VisucalStudio C++.net 2003 の使い方

ー Win32 コンソールアプリケーションの作成とデバッグ方法 ー

改訂履歴
平成17年7月1日 初版

作成
遠藤修平,川本公章,木下雅也,橋本和紀,福田拓也
監修
藤尾三紀夫

ー CONTENTS ー

VisualStudio のビルドとデバッグ方法

ビルドの方法

（1）ビルドの実行
（2）コンパイル時のエラー対応

デバッグの方法

（1）ブレークポイントの設定
（2）デバッグの実行
（3）デバッグの停止
（4）デバッグのリスタート
（5）デバッグのステップ実行
（6）デバッグ位置へのジャンプ
（7）指定行まで実行
（8）条件付きブレークポイント
（9）ウォッチ
（10）変数のデータ変更

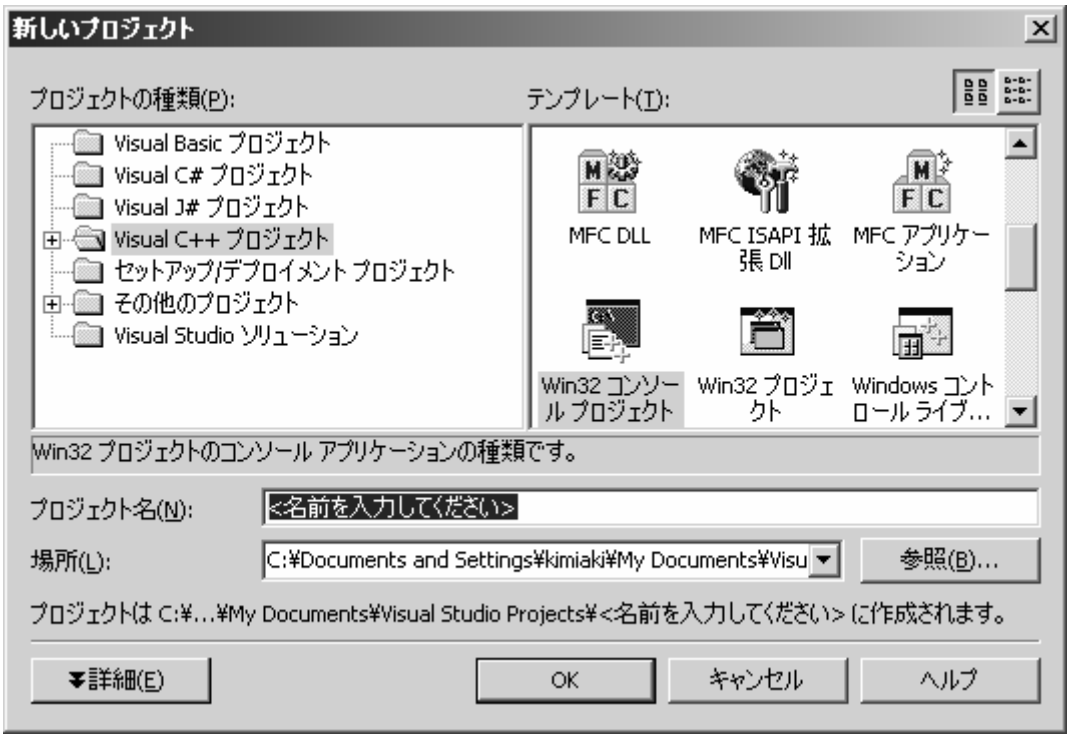
Win32 コンソールアプリケーションプロジェクトの作成

（1）Win32 コンソールアプリケーションプロジェクトの作成

[ファイル]-[新規作成]-[プロジェクト]または、ツールバーの

を選ぶ。

次にプロジェクトの種類から「Visual C++ プロジェクト」を選び、テンプレートから「Win32 コンソールプロジェクト」を選択し、プロジェクト名を入力する。



次に出てくる「Win32 アプリケーションウィザード」では、完了ボタンを押す。

（2）コンソールアプリケーションのためのコードの追加

次のコードを追加する。

```
#include <conio.h>
```

（kbhit 関数を使うため）

```
while(!kbhit());
```

（キーが押されるまでループする。）

これはコンソールアプリケーションでは終了時にコンソールウインドウが閉じてしまうため、キーを押すまで終了しないようにするための処置です。

```
// project.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
//

#include "stdafx.h"
#include <conio.h>

int _tmain(int argc, _TCHAR* argv[])
{

    while(!kbhit());

    return 0;
}
```

（3）1 から n まで足し合わせるプログラムのコーディング

次のようなコードを作成する。

```
// project.cpp : コンソール アプリケーションのエントリ ポイントを定義します。
//

#include "stdafx.h"
#include <conio.h>

int _tmain(int argc, _TCHAR* argv[])
{
    int i, a, n;

    printf("整数を入力してください：");
    scanf("%d", &n);
    a = 0;
    for (i=1; i<=n; i++) {
        a += i;
    }
    printf("１から%dまでの和は%dです¥n", n, a);

    while(!kbhit());

    return 0;
}
```

プログラムを実行する方法は、「F5」を押すまたは[デバッグ]-[開始]またはツールバーの

を選ぶ。

エディタの使い方

（1）行の選択

・左端をクリックすると行全体が選択される
・複数行はドラッグ又は、開始行の選択後、shift を押して最終行を選択でも可

（2）語句の選択

・カーソルによるドラッグでも選択できるが、単語の区切りであればダブルクリックで選択できる
・最初の位置にカーソルを移動後、shift を押して最終位置にカーソルを合わせることで選択できる

（3）コピー

・コピーしたい範囲を選択した後、Ctrl+C
・コピーしたい範囲を選択した後、Ctrl キーを押しながらコピー先にドラッグ

（4）ペースト

・ペースとしたい位置にカーソルを置いた後、Ctrl+V

（5）移動

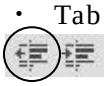
・移動したい範囲を選択した後、Ctrl+X

・移動したい範囲を選択した後、移動先にドラッグ

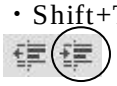
（6）複数行のインデント挿入と削除

複数(1 行でもよい)を選択し、

・ Tab キーで右シフト（[編集]-[詳細]-[行インデント]またはツールバー）



・Shift+Tab キーで左シフト（[編集]-[詳細]-[行インデント解除]またはツールバー）



（7）複数行のコメントと解除

複数行(1 行でもよい)を選択後

・コメントを付ける

[編集]-[詳細]-[選択範囲のコメント]またはツールバー

・コメントを解除

[編集]-[詳細]-[選択範囲のコメント解除] またはツールバー

（8）ブックマーク

・ブックマークの追加または削除

印を付けたい行を選択した後、[編集]-[ブックマーク]-[ブックマーク設定/解除]またはツールバー

・ブックマークへのジャンプ

前へ移動は[編集]-[ブックマーク]-[次のブックマーク]またはツールバー

後へ移動は[編集]-[ブックマーク]-[次のブックマーク]またはツールバー

・全てのブックマークの解除

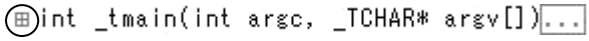
[編集]-[ブックマーク]-[ブックマークをクリア]またはツールバー

（9）アウトライン(複数行をまとめて表示/非表示する)

