○「 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
 - 1節: CTL と DLL のリンク法
 - 1 1 : CTL から DLL を呼び出す
 - 1-2: CTL と DLL 間の「変数型」対応
 - 2節: DLL 開発の基礎
 - 2-1:開発環境
 - 2-2:開発の基本手順
 - 2-3: 頒布方法
- 2章: DLL を作る・呼ぶ
 - 1節:数値編
 - 1-1:ステップ1;1本のプログラムを作成
 - 1-2:ステップ2; main()とDLL部に分割
 - 1-3: ステップ3; DLL 作成と Win 上での動作確認

1 - 4 : ステップ 4 ; MQL4 による DLL 動作確認

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。
 - 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)
   ł
         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("平均值= %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
```

```
5/36
```

(2) VC++ 2010 を立ち上げる

■ スタート ページ - Microsoft Visual O ファイル(E) 編集(E) 表示(Y) 字(約の(E)	++ 2010 Express - ツールローウィンドスの ヘルズ氏		<u>-18</u>
121-0-03 8 2 8 4 5 1	9-6-11	- 3	• 1 4 3 3 4 8 2 4
■ 2012-022/20170-0	スタートページ × Visual C++・2010 E この Visual C++・2010 E この おしいつのメックト この オロジェクト この 近く使った プロジェクト	xpress 19年の開始 最新ニュース よたそ 学習 アゥブルード 「「「「「「「」」」」 「「」」」 最初のアブリケー3	Wale C++ 2010 Dicress ALPC2 Stand C++ 2010 Dicress ALPC2 Stand C++ 2010 Dicress T L. Wakes Hard S+/ C+/ 2018 Dicress T L. Wakes Hard S+/ C+/ 2018 Dicress T Market Hard S+/ C+/ 2018 Dicress T Compared T Stand C++ 2010 Dicress ALPC2 Market Hard S+/ C+/ Dicress T Displement Hard S+/ C+/ Displement Hard S+/ C+/ Displement Hard S+/ Displement Hard S+/ Dis
	マプロジェクト55から込み的にページを開ける マンス3ートアップ4日にページを使示する	Visual C++ 2010	Express Offi Mati
华储无了			

(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] ⇒

	Diff.Park(1)	• 🐷	プロジェクト(円).		Otrl+Shift+N	- R 🕾 🖉 🖉 🚽	
	MR(Q)	• 9	77(1KE).		OWI+N		
	開じる(Q)		現存のコードからプロジ	19下奏作693(E)。			
ĩ	シリューションを閉じる(1)						
1	違規に打たファイルを上書き保存(S) Otri-S 通規したファイルに名前を付けて保存(A)。	al	C++·2010 Ex	oress			
1	すべてを操作(L) Old+Shill+S			的電力開始			
Ľ,	ページ設定(1).	sph.		I FREWIMIND	10(4/ /		
k,	FINEKE). Owi-P	-		よたその学習	アップクレード		
	將了(g) Alt+F4	005		2			
						ロジュクトを見つけたりできます。 和愛問発信をロサラーニング センター Codewifion Visual C++ 2010 Express の23時日	
				the second se	最初のアプリケ	一ジョンをすぼやく作成する	
					Visual 0++ 20	0 Express の新機能	
	同 プロジェクト 読み 同 フゥート アップログ	込み彼に こが一のた	バージを開じる 読み示する				

- <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]



in32 アブリケーション ウィザー Win32	ドー houknohi アプリケーション ウィリードへようごそ	
戦業 アカリケーションの新定	現在のけのジェク145足 ・ エンソール アラガケーション 現在の5002を有効にすめは、第 イをジックして代えい。 プロジェクトの作時消息、プロジェクトのreadmetert ファイルにでクロジェクトの情報と生成ファイルに使まる情報を参照して (1)。	(22)
		æ

<2> [アプリケーションの設定] チェックボックス

Win82 アプリケーション ウィザー アプリケ C:ヽ_	ド - hoikinchi いうヨンの設定		<u>?</u> x
場要 37597-932の設定	 アブリーションの特徴 「Minane アブリケーション(型) C エンノール アブリケーション(型) C ロレジ C ロレジ C スクチャク ライブラリ(型) 1000 イブラゴン Minade アンゴン 1100 イブラゴン Minade アゴニン・イルロ島とへのタービク 	党道ヘッダー ファイルを送加 「「 ATU(20) 「「 YFO(20)	
		<u> (10) (10) 77 *1</u>	even

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]

💐 heikinchi - Microsoft '	Visual C++ 2010 Exp	r688				_ 0 ×
77イル(日) 編集(日) 表示()	0 JOI19KD FIR	99(D) 9-1KD	50150 WORLD	0		
1 🖾 • 🖾 • 🖾 🖉 🖉	2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	P Deblac	* 9652	* 2	- 蜀家於要問。	
[™]						🎌 ৩–০০৯৬৩৯
J	iEat(D)		画 新しい項目(図)	Ctrl+Shift+A		
	🖌 funutio(D)	OH+X		Shifi+All+A		
	13 JK-(V)	OFI+C	当新しいフィルター(E)			
	NUTUR)	Of/HV I	😽 552(Q)-			
	 A 単体の 本前の家軍(M) 	E2				
	(R) 70175-(R)					
		_				
1						
JUNITY heikinghi Officia	成功しました。					

・[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?」

Visual C++ 2008 Express Edition	X
このプロジェクトは変更されています(T):	
heikinchi - Debug Win32	_
) ビルドしますか?	
【まいパタパー】 いいえ(N) キャンセル	
□ 今後このダイアログを表示しない(D)	

[はい]をクリック ⇒エラーが無ければ、結果が表示されます。

📾 c:¥Documents and Settings¥1¥My Documen 💶 🗆 🗙
「= 2.5」が表示されたらokです 📃
平均値= 2.500000

※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)」

 \Rightarrow

[___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「D11main()」 が必要です、挿入する位置はどこでも構いません。

