# ○「　MQL4 用 DLL の作り方・呼び方　」　　＜VC++2010 バージョン＞

前書き；　なぜ DLL を作るか？！、どうやって作って、どうやって呼び出すのか？？

・MQL4 では、関数を C 言語と略同じ文法のコードで組めますので、改めて DLL を使う意味があるのか疑問に思うところです。
しかしながら、C 言語の機能と過去の遺産を利用すると、実は MQL4 では組めない機能を発揮することが出来ます。

・MT4 には DLL を作るための C 言語「サンプル・コード」が「experts\samples」に提供されていますが、初心者にとっては「これをどうすりゃいいのさ？」と言うのが本音でしょう。(不親切！ですよね)
小生は C 言語に関しては「初心者以上、中級未満」、C++言語に至っては「超初心者」ですが、以前に他のトレードシステム用に悪戦苦闘の末に「自己流」で DLL 開発手法を確立した経緯があり、今回は MQL4 用 DLL 開発にも適用してみました。

・本稿では、簡単な DLL を小生の「自己流」で開発する方法を解説します、また C++コンパイラは「VC++2010 Express Edition」を前提としました。
小生は「VC++2008 Express Edition」の方が使い慣れているのですが、現時点 (2011 年) でダウンロード可能なのは「VC++2010 Express Edition」のためです。

・諸兄の大いなる活用を期待します

---

＜目次＞



---

# １章：基礎知識

・MQL4 では C++言語で作成した DLL（ダイナミック・リンク・ライブラリ）をインポートして、すなわち DLL 中の関数を呼出すことで機能拡張することができます。本稿では、C 言語および C++言語の基礎的（初歩的）な知識のみを前提として、DLL の開発方法を解説しています。

## １．MQL4 と DLL のリンク方法

### １－１．CTL から DLL を呼出す

MQL4 で DLL 中のを利用するには「#import」宣言を使います。

書式；

```
#import "DLL 名称"
    変数型　関数名(引数)；
#import
```

解説；

・DLL は「experts\liblaries」フォルダ内に入れます。
・関数名は、DLL 中で宣言した名称をそのまま使うことが必須です。
・import 宣言をした DLL の関数は、他の MQL4 関数と同様に扱うことができます。
・import 宣言（DLL）は指標（indicator)、EA（strategy）およびスクリプト（script）で使用することができます。

### １－２．MQL4 と DLL 間の「変数型」対応

・MQL4 の文字コードは「ANSI（ASCII)」です。
一方、MQL5 では「Unicode」が使われているそうですが、現時点では小生は確認していません。言い換えると、MQL4 は C 言語、MQL5 は C++言語の文字コードが使えると考えれば良いと考えます。

## ２．DLL 開発の基礎

### ２－１．DLL 開発環境

・最低限、C++コンパイラが必要です。フリー版（無償）のものが機能限定ながら多々存在しており、DLL 開発を試すには無償（フリー）版のものでも充分です。
本稿で紹介する DLL 作成方法は、取扱いが簡単で、また無償でダウンロードできる「VC++ 2010 Express Edition」を使った方法で解説しますが、他のコンパイラでも基本は同じはずです。（小生は確認していません）

※「VC++ 2010」はマイクロソフト株式会社の商標です
※VC++無償版で開発したソフトの販売は禁止されています、販売用の DLL を VC++で制作する場合は有償版を使用してください。（マナーは守りましょう）

・C++コンパイラ無償版のインストール方法については、専門書やネット上に情報が溢れていますので、そちらを参照ください。
（インストール後にハローワールドなどの表示ソフト作成により正常動作を確認してください）
・本稿では、正常にインストールされたものとして記述を続けます。
※本稿で紹介し、無償ダウンロード提供可能なDLLは「Windows XP以降のOS」上で動作いたします。(VC++2010 Express Editionの制限事項)

## ２－２．開発の基本手順

DLL開発手順は、下記の「４ステップ」手順を踏むことが効率的であり推奨します。

ステップ１
・Win上で動作する「１本につながったプログラム」を作成・・Debugモード
Debug版で動作確認

ステップ２
・Win上で動作する「main()」部と「DLL」部に分割・・Debugモード
［手順Ａ］・・DLL化する関数部分の切り出し
［手順Ｂ］・・DLL呼出側main()部のまとめ
［手順Ｃ］・・Debug版で組合せ動作確認

ステップ３
・Win版「.exe」とDLL（Release版）を作成し動作確認・・Releaseモード
［手順Ａ］・・DLL化する関数部分のRelese版をビルド
［手順Ｂ］・・DLL呼出側main()部のRelease版実行ファイル「.exe」をビルド
［手順Ｃ］・・上記のRelease版で組合せ動作確認

ステップ４
・「MQL4（呼出側）コード」と「DLL（Release版)」による動作確認

## ２－３．頒布方法

・VC++ 2010 Express Editionで開発したDLLの頒布と動作環境について；
外部頒布（Release）可能なDLLには下記の２種類があります。
① CLRアプリ
② Win32アプリ
本書で解説するDLLは「Win32アプリ」となります。
・Release版が頒布された側の必要な動作環境；
①「CLRアプリ」ではランタイムをインストールすることが必要になります
②「Win32アプリ」の場合は、開発時の設定が適切であれば、
ランタイム等をインストールする必要はありません。

# ２章：DLL を作る・呼ぶ

［解説内容］

・MQL4 から DLL に渡すデータ型としては、「数値、配列、文字、論理値」が可能ですが、本稿では最も基本的で応用範囲の広い「数値」に限定して解説します。
（「配列、文字、論理値」については別途解説予定）

［解説上の注意点］

・C、C++言語の記述法や DLL の作成方法は色々ありますが、本稿では多少本来の記述方法・使い方から外れても、初心者に最も判りやすいと思われる下記方法を採用しました。
① C++のソースコード等はエディタ等で事前に作成済みであると仮定しました。
VC++ 2010 を立ち上げてから、新規に作成するのが本来の方法ですが、
判り易さを優先したため。
②自前のヘッダファイル等は作らず、全てメインのコード中に記述する方法で解説します。（ソースコードを読み易くするため）

・本稿では、C 言語・C++言語についてのごく初歩的な知識のみを前提にしていますが、これらについてさえ不明な場合は、他の参考書・ネット上の参考情報を参照ください。（多すぎる程の情報があります）

［解説画面］

・以下の解説では、本稿記載時点で最新バージョンである VC++2010 Express Edition をダウンロードしたときのデフォルト画面で説明しますが、一つ前のバージョンである VC++2008 Express Edition を用いた場合も略同一操作になります。
VC++2010 の［ツール］―［設定］でデフォルトの「基本設定」から、「上級者用の設定」に変更すると、VC++2008 の画面（メニュー表示）と殆ど同一になります。
小生は、VC++2008 の画面（VC++2010；上級者用の設定）に慣れることを推奨します。

## １．数値編；　受け渡すデータが数値の場合

DLL 仕様：４つの数値（double）を引数として受け取り、その平均値（数値）を返す

### １－１．ステップ１；１本のプログラムを作成

・まず Debug モードで、Win 上で動く１本につながったプログラムを作成します。

（１）下記ソースコード「heikinchi.cpp」をエディタ等で事前に作成準備し、
適当なホルダ中に入れる

```
//ファイル名「heikinchi.cpp」
//インクルード・ヘッダ類
#define WIN32_LEAN_AND_MEAN //無くてもＯＫ
#include <windows.h>
#include <stdio.h>
#include <conio.h> // _getch()を使用するのに必要

//プロトタイプ宣言
double heikin(double, double, double, double) ;

int main(void)
{

        double open,high,low,close,kaitou;
        open=2.0;
        high=3.5;
        low=1.5;
        close=3.0;

        kaitou=heikin(open,high,low,close);

        printf("「= 2.5」が表示されたらＯＫです¥n ");
        printf("平均値＝  %lf¥n",kaitou);

        _getch( ) ; // キーのどれかが押されるまで終了しないために設定

        return(0);
}

//・・・ここから下が DLL 化対象・・・

double heikin(double a,double b,double c,double d){

        double kekka;
        kekka=(1.0/4.0)*(a+b+c+d);
        //「kekka=(1/4)*(a+b+c+d);」だと「0」になる！
        //printf("関数側での平均＝%lf¥n",kekka);//デバッグ用に使った
        return(kekka);

}
```

※補足説明１；

以下の形に配置しておくのがポイントです

インクルードするヘッダ類

プロトタイプ宣言・・main()の中で動作確認する関数（DLL 化対象）

main(){　・・・　}

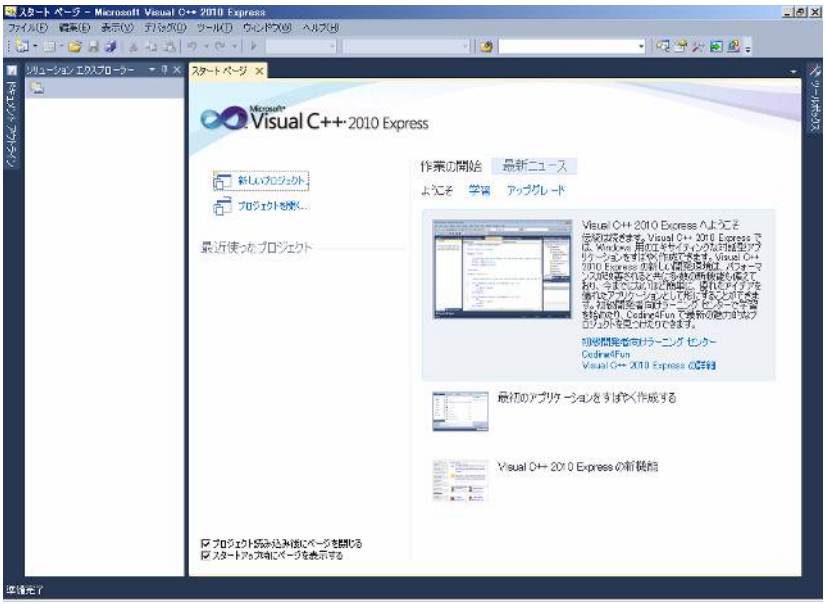
関数郡・・DLL 化対象

これは動作確認後に、main()部と DLL 部（関数郡）に容易に分離可能とするためです。

※補足説明２；

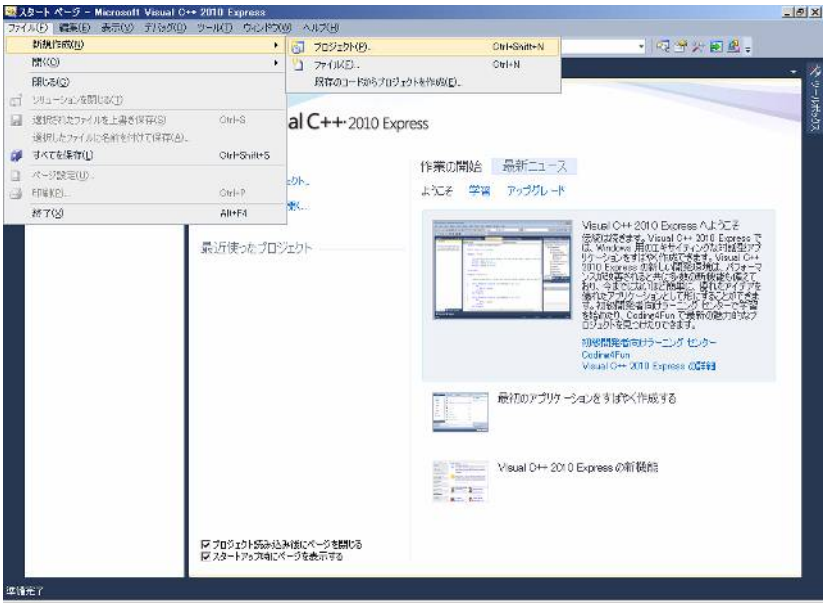
C++言語コンパイラでは、C 言語表記も使えるので慣れている表記法を使いましょう