非表示にする場合は－を押す

```
int _tmain(int argc, _TCHAR* argv[])
{
    int i, a, n;
```

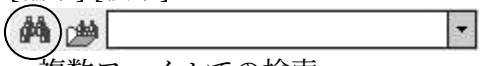
表示したい場合は＋を押す



（10）検索

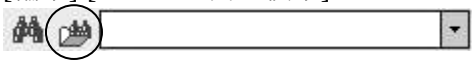
・単一ファイル(現在アクティブなファイル)での検索

[編集]-[検索]またはツールバー



・複数ファイルでの検索

[編集]-[ファイル内の検索]またはツールバー



（11）置換

・単一ファイルでの検索

[編集]-[置換]またはツールバー

・複数ファイルでの検索

[編集]-[ファイル内の置換]またはツールバー

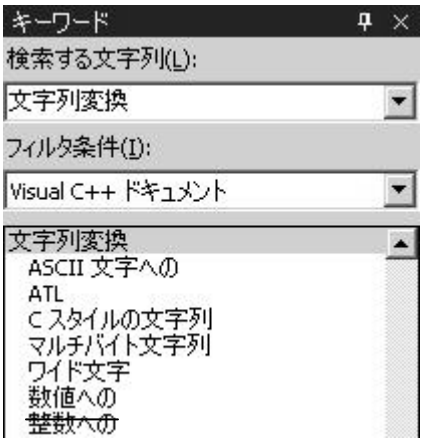
（12）関数のヘルプ参照

printf など関数の説明を知りたい場合は printf を選択(ダブルクリックでも選択できる)後 F1 キーを押すと説明画面が出てくる

（13）関数の探し方

ASCII 数字から int に変換する関数がわからない場合

[ヘルプ]-[インデックス]で ”文字列変換” を入力後　フィルタを C++ に設定。検索結果から “整数への” を選ぶと atoi が出てくる。



（14）関数への参照

・関数が宣言されている位置(.h)にジャンプする

　関数を選択後、[マウスの右クリック]-[宣言への移動]

・関数が定義されている位置(.cpp)にジャンプする

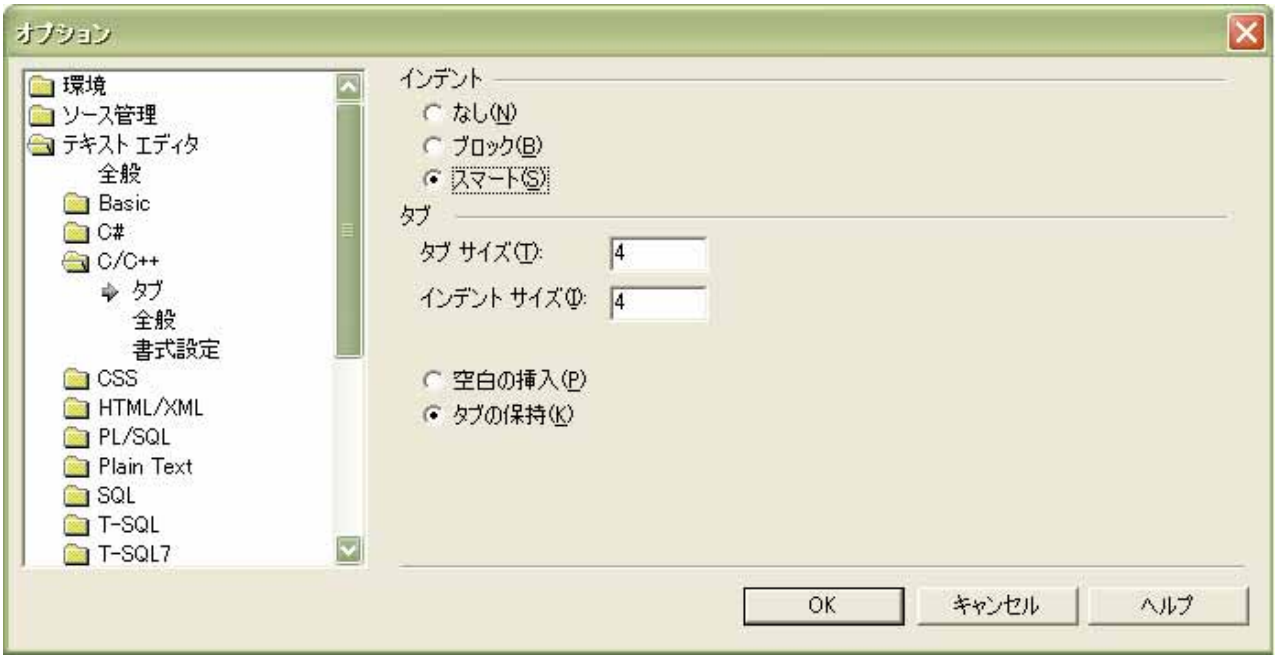
　関数を選択後、[マウスの右クリック]-[定義への移動]

・関数が使われている位置にジャンプする

　関数を選択後、[マウスの右クリック]-[参照へジャンプ]

（15）エディタのカスタマイズ

[ツール]-[オプション] を選ぶ。



出てきた画面の[テキストエディタ]-[C/C++]-[タブ]でインデント、タブの設定ができる。

（16）括弧の対応表示

'}' や ')' がどのカッコと対応しているかを調べるには、調べたい括弧を選択した状態で Ctrl+] または Ctrl+Shift+]を入力。

（17）コメント付きブックマーク（コメントタスク）

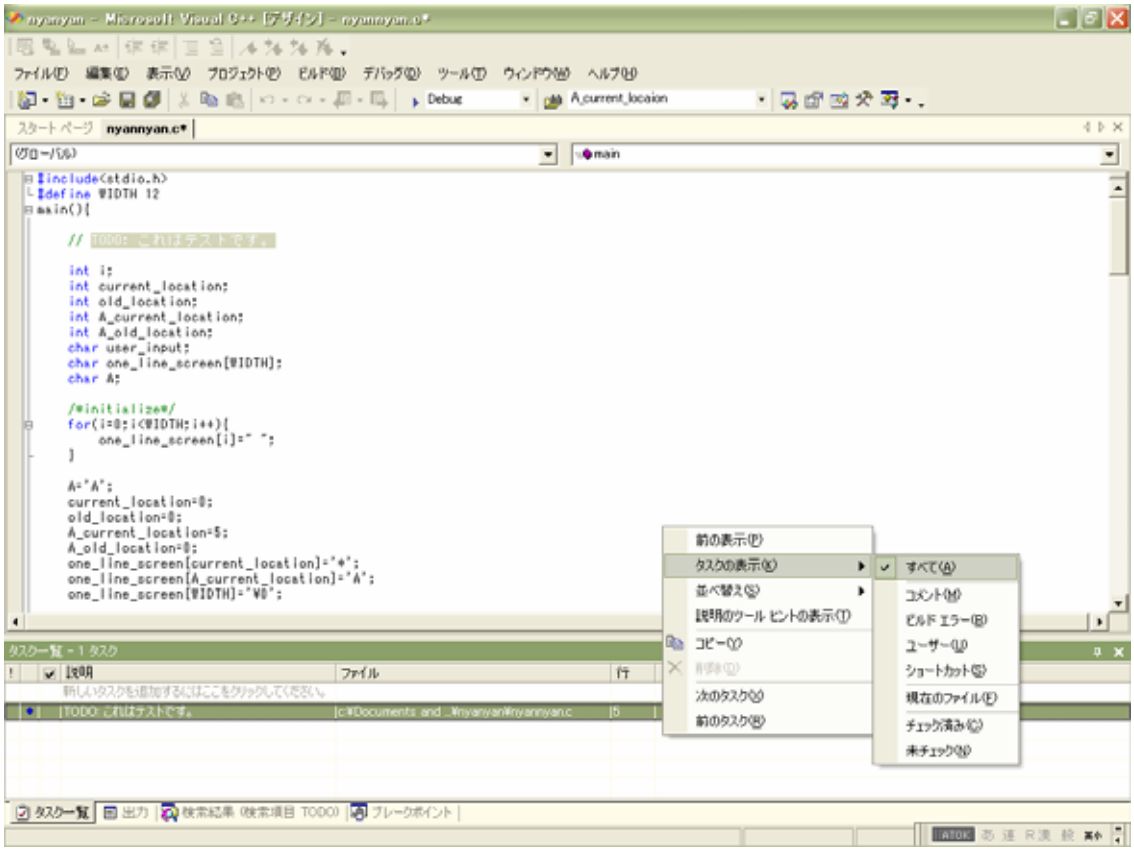
//の後に “TODO: コメント文” を書くと、タスクとして登録される

```
// TODO: これはテストです。
```

