インテル® C++ コンパイラー 10.1

日本語版 スペシャル・エディション

ー 入門ガイド ー

エクセルソフト株式会社
www.xlsoft.com

ー　目次　ー

1．はじめに

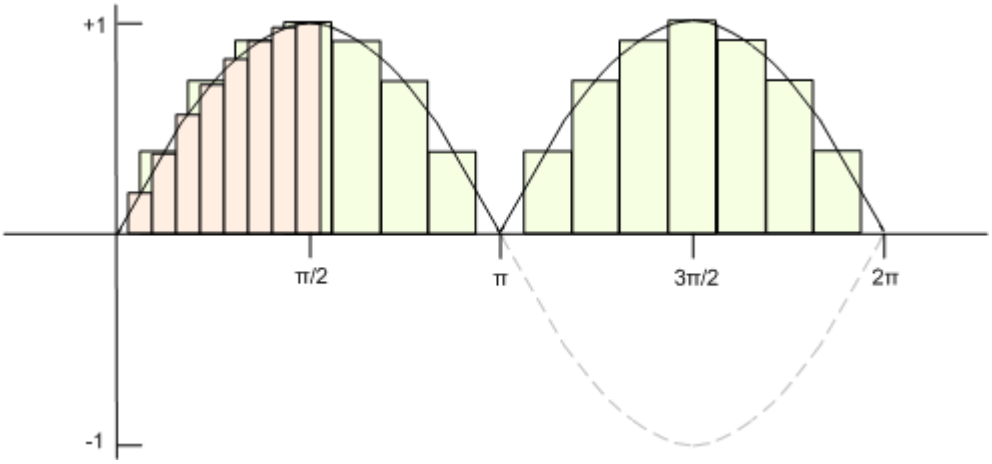
インテル® C++ コンパイラーのインストールが完了したら、適切なインストール、設定、およびコンパイラーの基本動作を確認するため、本ドキュメントで説明する動作検証を行ってください。この動作検証は、コマンドラインおよび Microsoft* Visual Studio* 統合開発環境 (IDE) を使用して行います。なお、この検証では IA-32 対応アプリケーション用インテル(R) C++ コンパイラー、および以下に示すサンプルコードを使用します。

<install-dir>¥Compiler¥C++¥10.1.xxx¥samples¥optimize¥int_sin.c

> **注：** <Install-Dir> は、インテル® C++ コンパイラーのインストール・パスです。デフォルトでは、"C:¥Program Files¥Intel" となります。また、"xxx" はマイナーバージョンを示しています。

1－1．検証用サンプルコード

検証用サンプルコードは、1 サイクル 2π ラジアンの正弦曲線の絶対値を積分する数値演算プログラムです。次の図は、計算に使用される方法を示しています。この方法は、曲線と上辺の中央部分が一致するように長方形を連続的に追加します。長方形の数が増えると (長方形の幅が狭くなると)、計算される領域は 4 (4.0) に近づきます。次の図は、2^4 内点と 2^5 内点の最初の 8 片で何が計算されているかを示しています。



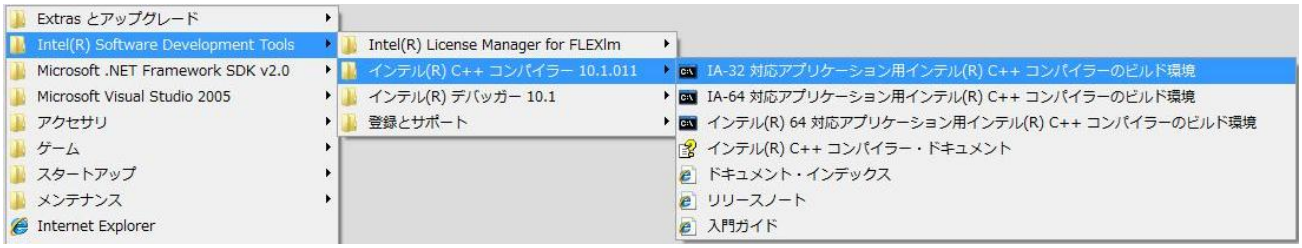
検証用サンプルコードをコンパイルして実行し、出力が既知の正しい値である 4 に収斂するかどうかをチェックすることでコンパイラーが適切にインストールされたかどうかを確認できます。またこのサンプルコードには、stdio、stdlib、時間ライブラリーに加えてインテルの数値演算ライブラリーが含まれているため、インテル・ライブラリーが正しくインストールされているかどうかも確認することができます。サンプルコード内の時間関数はプログラム実行の開始から終了までを測定したアプリケーション・クロックの数を返します。

2. コンパイラーの実行 (コマンドライン)

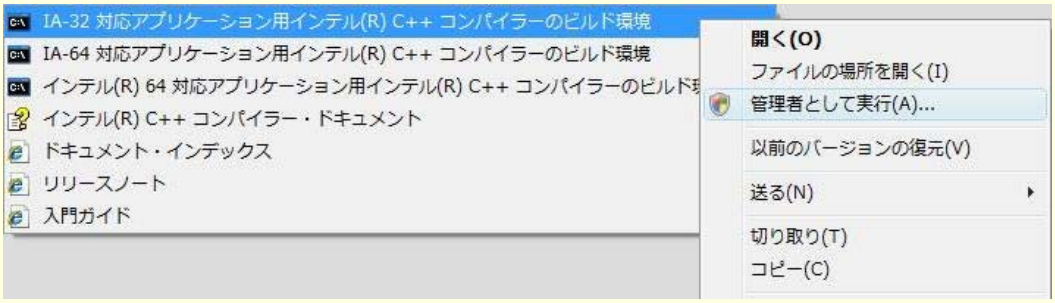
インテル® C++ コンパイラー 10.1 Windows 版は、icl コマンドを使用してコマンドラインから実行できます。ここでは、前述した int_sin.c 検証用サンプルコードを使用します。作業の大部分をコマンドラインからではなく、Microsoft* Visual Studio* IDE を使用して行っている場合でも、このセクションをスキップせず動作検証を行うことをお勧めします。

それでは、以下の手順に従ってコマンドラインにおける動作検証を行ってください。

1. Windows スタートメニューから [プログラム] > [Intel(R) Software Development Tools (インテル(R) ソフトウェア開発ツール)] > [インテル(R) C++ コンパイラー 10.1.xxx] > [IA-32 対応アプリケーション用インテル(R) C++ コンパイラーのビルド環境] を選択して、コマンド・ウィンドウを開きます。このウィンドウでは、環境変数（PATH、LIB、INCLUDE）の設定が完了しています。これは、コマンドウィンドウ起動時に、“<install-dir>¥Compiler¥C++¥10.1.xxx¥bin¥iclvars.bat” が内部で実行されるからです。



注：Microsoft* Windows Vista* を使用している場合は、下図のようにショートカットを右クリックして表示されるメニューから “管理者として実行” を選択してください。



2. カレントディレクトリを int_sin.c 検証用サンプルコードが存在するフォルダーまで移動します。

```
prompt> cd <install-dir>¥Compiler¥C++¥10.1.xxx¥samples¥optimize
```

２－１．コンパイル（最適化オプションなし）

最初に、最適化オプションを使用しないでコンパイルし、パフォーマンスの基準を確認します。次のようにインテル C++ コンパイラーを実行し、サンプルコードをコンパイルしてください。

```
prompt> icl /Od int_sin.c
```

注：インテル® C++ コンパイラーは、特にオプションを指定しない場合、デフォルトで最適化オプション（/O2）が有効となります。そのため、最適化なしでコンパイルする場合はオプション（/Od）を付加してデフォルトの最適化オプションを無効にする必要があります。なお、/Od などの “O” は大文字アルファベットのオーです。これは、Optimization（最適化）の頭文字です。

また、以下のように /Zi デバッグ・オプションを使用してもかまいません。この場合は、デフォルトの最適化オプションが /O2 から /Od に自動で変更されます。

```
prompt> icl /Zi int_sin.c
```

注：これらのコンパイルが正常終了しない場合は、Visual Studio* などのビルド環境が正しくインストールされていない可能性があります。インテル® C++ コンパイラーのサポートするビルド環境を確認の上、コンパイラーのコンポーネントをすべてアンインストールしてビルド環境の再インストールからやり直してください。

２－２．実行/プログラムの検証

実行プログラムは、サンプルコードと同じディレクトリに int_sin.exe という名前で生成されます。次のようにプログラムを実行します。

```
prompt> int_sin
```