（２）VC++ 2010 を立ち上げる




（３）「新規プロジェクト；heikinchi」を作成する

［ファイル］―［新規作成］―［プロジェクト］⇒

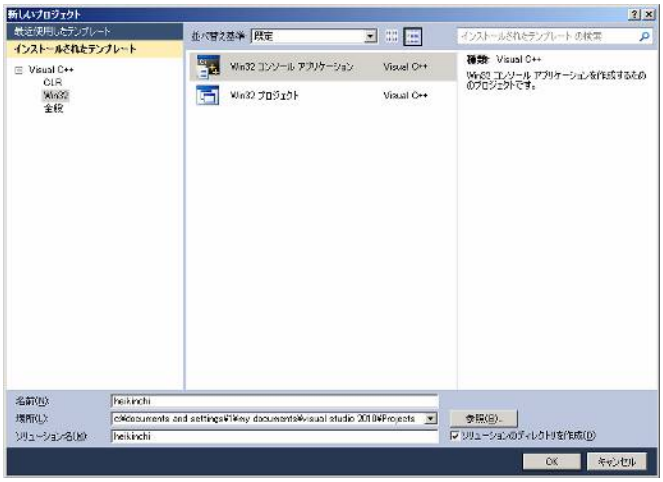



〈1〉［新しいプロジェクト］設定画面

・「インストールされたテンプレート：Win32」

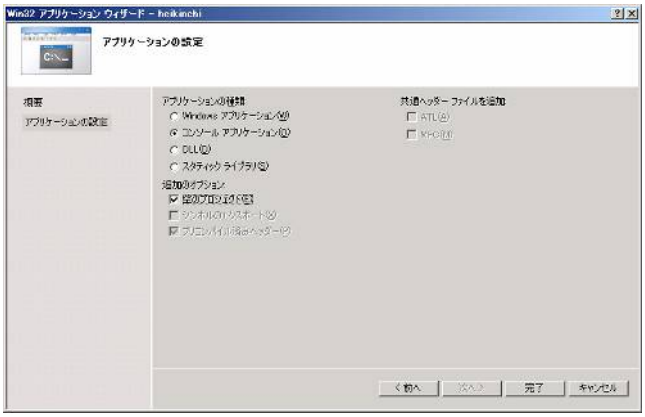
⇒「Win32 コンソールアプリケーション」

・「名前：heikinchi 」を打ち込む　⇒［OK］




［次へ>］をクリック

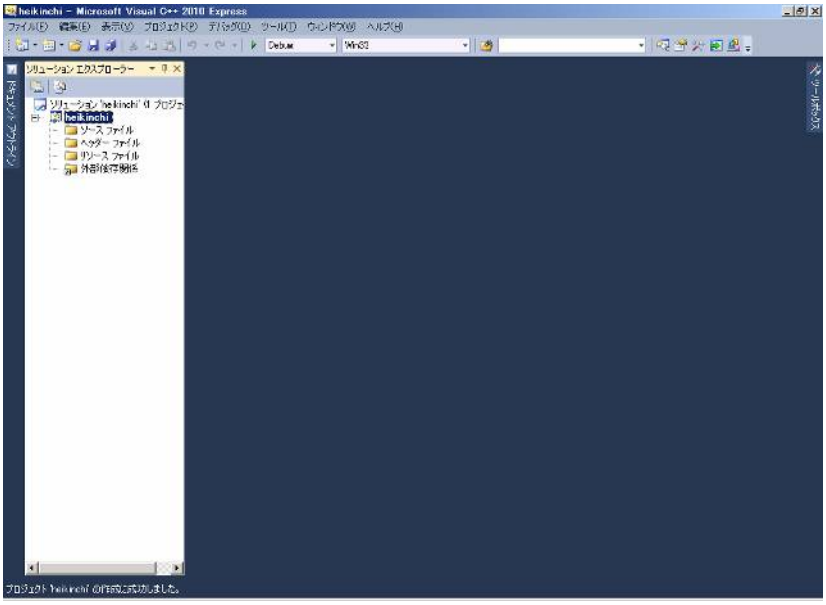

〈2〉［アプリケーションの設定］チェックボックス




設定：

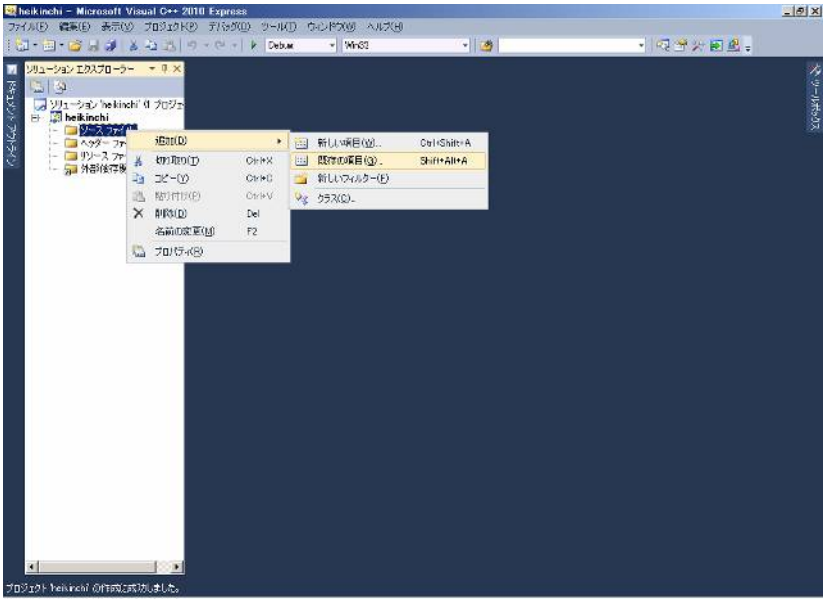
・「アプリケーションの種類：コンソールアプリケーション」

・「追加のオプション：空のプロジェクト」→［完了］

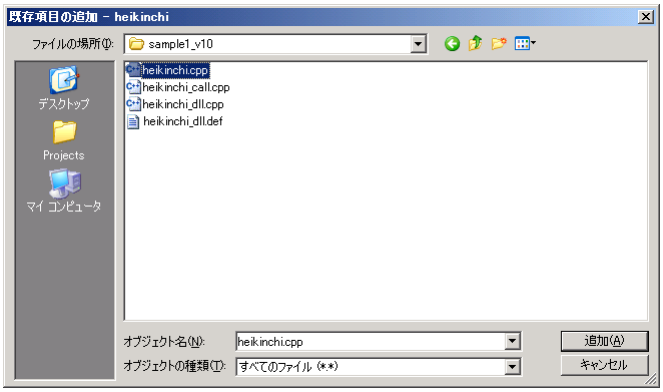


（４）「heikinchi.cpp」をプロジェクトに参加させる

［ソースファイル］上で、右クリック→［追加］―［既存の項目］

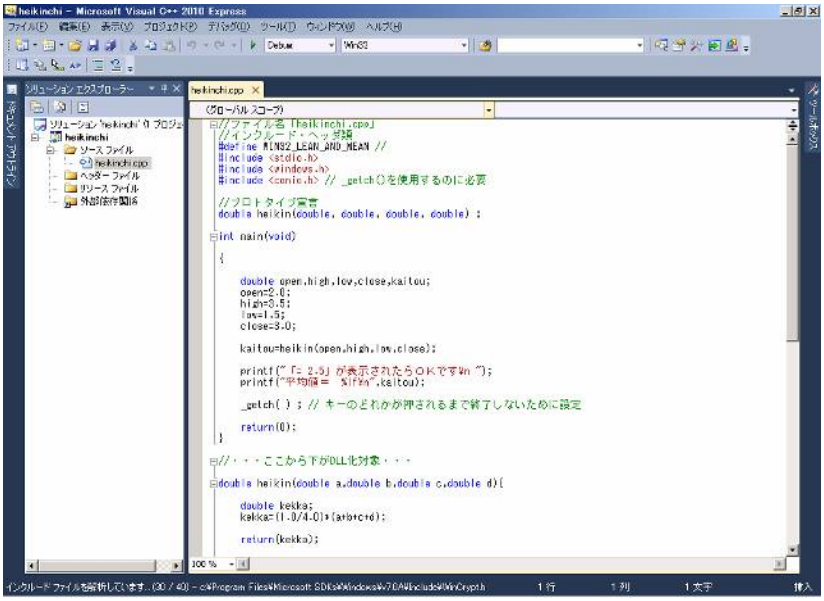


・［既存項目の追加］ウインドウで、「heikinchi.cpp」を指定する

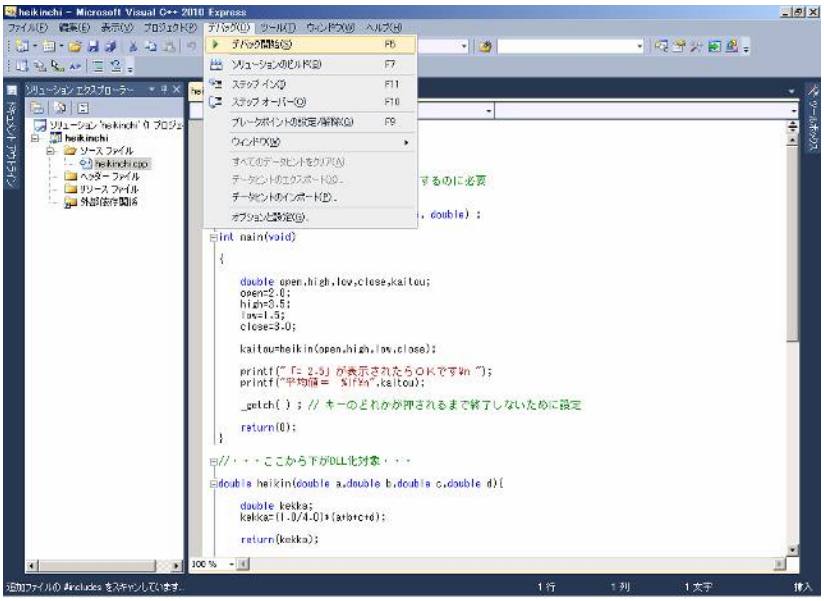



※注意；VC++2010 用に準備する「ソースファイル」は、文字コードを「Shift-JIS」で作成すること。（UTF-8N コードを使うと文字化けが発生する）

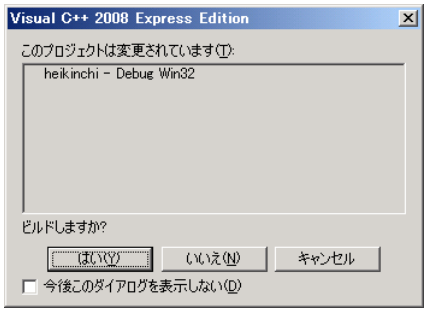
（５）ソースファイル・ホルダに格納された「heikinchi.cpp」をクリックし内容確認