登録されている物を見るには [表示]-[その他のウインドウ]- [タスク一覧]を見る。

ここで、[タスク一覧]で右クリックし、[タスクの表示]-[すべて]を選択する。

表示されたコメントをダブルクリックするとその行にジャンプする。



カスタムトークンとして TODO, HACK, UNDONE はデフォルトで登録されている。
新規にカスタムコメントトークンを登録するには　[ツール]-[オプション]-[環境]-[タスク一覧]-[コメントトークン]領域で[名前ボックス]に書込登録する。あわせて優先順位も設定。

その後、[追加]をクリック。

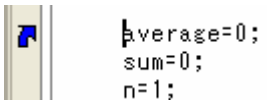
これで追加された。[OK]をクリックして完了。

（18）コードの行へのショートカット(コメント)の追加と削除

通常のブックマークは Vstudio を終了すると消えてしまう。終了しても消えないようにするにはショートカットを利用する。

・設定

カーソルをコード行においた後､[編集]-[ブックマーク]-[タスク一覧へのショートカットの追加]をクリック



すると、設定したコード行の左端に図のようなマークが現れる。

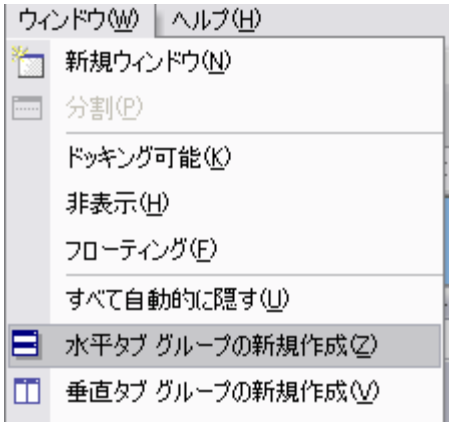
・　削除

カーソルをコード行においた後、 [編集]-[ブックマーク]-[タスク一覧へのショートカットの削除]をクリック。これで削除される

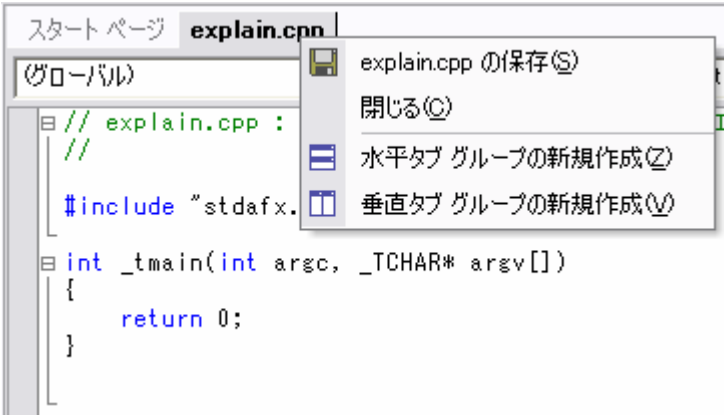
画面の表示設定

（1）複数ファイルの表示

[ウインドウ]-[水平(垂直)タブ グループの新規作成]を選ぶ。



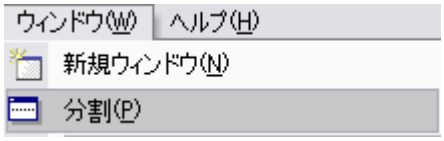
又は、タブを右クリックして[水平(垂直)タブグループの新規作成]を選ぶ。



また、クリックしたままドラッグして移動させたい位置にタブを持っていくことでも出来る。

