Help

```
#include  "bs1d_std.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2014+2) //The "#els
static int CHK_OPT(TR_ChangPalmer)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(TR_ChangPalmer)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

int ChangPalmer(int am, double s0, NumFunc_1 *pf, double t, double r, double div
                int N, double *ptprice, double *ptdelta)
{
  double  h, mu, u, d, scan, p, q, lowerstock, stock, R;
  double  *P;
  int     i, j;
  double lambda;
  int j0;
  double u0, d0;
  double K0;

  /*Strike*/
  K0 = pf->Par[0].Val.V_PDOUBLE;

  /*Price, intrisic value arrays*/
  P = (double *)malloc((N + 1) * sizeof(double));

  /*Up and Down factors*/
  h = t / (double)N;
  u0 = exp(sigma * sqrt(h));
  d0 = 1. / u0;
  j0 = (int)((log(K0 / s0) - N * log(d0)) / (log(u0 / d0)));
  lambda = (log(K0 / s0) - (2.*j0 - 1 - N) * sigma * sqrt(h)) / (N * sigma * sig
  lambda = (2.*((log(K0 / s0) - N * log(d0)) / (log(u0 / d0)) - j0) + 1) / (sigm
  mu = lambda * sigma * sigma;
```

```
u = u0;
d = 1. / u;
u *= exp(mu * h);
d *= exp(mu * h);
scan = u / d;

/*Discounted Probability*/
R = exp((r - divid) * h);
p = (R - d) / (u - d);
q = 1.0 - p;
p *= exp(-r * h);
q *= exp(-r * h);

/*Terminal Values*/
lowerstock = s0;
for (i = 0; i < N; i++)
  lowerstock *= d;

stock = lowerstock;
for (i = 0; i <= N; i++)
  {
    P[i] = (pf->Compute)(pf->Par, stock);
    stock *= scan;
  }

/*Backward Resolution*/
for (i = N; i > 1; i--)
  {
    lowerstock /= d;
    stock = lowerstock;
    for (j = 0; j < i; j++)
      {
        P[j] = q * P[j] + p * P[j + 1];
        if (am)
          {
            P[j] = MAX((pf->Compute)(pf->Par, stock), P[j]);
          }
        stock *= scan;
      }

  }
```

```
  lowerstock /= d;
  stock = lowerstock;

  /*Delta*/
  *ptdelta = (P[1] - P[0]) / (stock * u - stock * d);

  /*First time step*/
  P[0] = q * P[0] + p * P[1];
  if (am)
    {
      P[0] = MAX((pf->Compute)(pf->Par, s0), P[0]);
    }

  /*Price*/
  *ptprice = P[0];

  /*Memory desallocation*/
  free(P);

  return OK;

}

int CALC(TR_ChangPalmer)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;
  double r, divid;

  r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
  divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);
  return ChangPalmer(ptOpt->EuOrAm.Val.V_BOOL, ptMod->S0.Val.V_PDOUBLE, ptOpt->P
}


static int CHK_OPT(TR_ChangPalmer)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "PutAmer") == 0) || (strcmp(((Option *)Opt)
    return OK;
```

```
  return WRONG;
}

#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->init = 1;

      Met->Par[0].Val.V_INT2 = 100;
    }

  return OK;
}

PricingMethod MET(TR_ChangPalmer) =
{
  "TR_ChangPalmer",
  {{"StepNumber", INT2, {100}, ALLOW}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(TR_ChangPalmer),
  {{"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PR
  CHK_OPT(TR_ChangPalmer),
  CHK_tree,
  MET(Init)
};
```