```
BOOL APIENTRY D11Main (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 D11Main (HANDLE hModule, DWORD ul_reason_for_call, LPVOID 1pReserved)
  {
//----
   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 ***」 エクスポートする***関数が「***」と言う名前で DLL ファイルに定義される・・必須 ;「VERSION ***」 DLL のバージョン、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



[次ヘ>] 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

備要 アプリケーションの続定	アガリテンシムの通知 「Windows アガリテンション(2) 「ロンシードアガリテンション(2) 「ロンシードアガリテンション(2) 「ロンシードアガリテンション(2) 「ロンシートアガリト 「ロンシートランション(2) 「ロンシートランション(2) 「コンンドリートラン・2) 「コンンドリートラン・2)	共通へるテーライルもあ200 「「ATL-90 「 ATL-90 「 ATL-90

- 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

heikinchi_dll - Microsoft Visual C	++ 2010 Express				10 X
ワイル(E) 編集(E) 表示(E) プロジェクト	(P) 开行外国) 步一队(D)	BURNA WORK			
(1)・回・回 日前、日本(1)	19 - Ci - 1 Deb.ac	* 90582	· · · · ·	• 122 37 27 28 4	
2012-932 128270-9 ● ● ● > ● ③ ● ③ ● ③ ● ○ ● ○ ● ○ ● ○ ● ○ ● ○ ● ○ ● ○					
	89) Bhiotrig		• \$ \$ 4 4	- 0 - 1	×
• J					

(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]

リューション エクスプロージ)	- 4 ×						
🔁 😡 🗔 Wit-Sex heiki	nchi t	uf 0 703						
🖨 🛄 heikinchi dl	-							
		iSto(D)	*		新以如何目的。	ChritShift+A		
	do	RIONED(I)	CHI+X	-	原作の項目(9)。	Shift+Alt+A		
	-	⊐e~(y)	Ctri+D		教しいフィルター(ビ			
	3	MSUPPORED	C4/HV	43	クラス(①).			
	^	-2.新小王軍(10)	20					
	En.	danta (n)	rx .					
3	-23	70777420						
		出力 1000-3	na (0)			1551		- 9
		出力 出力和	৫৯৯৩			• [3]	원구(순)] 2	- 1
		出力 出力元	0.\$ (2)		_	•131	월(종) 교	• 4
		цал Цалт	2호규(일)			• 191	2213 III.	• 4
		<u>ಆರಾ</u> ಚನಾನಾ	০৯নংহা			• [9]	23 3 I	- 1