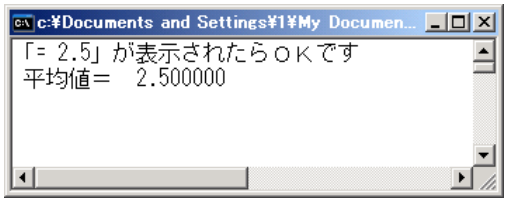
・［デバッグ］―［デバッグ開始］（又は、 をクリック）



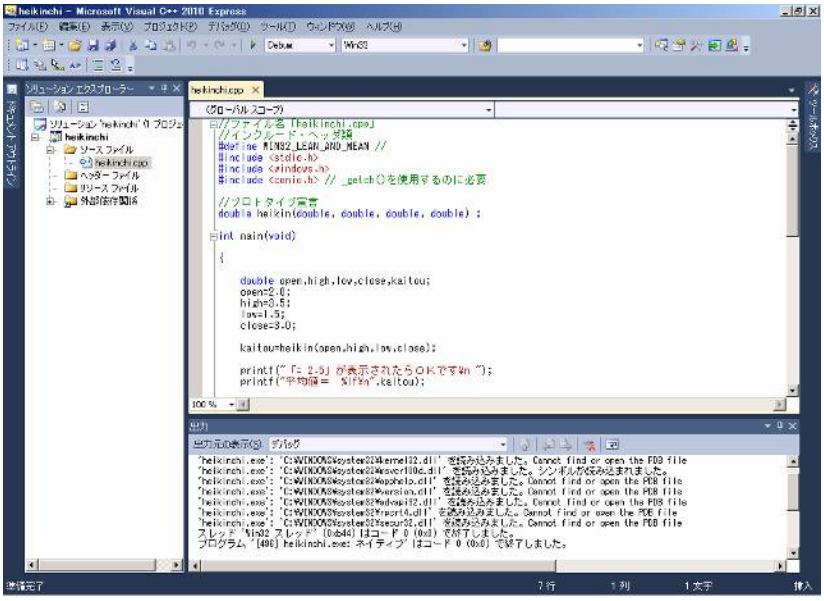
⇒　・「プロジェクトは変更されています、ビルドしますか？」



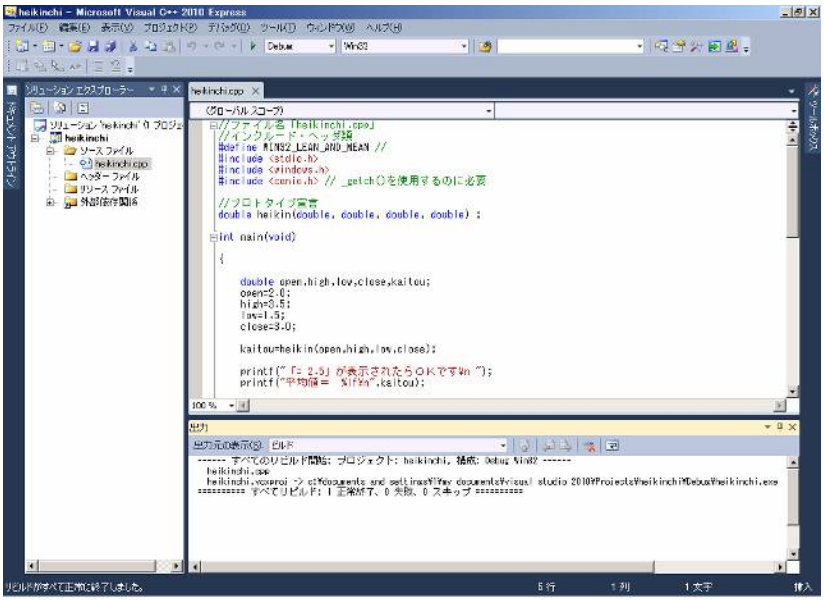

［はい］をクリック　⇒エラーが無ければ、結果が表示されます。




※このとき、デバッグ状況の VC++画面は下記のようになります。



※［ビルド］－［ソリューションのビルド］を実行してみると下記の様に表示されます。(VC++2010 基本設定では［ビルド］－［ソリューションのビルド)



１－２．ステップ２；main() と DLL 部に分割

・Win で動く main()と DLL に分割します（Debug モード）

・再確認；下記の手順で解説します。

［手順Ａ］・・・DLL 化する関数部分の切り出し

［手順Ｂ］・・・DLL 呼出側 main()部のまとめ

［手順Ｃ］・・・Debug 版で組合せ動作確認

［手順Ａ］・・・DLL 化する関数部分の切り出し

（１）先ず、「ステップ１」で作成したプログラム・コードの中から、下記の手順で「関数部」→「①DLL 本体；heikinchi_dll.cpp」を、切り出す。

（関数部分を下記の手順で修正していきます）

関数部分　＜ステップ１；コードより＞

```
//・・・ここから下が DLL 化対象・・・

double heikin(double a,double b,double c,double d){

        double kekka;
        kekka=(1.0/4.0)*(a+b+c+d);
        return(kekka);
}
```

<1>ヘッダ部を追加

```
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
```

※「#define WIN32_LEAN_AND_MEAN」は、余計なインクルードはしないことを指示しているだけなので、無くてもＯＫ。

<2>関数部を修飾（２箇所）

・先頭に「 __declspec(dllexport) 」を追記

・戻り値の型名の後に「 __stdcall 」を追記

「double heikin(double a,double b,double c,double d)」

⇒

「__declspec(dllexport) double __stdcall heikin(double a,double b,double c,double d)」

<2>DLL 用の main を添付

※C 言語のプログラムには「main()」であるように、DLL にも下記の「Dllmain()」が必要です、挿入する位置はどこでも構いません。

```
  BOOL APIENTRY DllMain(HANDLE hModule,DWORD ul_reason_for_call,LPVOID lpReserved)
   {
//----
    switch(ul_reason_for_call)
      {
       case DLL_PROCESS_ATTACH:
       case DLL_THREAD_ATTACH:
       case DLL_THREAD_DETACH:
       case DLL_PROCESS_DETACH:
          break;
      }
//----
    return(TRUE);
  }
```

<4>最終コード・・「①DLL 本体；heikinchi_dll.cpp」

※<1>～<3>までの処理をした結果のコードです

```
__declspec(dllexport) double __stdcall heikin(double a,double b,double c,double d){

        double kekka;
        kekka=(1.0/4.0)*(a+b+c+d);
        return kekka;

}
BOOL APIENTRY DllMain(HANDLE hModule,DWORD ul_reason_for_call,LPVOID lpReserved)
  {
//----
    switch(ul_reason_for_call)
      {
       case DLL_PROCESS_ATTACH:
       case DLL_THREAD_ATTACH:
       case DLL_THREAD_DETACH:
       case DLL_PROCESS_DETACH:
          break;
      }
//----
    return(TRUE);
  }
```

（２）「②定義ファイル；heikinchi_dll.def」を準備します

※これは、DLL からエクスポートする関数名称を定義するためのファイルです。

②「keikinchi_dll.def」

```
;コメント欄
;ファイル名「keikinchi_dll.def」
```

```
;「EXPORTS ***」ｴｸｽﾎﾟｰﾄする***関数が「***」と言う名前でDLLﾌｧｲﾙに定義される・・必須
;「VERSION ***」DLLのﾊﾞｰｼﾞｮﾝ、在っても無くても良い

LIBRARY keikinchi_dll;DLLの内部名称

EXPORTS
      heikin

VERSION 1.0
```

