Help

```
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
#else

/// \ file numint.hpp
/// \ brief template class for numerical integration
/// \ author  M. Ciuca (MathFi, ENPC)
/// \ note (C) Copyright Premia 8 - 2006, under Premia 8 Software license
//
//  Use, modification and distribution are subject to the
//  Premia 8 Software license

#ifndef _NUMINT_H
#define _NUMINT_H

#include <stdexcept>

using namespace std;

template<class T> class NumInt
{
  T *_pt;
  double _integrationStep;
public:
  NumInt(T *pt = 0, double integrationStep = 1.e-03)  throw(logic_error) :
    _pt(pt), _integrationStep(integrationStep)
  {
    if (integrationStep <= 0)
      throw logic_error("Numerical integration: the integration step must be"
                        " strictly positive!");
  }

  void setPt(T *pt)
  {
    _pt = pt;
  }

  void setIntegrationStep(double integrationStep) throw(logic_error)
  {
    if (integrationStep <= 0)
```

```
      throw logic_error("Numerical integration: the integration step must be"
                        " strictly positive!");
    _integrationStep = integrationStep;
  }

  typedef double (T::*PtrF)(double);
  double compute(PtrF f, double a, double b);
};

template<class T>
double NumInt<T>::compute(PtrF f, double a, double b)
{
  if (a == b)
    return 0.;
  if (a > b)
    return - compute(f, b, a);


  double h = _integrationStep; //RIEMANN_NUM_INTEGR_PRECISION;
  int step_no = (int) floor((b - a) / h) ;
  double xi = a;
  double sum = 0;
  for (int i = 1; i < (step_no + 1); i++)
    {
      if (xi + h > b)
        return sum;
      sum += (_pt->*f)(xi + h) * h;
      xi += h;
    }
  return sum;
}




#endif
#endif //PremiaCurrentVersion
```