

[Help](#)

```

#define WITH_formula 1
#include "bs1d_lim.h"

static int CallDownOut_ReinerRubinstein(double s, double k, double l, double reb
{
    int phi, eta;
    double A, B, C, D, E, F;
    double dA, dB, dC, dD, dE, dF;

    phi = 1;
    eta = 1;
    formula(s, k, r, divid, sigma, t, l, rebate, phi, eta, &A, &B, &C, &D, &E, &F,
            &dA, &dB, &dC, &dD, &dE, &dF);
    if (k >= l)
    {
        *ptprice = A - C + F;
        *ptdelta = dA - dC + dF;
    }
    else
    {
        *ptprice = B - D + F;
        *ptdelta = dB - dD + dF;
    }

    return OK;
}

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

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
    divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);
    limit = ((ptOpt->Limit.Val.V_NUMFUNC_1)->Compute)((ptOpt->Limit.Val.V_NUMFUNC_
    rebate = ((ptOpt->Rebate.Val.V_NUMFUNC_1)->Compute)((ptOpt->Rebate.Val.V_NUMFU

    return CallDownOut_ReinerRubinstein(ptMod->S0.Val.V_PDOUBLE, (ptOpt->PayOff.Val

```

```

limit, rebate, ptOpt->Maturity.Val.V_DATE
&(Met->Res[0].Val.V_DOUBLE), &(Met->Res[1]
}

static int CHK_OPT(CF_CallDownOut)(void *Opt, void *Mod)
{
    Option *ptOpt = (Option *)Opt;
    TYPEOPT *opt = (TYPEOPT *) (ptOpt->TypeOpt);

    if ((opt->Parisian).Val.V_BOOL == FALSE)
        return strcmp(((Option *)Opt)->Name, "CallDownOutEuro");
    return WRONG;
}

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

PricingMethod MET(CF_CallDownOut) =
{
    "CF_CallDownOut",
    {" ", PREMIA_NULLTYPE, {0}, FORBID}},
    CALC(CF_CallDownOut),
    {"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PR
    CHK_OPT(CF_CallDownOut),
    CHK_ok,
    MET(Init)
} ;

```