各計算で消費される実行時間 (プロセッサー・クロック・サイクルの数) は、内点の数が増えると、計算された整数値 4.0 に近く (または等しく) なります。プログラムを実行すると、次のような出力が表示されます。

```
    Number of Interior Points | Computed Integral |
------------------------------------------------
                4             |   3.141593e+000   |
------------------------------------------------
                8             |   3.792238e+000   |
------------------------------------------------
               16             |   3.948463e+000   |
```

32	3.987141e+000
64	3.996787e+000
128	3.999197e+000
256	3.999799e+000
512	3.999950e+000
1024	3.999987e+000
2048	3.999997e+000
4096	3.999999e+000
8192	4.000000e+000
16384	4.000000e+000
32768	4.000000e+000
65536	4.000000e+000
131072	4.000000e+000
262144	4.000000e+000
524288	4.000000e+000
1048576	4.000000e+000
2097152	4.000000e+000
4194304	4.000000e+000
8388608	4.000000e+000
16777216	4.000000e+000
33554432	4.000000e+000
67108864	4.000000e+000

Application Clocks = 1.095100e+004

２－３．コンパイル（最適化オプションあり）

インテル® C++ コンパイラーにはたくさんの最適化オプションが用意されています。これらの最適化オプションを使用してプログラムのパフォーマンスを向上させることができます。ここでは、次のようにデフォルトの最適化オプションを使用してコンパイラーを実行します。

```
prompt> icl int_sin.c
```

デフォルトでは、コンパイラーはコードの実行速度を優先するレベル 2 の最適化 (/O2) を行います。

２－４．実行/パフォーマンスの比較

次のように、最適化された int_sin プログラムを実行します。

```
prompt> int_sin
```

最適化を行わなかった場合と、アプリケーション・クロックの数を比較します。実際の時間の差は使用するアーキテクチャーに依存します。

```
                :
                :
-------------------------------------
     2097152        |    4.000000e+000    |
-------------------------------------
     4194304        |    4.000000e+000    |
-------------------------------------
     8388608        |    4.000000e+000    |
-------------------------------------
    16777216        |    4.000000e+000    |
-------------------------------------
    33554432        |    4.000000e+000    |
-------------------------------------
    67108864        |    4.000000e+000    |

    Application Clocks      = 3.962000e+003
```

注：この例における (最適化なしから最適化ありにした場合の) 実行時間の大幅な向上はすべてのプログラムにあてはまるものではありませんが、通常は、適切な最適化を行うことで、インテル(R) プロセッサー上で実行するプログラムの実行時間を向上できます。インテル® C++ コンパイラーは、デフォルトでは /O2 レベルでプログラムを最適化する点に注意してください。

２－５．コンパイル（最適化オプションあり）－ その２

ここでは、さらに有効な最適化オプションを２つ試してみます。
１つ目は、/O2 よりもさらに強力な /O3 オプション。
２つ目は、ベクトライズと呼ばれるオプション /QxP です。このベクトライズ・オプションは使用するシステムに搭載されるCPU に対応して、オプション名の最後のアルファベット（この例では “P”）が、S、T、O、B、Nなどに変わります。例えば、この例のように /QxP を使用した場合は、一般的な インテル® Pentium 4 または Pentium D を搭載したシステム、 /QxT を指定した場合は、インテル® Core™2 Duo などのシステム、また /QxS は最新インテルCPU（SSE4 機能などをもつCPU）に対応します。この例では /QxP を使用しますが、ご使用の CPU に対応したオプションを使用してもかまいません。ベクトライズ・オプションに関する詳細はインストールされるコンパイラー・マニュアルを参照してください。

```
prompt> icl /O3 /QxP int_sin.c
```

注：このベクトライズ・オプションは、指定した CPU の “SSE 命令” を使用したコードを生成します。例えば、/QxT を使用した場合は、インテル® Core™2 Duo プロセッサーの SSE3 命令などが使用されます。そのため、/QxT で生成された実行プログラムは、インテル® Core™2 Duo 以上のプロセッサーを搭載したシステムでは実行可能ですが、一般的なインテル® Pentium 4 プロセッサーのシステムでは実行させることができませんので注意が必要です。

２－６．実行/パフォーマンスの比較 － その２

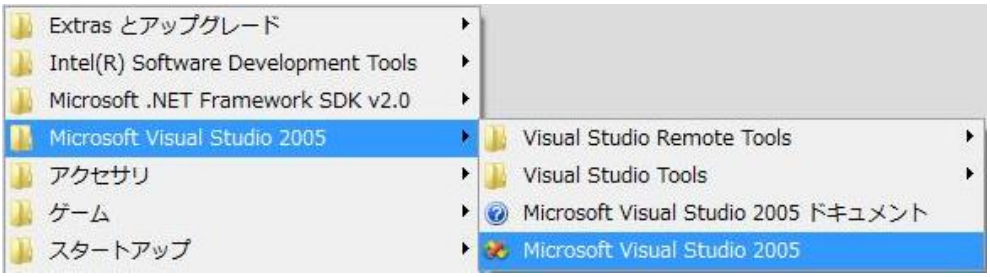
それでは、実際にベクトライズされた int_sin プログラムを実行し、結果を比較してください。

```
prompt> int_sin

            :
            :
---------------------------------------
     8388608        |    4.000000e+000    |
---------------------------------------
    16777216        |    4.000000e+000    |
---------------------------------------
    33554432        |    4.000000e+000    |
---------------------------------------
    67108864        |    4.000000e+000    |

    Application Clocks     = 1.981000e+003
```

3. コンパイラーの実行（Microsoft* Visual Studio* IDE）

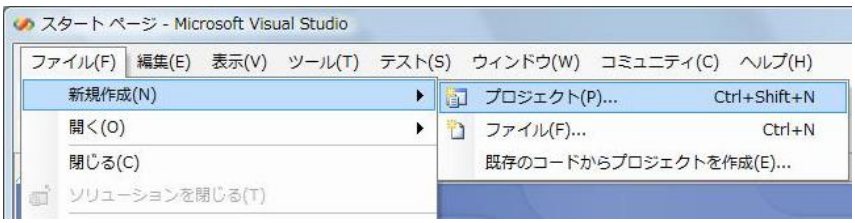
インテル® C++ コンパイラーをMicrosoft* Visual Studio* 環境で使用する手順を説明します。Microsoft* Visual Studio* 環境を使用する場合は、まずプロジェクトを作成し、ビルド環境を設定する必要があります。ここでは、Microsoft* Visual Studio* 環境として、Visual Studio 2005（以下、VS2005）を使用して説明します。なお、サンプルコードはコマンドライン同様、“int_sin.c” を使用します。
それでは、以下の手順にしたがって Microsoft* Visual Studio* 環境における動作検証を行ってください。

1. まず、Windows スタートメニューから VS2005 を起動します。

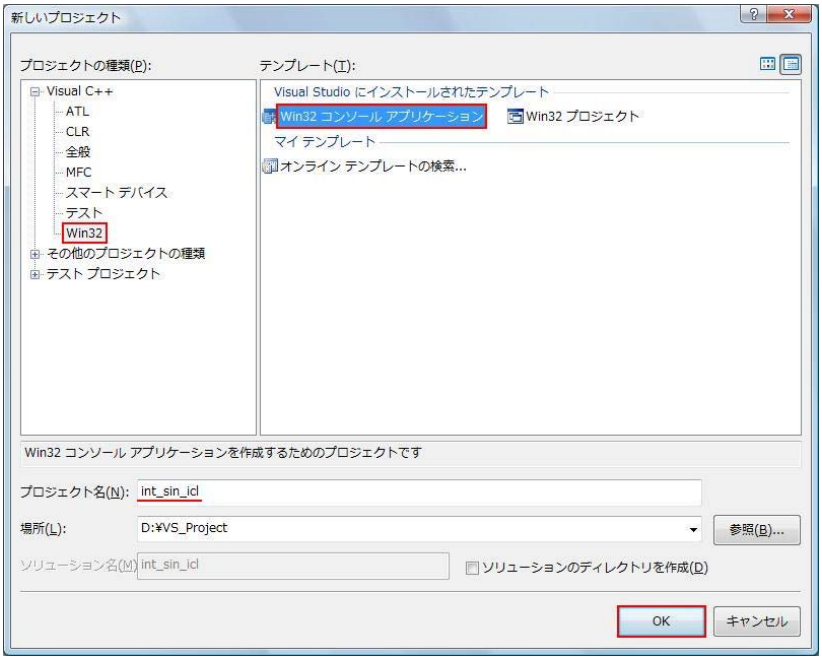


注：Microsoft* Windows Vista* を使用している場合は、下図のようにショートカットを右クリックして表示されるメニューから “管理者として実行” を選択してください。