（３）VC++ 2008（2010）を立ち上げる

（４）「新規プロジェクト；heikinchi_dll」を作成する

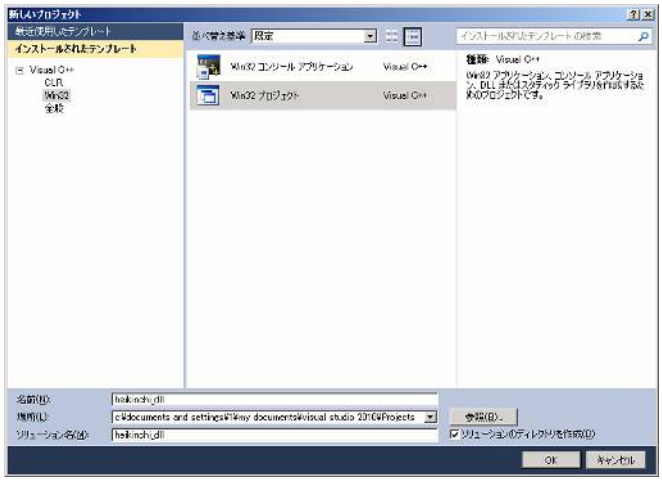
［ファイル］―［新規作成］―［プロジェクト］⇒

〈1〉［新しいプロジェクト］設定画面

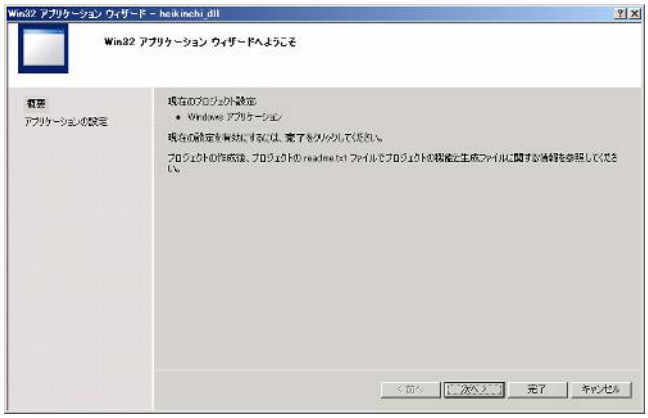
・［インストールされたテンプレート：Win32］

→「Win32 プロジェクト」を選択

・「名前：heikinchi_dll」を打ち込む

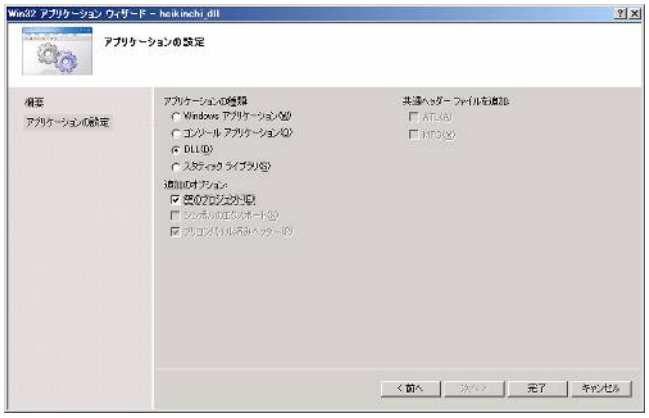



［OK］→



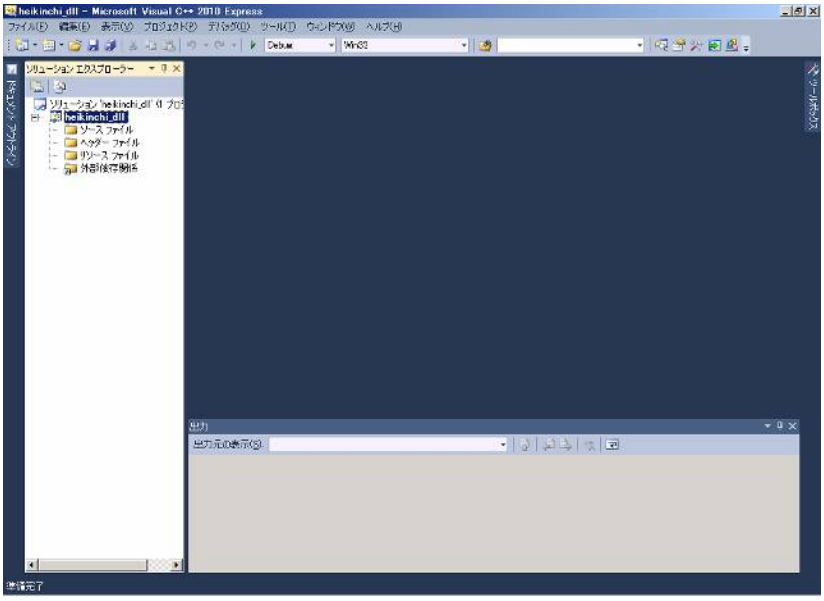

［次へ＞］をクリック　　→

〈2〉［アプリケーションの設定］チェックボックス



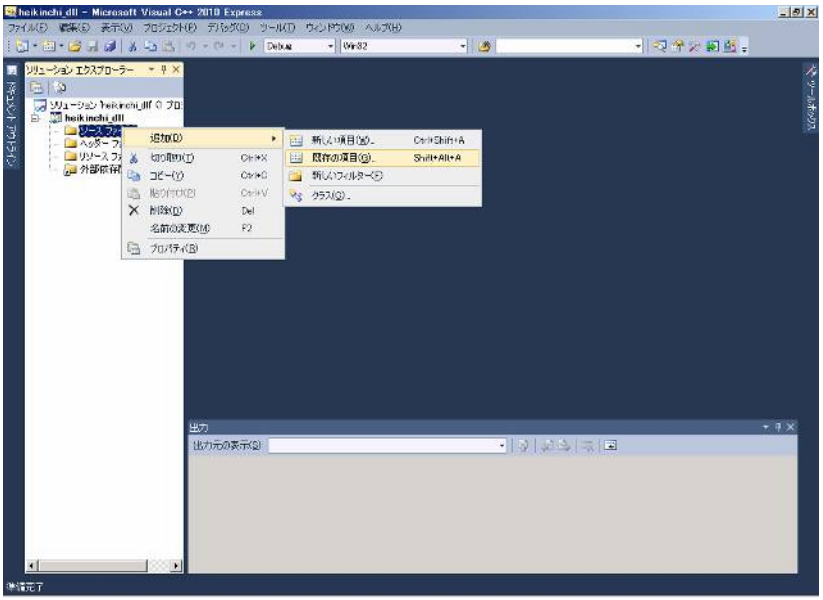

設定：　・「アプリケーションの種類：DLL」

・「追加のオプション：空のプロジェクト」→［完了］をクリック



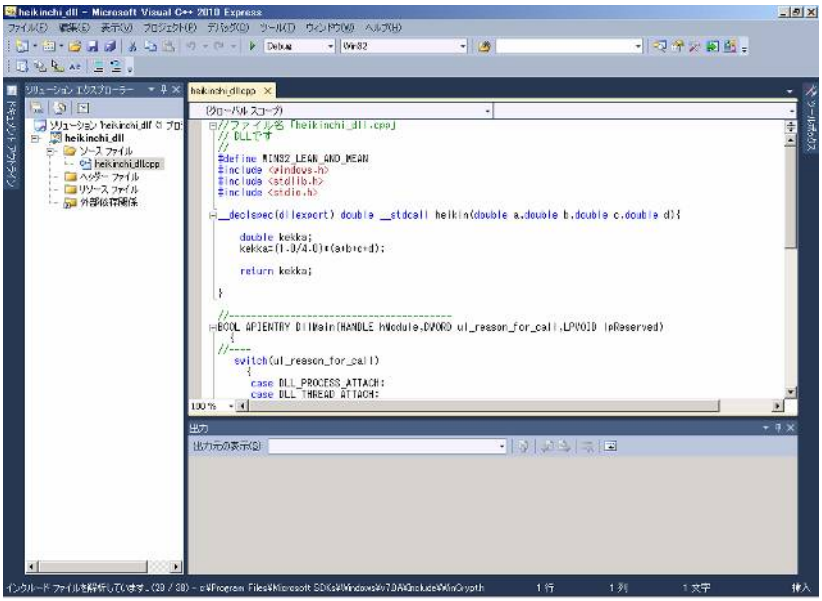

（５）「heikinchi_dll.cpp」をプロジェクトに参加させる

［ソースファイル］上で、右クリック→［追加］―［既存の項目］



・[既存項目の追加] ウインドウで、「heikinchi_dll.cpp」を指定する

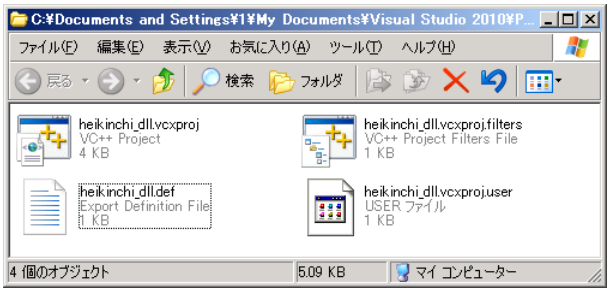
（６）ソースファイル・ホルダに格納された「heikinchi_dll.cpp」をクリックし、内容確認を確認する



・「My Documents\Visual Studio 2010\Projects」中に「heikinchi_dll」フォルダが作成されます。

（７）定義ファイル「keikinchi_dll.def」を、「(プロジェクト名).vcxproj」が存在するフォルダー内にコピーします。本件では、下記のフォルダ内となります。

・「My Documents\Visual Studio 2010\Projects\ heikinchi_dll \ heikinchi_dll」




（８）「keikinchi_dll のプロパティ」設定を行います

［プロジェクト］－［keikinchi_dll のプロパティ」で表示される

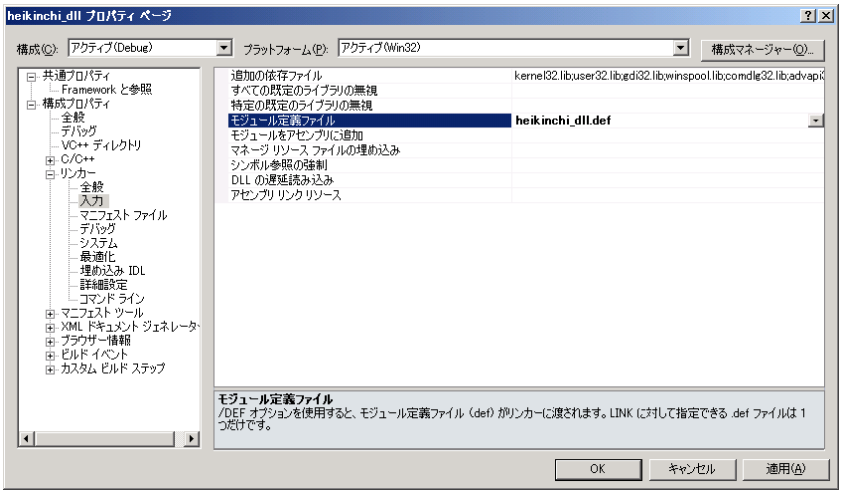
「keikinchi_dll プロパティページ」で、

・［構成プロパティー］―［リンカ］―［入力］―［モジュール定義ファイル］

に「heikinchi_dll.def」と手入力する。

（関数名の定義ファイル名を記入する）

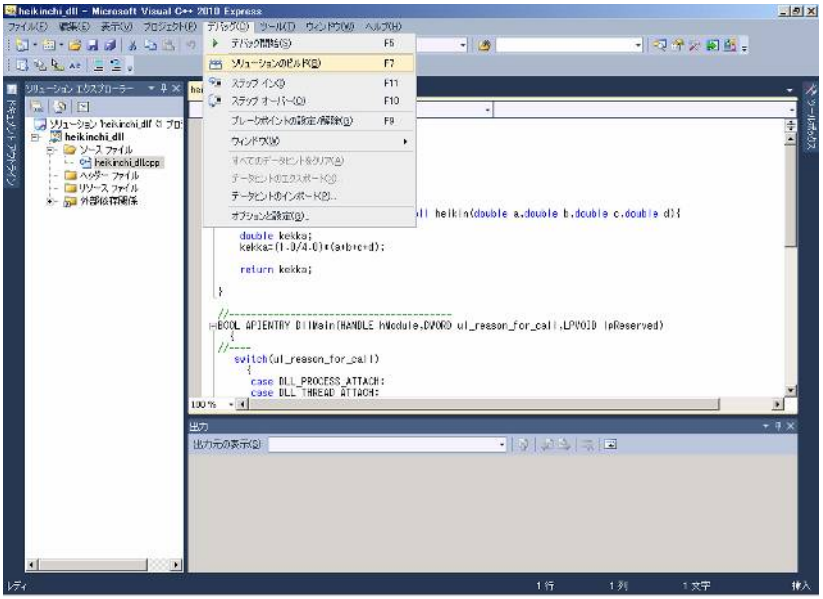
・［適用］→［OK］



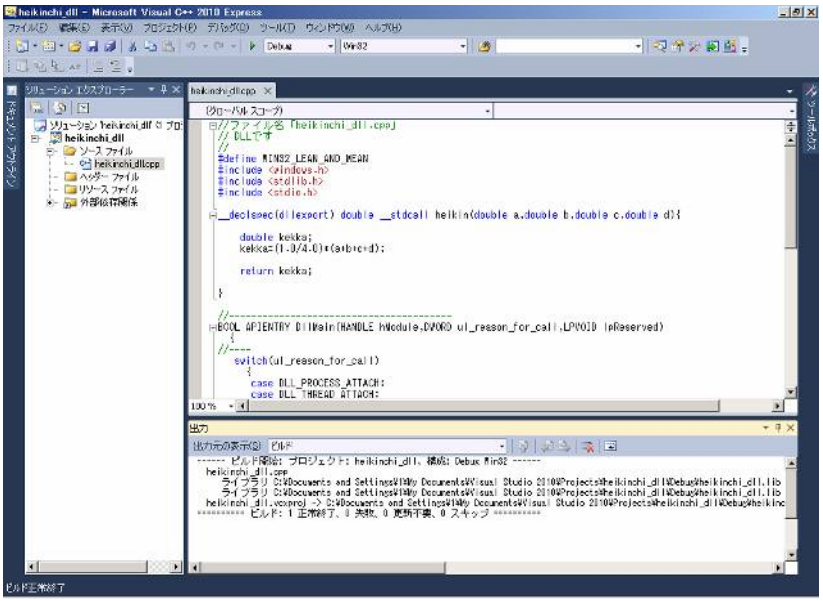

（9）「ビルド（コンパイル、リンク)」を行う

〈1〉ビルドし、正常終了することを確認する

・[デバッグ] ー [ソリューションのビルド]

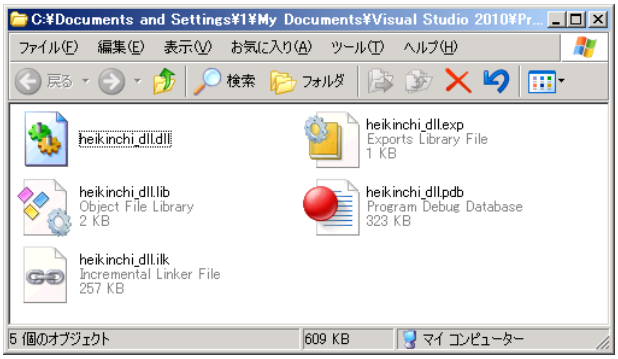



・エラーが無いことを確認する⇒「====ビルド：1 正常終了、0 失敗、…====」



〈2〉生成ファイルを確認する

・「My Documents\Visual Studio 2010\Projects\ heikinchi_dll \Debug」
ディレクトリー中に、「heikinchi_dll.dll」と「heikinchi_dll.lib」が生成されていること。




（１０）[ファイル]―[ソリューションを閉じる]で、このプロジェクトを終わらせる。

※補足説明１；DLL コード

・通常、VC++2010 では、C 言語形式「.c」も C++言語形式「.cpp」
ファイルも扱えます、また「.cpp」ファイルにC言語形式の記述しても
適切に処理されます。

・DLL を作成する際、C 言語のリンカでは、関数名はそのままエクスポートする関数名となりますが、C++言語のリンカでは、勝手に変更（修飾）されてしまいますので、そのままでは「DLL」と「呼び出し側（アプリ）」を切り離すことができません、つまり DLL を修正するたびにアプリ側も修正することになってしまいます。
そこで、VC++2010 を使って DLL を開発する場合は、関数定義ファイル「.def」を使い、DLL からエクスポートする関数名を修飾されないようにすることになります。

・関数名が修飾されないようにする方法としては、「extern ″C″ 」宣言（C 言語形式のリンカ指定）と関数定義ファイルを組み合わせて使う方法もありますが、ここではMQL4「experts/samples」にある DLL 作成サンプル・コードに示された方法を採用することにしました。
・「DllMain」は下記の理由で記載します。
通常、C 言語のコードを書くときは必ず「main(){}」を含むエントリー・ポイントとなる記述が必須になります。（普通、無条件に書いているはず）
DLL 作成でも、DLL エントリー・ポイント（DLL の初期化・終了処理を行なう関数）が必要であり、この関数名が通常は「DllMain」となります。
リンカによっては、「DllMain」関数の記載が省略されていると、自動的に C ランタイム・ライブラリから、初期化に成功した旨の「TRUE」を返す「DllMain」関数をリンクする場合があるようですが、ここでは省略はせず、やはり MQL4「experts/samples」にある DLL 作成サンプル・コードに示された方法を採用することにしました。

※補足説明２；モジュール定義ファイル「.def」
・最低限必要な記述は「EXPORTS　heikin 」のみです、ここで指定する関数名「heikin」が外部にエキスポートされ、他のアプリソフトからも呼出可能になります。
・「;」の後の一行はコメント欄になります。
・「LIBRARY keikinchi_dll」は無くても構いません。
・「VERSION 1.0」記述は在っても無くてもよいのですが、一応管理用に付けておきました。
・関数が増えた場合の対応方法
例えば、「heikin」に加えて「kasan」と「hikizan」と言う名称の関数が増えたときは、下記の様に追加記述していくだけで済みます。

```
EXPORTS
        heikin
        kasan
        hikizan
```

