```cpp
#include "PID.h"
#include<iostream>
#include<vector>
#include<string>
P_ctrl::P_ctrl(const float& P) :
    m_P(P)
{/*constructor code*/}

float P_ctrl::calc_ctrl_P( float error) {
    return (float)(m_P * error);
    std::cout<<"hello i am in pid"<<std::endl;
}

PI_ctrl::PI_ctrl(const float& P,
                 const float& I,
                 const float& dT) :
    P_ctrl(P),
    m_I(I),
    m_integrator(0.0),
    m_dT(dT)
{/*constructor code*/}

float PI_ctrl::calc_ctrl_I( float error) {
    float m_dT=1;
    m_integrator += m_I * error * m_dT;
    return m_integrator;
}

float PI_ctrl::calc_ctrl_PI( float error) {
    return PI_ctrl::calc_ctrl_P(error) +
           PI_ctrl::calc_ctrl_I(error);
}

PID_ctrl::PID_ctrl(float P, float I, float D, float dT, std::string ID)
    :
    PI_ctrl(P, I, dT), // PI_ctrl will automatically construct inherent P controller
    m_D(D),
    m_p_error(0.0)

{/*constructor code*/}


float PID_ctrl::calc_ctrl_D( float error)
{
    float m_dT=1;
    float  ret_val = m_D*(error) / m_dT;
    m_p_error = error;
    return ret_val;
}

float PID_ctrl::calc_ctrl_PID( float error)
{
    return PI_ctrl::calc_ctrl_PI(error) +
           PID_ctrl::calc_ctrl_D(error);
}

std::string PID_ctrl::getName(){
return newName;
}
std::string PID_ctrl::setName(std::string name){
newName=name;
}
```