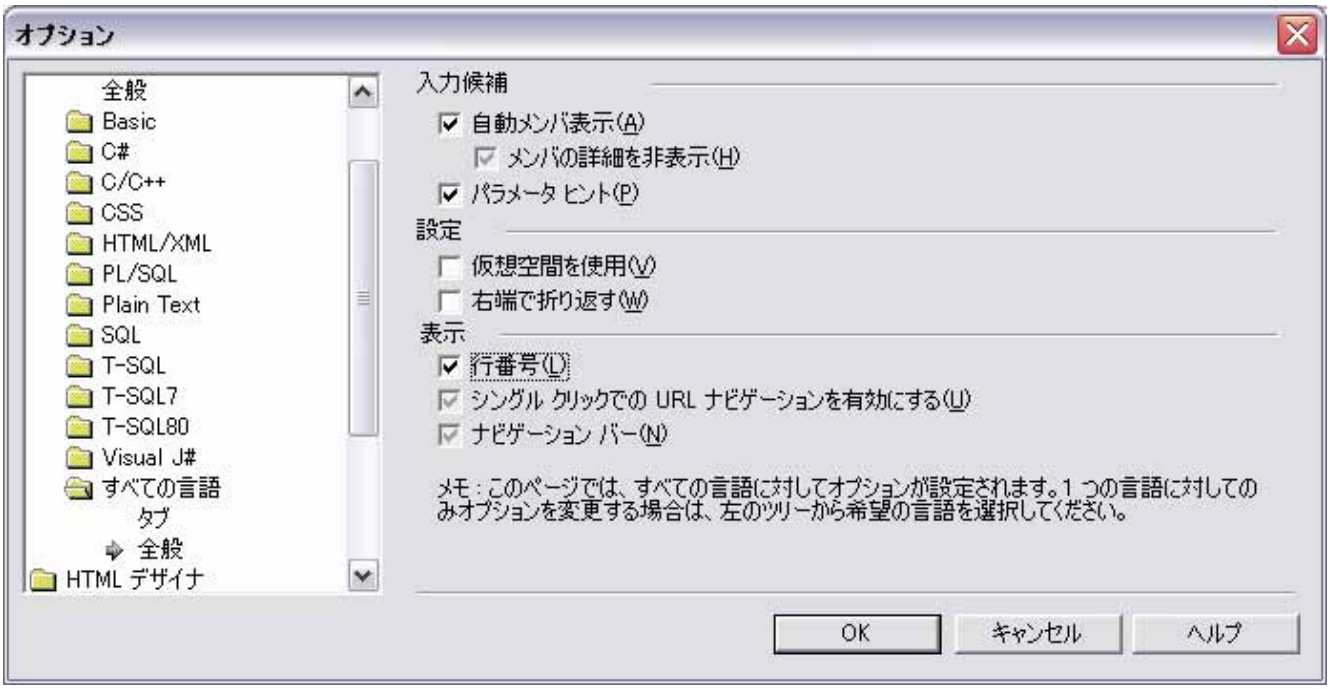
（2）大きなファイルの分割表示

[ウインドウ]-[分割]を選ぶと今選んでいるウインドウが分割表示される。



（3）行番号の表示と行へのジャンプ

[ツール]-[オプション]を押した後に「テキストエディタ」-「すべての言語」-[全般]を選び、「行番号」のチェックボックスを ON にする。

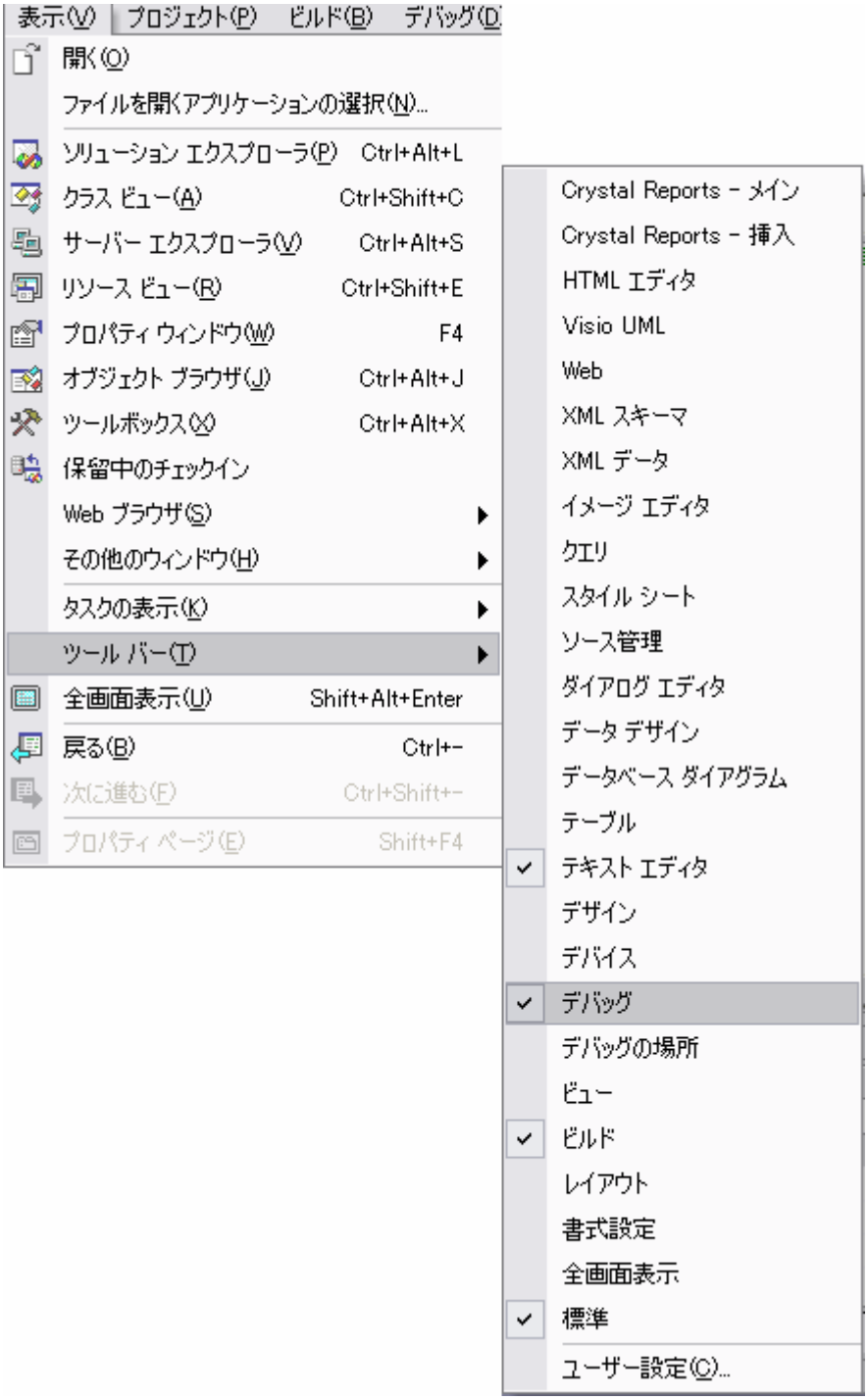


すると、特定の行へ行きたいときは[編集]-[ジャンプ]を選び行きたい行の行番号を指定する。また、[編集]-[ジャンプ]の変わりに Ctrl+G を押しても出来る。

（4）ツールバーの表示

必要なツールバーを画面に表示させる

[表示]-[ツールバー]から画面上に置いておきたいものを選ぶ。

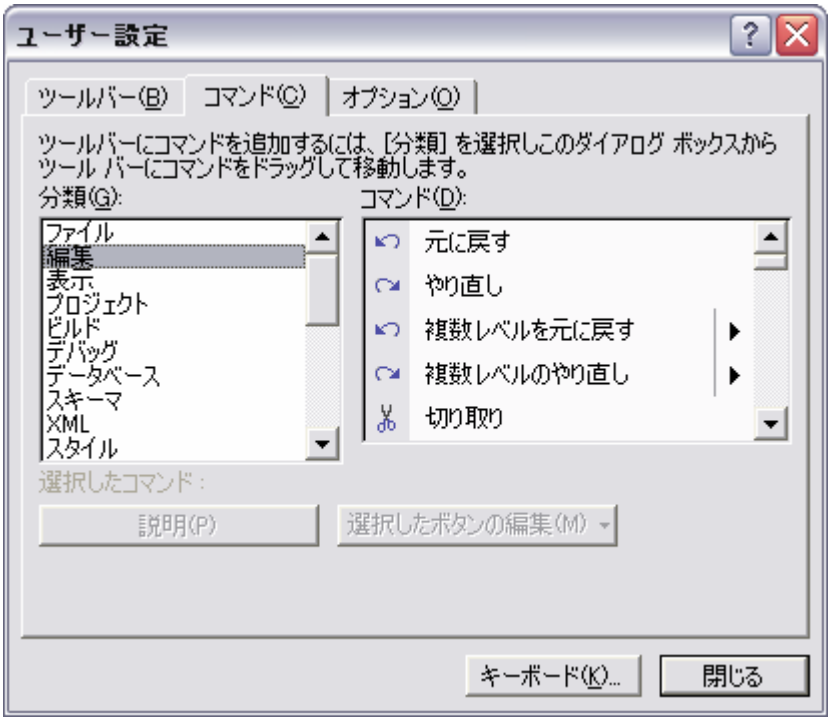


通常は「テキストエディタ」、「デバッグ」、「ビルド」、「標準」を有効にしておく

（5）ツールバーのカスタマイズ

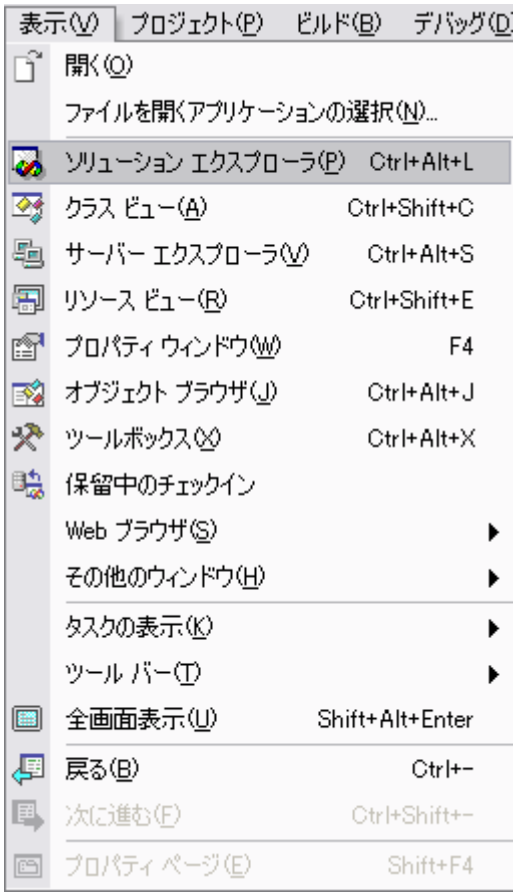
よく使うコマンドはメニューからではなくツールバーに登録しておく方が効率がよい

[ツール]-[カスタマイズ]-「コマンド」でコマンド内から必要なツールバーのボタンを選択し、画面上のツールバーにドラッグする。



（6）便利なウインドウ

[表示]から表示したいウインドウを選択する。

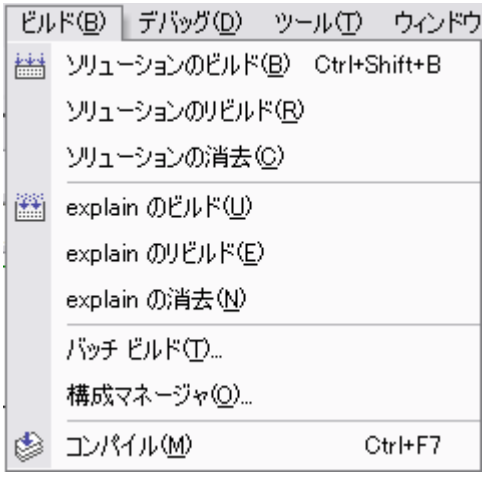


・ソリューションエクスプローラ(*.cpp, *.h ファイルの一覧表示)
・クラスビュー(クラスの一覧表示,メンバ関数,メンバ変数も表示,直接ジャンプ(指定した場所へ移動すること)することができる)
・オブジェクトブラウザ(グローバル変数や定数の表示)
・リソースビュー(メニューやダイアログ、アイコンなどの一覧)
・プロパティウインドウ(ダイアログ等のプロパティ表示)