2. VS2005 のメニューから、[ファイル] - [新規作成] - [プロジェクト] を選択して [新しいプロジェクト] ダイアログを表示します。下図に示すように、[プロジェクトの種類] で [Win32] を選択し、 [テンプレート] で [Win32 コンソールアプリケーション] を選択します。プロジェクト名として int_sin_icl を指定して [OK] ボタンをクリックします。なお、プロジェクトを作成する “場所” は任意でかまいません。

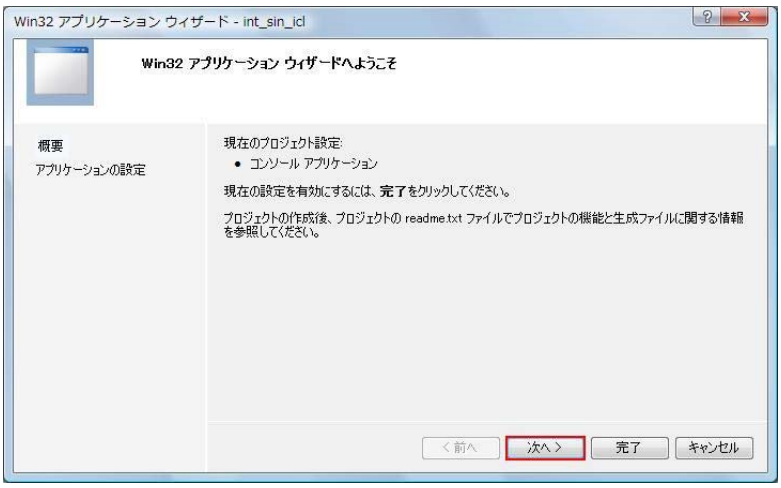


図：新しいプロジェクト



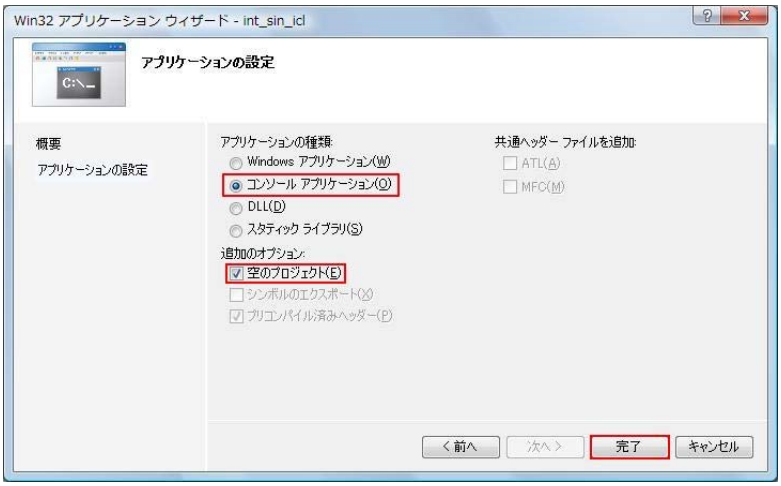
3. [Win32 アプリケーション ウィザード] が表示されるので、[次へ] をクリックします。

図：Win32 アプリケーション ウィザード

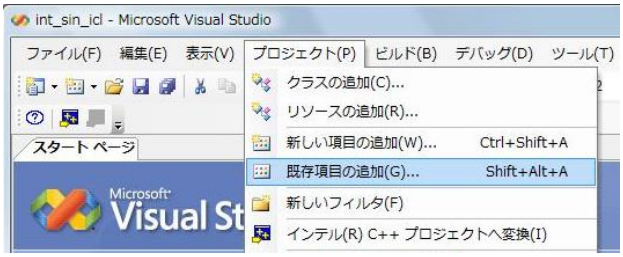


4. [アプリケーションの設定] ダイアログで、下図のように [コンソール アプリケーション] および [空のプロジェクト] が選択されていることを確認し、[完了] をクリックして "int_sin_icl" プロジェクトの作成を完了します。

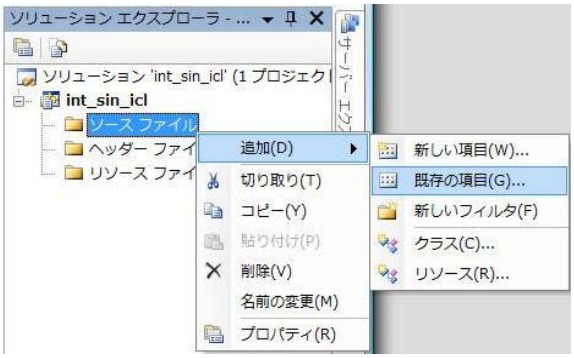
図：アプリケーションの設定



5. 作成したプロジェクトにサンプルコード（int_sin.c）を追加します。メニューから [プロジェクト] - [既存項目の追加...] を選択、または [ソリューション エクスプローラ] から “ソースファイル” を右クリックして表示されるメニューから [追加] - [既存の項目] を選択します。



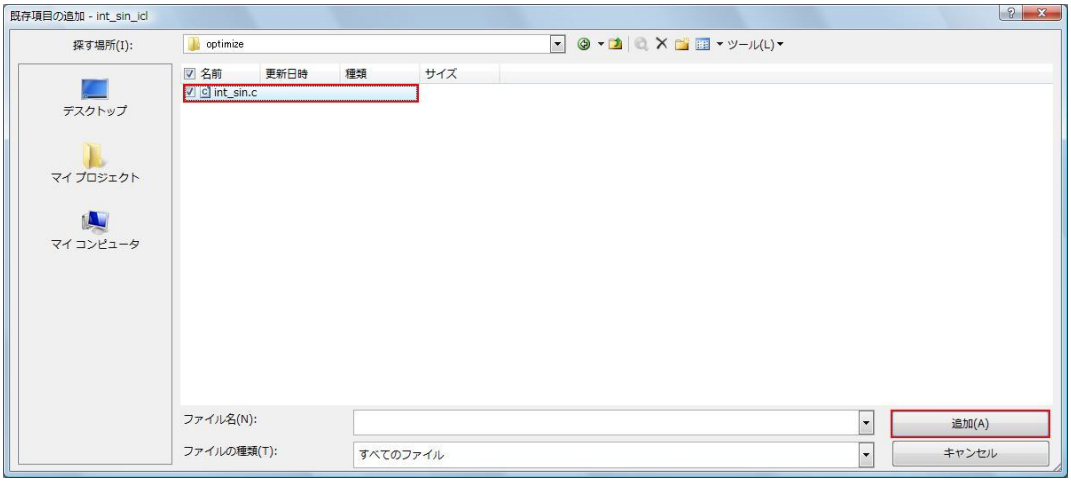
または、



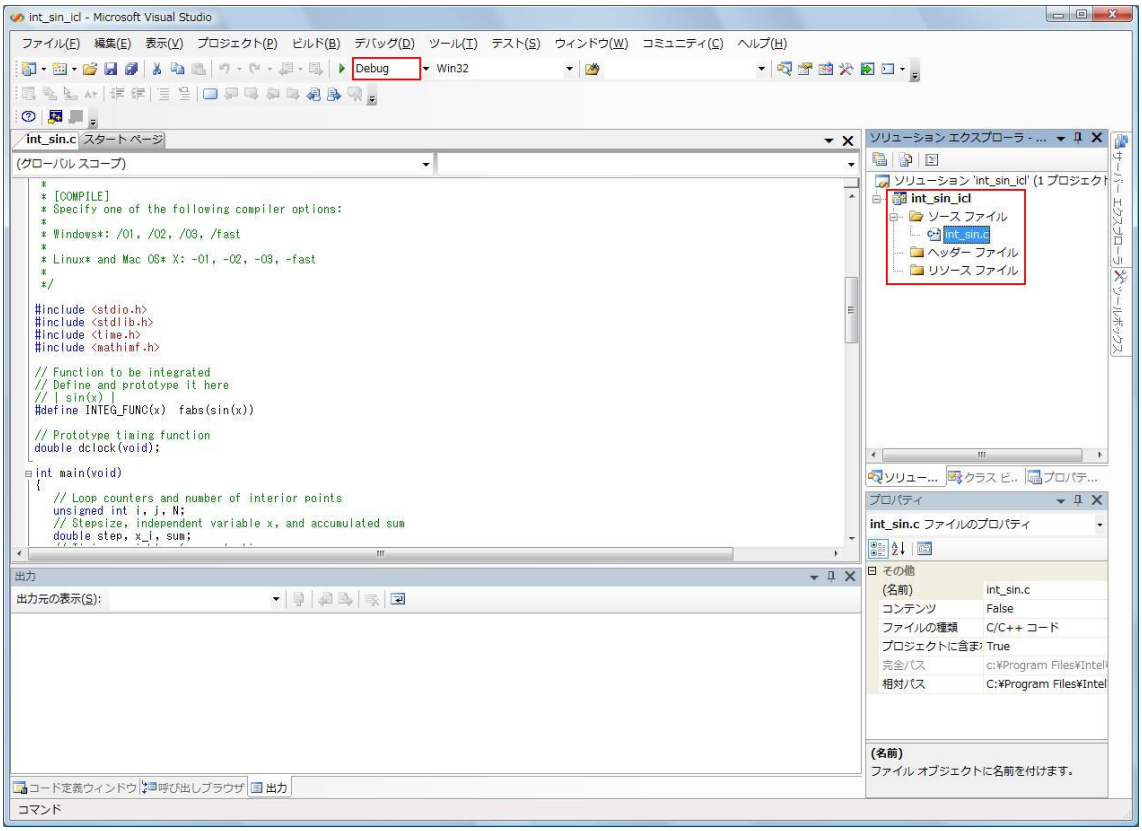
表示される [既存項目の追加] ダイアログで以下のサンプルコードを選択して [追加] ボタンをクリックします。