・その他、各種の指定が可能ですが、詳細は Web 上などで参照ください。

［手順Ｂ］・・・DLL 呼出側 main()部のまとめ
・次に、「ステップ１」で作成した main()部を、DLL 呼出側としてまとめます。
呼出側のソースコードとして下記の「heikinchi_call.cpp」を準備し、適当なホルダ中に入れる。

```
//ファイル名「heikinchi_call.cpp」
//  DLL を呼出す側
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h> // _getch()を使用するのに必要

//暗黙的（静的）な DLL リンクを採用
//下記「″ ″」部には、dll をビルドしたときのﾗｲﾌﾞﾗﾘ名を記載する
#pragma comment(lib,"heikinchi_dll.lib")

//プロトタイプ宣言
__declspec(dllimport) double __stdcall heikin(double, double, double, double);

int main(void)
{
        double open,high,low,close,kaitou;
        open=2.0;
        high=3.5;
        low=1.5;
        close=3.0;

        kaitou=heikin(open,high,low,close);

        printf("「= 2.5」が表示されたらＯＫです¥n ");
        printf("平均値＝  %lf¥n",kaitou);

        _getch( ) ; // キーのどれかが押されるまで終了しないために設定
        return(0);
}
```

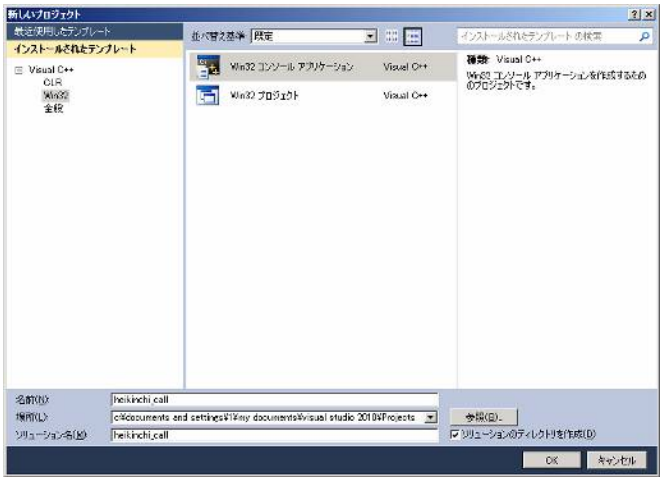
［手順Ｃ］・・・Debug 版で組合せ動作確認

（１）VC++ 2008 (2010) を立ち上げる

（２）「新規プロジェクト；heikinchi_call」を作成する

　［ファイル］―［新規作成］―［プロジェクト］⇒

〈1〉［新しいプロジェクト］設定画面




設定；

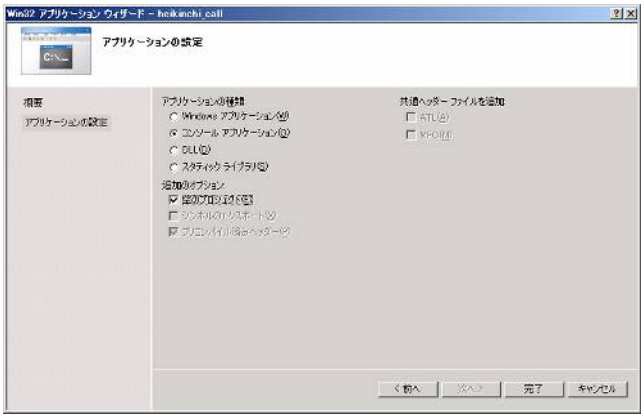
・「インストールされたテンプレート：Win32」

→「Win32 コンソールアプリケーション」

・「名前：heikinchi_call 」を打ち込む　→［OK］→


［次へ>］をクリック　→

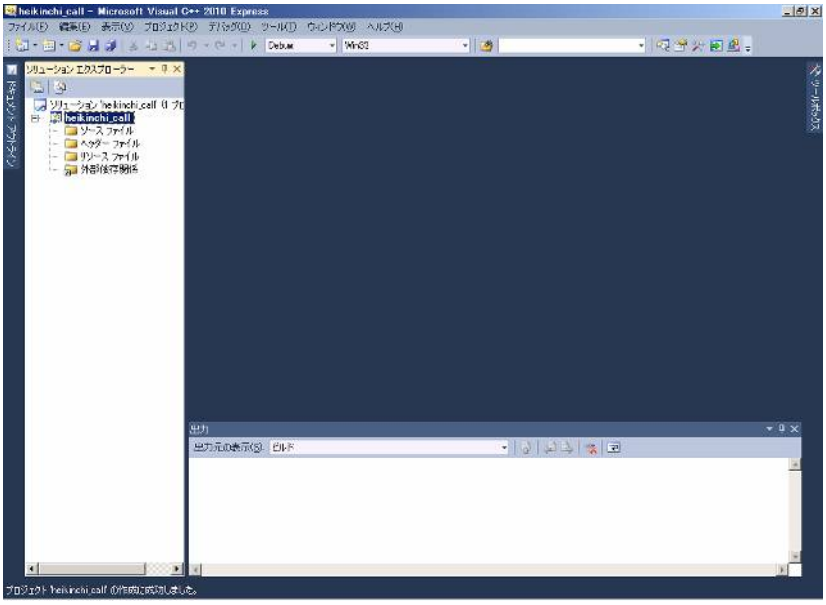
〈2〉［アプリケーションの設定］チェックボックス




設定；

・「アプリケーションの種類：コンソールアプリケーション」

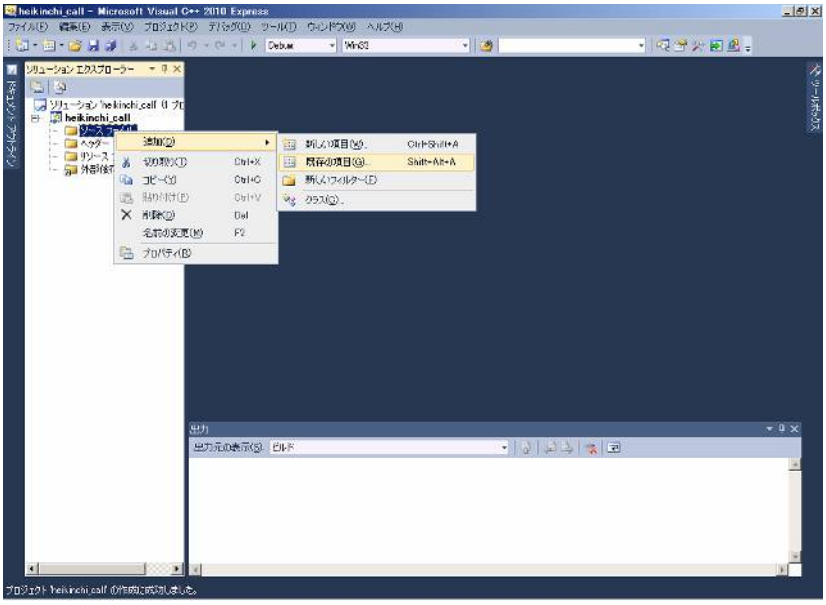
・「追加のオプション：空のプロジェクト」→［完了］→



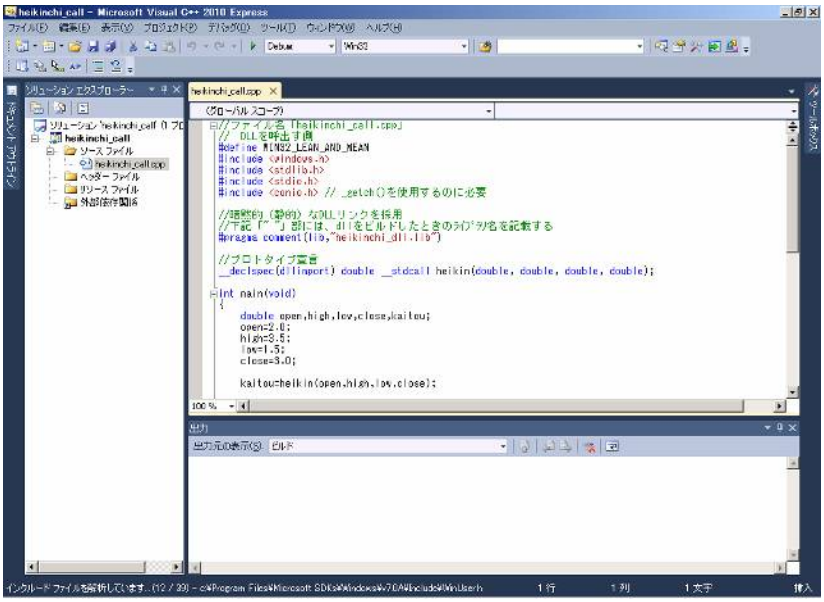
（3）「heikinchi_call.cpp」をプロジェクトに参加させる

［ソースファイル］上で、右クリック→［追加］―［既存の項目］

・［既存項目の追加］ウインドウで、「heikinchi_call.cpp」を指定する

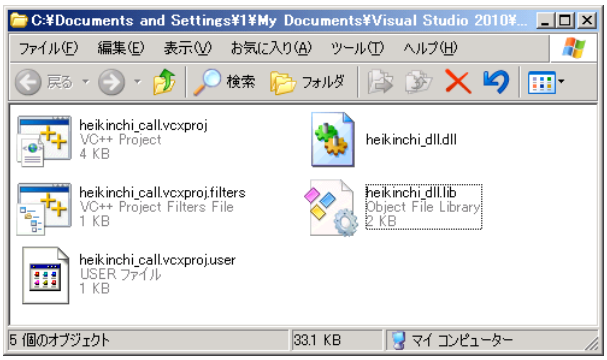


（4）ソースファイル・ホルダに格納された「heikinchi_call.cpp」をクリックし、内容確認を確認します。



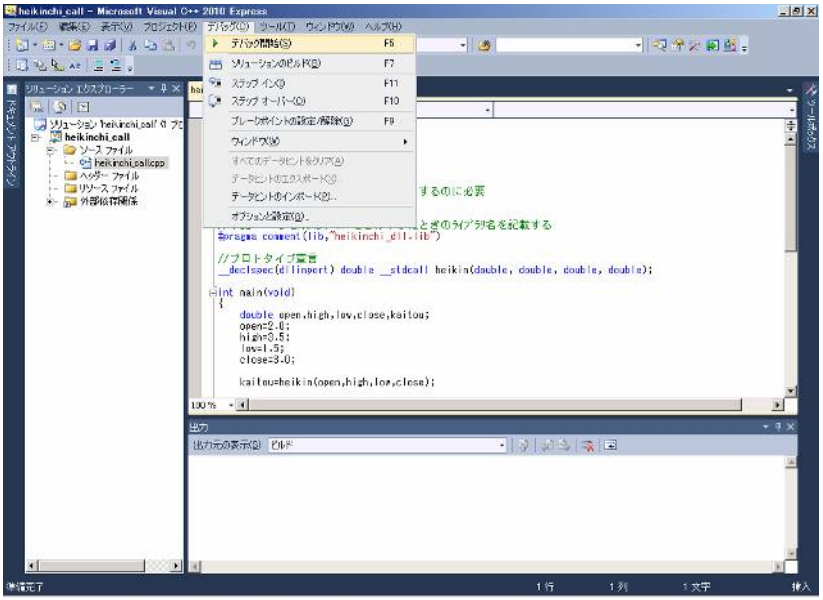

（５）「My Documents\Visual Studio 2010\Projects」ディレクトリー内に、「heikinchi_call」ホルダが作成されていることを確認する。

（６）「手順Ａ」で生成された「keikinchi_dll.dll」と「keikinchi_dll.lib」を、「heikinchi_call」ホルダ中の更に「(プロジェクト名).vcxproj」が存在するホルダー内にコピーする。

