue, double _Gap, int _Num)
        {
            double[] Result = new double[_Num];
            if (_startingValue == 0)  //**********끝단에서의 모멘트 암
            {
                for (int i = 0; i < _Num; i++)
                {
                    Result[i] = _startingValue + i * _Gap; //등차수열 일반항
                }
                return Result;
            }
            else  //**************************** 특정 지점에 대한 모멘트암
            {
                for (int i = 0; i < _Num; i++)  //끝단에서의 모멘트 암
                {
                    Result[i] = i * _Gap - _startingValue; //등차수열 일반항
                }
                return Result;
            }
        }
```

▲ Calc_Lever 함수 정의

static double[] Moment(double[] _DataArr, double[] _LeverArr)

- 특정 1 차원 배열의 모멘트값 계산을 위해 배열의 요소를 순회하며 모멘트 팔과의 곱을 수행하고 그 결과 배열을 반환하는 메서드. 심슨적분을 위해 모멘트 합 값이 아닌 배열을 반환하도록 하였다.
- 이 때 _DataArr 과 _LeverArr 의 배열 크기는 달라도 곱셈이 가능하며 계산 기준은 _DataArr 크기에 맞춰 행렬곱이 이루어진다.

```
static double[] Moment(double[] _DataArr, double[] _LeverArr)
        {
            double[] Result = new double[_DataArr.Length];
            for (int i = 0; i < _DataArr.Length; i++ )
            {
                Result[i] = _DataArr[i] * _LeverArr[i];
            }
            return Result;
        }
```

▲ Moment 함수 정의

static double Calc_TPC(int _WL)

- WL 인자로 받은 특정 워터라인에서의 수선면적 값을 가지고 TPC 를 계산하는 메서드

```
static double Calc_TPC(int _WL)
        {
            double TPC;
            TPC = WaterPlaneArea[_WL] * 1.025 / 100;
            return TPC;
        }
```

▲ Calc_TPC 함수 정의

static double Calc_VCB(int _WL)

- _WL 인자로 받은 특정 워터라인에서의 높이방향 부력중심을 계산하는 메서드
- 모멘트 팔 값을 내부 배열 double[] Lever 에 저장하고 워터라인 간격 값을 Row[1]을 통해 읽어와 Lever 배열을 생성한다. 그리고 모멘트 곱 및 심슨적분 메서드를 내부에서 호출하여 VCB 값을 반환한다.

```
static double Calc_VCB(int _WL)
        {
            double[] Lever = new double [NumberOfWaterLine];
            Lever = Calc_Lever(0, File_IO.Row[1], NumberOfWaterLine);
            double[] Result = Moment(WaterPlaneArea, Lever);
            double VCB = Simpson_Rule(Result, File_IO.Row[1], _WL + 1)
                                        / Volume[_WL];
            return VCB;
        }
```

▲ Calc_VCB 함수 정의

static double Calc_LCB(int _WL)

- _WL 인자로 받은 특정 워터라인 에서의 길이방향 부력중심을 계산하는 메서드
- _WL 인자로 받은 특정 워터라인 에서의 각 스테이션 별 면적을 심슨적분을 통해 스테이션의 개수만큼의 크기를 가진 StationArea_WL 이라는 내부 배열을 생성하여 저장한다.
- 모멘트 팔 값을 내부 배열 double[] Lever 에 생성하며 이때 배열의 크기는 인자로 받은 _WL 만큼의 크기로 지정하고 스테이션 간격 값을 읽어와 Lever 배열을 생성한다.

- 그리고 StationArea_WL 과 Lever 간의 모멘트 곱 및 심슨적분 메서드를 내부에서 호출하여 LCB 값을 반환한다.