<install-dir>¥Compiler¥C++¥10.1.xxx¥samples¥optimize¥int_sin.c

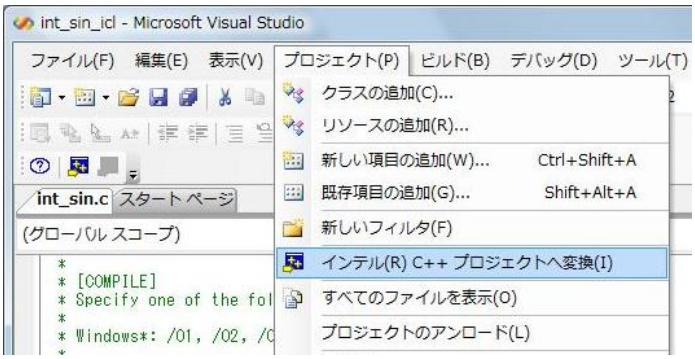
図：既存項目の追加



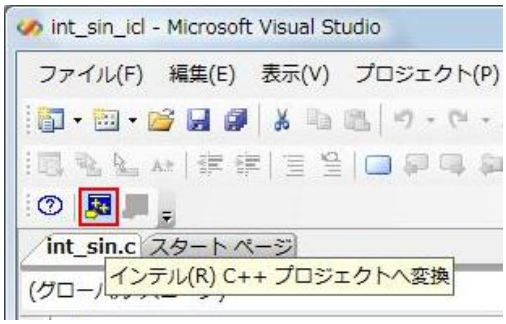
6. 新しいプロジェクト int_sin_icl の “ソースファイル” に、サンプルコード “int_sin.c” が追加されたことを確認します。また、プロジェクト構成が “Debug” に設定されていることも確認してください。



7. 次に、プロジェクトの変換を行います。 [プロジェクト] メニューから [インテル(R) C++ プロジェクトへ変換] を選択、またはインテル(R) C++ ツール バーの [インテル(R) C++ プロジェクトへ変換] ボタンをクリックします。



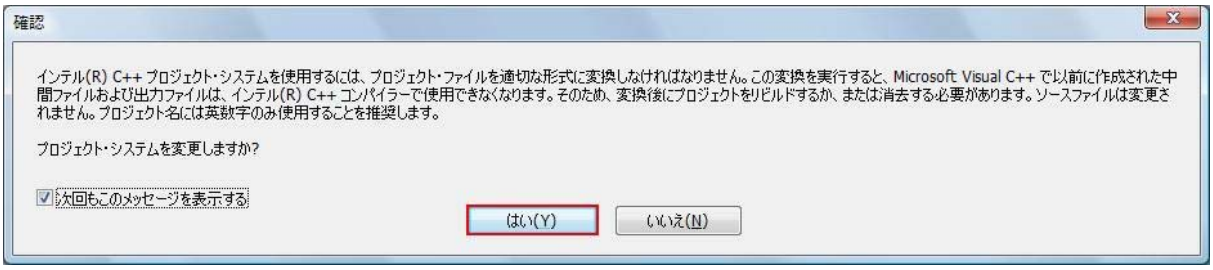
または、



注 : [ソリューションエクスプローラ] からプロジェクトを右クリックして表示されるメニューから、[インテル(R) C++ プロジェクトへ変換] を選択することもできます。

8. 表示される [確認] ダイアログで [はい] をクリックします。 変換が成功すると Microsoft Visual C++ プロジェクトがインテル C++ プロジェクトに変換され、新しいインテル C++ プロジェクト・ファイル (.icproj) が作成されます。また、ソリューション エクスプローラー内にインテル(R) C++ プロジェクトのロゴが表示されます。

図：確認

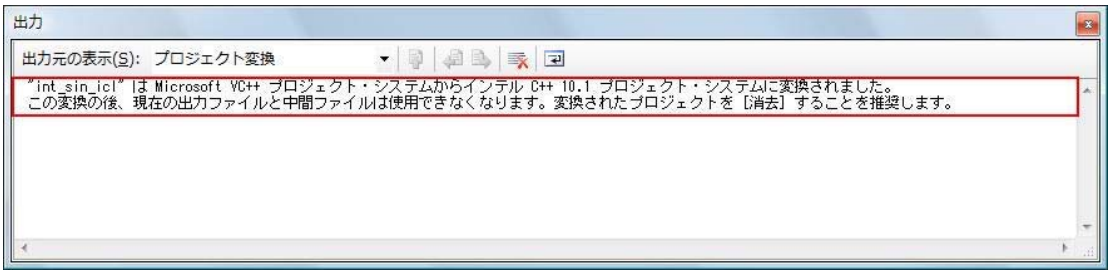


図：Microsoft Visual C++ プロジェクト

プロジェクトの変換

図：インテル C++ プロジェクト

プロジェクトの変換の結果は、[出力] 画面にも表示されます。

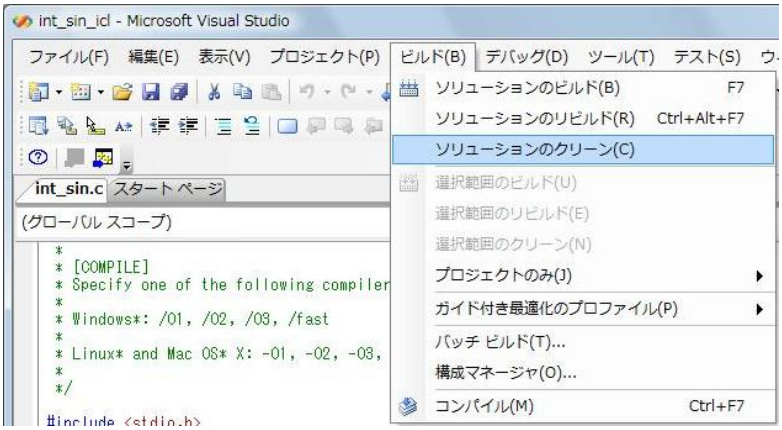


注：インテル® C++ コンパイラーでは、.NET プロジェクトのようなマネージドコードを生成するプロジェクトをサポートしていないため、これらのプロジェクトは変換できません。

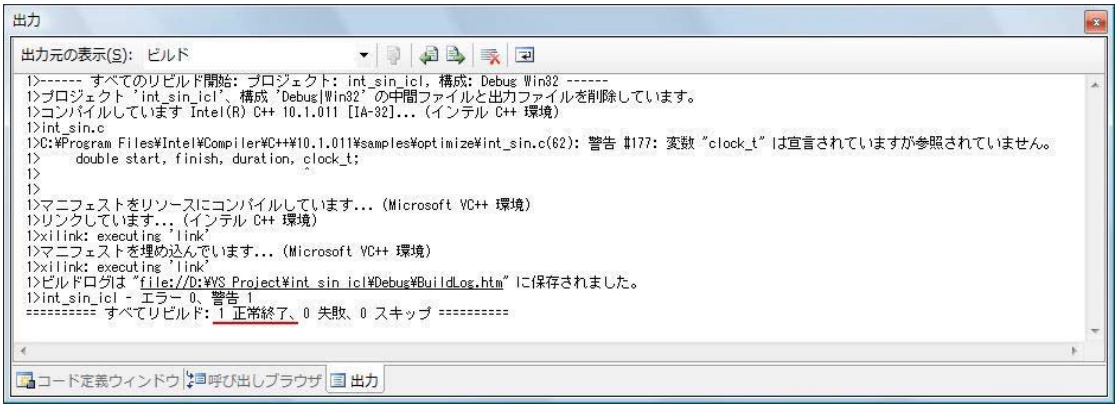
３－１．ビルド（最適化オプションなし）

まず、最適化オプションなしでビルドを行います。次の手順を実行します。

1. 作成したプロジェクトをビルドする前に、プロジェクトの内容を初期化します。VS2005 のメニューから、[ビルド] - [ソリューションのクリーン] を選択します。