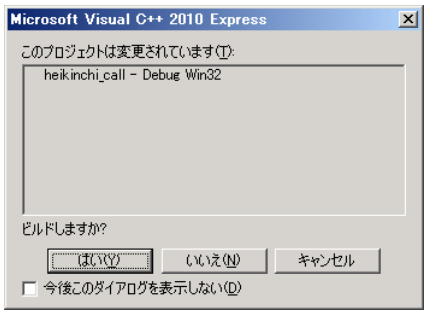


（７）「デバッグ」を開始する

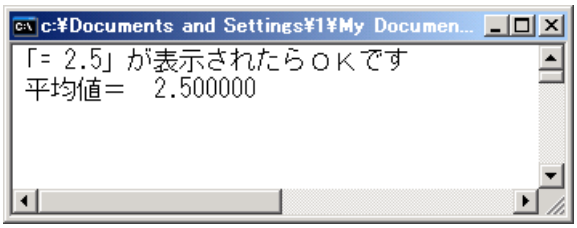
〈1〉［デバッグ］－［デバッグ開始］（又は、  をクリック）



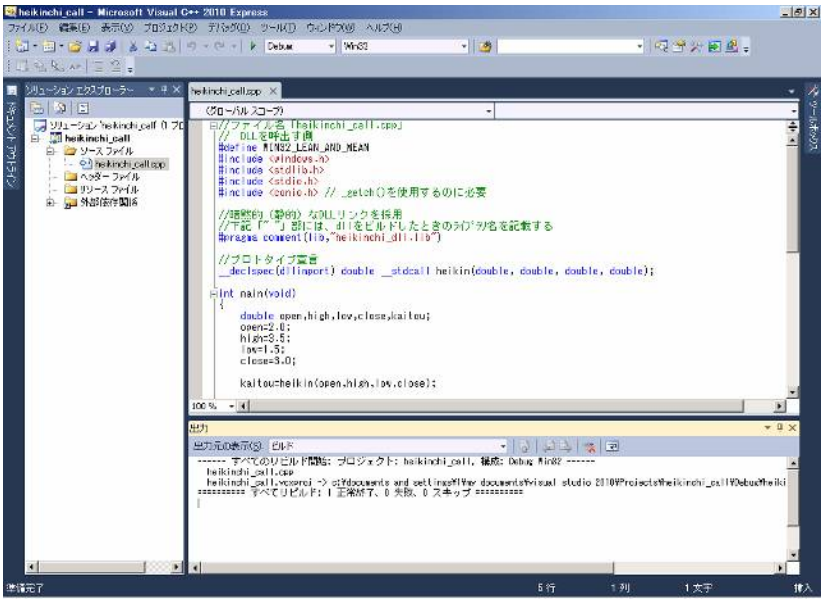

〈2〉「プロジェクトは変更されています、ビルドしますか？」




［はい］⇒エラーが無ければ、「作成手順手１」と同様の結果が表示されます。




※このとき、ビルド状況は下記の様に表示されます。




※補足説明１；　暗黙的 DLL リンク

・DLL を Win アプリから呼び出す方法は、下記の２種類があります。

①暗黙的（静的）リンク；

リンカーでリンクした時点で DLL の種類や関数は固定

②明示的（動的）リンク；

DLL の種類や関数を「動的」に変更できる

・当然、MQL4 での DLL 呼出し方法は「明示的リンク」によるのですが、C++言語での記述法が複雑なため本稿の Win 上での「main()から DLL 呼び出し」では、はるかに記述が簡単な「暗黙的リンク」を採用しています、このための記述が下記の部分です。

#pragma comment(lib,"heikinchi_dll.lib")

※補足説明２；　プロトタイプ宣言

・「暗黙的リンク」と合わせて、DLL 中の関数をプロトタイプ宣言する部分が下記です。

```
__declspec(dllimport) double __stdcall heikin(double, double, double, double);
```

※補足説明３；　暗黙的 DLL リンク方法採用の利点

・リンクする時点で、main()側から呼出可能な DLL と関数が固定されてしまうと言う制限があるにしても、「暗黙的 DLL リンク」には、お気づきの様に極めて優れた特徴があります。

すなわち、１本の動作する Win プログラムから「DLL 部」を作り、動作確認するには、下記の様に単純に分割し、数行のみ追加すれば良いのです。

・本例では、下記手順で済んでしまいます。

〈1〉１本の動作する Win プログラム（heikinchi.cpp）で、「**太字**」部分を追加し、
〈2〉「細字」のプロトタイプ宣言部を削除した後、「呼出」部分と「DLL」部分に分割し、
〈3〉後は、DLL 用に簡単な内容のモジュール定義「.def」ファイルのみ準備すれば完了。

```
//ファイル名「heikinchi.cpp」
//インクルード・ヘッダ類
#define WIN32_LEAN_AND_MEAN //無くてもＯＫ
#include <windows.h>
#include <stdio.h>
#include <conio.h> // _getch()を使用するのに必要

//下記「″ ″」部には、dll をビルドしたときのﾗｲﾌﾞﾗﾘ名を記載する
#pragma comment(lib,"heikinchi_dll.lib")

//プロトタイプ宣言
__declspec(dllimport) double __stdcall heikin(double, double, double, double);

//プロトタイプ宣言
//double heikin(double, double, double, double) ;←●DLL 呼出 main()化するときに削除します


int main(void)
{

    double open,high,low,close,kaitou;
    open=2.0;
    high=3.5;
    low=1.5;
    close=3.0;

    kaitou=heikin(open,high,low,close);

    printf("「= 2.5」が表示されたらＯＫです¥n ");
    printf("平均値＝  %lf¥n",kaitou);

    _getch( ) ; // キーのどれかが押されるまで終了しないために設定

    return(0);
}
```

**－－－－－－－－　ここで分割（上側は呼出側、下側は DLL）　－－－－－－－－**

```
//・・・ここから下が DLL 化対象・・・
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <stdlib.h>
#include <stdio.h>


__declspec(dllexport) double __stdcall heikin(double a,double b,double c,double d){

//double heikin(double a,double b,double c,double d){ ←・DLL 化するときに削除
```

```
    double kekka;
    kekka=(1.0/4.0)*(a+b+c+d);
    //「kekka=(1/4)*(a+b+c+d);」だと「0」になる！
    //printf("関数側での平均＝%lf¥n",kekka);//デバッグ用に使った
    return(kekka);

}

BOOL APIENTRY DllMain(HANDLE hModule,DWORD ul_reason_for_call,LPVOID lpReserved)
  {
//----
   switch(ul_reason_for_call)
     {
      case DLL_PROCESS_ATTACH:
      case DLL_THREAD_ATTACH:
      case DLL_THREAD_DETACH:
      case DLL_PROCESS_DETACH:
         break;
     }
//----
   return(TRUE);
  }
```

１－３．ステップ３；DLL 作成と Win 上での動作確認

・Win 版実行形式「.exe」と Release 版の DLL を作成し動作確認します

・再確認；下記の手順で解説します。

［手順Ａ］・・・DLL 化する関数部分の Relese 版をビルド

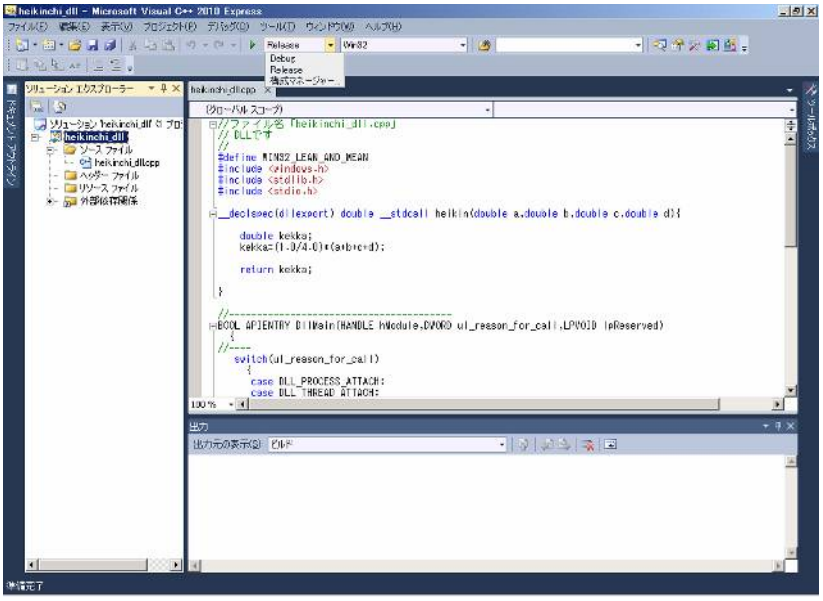
［手順Ｂ］・・・DLL 呼出側 main()部の Release 版実行ファイル「.exe」をビルド