ビルドの方法

（1）ビルドの実行

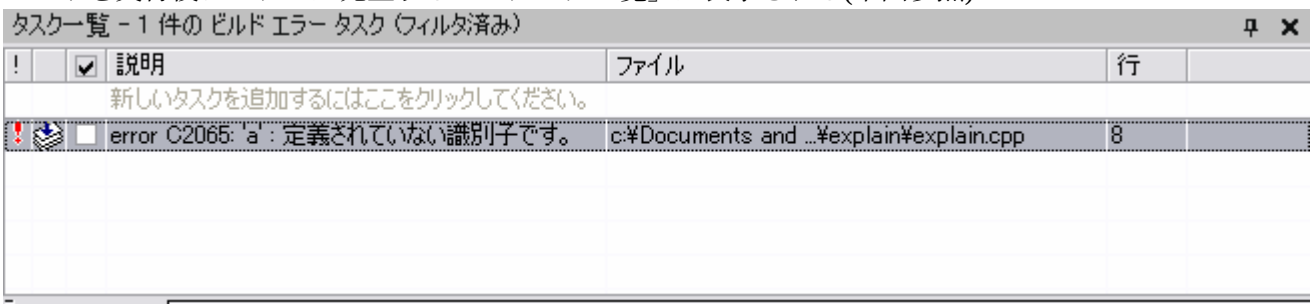
[ビルド]-[(プロジェクト名)のビルド]を押す。下図の場合は「explain のビルド」を押す。



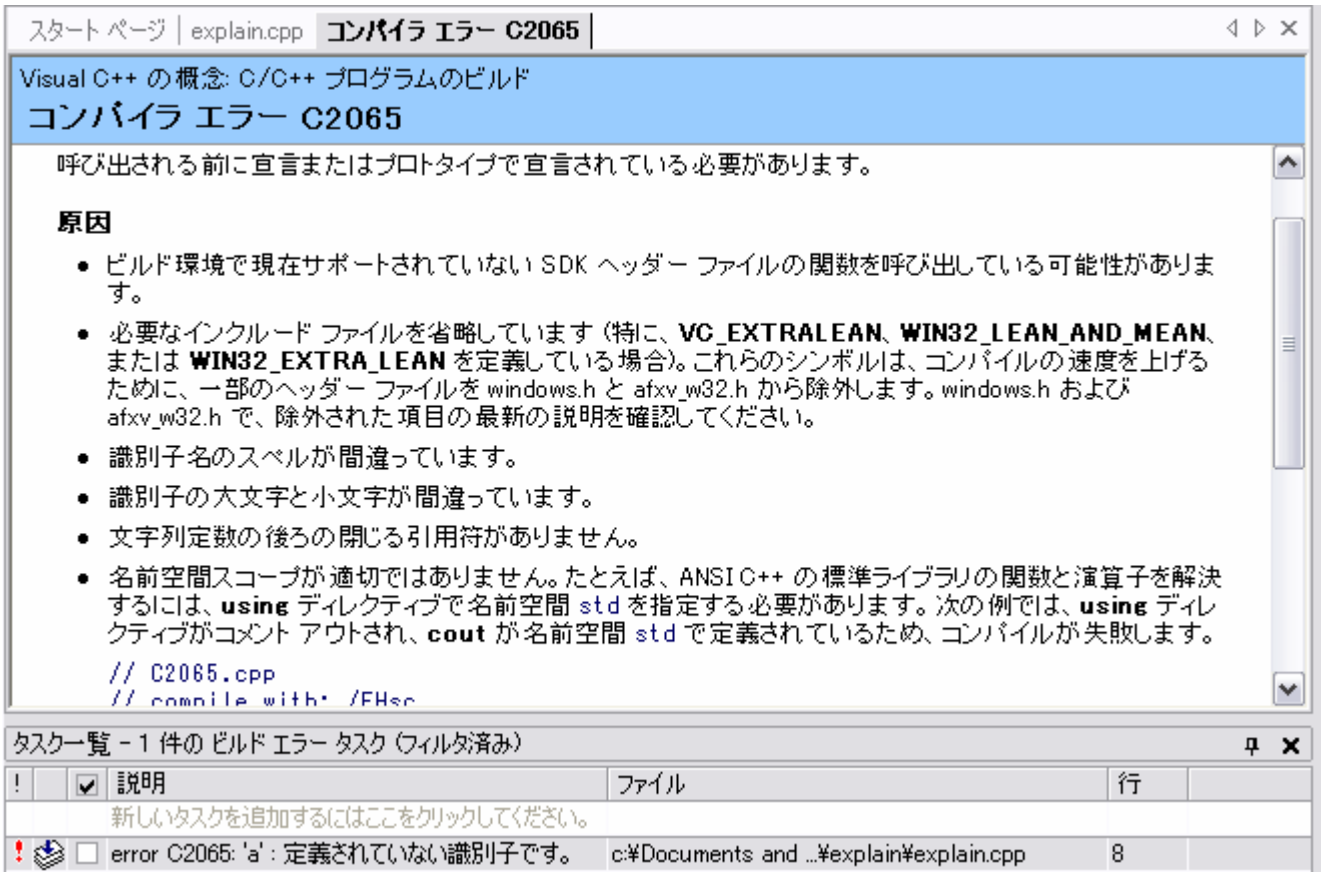
またはツールバーのビルドを押す。

（2）コンパイル時のエラー対応

ビルドを実行後にエラーが発生すると「タスク一覧」が表示される(下図参照)



- エラー行(上図で選択されている部分)をダブルクリックするとエラー行へジャンプ
- エラー行のエラー番号を選択して F1 でエラーの詳細説明ウインドウが現れる



・どのファイルのコンパイル時にエラーが発生したかなどの詳細は「出力」画面を見るとわかる

デバッグの方法

（1）ブレークポイントの設定

・設定（丸い印●が付く）。あるいはツールバー
停めたい行の左端をクリックまたは F9、ツールバーおよびメニューからも設定可能

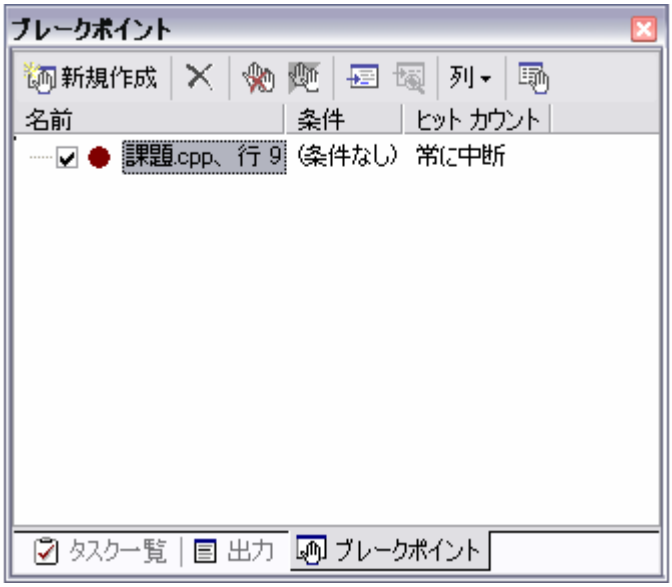
・解除（丸い印●が消える）。あるいはツールバー
解除したい行の左端をクリックまたは F9、ツールバーおよびメニューからも設定可能

・一時解除（丸い印○になる）
削除ではなく一時的な解除を実行するには、一時解除したい行でマウスの右クリック[ブレークポイントの無効化]をクリック。あるいはツールバー

・全部解除
全てのブレークポイントを解除するには[デバッグ]-[すべてのブレークポイントを解除]