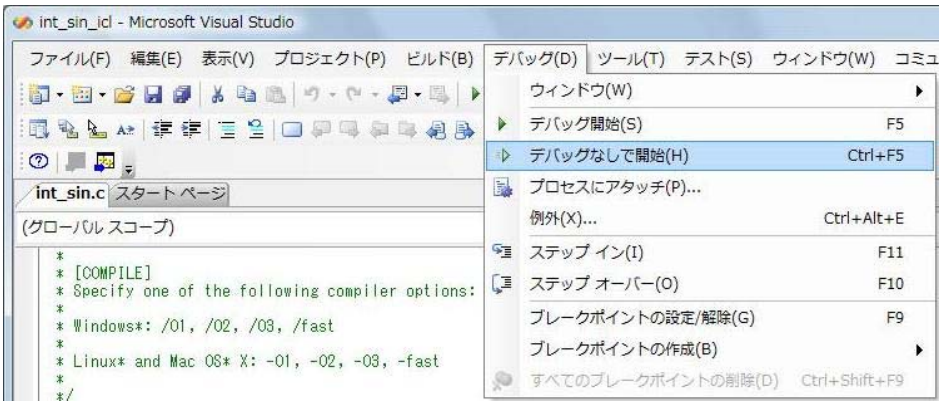
1. 次に、[ビルド] - [ソリューションのビルド] を選択します。ビルド結果が表示されます。正常終了していることを確認してください。

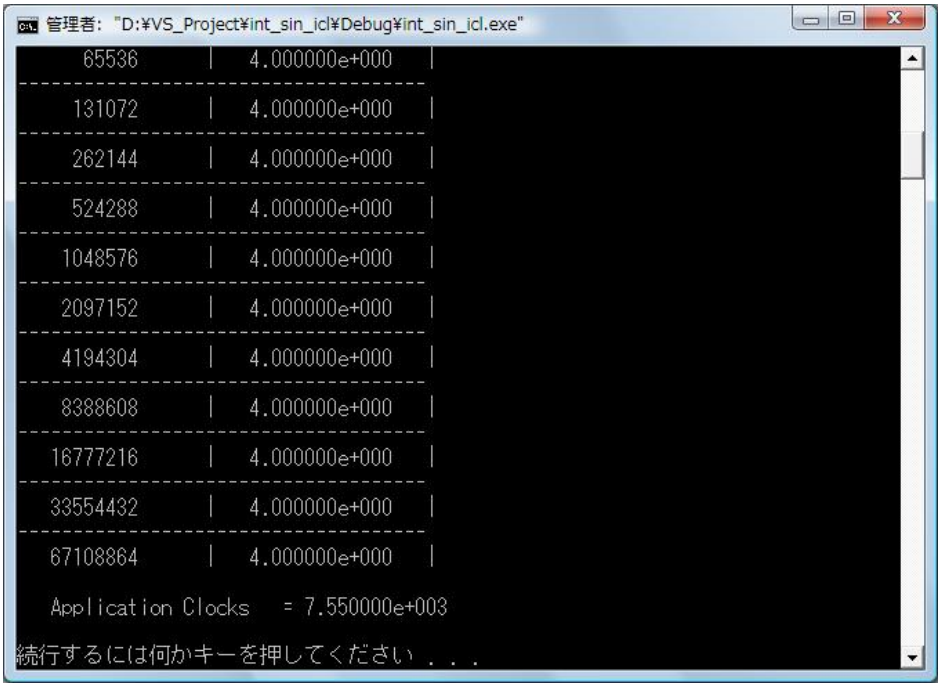


> 注：プロジェクトには通常、Debug 構成と Release 構成という２種類のプロジェクト構成（ビルド設定環境）が用意されています。一般的に開発中のプロジェクトは Debug 構成で作業を行い、開発が完了した製品を Release 構成でビルドします。デフォルトのプロジェクト構成は Debug 構成で、プロジェクトは最適化なし、シンボリック・デバッグ情報付きでビルドされます。これはコマンドラインから、 icl /Od /Zi int_sin.c と入力した場合とほぼ同じです。

３－２．実行/プログラムの検証

1. VS2005 のメニューから、[デバッグ] - [デバッグなしで開始] を選択します。コマンド・ウィンドウにプログラムの実行結果が表示されます。





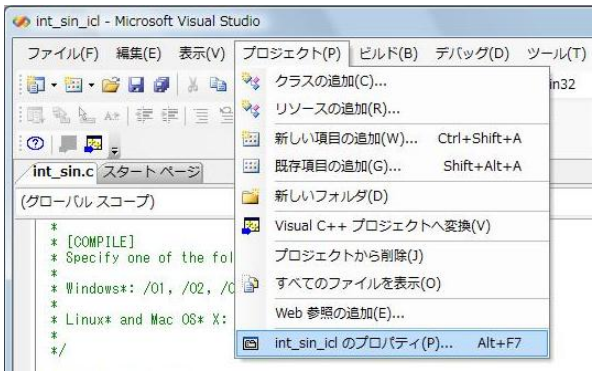
2. プログラム実行に使用された CPU 時間をメモします。

> 注： Microsoft* Visual Studio* 環境による実行が正常に行われない場合は、環境設定に問題があります。一度すべてのインテル® C++ コンパイラーのコンポーネントをアンインストールして、以下の『インストールガイド』を参考に、カスタムインストールをお試しください。
> http://jp.xlsoft.com/documents/intel/cwin/ICC_J_Install.pdf

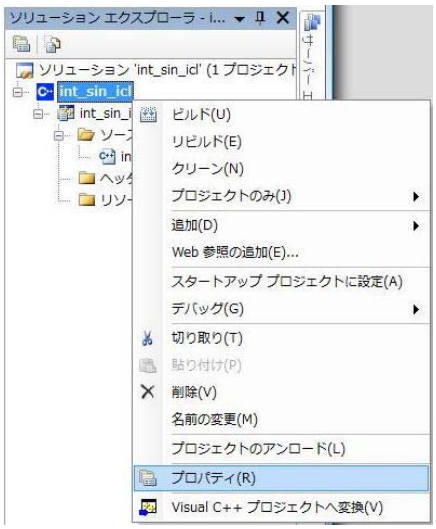
３－３．コンパイル（最適化オプションあり）

今度は、最適化オプションを使用してビルドを行います。次の手順を実行します。

1. まず、[プロパティ ページ] ダイアログを表示させます。VS2005 のメニューから [プロジェクト] - [プロパティ] を選択します。または [ソリューション エクスプローラ] 内のプロジェクト（int_sin_icl）を右クリックして [プロパティ] を選択します。