［手順Ｃ］・・・Release 版で組合せ動作確認

［手順Ａ］・・・DLL 化する関数部分の Relese 版をビルド

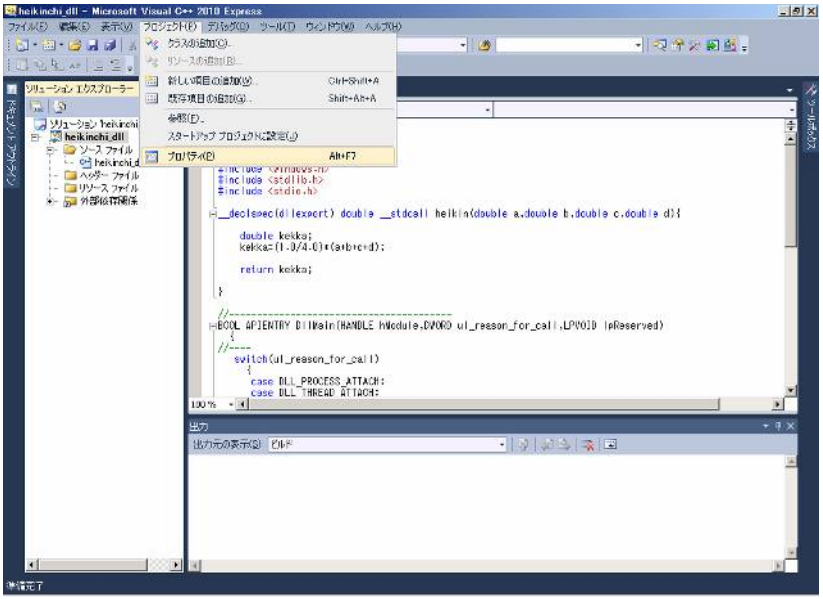
※先ず DLL 側から、頒布形式である Release 版にビルドし直します。

（１）「heikinchi_dll」プロジェクトを開き、開発モードを［Debug］→［Release］へ変更します




（２）「heikinchi_dll のプロパティ」設定を行います

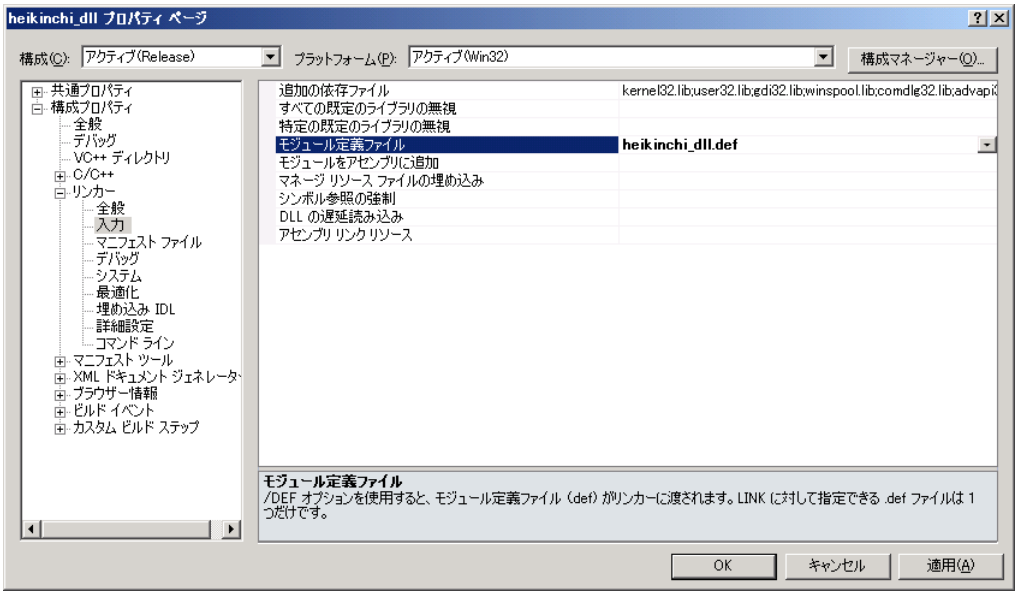
［プロジェクト］－［プロパティ」⇒



表示された「keikinchi_dll プロパティページ」での設定

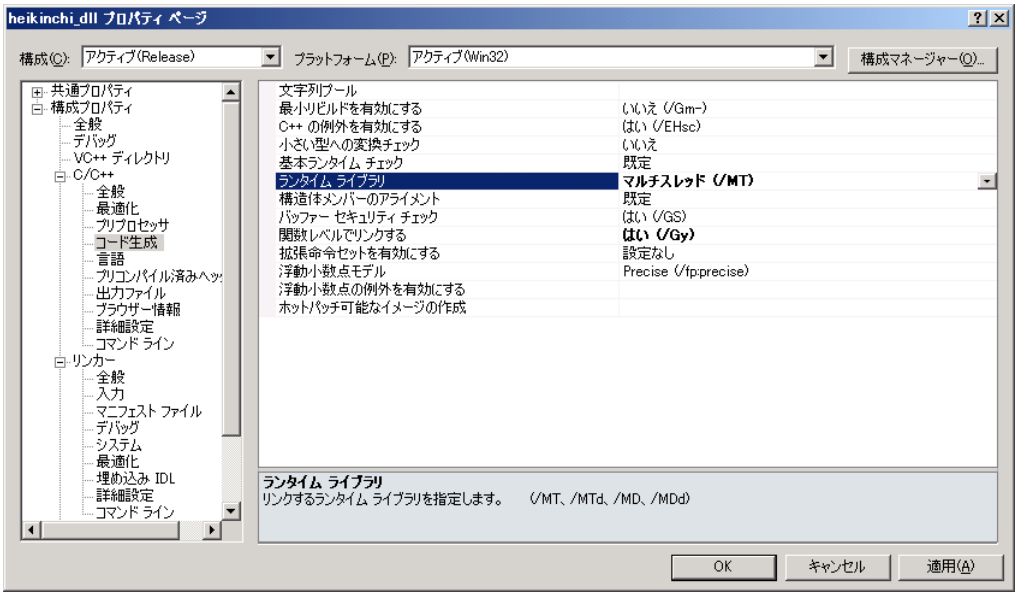
〈1〉［構成プロパティー］―［リンカ］―［入力］―［モジュール定義ファイル］に「keikinchi_dll.def」と手入力する。

・［適用］→［OK］




〈2〉［構成プロパティー］―［C/C++］―［コード生成］―［ランタイム ライブラリ］で「マルチスレッド(/MT)」を選択

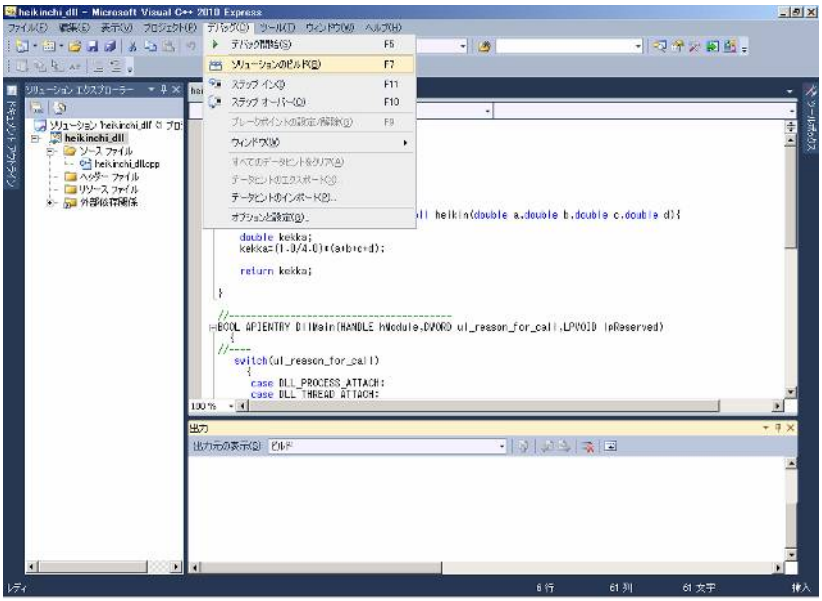
・［適用］→［OK］



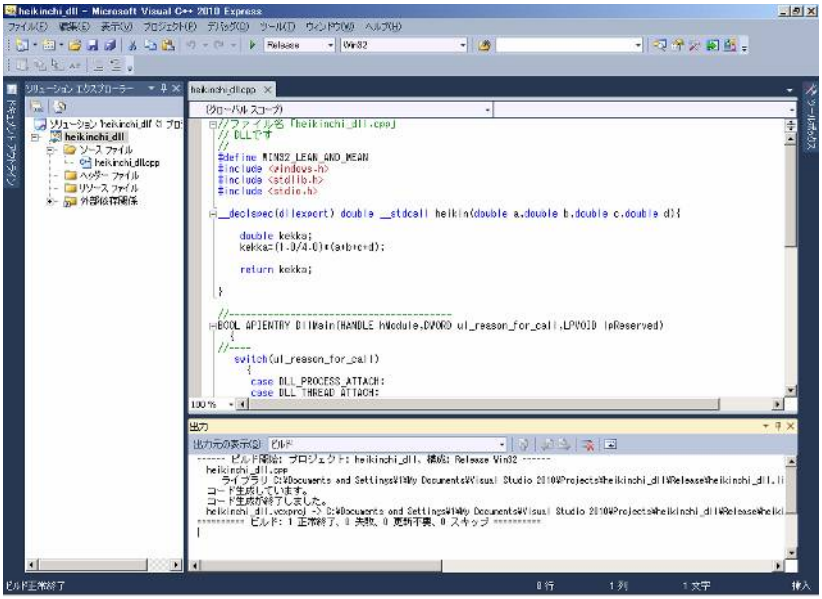

（3）「ビルド（コンパイル、リンク)」を行う

〈1〉ビルドし、正常終了することを確認する

・［デバッグ］ー［ソリューションのビルド］



・エラーが無いことを確認する⇒「====ビルド：1 正常終了、0 失敗、…====」



〈2〉生成ファイルを確認する

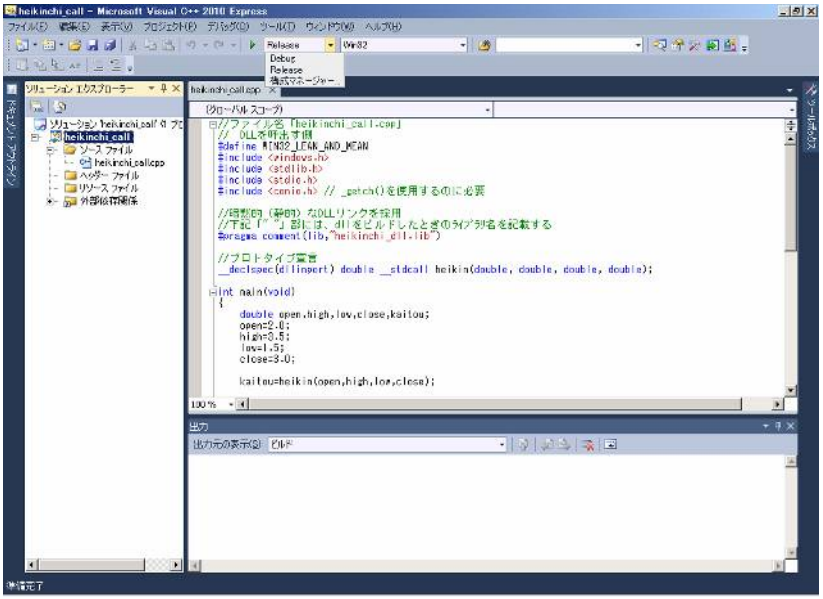
・「My Documents\Visual Studio 2010\Projects\ heikinchi_dll \Release」ディレクトリー中に、「keikinchi_dll.dll」と「keikinchi_dll.lib」が生成されていること。