・全部一時解除
全てのブレークポイントを一時解除するには[デバッグ]-[すべてのブレークポイントを無効にする]

・これらはブレークポイントウインドウでも設定可→[デバッグ]-[ウィンドウ]-[ブレークポイント]



（2）デバッグの実行

・デバッグモードでの実行
[デバッグ]-[開始]または F5
・通常モードでの実行
[デバッグ]-[デバッグ無しで実行]または Ctrl+F5

（3）デバッグの停止

デバッグを終了するには [デバッグ]-[デバッグの停止]またはツールバー

（4）デバッグのリスタート

デバッグを最初からやり直すには[デバッグ]-[再起動]またはツールバー

（5）デバッグのステップ実行

・ステップイン(F11)
・ステップアウト(Shift+F11)
・ステップオーバ(F10)

（6）デバッグ位置へのジャンプ

デバッグ中にどこで停まっているかわからなくなった場合、どこでもいいのでマウス右クリックして[次のステートメントの表示]をクリック、またはツールバー

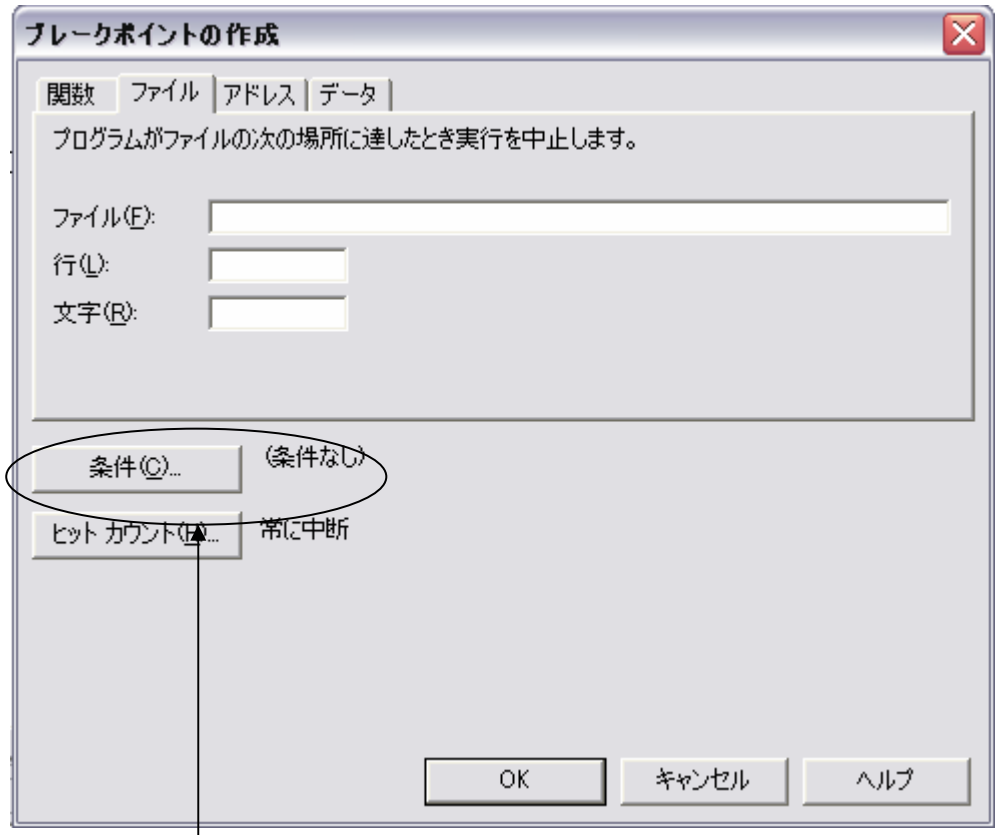
（7）指定行まで実行

デバッグで停めたい位置にカーソルを置いて、マウスを右クリック[カーソル行まで実行]を選択
このとき、ブレークポイントを設定する必要はない

（8）条件付きブレークポイント

・指定行で指定条件によって停める