```
static double Calc_LCB(int _WL)
        {
            double[] StationArea_WL = new double[NumberOfStation];
            for (int i = 0; i < NumberOfStation; i++)
            {
                double[] Line = GetLineFromOffset(Offset, i, 0);
                StationArea_WL[i] = 2*Simpson_Rule(Line, File_IO.Row[1],_WL+1);
            }
            double[] Lever = new double[NumberOfStation];
            Lever = Calc_Lever(0, LBP / (NumberOfStation - 1),
                    NumberOfStation);
            double[] Result = Moment(StationArea_WL, Lever);
            double LCB = Simpson_Rule(Result, LBP / (NumberOfStation - 1),
                        NumberOfStation) / Volume[_WL];
            return LCB;
        }
```

▲ Calc_LCB 함수 정의

static double Calc_LCF(int _WL)

- _WL 인자로 받은 특정 워터라인에서의 부면심의 길이방향 위치를 계산하는 메서드
- Offset 배열로부터 스테이션의 개수만큼의 크기를 가진 Line 이라는 내부 배열을 생성하여 Offset 의 반폭값을 전폭값으로 확장하여 저장한다.
- 모멘트 팔 값을 내부 배열 double[] Lever 에 저장하고 스테이션 간격 값을 읽어와 Lever 배열을 생성한다. 그리고 모멘트 곱 및 심슨적분 메서드를 내부에서 호출하여 LCF 값을 반환한다.

```
static double Calc_LCF(int _WL)
        {
            double[] Lever = new double[NumberOfStation];
            Lever = Calc_Lever(0, LBP / (NumberOfStation - 1), NumberOfStation);
            double[] Line = GetLineFromOffset(Offset, _WL, 1);  //WL에 따라 오프셋 값을 다르게 읽을 예정. -> 심슨룰시 NumberOfStation 써도 무방.
            for (int i =0; i< Line.Length;i++)  //반폭을 전폭으로 확장
            {
                Line[i] = 2 * Line[i];
            }
            double[] Result = Moment(Line,Lever);
            double LCF = Simpson_Rule(Result, LBP / (NumberOfStation - 1), NumberOfStation) / WaterPlaneArea[_WL];
            return LCF;
        }
```

▲ Calc_LCF 함수 정의

static double Calc_BM(int _WL)

- _WL 인자로 받은 특정 워터라인에서의 기선으로부터 횡메타센터까지 높이를 계산하는 메서드
- Offset 배열로부터 스테이션의 개수만큼의 크기를 가진 Line 이라는 내부 배열을 생성하여 Offset 의 반폭값 읽어들인 후 System 의 내부 메서드 Pow 를 이용하여 반폭의 3 승을 계산하여 Half_Breadth_POW3 배열에 할당한다.(관성모멘트 계산용)
- Half_Breadth_POW3 과 스테이션 간격 값을 읽어와 심슨적분 메서드를 내부에서 호출하여 BM 값을 반환한다.

```
static double Calc_BM(int _WL)
        {
            double[] Half_Breadth_POW3 = new double[NumberOfStation];
            //반폭의 3승

            double[] Helf_Breadth = GetLineFromOffset(Offset, _WL, 1);
//WL에 따라 오프셋 값을 다르게 읽을 예정. -> 심슨룰시 NumberOfStation 써도 무방.
            for (int i = 0; i < NumberOfStation; i++)
            //반폭을 전폭으로 확장 + 반폭 3승(수선면 관성모멘트 유도용)
            {
                Half_Breadth_POW3[i] = Math.Pow(Helf_Breadth[i], 3);
                //System.Math 내장 Method.
            }
            double BM = 2 * Simpson_Rule(Half_Breadth_POW3,
            LBP / (NumberOfStation - 1), NumberOfStation) / (3 * Volume[_WL]);
            return BM;
        }
```

▲ Calc_BM 함수 정의

static void Main(string[] args)

- 본 프로그램에서 사용 할 Offset Data 가 있는 파일 명을 지정하여 계산에 필요한 데이터들을 가져오는 과정 및 계산하는 과정 File_IO 클래스를 이용하여 출력하는 과정 등 선박계산 전반의 모든 과정이 담겨있음
- 메인 메서드에서 대상 선박의 수선면적 계산과 스테이션 면적 계산, 볼륨계산이 이루어지며 이 과정은 별도의 메서드로 구현하지 않고 메인 메서드에 포함시켰다. 그리고 그 결과 값을 static 선언을 한 배열에 정적으로 할당하여 각종 선박계산 메서드에서 바로 호출할 수 있도록 구현하였다.
- 각종 선박계산 메서드에서는 출력하고자 원하는 워터라인의 값을 입력하면 해당 메서드에서 결과값을 반환하는 형식으로 구현하였으며 반복문을 통해 모든 워터라인에서의 유체정역학적 계수 값들을 계산하도록 하였다.(단, 심슨 적분이 불가능한 3번째 이하 지점의 워터라인에 대해서는 계산을 수행하지 않았다)