※この様にして作成する DLL は当然ですが CTL のみではなく、他のアプリソフト（例えば、エクセルの VBA など）からも呼び出す事が可能です。

［手順Ｂ］・・・DLL 呼出側 main()部の Release 版実行ファイル「.exe」をビルド

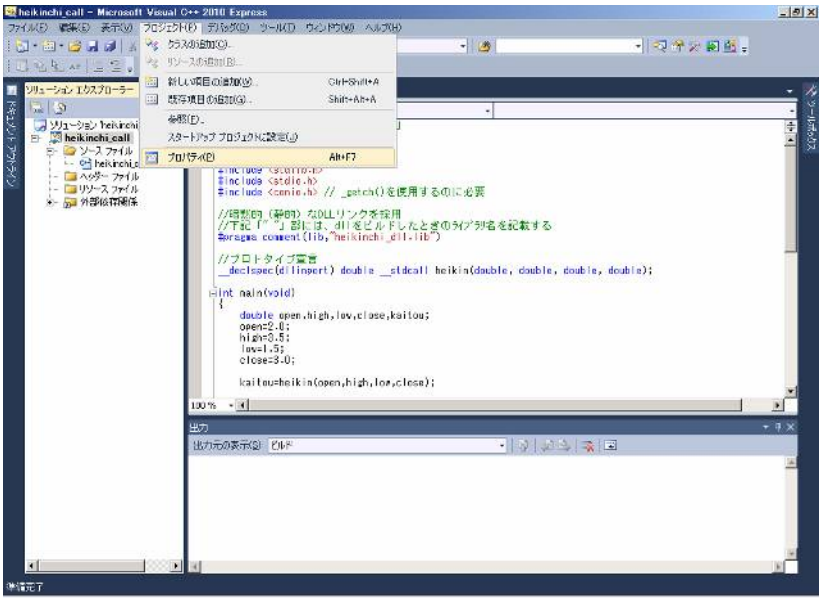
次に、呼出側も Release 版にビルドし直します。

（１）「heikinchi_call」プロジェクトを開き、開発モードを［Debug］→［Release］へ変更します



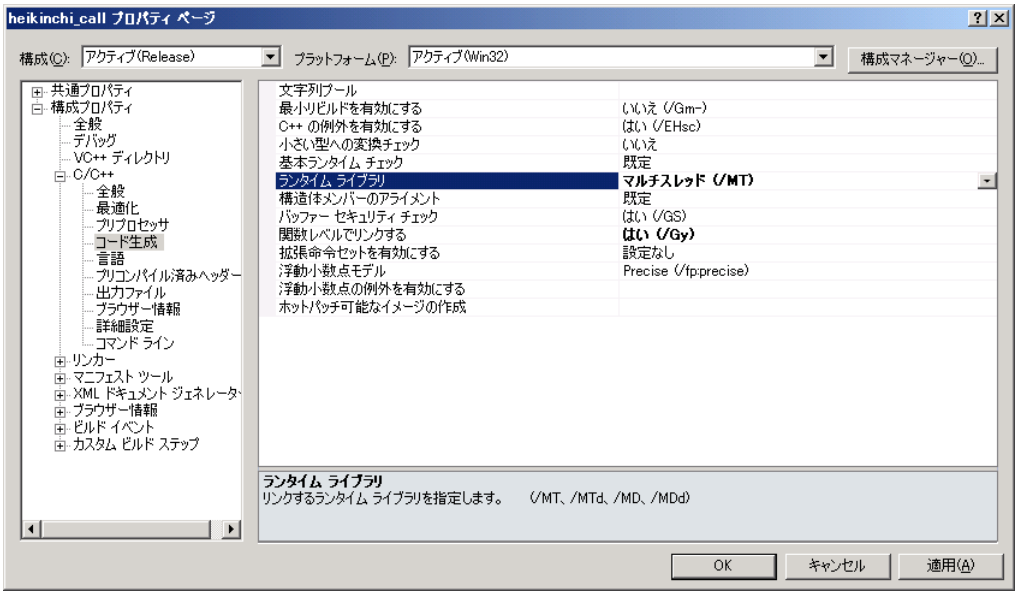

（２）「keikinchi_call のプロパティ」設定を行います

［プロジェクト］－［プロパティ」⇒



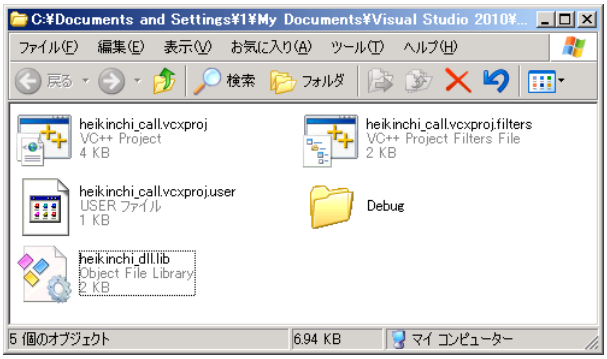
「heikinchi_call プロパティページ」で、

・[構成プロパティー] ― [C/C++] ― [コード生成] ― [ランタイム ライブラリ] で「マルチスレッド(/MT)」を選択




（３）手順Ａで作成された「keikinchi_dll.lib」を「.vcxproj」があるフォルダー中に入れる。

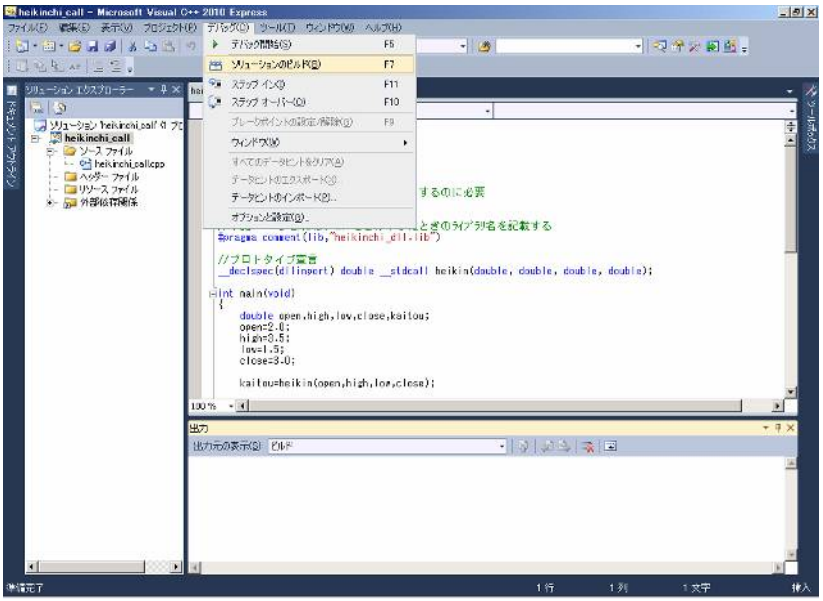
※事前に、デバックで作成して入れておいた「lib と dll」は捨てる。



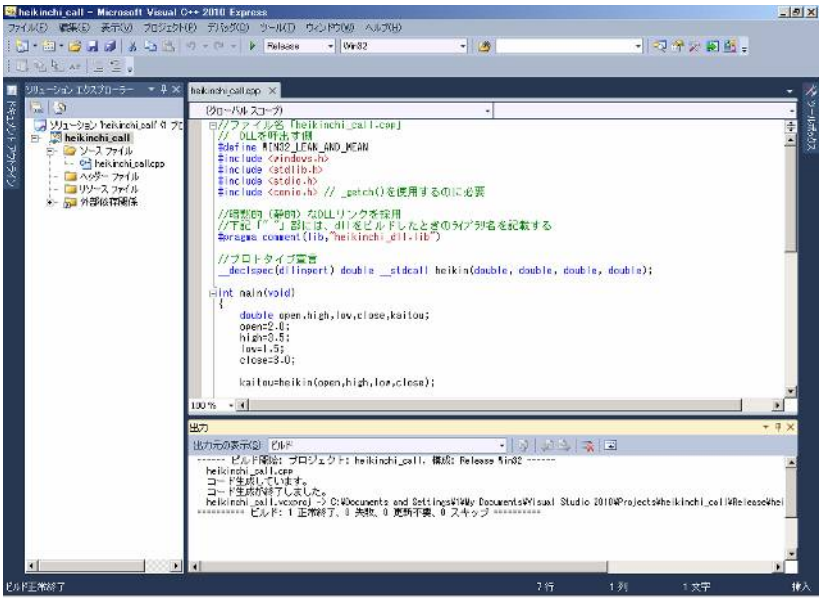

（４）「ビルド（コンパイル、リンク)」を行う

〈1〉ビルドし、正常終了することを確認する

・[デバッグ] ― [ソリューションのビルド]

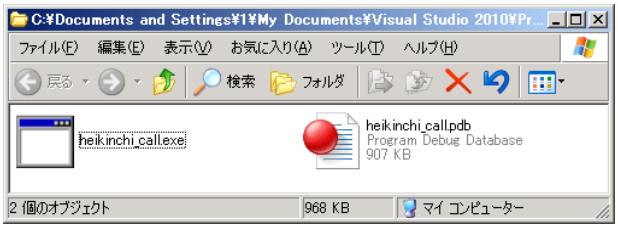


・エラーが無いことを確認する⇒「====ビルド：1 正常終了、0 失敗、…====」



〈2〉生成ファイルを確認する

・「My Documents\Visual Studio 2010\Projects\ heikinchi_dll \Release」ディレクトリー中に、「keikinchi_call.exe」が生成されていること。



［手順Ｃ］・・・Release 版で組合せ動作確認

※Release 版の動作確認を行うため、VC++ 2010 がインストールされていない
パソコン上で、動作確認を行う。（これ、大切です）
VC++ 2010 がインストールされてパソコン上で動作確認すると、本来単独では
動かない（配布はできない）はずのデバッグモードの DLL でも一見正常に動いて
しまうので、要注意！

（１）「手順Ｂ」で生成された「keikinchi_dll.dll」と「keikinchi_call.exe」を、
VC++ 2008（2010）がインストールされていないパソコン上の任意ホルダー
（同一の）にコピーする。

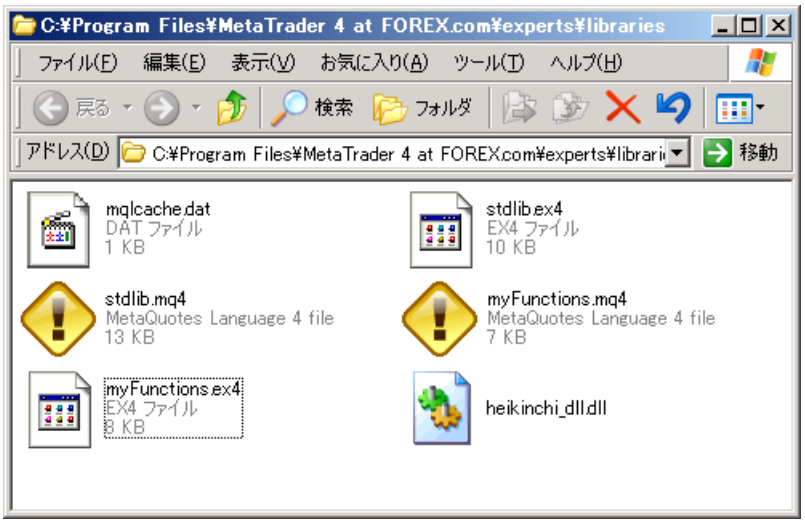

（２）「keikinchi_call.exe」をダブルクリック
⇒プログラムが実行され、「ステップ２」と同一結果が表示される。

## １－４．ステップ４；MQL4 による DLL 動作確認

・MQL4（呼出側）と DLL（Release 版）による動作確認を行う。

### （１）DLL の準備

・DLL(heikinchi_dll.dll)を「experts\libraries」ディレクトリー中にコピーする。




### （２）MQL4 コードの準備

・DLL を呼び出し、動作確認するための MQL4 コード（例）を準備する

・別ウインドウに表示する場合は、

「①」のコメントを外し、「②」をコメントアウト

・同一ウインドウに表示する場合は、

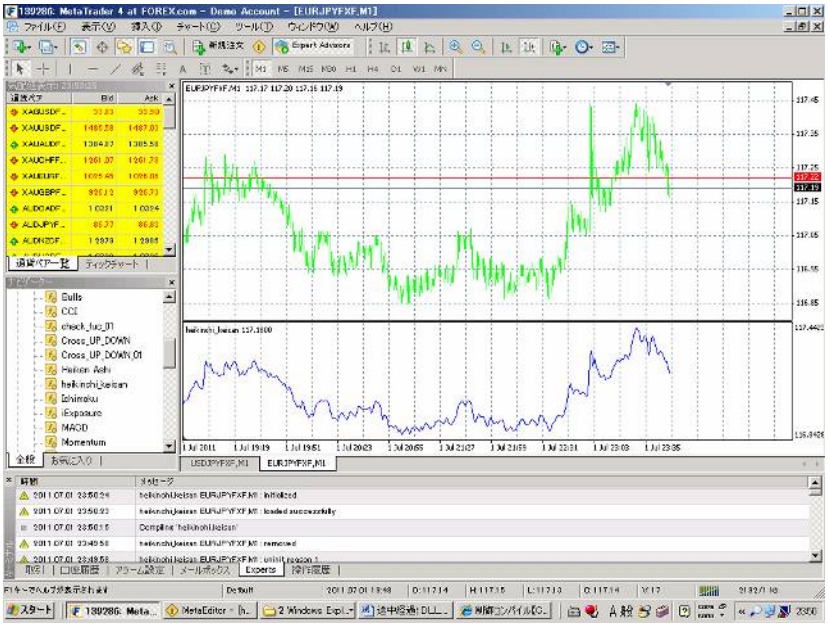
「②」のコメントを外し、「①」をコメントアウト

```
//          heikinchi_keisan.mq4     indicator
//

#import "heikinchi_dll.dll"
    double heikin(double,double,double,double);
#import

//#property indicator_separate_window //←①別ウインドウに表示
#property indicator_chart_window //←②同一ウインドウに表示
#property indicator_buffers 1

double HEIKIN[];

int init()
  {
   SetIndexStyle(0,DRAW_LINE,STYLE_SOLID,1,Blue);
   SetIndexBuffer(0,HEIKIN);
   IndicatorBuffers(1);
   return(0);
  }

int deinit()
```

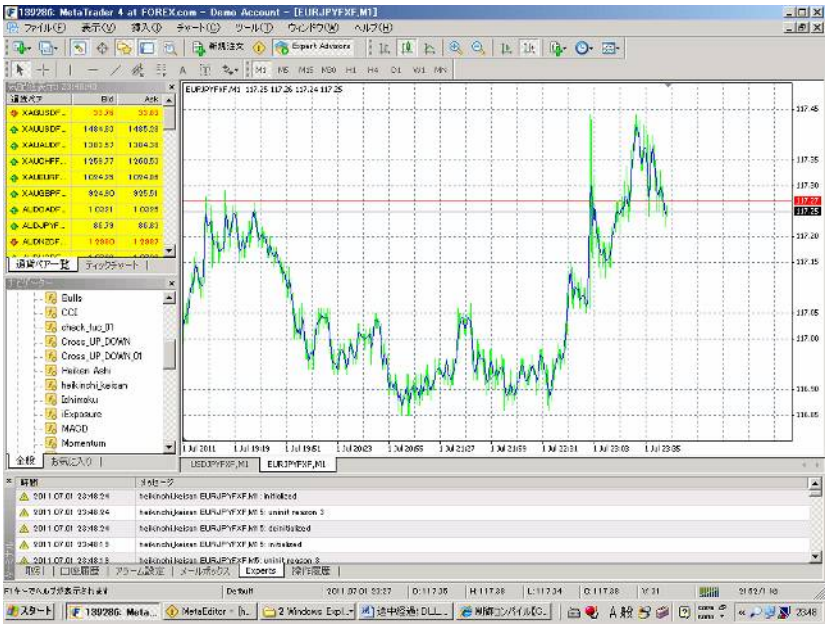
```
  {
   return(0);
  }

int start()
  {

    int limit=Bars-IndicatorCounted();

   ArraySetAsSeries(HEIKIN,true);

   for(int i=limit-1;i>=0;i--)
   {
       HEIKIN[i]=heikin(Open[i],Low[i],High[i],Close[i]);
   }

   return(0);
  }
```

（３）実行確認

①［別ウインドウに表示］する場合



②［同一ウインドウに表示］する場合



以　上