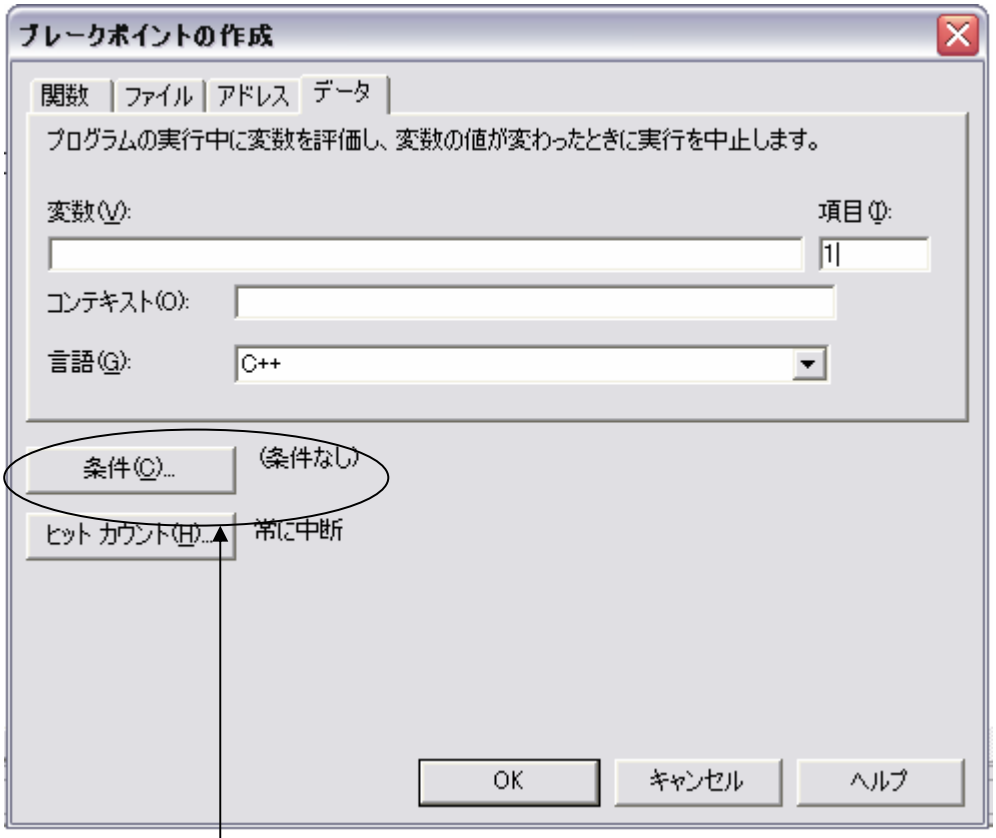
[ブレークポイント]―[ブレークポイントの作成]―ファイルタブ



ここでは「条件なし」となっているが、条件スイッチをクリックし自分で条件を定めることができる。

・変数の値の指定条件によって停める

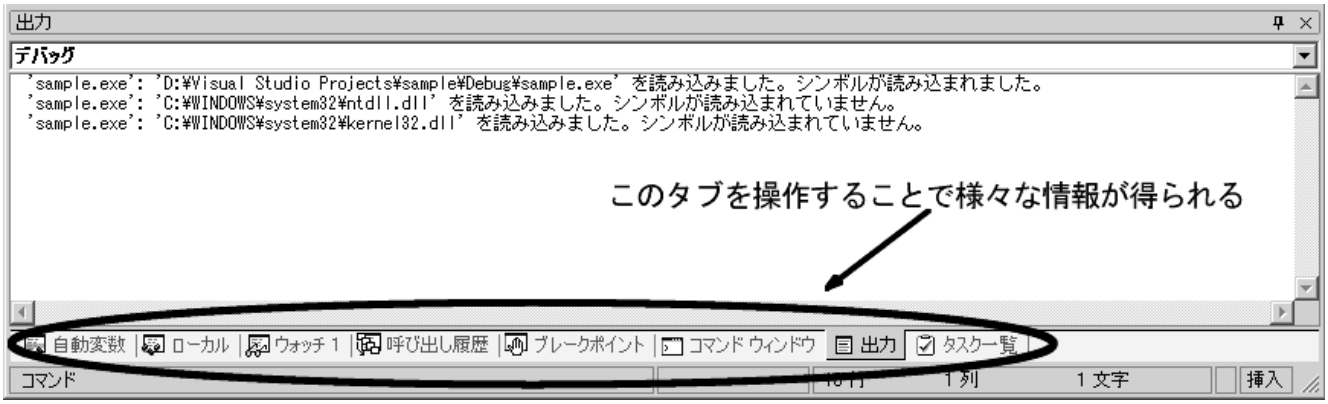
[ブレークポイント]―[ブレークポイントの作成]―データタブ



ここでは「条件なし」となっているが、条件スイッチをクリックし条件を自由に決めることができる。

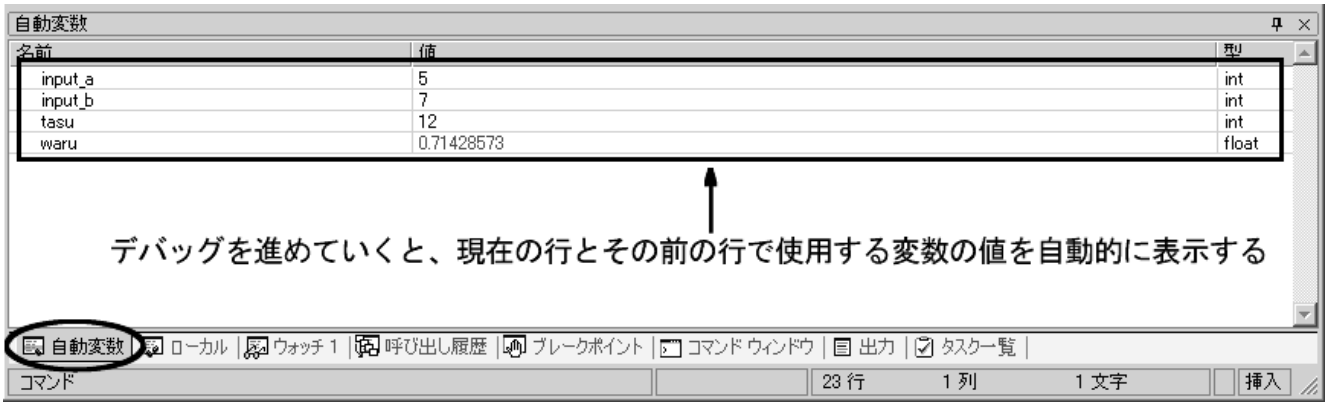
（9）ウォッチ

デバッグ中にウインドウの下半分に表示される画面を使って実行途中の変数の値等を見ることができる。なお、それぞれの値はプログラムが進行するのに応じて動的に変化していく。



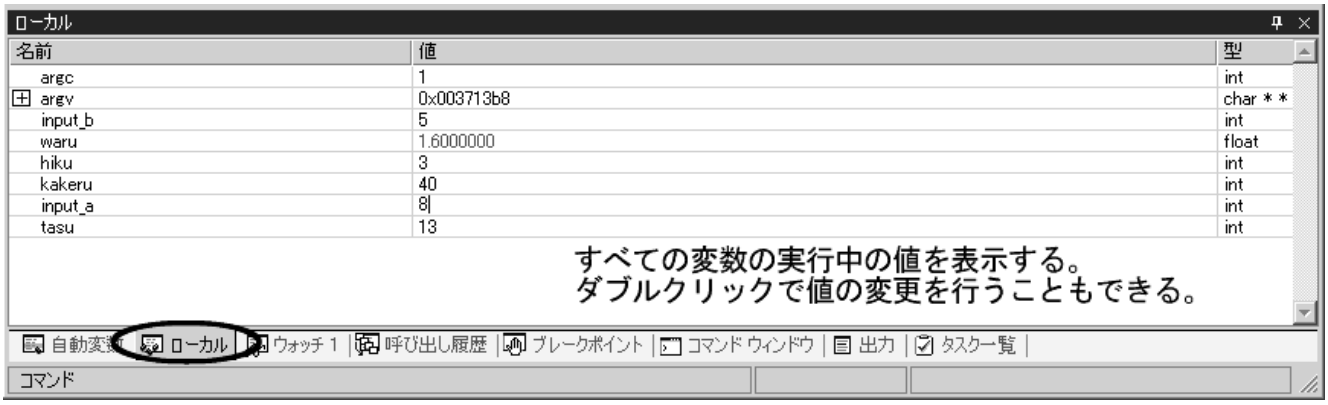
※上の画面を使わないでカーソルを用いた方法もある。ソース上の値を見たい変数の上にカーソルを持って行くと右図のように実行途中の変数の値を表示できる。

・自動変数



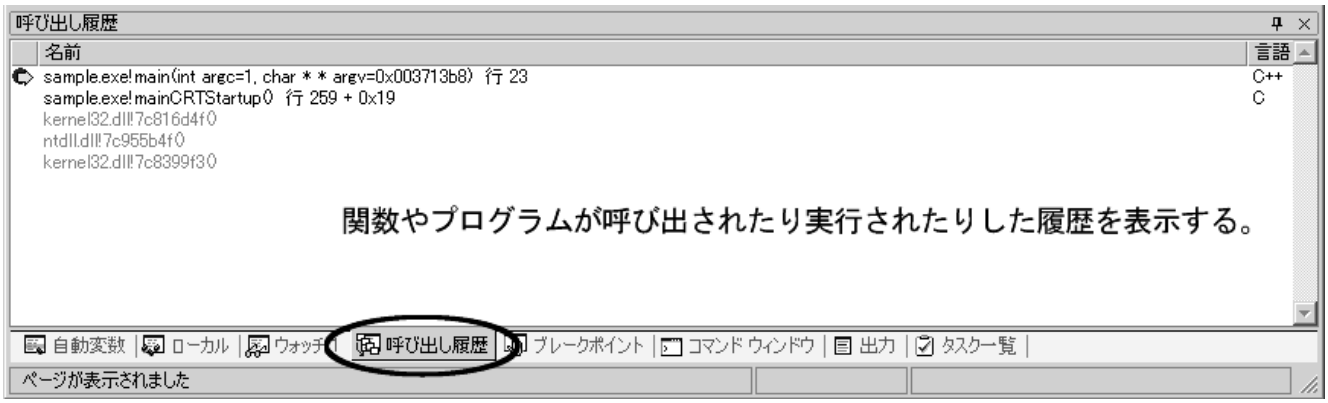
アクティブな行とその前の行の情報を表示する。
表示されていない場合はメニューの[デバッグ]-[ウインドウ]-[自動変数]を選択する。