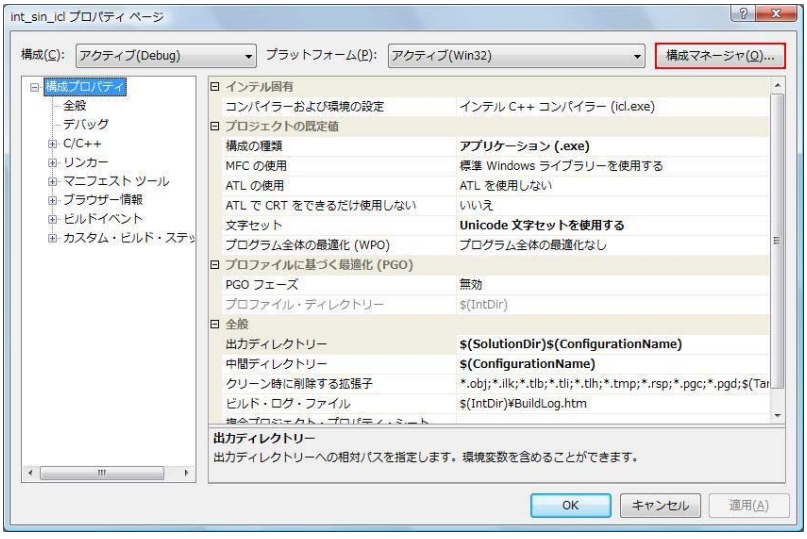


または、



表示される [プロパティ ページ] ダイアログ で、[構成マネージャ...] ボタンをクリックして、 [構成マネージャ] ダイアログを表示させます。

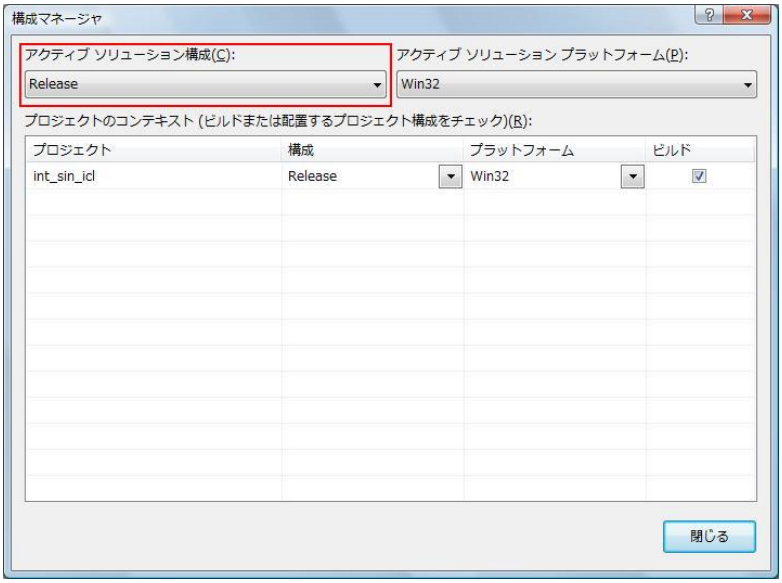
図：プロパティ ページ



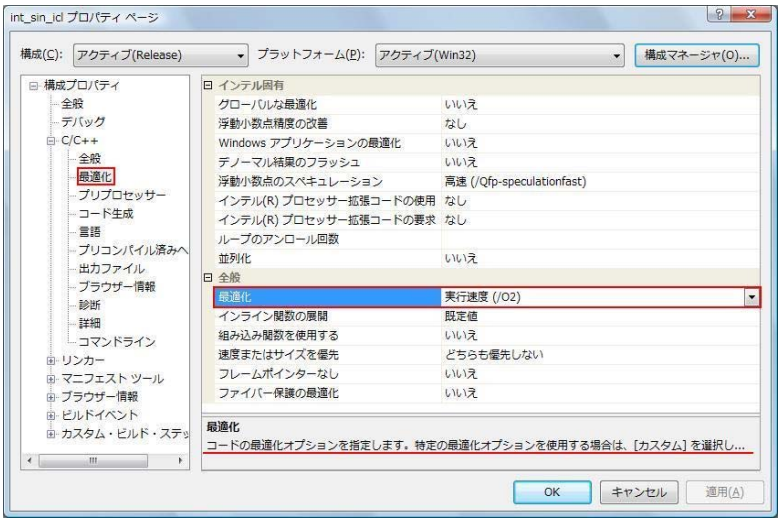
注：この [プロパティ ページ] ダイアログは、コンパイル・オプションの設定やインクルードファイル、ライブラリーファイルの指定など、プロジェクトのビルドに関するさまざまな設定を行う非常に重要なダイアログです。また、この [プロパティ ページ] は、Debug／Release それぞれのプロジェクト構成において個別のプロパティページを持っています。

2. [構成マネージャ] ダイアログで、[アクティブ ソリューション構成] を “Debug” から “Release” に変更し、 [閉じる] ボタンをクリックします。

図：構成マネージャ



3. [プロパティ ページ] ダイアログに戻り、[構成プロパティ] - [C/C++] - [最適化] を選択して、[最適化] が “実行速度(/O2)” に設定されていることを確認します。

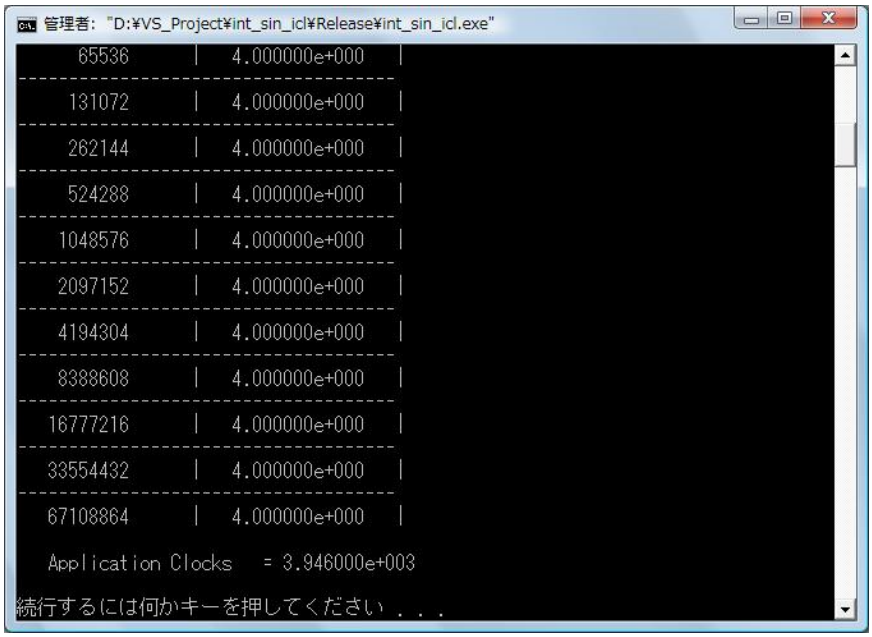


4. [ビルド] - [ソリューションのクリーン] を選択してから、[ビルド] - [ソリューションのビルド] を選択して、“Release” 構成で int_sin_icl プロジェクトをビルドします。

> 注：プロジェクト構成を Debug 構成から Release 構成に変更すると、最適化が有効となります。Release 構成において、デフォルトの最適化オプションは “実行速度”、つまり “/O2” が設定されています。

３－４．実行/パフォーマンスの比較

1. [デバッグ] - [デバッグなしで開始] を選択します。コマンド･ウィンドウに次の出力が表示されます。



2. 最適化を行った場合の CPU 時間をメモして、最適化を行わなかった場合と比較します。

> 注：この例における (最適化なしから最適化ありにした場合の) 実行時間の大幅な向上はすべてのプログラムにあてはまるものではありませんが、通常は、適切な最適化を行うことで、インテル(R) プロセッサー上で実行するプログラムの実行時間を向上できます。インテル® C++ コンパイラーは、デフォルトでは /O2 レベルでプログラムを最適化する点に注意してください。

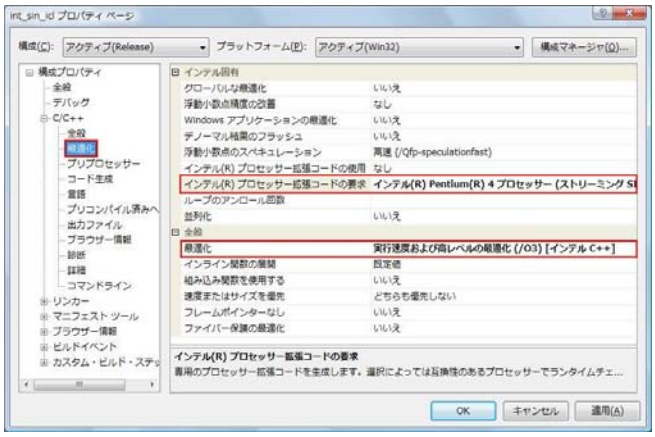
３－５．コンパイル（最適化オプションあり）ー その２

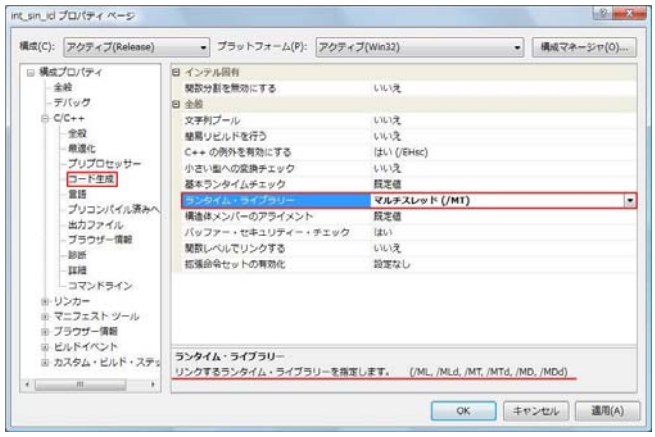
コマンドライン同様、ここでもさらに最適化を行ってみましょう。使用する最適化オプションは、/O3 およびベクトライズ・オプションの /QxP を適用します。

1. プロジェクトの [プロパティ ページ] ダイアログを開き、[構成プロパティ] - [C/C++] - [最適化] を選択して、下図のように [最適化] の値を "/O3"、 [インテル(R) プロセッサー拡張コードの要求] を "/QxP" に設定します。なお、ベクトライズの最適化オプションは、ご使用の CPU に対応したオプションを使用してもかまいません。ベクトライズ・オプションに関する詳細はインストールされるコンパイラー・マニュアルを参照してください。

