```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RealTimeBridge
{

    class RealTime9_13
    {
        // Input Parameters are valued from db
        struct InputParams
        {
            public double C_D0WA; // “Drag planking” coefficient (D=0) → CD0wa (parameter)
            public double C_D1WA; // “Drag planking” coefficient (D=1) → CD1wa (parameter)
            public int D;          // The presence (D=1) or not (D=0) of debris on the stack base
            public double IDRO1;  // Height of the water → [IDRO1]
            public double IDRO2;
            public double SONAR1; // Height of the bottom of the river -> [SONAR1] : I assume SONAR1 is calculated before and is available as a parameter to this 
section
            public double bottom_ref; // The bottom_ref is a parameter
            public double c; public double D_Pile; // c and Dpile are parameters
            public double Water_Density; // Water density → p (parameter)
            public double Water_Speed; // Water speed → Vwater (see point before) from the previous section
            public double Beta_A;
        }
        struct Result_WaterThrust
        {
            public double S_water;
            public double As;
        }
        //  To calculate the water thrust should call this function with input parameters and it returnes the Result Structure
        public Result_WaterThrust Water_Thrust(InputParams params_input)
        {
            double C_DWA = 1; // “Drag planking” coefficient
                                                                            //اگر مقدار دهی نگردد برنامه خطا میگیرد به همین دلیل مقدار دهی شده ولی در طول برنامه حتما مقدار صحیح خود را بدست میآورد
            double h_s = 1;
            double b_s ;
            double As = 0;
            double S_Water = 0;

            Result_WaterThrust Result = new Result_WaterThrust();

            if (params_input.SONAR1 <= params_input.bottom_ref)
            {
                h_s = params_input.IDRO2 - params_input.bottom_ref;
            }
            else if (params_input.SONAR1 > params_input.bottom_ref)
            {
                h_s = params_input.IDRO2 - params_input.SONAR1;
            }
            else if (params_input.SONAR1 == params_input.bottom_ref) // This criteria was not found in the document, hence we assume that is equal to the first 
criteria
            {
                h_s = params_input.IDRO2 - params_input.bottom_ref;
            }
            if (params_input.D == 0)
            {
                C_DWA = params_input.C_D0WA;
                b_s = params_input.c;
                As = b_s * h_s;
                S_Water = (0.5) * C_DWA * params_input.Water_Density * As * (params_input.Water_Speed * params_input.Water_Speed);
            }
            else if (params_input.D == 1)
            {
                C_DWA = params_input.C_D1WA;
                b_s = 2 * params_input.D_Pile;
                As = b_s * h_s;
                S_Water = (0.5) * C_DWA * params_input.Water_Density * As * params_input.Beta_A * (params_input.Water_Speed * params_input.Water_Speed);
            }
            Result.S_water = S_Water;
            Result.As = As;
            return Result;
        }

        // To Calculate the Weight of the structure
        struct InputParamsWeight
        {
            public double P_p1; // Load on the stack → PP (parameter)
            public double P_pu; // Weight of pulvino → Ppu (parameter)
            public double P_tp; // Weight of trunk pylon → Ptp (parameter)
            public double P_b; // Weight of the beam → Pb (parameter)
            public double P_p2; // Weight per meter of the single pylon → Pp (parameter)
            public double h_beam; // Height of the beam → hbeam (parametro)
            public double SONAR1; // Height of stack portion exposed → to be evaluated with [SONAR1] : I assume SONAR1 is calculated before and is available as a 
parameter to this section
        }

        public double Weight_of_Structure(InputParamsWeight params_input)
        {
            double PP_structure = params_input.P_p1 + ((2 * params_input.P_pu + 6 * params_input.P_tp) + 6 * (params_input.P_p2 * (params_input.h_beam - 
params_input.SONAR1)));
            return PP_structure;

        }
    }
}
```