- ・[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

💐 heikinchi_dll – Microsoft Visual C	++ 2010 Express			_ 5 >
ファイル(日) 東東(日) 表示(の) クロジェクト	(E) FREED SHALL DURING AND B			
i 🔂 • 🗇 • 🥔 🚚 🥔 & -5 🔠	*7 = C ² = ▶ Debus = Win82	- 10	· 🔍 🕈 🗶 🖬 🔮 🚽	
13 2 2 Ar = 2 .				
🔟 90a-9a9 16270-5- 🔹 🖗 🗙	hakinshi dilepa 🗙			- /
X 🛄 🧐 🖂	(ジローバル スコープ)	1		- 1
 ○ 別シーション beitwood aff で プロ: ○ De beitwood aff で プロ: ○ シーン フィイル ○ ローン・フィール - ○ ハジー フィイル - ○ ハジー フィイル - ○ ハジー フィイル - ○ ハジー スパー - ○ ハジー スパー 	<pre>Bit Constant State State</pre>	eli heikin(double a.double b e.00000 ui_reesen_for_celi.U	.dcuble c.double d){ PVOID (pReserved)	
	<u>出力</u>			- 9 ×
	出力元の表示(2)	- 3 23		
インウルード ファイルを解析しています。(20 / 38	-) - oʻ¥Frogram Files¥Miprosoft SDKs¥Windows¥v7DAVG	nskude¥MinGrypth 1117	1列 1文字	#入

- ・「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]

eikinchi_dll ナロパティ ページ			?
構成(<u>C</u>): アクティブ(Debug)	▼ プラットフォーム(P): アクティブ(Win32)	_	構成マネージャー(0)
	追加の依存ファイル すべての既定のライブラリの無視 特定の既定のライブラリの無視	kernel32.lib;user32.lib;gdi32.lib;winsp	ool.lib;comdlg32.lib;advaj
 - 全貌 - デバッグ - アバッグ - マントッティレクトリ - マントント - マンカー - マンカント - マンカスト ファイル - マンカスト ファイル - マンカスト ファイル - デバッグ - シンテム - マンカスト ファイル - デバッグ - シンテム - マンカスト ファイル - デジッグ - マンカント - マンカスト ファイル - デバッグ - マンカスト - マンカスト - マンカスト - マンカスト - マンファント - アンパント - マンアンド - マンド - デン - マンド - デン - アンド - アン - アンド - アン - アン	まジュールをやとっかしご意加 モジュールをやとっかしご意加 マネージリソースファイルの埋め込み シンボル参照の達制 DLLの遅度法が込み アセンブリリンクリソース	heikinchi_dll.def	
	モジュール定義ファイル /DEF オブションを使用すると、モジュール定義ファイル(っだけです。	idef) がリンカー(ご渡されます。 LINK (こ対して指定	ごできる .def ファイルは 1
	<u></u>	OK ***	セル (適用(<u>A</u>)

- (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] [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。
- ・「D11Main」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「D11Main」となります。
 リンカによっては、「D11Main」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「D11Main」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_dl1」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 <stdlib.h> #include <stdlib.h>

//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ブラリ名を記載する
#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

