

[Help](#)

```
#include "doublim.h"

static NumFunc_1 put =
{
    Put,
    {"Strike", PDOUBLE, {100}, ALLOW, SETABLE}, {" ", PREMIA_NULLTYPE, {0}, FORBI
    CHK_call
};

static NumFunc_1 const_Re =
{
    Const,
    {"Const Rebate", DOUBLE, {100}, ALLOW, SETABLE}, {" ", PREMIA_NULLTYPE, {0},
    CHK_ok
};

static NumFunc_1 constdoublestep_Low =
{
    Const,
    { {"Lower Limit 1", PDOUBLE, {100}, ALLOW, SETABLE},
      {"Lower Limit 2", PDOUBLE, {0}, ALLOW, SETABLE},
      {" ", PREMIA_NULLTYPE, {0}, FORBID, SETABLE}
    },
    CHK_call
};

static NumFunc_1 constdoublestep_Up =
{
    Const,
    { {"Upper Limit 1", PDOUBLE, {100}, ALLOW, SETABLE},
      {"Upper Limit 2", PDOUBLE, {0}, ALLOW, SETABLE},
      {" ", PREMIA_NULLTYPE, {0}, FORBID, SETABLE}
    },
    CHK_call
};

static TYPEOPT TwoDoubleStepPutOutEuro =
{
```

```

/*PayOff*/          {"PayOff", NUMFUNC_1, {0}, FORBID, SETABLE},
/*Rebate*/          {"Const Rebate", NUMFUNC_1, {0}, FORBID, SETABLE},
/*LowerLimit*/      {"Lower Limit", NUMFUNC_1, {0}, FORBID, SETABLE},
/*UpperLimit*/      {"Upper Limit", NUMFUNC_1, {0}, FORBID, SETABLE},
/*Maturity*/        {"Maturity", DATE, {0}, ALLOW, SETABLE},
/*DateBetween0andMaturity*/ {"DateBetween0andMaturity", DATE, {0}, ALLOW, SETAB
/*OutOrIn*/         {"Out", BOOL, {OUT}, FORBID, UNSETABLE},
/*Parisian*/        {"Parisian", BOOL, {FALSE}, FORBID, UNSETABLE},
/*TwoDoubleStep*/   {"TwoDoubleStep", BOOL, {TRUE}, FORBID, UNSETABLE},
/*RebNo*/           {"Rebate", BOOL, {REBATE}, FORBID, UNSETABLE},
/*EuOrAm*/          {"Euro", BOOL, {EURO}, FORBID, UNSETABLE}
};

```

```

/* For double parisian options, the same delay must be used for lower and
 * upper barriers. The value of the delay is deduced from the one associated
 * to the Lower barrier Numfunc */

```

```

static int OPT(Init)(Option *opt, Model *mod)
{
    TYPEOPT *pt = (TYPEOPT *) (opt->TypeOpt);

    if (opt->init == 0)
    {
        opt->init = 1;
        opt->nvar = 11;
        opt->nvar_setable = 5;

        pt->PayOff.Val.V_NUMFUNC_1 = &put;
        pt->Rebate.Val.V_NUMFUNC_1 = &const_Re;
        pt->LowerLimit.Val.V_NUMFUNC_1 = &constdoublestep_Low;
        pt->UpperLimit.Val.V_NUMFUNC_1 = &constdoublestep_Up;

        (pt->EuOrAm).Val.V_BOOL = EURO;
        (pt->OutOrIn).Val.V_BOOL = OUT;
        (pt->RebOrNo).Val.V_BOOL = NOREBATE;
        (pt->Maturity).Val.V_DATE = 0.5;
        (pt->DateBetween0andMaturity).Val.V_DATE = 0.25;
        (pt->PayOff.Val.V_NUMFUNC_1)->Par[0].Val.V_PDOUBLE = 100.0;
        (pt->Rebate.Val.V_NUMFUNC_1)->Par[0].Val.V_PDOUBLE = 0.0;
        (pt->LowerLimit.Val.V_NUMFUNC_1)->Par[0].Val.V_PDOUBLE = 70.0;
        (pt->LowerLimit.Val.V_NUMFUNC_1)->Par[1].Val.V_SPDOUBLE = 75.0;
    }
}

```

```
        (pt->UpperLimit.Val.V_NUMFUNC_1)->Par[0].Val.V_PDOUBLE = 130.0;
        (pt->UpperLimit.Val.V_NUMFUNC_1)->Par[1].Val.V_SPDOUBLE = 125.0;
    }
    if ((pt->Rebate.Val.V_NUMFUNC_1)->Par[0].Val.V_PDOUBLE > 0.0)
        (pt->RebOrNo).Val.V_BOOL = REBATE;

    return OK;
}

MAKEOPT(TwoDoubleStepPutOutEuro);
```