また、[構成プロパティ] - [C/C++] - [コード生成] を選択して、下図のように [ランタイム・ライブラリー] の値を “/MD” から “/MT” に変更してください。この変更によりリンクするランタイム・ライブラリーが動的ライブラリー（.dll）から静的ライブラリー（.lib）に変わります。本検証で使用しているサンプルコード（int_sin.c）はループ内でインテル数値演算ライブラリーを使用しており、このループをベクトライズするためにこの変更を行います。

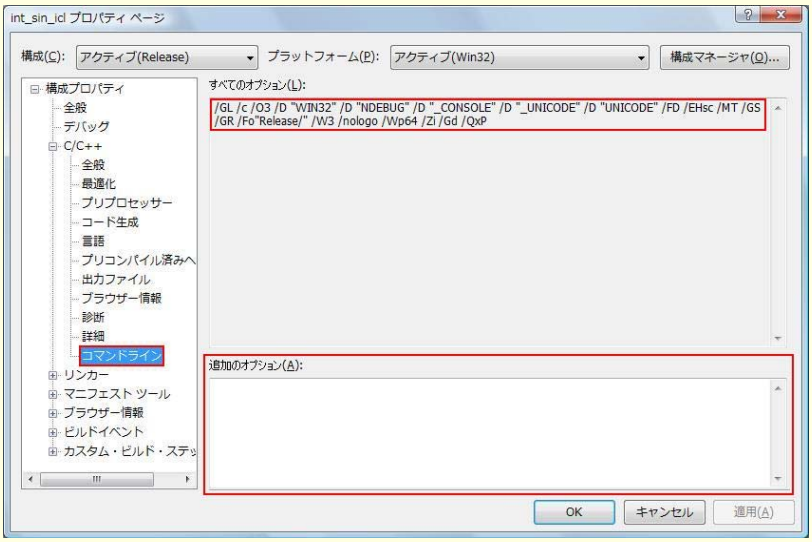
図：プロパティ ページ





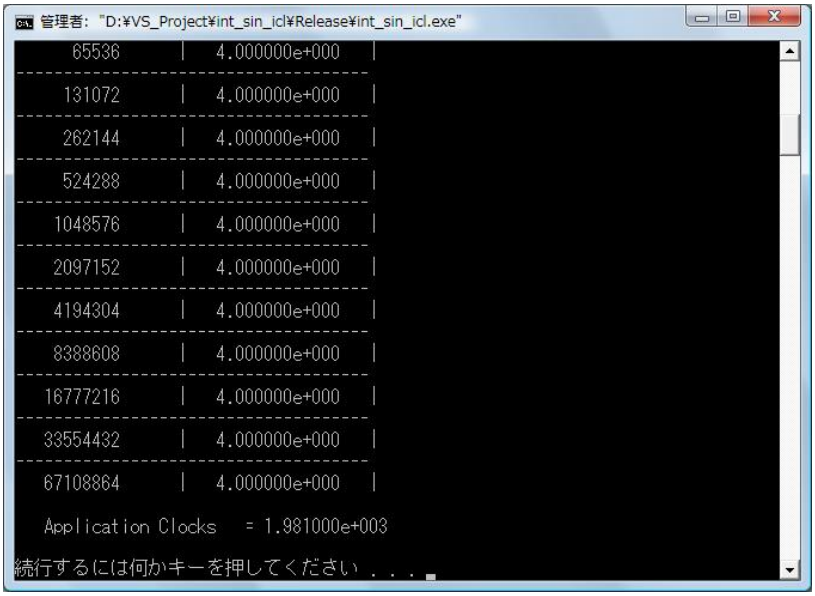
2. [ビルド] - [ソリューションのクリーン] を選択してから、[ビルド] - [ソリューションのビルド] を選択して、int_sin_icl プロジェクトをビルドします。

注：設定したすべてのコンパイル・オプションを確認する場合は、[プロパティ ページ] ダイアログから、[構成プロパティ] - [C/C++] - [コマンドライン] を選択してください。今回の例では、”/O3” および “/QxP” オプションが設定されていることが確認できます。なお、[追加のオプション] には手動でコンパイル・オプションを追加することができます。



３－６．実行/パフォーマンスの比較 ― その２

1. VS2005 のメニューから、[デバッグ] - [デバッグなしで開始] を選択してプログラムを実行します。



2. 最適化を行った場合の CPU 時間をメモして、結果を比較します。

> 注：正しく実行されない場合は、指定したベクトライズ・オプションに対し、システムの CPU が対応していない可能性があります。その場合はコンパイラー・ドキュメントを参考に、適切なコンパイル・オプションを適用してください。また、最適化オプション概要に関しては以下の『コンパイラー最適化ガイド』を参照してください。
> http://jp.xlsoft.com/documents/intel/compiler/qr_guide_jp.pdf

４．既存ソースのコンパイル

ここでは、既存のソースファイルをコンパイルする際のいくつかのポイントをご紹介します。

４－１．複数のソースコードをコンパイルする場合

コマンドラインから、複数のソースコードをコンパイルする場合は、以下の例のようにすべてのソースコードをコマンドラインに指定してコンパイルしてください。

```
prompt> icl    main.cpp sub1.cpp sub2.cpp
```

また、Visual Stuidio* IDE から、複数のソースコードをコンパイルする場合は、下図のようにすべてのソースコードを [ソリューション エクスプローラ] の “ソースファイル” に追加してください。

４－２．特定のヘッダー／ライブラリーファイルを使用する場合

特定のヘッダーおよびライブラリーファイルをコマンドラインから使用する場合は、一般的に環境変数を使用します。設定する環境変数は、以下の３種類です。

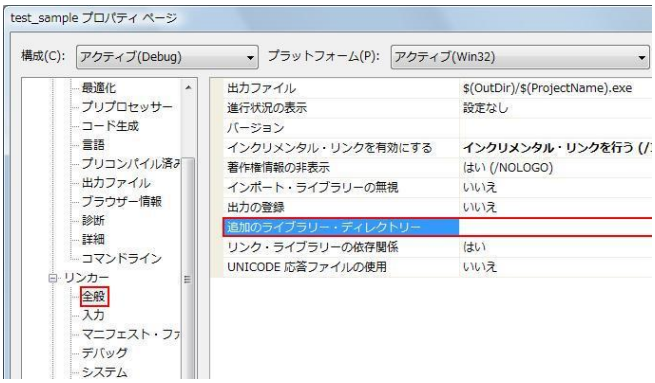
- INCLUDE：ヘッダーファイルの検索パス
- LIB：　　静的ライブラリーファイル（.lib）の検索パス
- Path：　　動的ライブラリーファイル（.dll）の検索パス

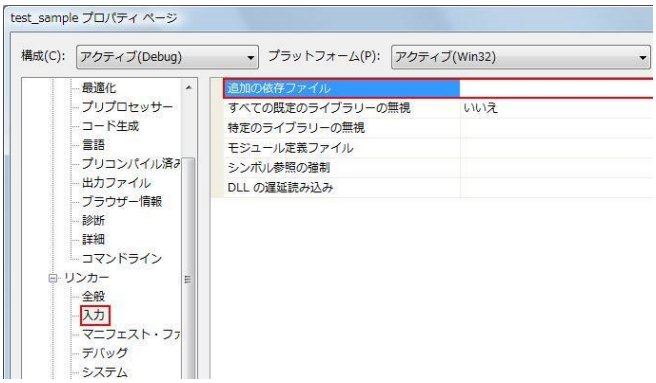
なお、これらの環境変数は、コマンドライン・ウィンドウ内で設定する環境変数、またはシステム環境変数が利用できます。環境変数の設定方法は別途、Microsoft* Windows のマニュアルを参照してください。

また、Visual Studio* IDE から使用する場合は、対象のファイルおよびパスの設定が必要となります。これらの設定は [プロパティ ページ] ダイアログで行います。

① ヘッダーファイル・パスの指定は、 [構成プロパティ] - [C/C++] - [全般] を選択して、[追加のインクルード・ディレクトリー] に設定してください。

② ライブラリーファイル・パスの指定は、 [構成プロパティ] - [リンカー] - [全般] を選択して、[追加のライブラリー・ディレクトリー] に設定してください。また、ライブラリーファイルの指定は、 [構成プロパティ] - [リンカー] - [全般] を選択して、[追加の依存ファイル] に設定してください。