・ローカル変数

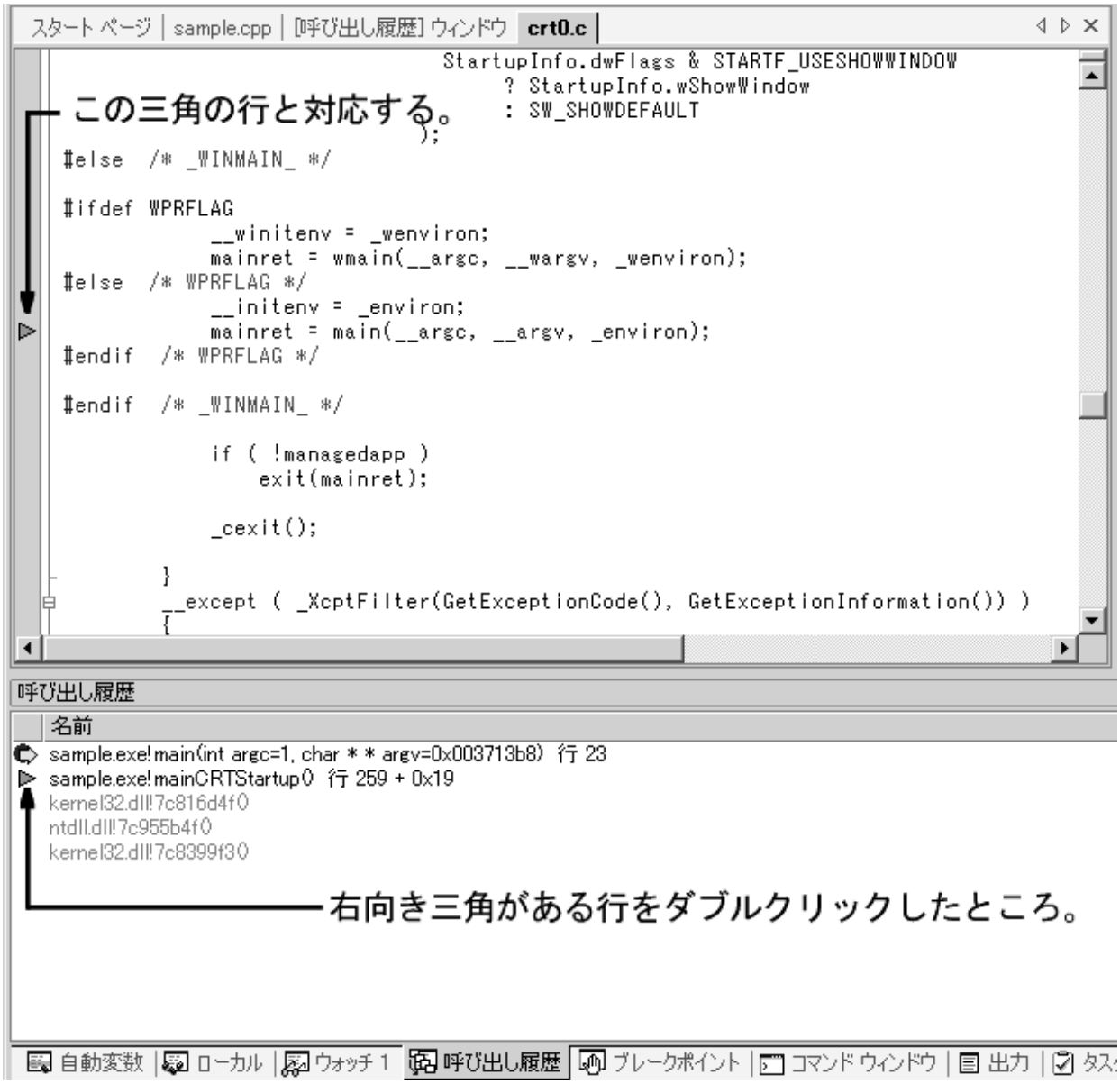


ブレークポイントで中断しているときにすべてのローカル変数の値や型を表示する。[値]をダブルクリックすることで、値の変更をすることもできる。
表示されていない場合は[デバッグ]-[ウインドウ]-[ローカル]を選択する。

・呼び出し履歴



関数やプログラムがどのような順序で呼ばれているかを表示する。
ダブルクリックによりその関数に移動できる。（図参照）

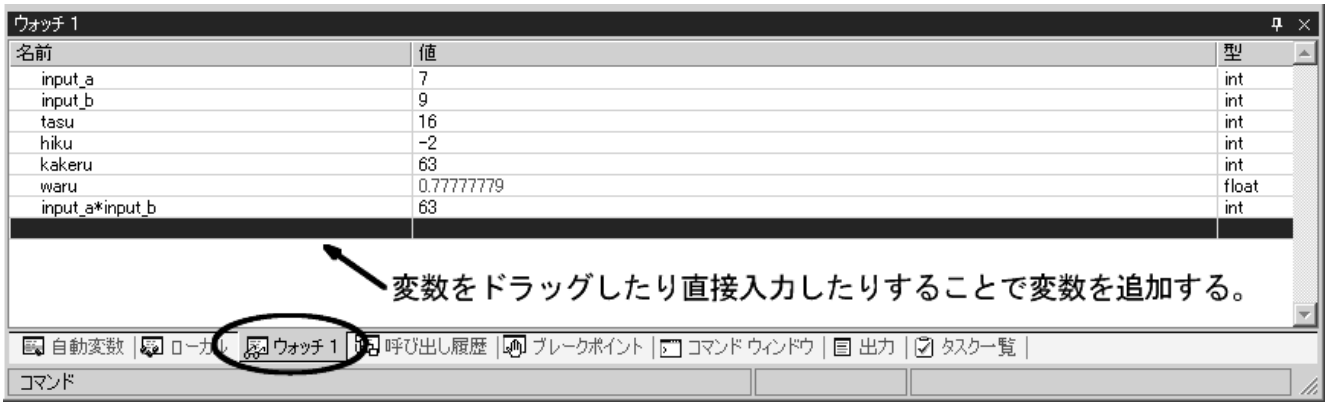


表示されていない場合は[デバッグ]-[ウインドウ]-[呼び出し履歴]を選択する。

・ウォッチ（全部で 4 画面表示することが可能）

ユーザが見たい変数を設定しその値を見ることができる。

ウォッチ画面の表示・追加は [デバッグ]-[ウインドウ]-[ウォッチ]-[ウォッチ 1]



ウォッチ画面へのデータ表示には変数名を選択してウォッチウインドウまでドラッグする。もしくは、マウスの右クリックで[ウォッチ式の追加]でも可能。なお、数式を入力してもよい。一時的ならクイックウォッチでもよい。なお、ウォッチ画面からの削除は変数等を選択した後[Del]を押す。

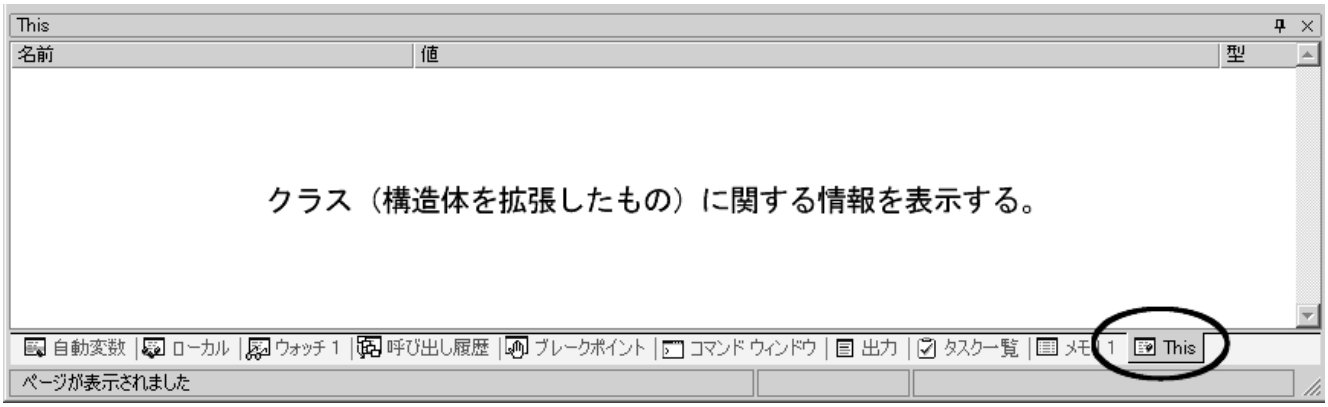
・メモリ

実際のコンピュータ内のメモリを表示する。配列の中身などの確認できる。

アドレスや表示方法等を変更できる。最大で４個まで表示ｓを作ることができる

メモリ画面の表示・追加は[デバッグ]-[ウインドウ]-[メモリ]を選択する。

・実行中のクラス情報



表示されていない場合は[デバッグ]-[ウインドウ]-[This]を選択する。

（10）変数のデータ変更

デバッグ途中で変数の値を変更したい場合は、ウォッチ画面で値をダブルクリックしたりすることで直接その変数のデータを書き換えることもできる。