しいプロジェクト	and the second se	and the second		? ×
t近使用したテンプレート	並べ替え基準 [既定	• 00 💼	インストールされたテンプレートの検索	Q
ストールされたテンプレート Visual C++	Win32 コンソール アウルケーション	/ Visual C++	₩棘 Visual C++	
015 Mis£ ≰€2	_==	Vizail C++	Wet2 エゾール アガローシュンを作る 077072357でき	t1260
r (Petilischi ca (chicasunen Yaz-8(2) (Petilischi ca 7 ;	ll Is and settings¥'l¥ny dacuments¥visual stud II	o 2018¥Frojeste 💌	<u>参稿(D)</u> アリリューションのアイレクド9名作成(D) OK キャ	501
・「イン」	ストールされた	テンプレ		2]
	→「Win32 コン	ノール	アプリケーシ	зン
•「名前	:heikinchi_ca	11」を	打ち込む -	→ [C
7-01. A.A. M. R. B.				alvi
Win32 779	りょうそう ウィザードへようごそ			<u> </u>
実 りり∽-9 ₈ ンの新定	現在の7月3121492 ・ エンジール アプリケーション 現在の1992年年時代にすめには、第17をクリックして プロジェントの作用時代、プロジェントのreadmeted Co	(ださい。 ア・イルでプロジェクトの株計	始生成27イルに勝くる情報を参照して383	6

[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

P795	ーションの設定		
項要 97997 – 9≟200281世	 アフリケーションが得きま Wrotexe アフリケーションが マリン・レッアフリケーションが マリン・レッアフリケーションが マリン・レッアフリケーションが マリン・レッアフリレ マリン・レッアフリレ アンボーレアフリレ アンボーレアフレン アンボーレアン アン アン	北山へタネーファイルを活知 「「 ホTL(<u>@)</u> 「 ∀r-○(<u>m</u>	
		(物) 2002 元7	キャンセル

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]→

g nerkincen can - Microsoft visuar	C++ 2010 Express				
カイル(日) 編集(日) 表示(日) プロジェク	KE) FIGS(D) U-IKD	0-0400 AUX(B)			
CI-E-CA 2 3 3 4 23	Debus	* WinS2	* 🥝	- 医单次医型+	
1 2012-922 123.70-9 ● ● × 3 3 - 2012-93/ Netkinchi cell (1 7 - 1 10 23.774 / - 1 10 - 2.774 / - 3 49 - 2.774 / - 3 5 - 2.774 /	C.				
	2010				- 0.54
	出力元の表示(5) ビルド		• N (43		
				8 H	
					10-0

- (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。

💐 heikinchi_call - Nicrosoft Visual C++ 2	1010 Express					_ 0
ファイル(日) 編集(日) 表示(分) プロジェクト(日)	ディをが回り、シール(日)	10000 AU2(10				
1 1 · 1 · 2 4 4 4 2 2 1 ·	Carl & Debus	* 90582	· 🦉		· 风出关图]	£ -
Ul № % * Ξ ≌ ÷						
📕 シリューション エクスプローラー 🔹 中 🗙 📷 k	inchi_call.pp ×					
7 E 9 E	(グローバル スコープ)		-			-
Y12-222 Vehichtorf 0 70 Y2-222 Vehichtorf 7 Wehichtor 1 201 W-2 704/A W-2 704/	/// ⇒ //.2 The // DLL20+/.1 The Bar inde Scattle The	Neinehi_cell.spaj #A. AND MEAN #A. AND MEAN #A. AND MEAN #A. AND MEAN #A. AND MEAN #A. All Sell, FU #A. All Sell, FU #A. All Sell, FU #A. All Sell, FU #A. And MEAN #A. And	用するのに必要 たと含の方が列名をi 1150 call heikin(double teu; 566):	icttr¥o , double, double	, double);	÷
100	× • 1					
4421						* 1 ×
	DODRAU/PL DAY		*	0 2 3 3	<u>-</u>	100
						×
			and the state state of the	10.04	Container)
インウルードファイルを解析しています。(12 / 39) - e/	¥Program Files¥Microsof	t SDKs¥Mindoks¥v7CA¥	Include¥WinUserh	1行	1列 1女子	182

- (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?」

Microsoft Visual C++ 2010 Express	×
このプロジェクトは変更されています(T):	
heikinchi_call - Debug Win32	
, ビルドしますか?	
【【(\\?)】 (いいえい) キャンセル 「 今後このダイアログを表示しない(<u>D</u>)	