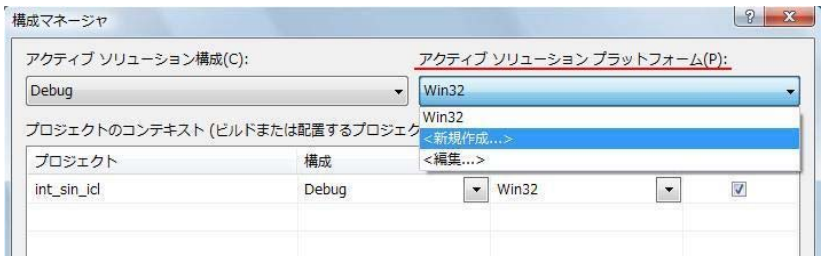


4－3．64 ビットアプリケーションの作成

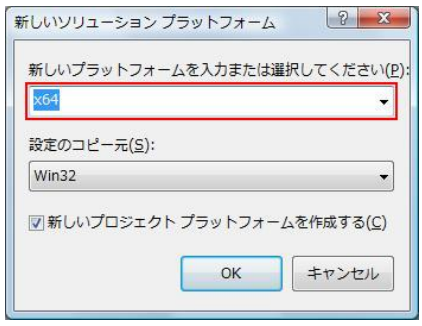
Visual Stuidio* IDEからインテル® C++ コンパイラー を使用して 64 ビット対応アプリケーションを作成することができます。ここでは、インテル® 64 アプリケーションの作成方法を説明します。

まず、[構成マネージャ] で、[アクティブ ソリューションプラットフォーム] から “<新規作成...>” を選択して [新しいソリューション プラットフォーム] 画面を開きます。同画面で、下図のように新しいプラットフォームを “Win32” から “x64” へ変更します。この操作で、ビルド環境が 64 ビット（インテル® 64）に変更されます。VS2005 のメイン画面に戻りビルドを実行します。

図：構成マネージャ



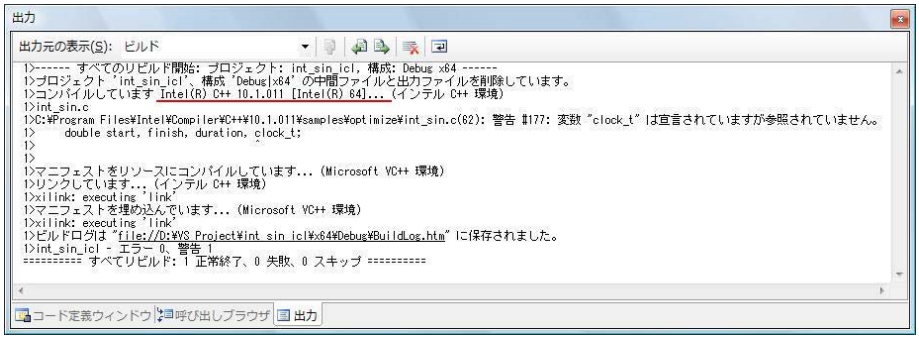
図：新しいソリューション プラットフォーム



ビルドの結果は、[出力] 画面にて確認することができます。

インテル® 64 用インテル® C++ コンパイラー が使用されていることを確認してください。

図：出力



注：64 ビットアプリケーションを作成する場合は、インテル® 64 用 ／ IA-64 用インテル® C++ コンパイラーのインストールはもちろん、リンク環境である Microsoft* Visual Studio* IDE 側でもそれぞれのアーキテクチャーに対応したライブラリー、その他必要なツールがインストールされている必要があります。たとえば、Visual Studio 2005 Professional Edition をインストールする際にカスタムインストールを選択し（またはアップデートで “機能の追加と削除” を選択し）以下の [セットアップ オプションページ] 画面にて、 “X64 コンパイラおよびツール” をチェックしてインストールする必要があります。

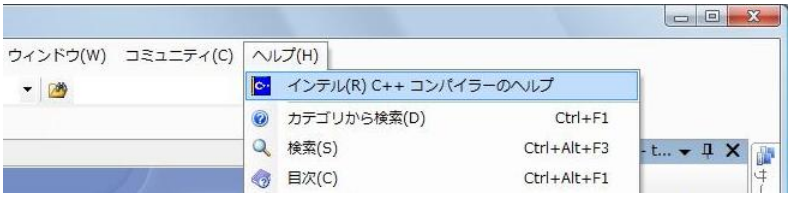
5． 追加情報

追加情報として、インテル® C++ コンパイラー関連事項をいくつかご説明します。

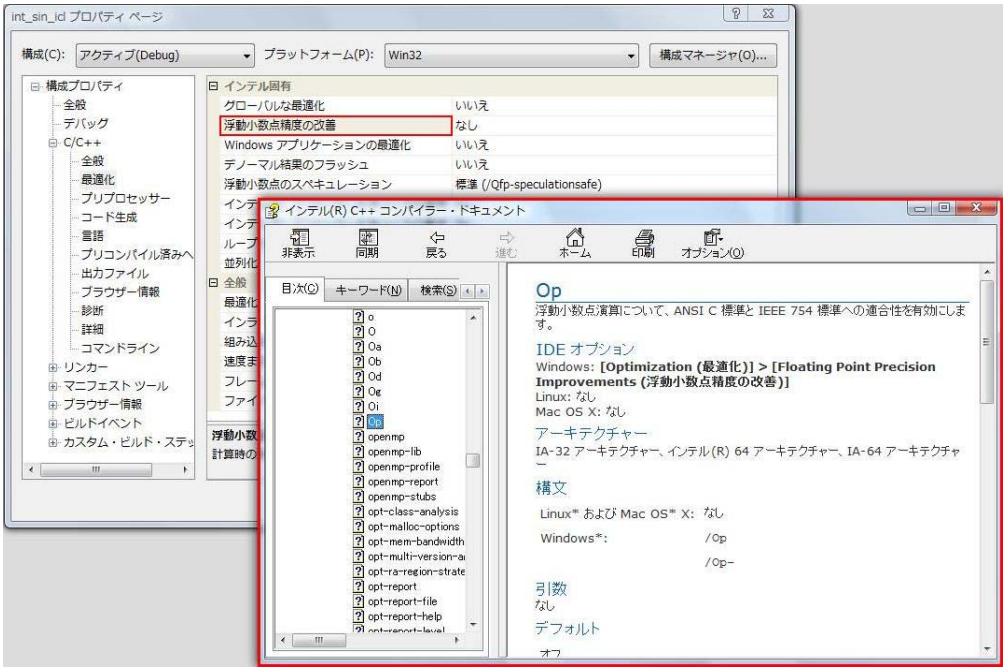
5－1． ドキュメントの参照方法

インストールされるすべてのドキュメントを参照するには、以下のように Windows スタートメニューから “ドキュメント・インデックス” を選択してください。このインデックスには、本製品に含まれているすべてのドキュメントへのリンクが含まれています。

Visual Studio* IDE からは、『インテル® C++ コンパイラー・ドキュメント』へのアクセスが可能です。



また、Visual Studio* IDE から、”F1” キーによる状況依存ヘルプ機能が使用できます。これは、例えば以下のようにプロジェクトの [プロパティ ページ] で [浮動小数点精度の改善] をクリックしてフォーカスをおいた状態で キーボードの “F1” キーを押下することにより、『インテル® C++ コンパイラー・ドキュメント』がポップアップされ浮動小数点演算に関するオプションの内容が表示されます。



５－２．サンプルコード

本ドキュメントでは、サンプルコード（int_sin.c）を使用しましたが、インテル® C++ コンパイラーには、この他にもたくさんのサンプルが用意されています。以下にその概要をご紹介します。

- PGO オプション用サンプル
- IPO オプション用サンプル
- OpenMP を使用したサンプル
- 組み込み関数を使用したサンプル
- ベクトライズ・オプション用サンプル

サンプルコードに関する詳細は、以下の “samples.htm” を参照してください。

<Install-Dir>¥Compiler¥C++¥10.1.xxx¥samples¥samples.htm

６． 最後に

インテル® C++ コンパイラーの動作検証が完了したら、次はさまざまな最適化オプションを試してアプリケーションのパフォーマンス向上を図りましょう。インテル® C++ コンパイラーには、本ドキュメントで使用した最適化オプションの他にも、たくさんのオプションが用意されています。その中でも、今後のマルチコアー・プロセッサーに対応する自動並列化オプション（/Qparallel）は是非試したいオプションです。最適化オプションの詳細はコンパイラー・ドキュメント、または以下の『コンパイラー最適化ガイド』を参照してください。
http://jp.xlsoft.com/documents/intel/compiler/qr_guide_jp.pdf

その他ご不明な点がありましたら、下記お問い合わせ窓口より弊社サポートまで御連絡ください。
https://www.xlsoft.com/jp/services/xlsoft_form.html

2008 年 4 月 1 日