[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。

Alter State 2010 10000 10000 10000 10000 Alter State 2010 Alter 2010	The rest of the second	++ 2010 Express	AND REAL AND REAL				- 01 2
COLORED COLORE	SAME WHEN BUILD DISISN		- Brance	- 100 I	100	n	
Win-Syl 22070-5- *** Windthenlage X Windthenlagel C/D - Ak 20-70 Windthenlagel F// 77 / 7.5 This initial isol Windthenlagel F// 77 / 7.5 This initial isol <		Second Providence	- The sec	- 11-24			
第二日 100<	a canta Na Williama 🕈						
CO-rNA 20-70 CO-rNA 20-rNA 20-rN	>リューション エクスプローラー * ギ ×	heikinchi_call.pp ×					
Syn-2-2/ Nethod yolf 0.75		(グローバル スコープ)					-
Augusta and a second and second and a	ジリューシュン * behndu cell ① 7C ジーン・ション・ション・ ジーン・ション・ ジーン・ション・ ジーン・ション・ ジーン・ション・ ジーン・ション・ ジーン・ション・ ジーン・ション・ ジーン・ション・ ジーン・		alkinoh (cal.cal) Call (A) (A) (A) (A) Call (A) (A) (A) Call (A) (A) (A) A) A) A) A) A) (A) (A) (A)	用するのに必要 とときのみ(パックを記載する lis ¹) ail heikin(double, double eu; ce):	.comble, comble);		46 × 1
the second secon		uno n					-
Byth		100 % • 14					
Berthalden (2) EUM Seine (2) EUM		出力					+ ū ×
 		三方元の表示(5) ビルド	Renter and State to be	v g y⊥ Handhi call tëttr Dahar ∎ice	수 % 코		
▲ ▲ 登議元7 5行 1列 1大字 ##		heikinchi call.com heikinchi call.com for the filter for the filt	roi -> cifdocuments and しド: I 正常が了、D 失敗	settinus#I¥no documents¥visuul 07≢oJ =======	studio 2010¥Projecta	fheikinchi_ox ¥0ebo	uðheiki
建罐完了 5行 1列 1大平 排入		•			-		
	準備完了			5 fj	1 91	1 女宇	# 入

※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);

※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ブラリ名を記載する
#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 mainO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("平均值= %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 D11Main (HANDLE hModule, DWORD ul_reason_for_call, LPVOID 1pReserved)
  {
//--
   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」⇒



27/36

```
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]

heikinchi_dll プロパティ ページ			<u>? ×</u>
構成(<u>C</u>): アクティブ(Release)	ブラットフォーム(P): アクティブ(Win32)	•	構成マネージャー()
 ● 共通プロパティ ● 構成プロパティ ● 構成プロパティ ● 全般 ● デバッグ ● VC++ ディレクトリ ● C/C++ ● リンカー ● 全般 ● スカ ● マニフェスト ファイル ● ブバッグ ● ジステム ● 漫画化 ● 受加 ● マニフェスト ファイル ● 実行い次グ ● マニアエスト ファイル ● デバッグ ● マニアエスト ファイル ● ブバッグ ● マニアエスト ファイル ● マニアエスト マール ● マニアエスト マール ● マニアエスト マール ● マニアエスト マール ● アンド ライン ● マニアエスト マール ● ガルド オペント ● カスタム ビルド ステップ 	注創加の依存ファイル すべての既定のライブラリの無視 特定の既定のライブラリの無視 モジュール定義ファイル モジュール定義ファイル モジュール定義ファイル モジュール定義ファイル モジュール定義ファイル モジュール定義ファイル モジュール定義ファイル	kernel32.libuser32.libugdi32.libwinspo	ol.lib;comdlg32.lib;advapi
X D	/DEF オブションを使用すると、モジュール定義ファイル (def) が つだけです。	リンカーに渡されます。LINK に対して指定	できる .def ファイルは 1
		OK ++>	ゼル 適用(<u>A</u>)

<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]

neikinchi_dll フロパティ ページ			<u ?
構成(C): アクティフ(Release)	▼ ブラットフォーム(P): //クティフ(Win32) 文字列プール		構成マネージャー(<u>0</u>)
直 構成プロパティ	最小リビルドを有効にする	(いいえ (/Gm-)	
全般	C++ の例外を有効にする	(はい (/EHsc)	
デバッグ	小さい型への変換チェック	いいえ	
	基本ランタイム チェック	既定	
□····································	ランタイム ライブラリ	マルチスレッド (/MT)	-
	構造体メンバーのアライメント	既定	
プリプロセッサ	バッファー セキュリティ チェック	(t) (/GS)	
- コード生成	関数レベルでリンクする	(IC) (/Gy)	
	拡張命令セットを有効にする	設定なし	
- プリコンパイル済みヘッ)子動小艇点モナル 2015年1月8日をおりまたもとのサラ	Precise Vtp:precise/	
- 出力ファイル)チ動小数点の例外を有効にする		
	ホットパッナー 前記 はイメーン のパドルス		
-マニフェスト ファイル			
デバッグ			
ーシステム			
最適化			
- 理例)へみ IDL	ランタイム ライブラリ		
	リンクするランタイム ライブラリを指定します。	VMT, ZMTd, ZMD, ZMDd)	
	L		
		OK	適用(<u>A</u>)

- (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] [ソリューションのビルド]

heikinchi_dll - Microsoft Visual C++ 2	2010 Express					_ 5
MULE (1946) AT() 705191()	T/9500 9-00 9-00	A.J. (H)	11000			
1.01.02.04.09.03.02.02.02 1.01.02.04.09.03.02.02.02	Allan-Park/DPJLB(R)	F7	10	•11-2	ou ×. €0 85 €	
90a-9a916270-5- • # × he	X797 1X0 X797 1X0 X797 1X0 X797 1X0	F11 F10				-
March 20	プレーカポイントの設定の解除(の)	FR				-
E Sheikinchi dl	242/83200					-
En Go ソース ファイル □ □ □ Philippin dillore	オペアのデータビント参考リア(A)					
- 📴 ヘッダー ファイル	7-962-H0192#-H00					- 1
トー 💷 リソース ファイル	テータビントのインボードロー					- 11
er ga /renkritista	オブションと設定(8)。)	II helkin(double a.doubl	le b.dcuble c.double d	03	- 11
	HEOUL AFJERINY DITUSIN(HAND , svitch(ul_resson_for_ca case DLL_PROCESS_ATT cose DLL_TREAD &TTA 076 - [4]	LE NNOdule (II) ACH: CH:	,DVUKD ul_resson_for_cal	(,L'YOID (pReserved)		<u>ح</u>
90	<i>л</i>					+ 9 ×
u L	いたの表示(2) ビルド		• 5 2	14 🐳 🖬		
						-
1						2

・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

💐 heikinchi_call – Microsoft Visual	C++ 2010 Express			_ 5 X
ファイル(E) 実験(E) 表示(y) プロジェク	F(E) FIGSTO SHALD SCOPED AND B			
1 ⊡•⊡•⊘⊿ ∦≱55 ⊂355× 22,	V - C - Felaces - We32 Dobug Release	- 0	· N (2) (4) (2) (2) (3) (4) (4) (4) (4) (4) (4) (4) (4) (4) (4	
🗾 シリューション エクスプローラー 🔹 🖡 🗙	構成マネージャー。 hakinchi calleop ×			- 1/2
ジリューション (シス) (ジー)	Tekenorycallep × 203-54 A 23-7) 203-54 A 23-7) 1012-54 25 (Teke 140 JC34) 2017 2 4 25 (Teke 140 JC34) 2017 2 4 25 (Teke 140 JC34) 2016 2 5 (Teke 140 JC34) 2016 2 5 (Teke 140 JC34) 2016 2 5 (Teke 140 JC34) 2017 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	- するのに必要 までの対プ列名を記載する b) heikin(double, double, d ;	ioubis, doubis);	• • • • • • • • • • • • • • • • • • •
	Inter - A	00 00		
	AVA 11 101			
		1 1 5 1 1 S 1		
	Downorking Un	- 19 1903		×
· 1				E.
準備完了				

(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

heikinchi_call プロパティ ページ			<u>? ×</u>
構成(n)・ アクティブ(Release)	▼ プラットフォー J. (P) アクティブ(Win32)		▼ 構成マネージャー(∩)
構成(2): アクティブ(Release)	▼ ブラットフォーム(P): アクラィブ(Win32) 文字列リプール 最小リビルドを有効にする O++ の例外を有効にする 小さい型への変換チェック 基本ランタイム チェック ランタイム ライブラリ 構造(キンパーのアライジント パッファー セキュリティ チェック 閲覧レベルでリンクする 拡張命令セットを有効にする 注発動・数点モデル	(ハハえ //Gm-) (はい //EHsc) (ハハえ 既定 マルチスレッド //MT) 既定 はい //GS) はい //GS) はにい //Gy) 設定なし Precise //fpprecise)	▲構成マネージャー(Q)
 フリコンパイル済めへ999 -ソコンパイル済めへ999 -ブラウザー情報 - 芋や細設定 -コマンドライン ワンカー マンコント ツール マンコント ジェネレータ・ ウブラザー情報 ビルドイペント ロカスタム ビルド ステップ 	7年907193次市の例外を有効にする 注撃的小教長点の例外を有効にする ホットパッチ可能なイメージの作成		
· •	ランタイム ライブラリ リンクするランタイム ライブラリを指定します。	(/MT、/MTd、/MD、/MDd)	
		OK	キャンセル 適用(<u>A</u>)

(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]

heikinchi_call - Microsoft Visual C++	 2010 Express 				
ウイル(日) 電振(日) 表示(の) プロジェクト(日)	デバック(D) シール(D) ウィンドウ(M)	ヘルプ(日)			
0.0.00	▶ 37(5)00086(S)	F5	- 3	- ସ୍ ମ ହ ଲ	<u>11</u> +
	※ ソリューションのビルド(E)	F7			
995-962 IOZ70-9- 🔹 🕂 🗙 🔥	al 🧐 2707 (1XQ)	F11			
12 (Q	- 💭 2797 1-11-00	F10			
💭 WI-930 heisinchi sall († 70	ブレークポイントの設定/解除(0)	F9			+
EF (2) Neikinchi cali EF (2) ソース 77イル	ウインドウンジ				-
- 😋 heikinchi pallopp	すべてのデータヒントをクリア(色)				
	データビントのエクスポートへの		するのに必要		
····································	テータビントのインボードビー		o environmente		
	オブションと設定(点)。		ときのライブジョ名を記載する		
	#pragma comment(lib, "heik	inchi dilla	ib")		
	dauble open,high,low, open2.0; high=3.5; low=1.5; close=3.0; kaitau=heikin(open,hi	close,kaito sh,los,clos	a);		×
9	57				- 9 ×
	出力元の表示(2) 2018		- 3 .23		
					14
					F
CTC 7			1行	1列 1文字	10

・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。

C:¥Docun	nents ar	nd Setting	s¥1¥j	デスクトップ	/¥●プロジュ	ኃኑ ኛ	刀稿¥章	t;DLLØf	'ፑり [<u>- 0 ×</u>
ファイル(E)	編集(E)	表示₩	お気()	こ入り(<u>A</u>)	ツール①	ヘルブ	Ω.			
🕞 戻る 🔹	•	🏂 🔎	検索	🔁 ७७	มร 🛛 😭	Ø	$\boldsymbol{\times}$	9	•	
heikinchi_dll.dll				heikinchi_callexe						
 2 個のオブジェク	ŀ				110	КВ	- 9	קר באני	ב-ק	

- (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。

📾 c:¥Documents and Settings¥1¥My Do 💶 💌
「= 2.5」が表示されたらokです 🛛 🔺
平均値= 2.500000 🚽
-

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



<u>以 上</u>