以下は、wxWidgets と Microsoft Visual C++ を使用してソフトウェアを開発する際に手助けとなる、幾つかの Tips と技巧です。

プリコンパイル済みヘッダファイルの使い方

wxWidgets ドキュメントは、wx/wxprec.h をプリコンパイル済みヘッダファイルとして使用することを提案する。ここでは、以下の利点を示す。

- プリコンパイル済みヘッダファイルとして、wx/wxprec.h 以上のものを使用することができる。
- " デバッグメモリアロケーション強化（以下参照）" のために必要な行を追加する場所が得られる。

プリコンパイル済みヘッダを使用するために、以下の設定を行う。：

1. ヘッダファイル stdwx.h を作成する。

2. このヘッダファイルに、プリコンパイルしたい全てのヘッダファイルを含める。stdwx.h は、例えば以下のようになる。：

```
#include "wx/wxprec.h"                  // wxWidgets のプリコンパイル済み／標準 ヘッダ
#include "wx/notebook.h"                // 必要な他のヘッダ
#include "wx/statline.h"
#include "wx/tglbtn.h"

// デバッグメモリアロケーションの強化
#ifdef _DEBUG
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK ,__FILE__, __LINE__)
#else
#define DEBUG_NEW new
#endif

#pragma warning (disable:4786)
#include <map>
#include <vector>

using namespace std;
```

3. C++ ファイル stdwx.cpp を作成し、以下の行を記述する。

```
#include “ stdwx.h ”
```

4. stdwx.h と stdwx.h をプロジェクトに追加する。

5. プロジェクトの設定を開く。

6. ' 全ての構成 ' を選択。

7. [C/C++]:[プリコンパイル済みヘッダー] を選択。

8. [プリコンパイル済みヘッダーファイル (.pch) を使用] を選択し、[このヘッダーまで] に stdws.h を入力。

9. プロジェクトファイルの一覧から、stdwx.cpp を選択。

10. [プリコンパイル済みヘッダーファイル (.pch) の作成] を選択肢、[このヘッダーまで] に stdwx.h を入力。

デバッグメモリアロケーション（メモリリークダンプ）にファイル名と行数を含める方法

wxWidgets は、Microsoft Visual C++ を使用しているときにデフォルトで有効になるでデバッグメモリアロケーションを持っている。もしアプリケーションがメモリリークを起こすと、プログラムを閉じたときに全てのリークされたメモリブロックがダンプされる。

```
Dumping objects ->
{5641} normal block at 0x00BB30A0, 100 bytes long.
 Data: <              > CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
```

これはすばらしいが、リークが発生した正確な位置を調べなければならない。あるいは、_CrtSetBreakAlloc(5641) を呼ぶ必要がある。ここで、5641 は、アロケーションダンプの括弧に記されたアロケーション番号である。

しかし、もっと簡単な方法がある。

1. プリコンパイルされるヘッダファイルに以下の行を追加する。：

```
#ifdef _DEBUG
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK ,__FILE__, __LINE__)
#else
#define DEBUG_NEW new
#endif
```

2. 全ての CPP ファイルに、最後の #include 行のすぐ後（最初の malloc/new の前に）に以下の行を記述する。

```
#ifdef _DEBUG
#define new DEBUG_NEW
#endif
```

上記では、以下のように表示される。：

```
Dumping objects ->
P:\SBKService\SBKMainFrame1.cpp(1036) : {5645} normal block at 0x00BB30F0
 Data: <              > CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD CD
```

メモリリークが発生したソースファイルと行番号が表示され、デバッグ出力ウィンドウでダブルクリックすることにより、ソース行にタグジャンプすることができる。

wxString の内容をデバッグウィンドウに自動的に表示する方法

プログラムをデバッグするとき、wxString 変数はデバッグウィンドウに以下のように表示される。：

```
[+] Variable   {...}
```

[+] をクリックして構造体を展開すると、m_pchData 変数を見ることができ、それが実際のデータへのポインタになっている。

これが面倒であり、解決方法を探した。AUTOEXP.DAT という興味深いファイルが存在し、これが望んでいたものであった。

1. AUTOEXP.DAT を探して開く。Version 6 を使用している場合には、\Program Files\Microsoft Visual Studio\Common\MSDev98\Bin ディレクトリにある。.NET Studio を使用している場合には、探す必要がある。

2. 以下の行を [autoexpand] セクション（または、ファイルの最後）に追加する。.NET 2003 を使用している場合には、[hresult] セクションに置かないように注意しなければならない。：

```
wxString=<m_pchData,st>
```

3. Visual Studio を再起動する。

これで全てである。これで、デバッグ時に文字列を表示すると、wsString 変数は以下のように表示される。：

```
[+] Variable   {“Contents of the string”}
```

特にこの変数の変化を見たいときに、多くのマウスクリックを減らすことができる。And while you are at it, read the comments in AUTOEXP.DAT. I am sure you can think of more variables you ’d like to handle this way.

While you are at it editing autoexp.dat...

ささいな関数へのステップインを防ぐ方法

デバッガでステップイン (F11) を実行し、引数に文字列を持つ関数にステップインしようとすると、デバッガは、まず文字列コンストラクタにステップインする。これは正しい動作であるが、面倒である。通常、文字列コンストラクタのデバッグは行わない。同じことは、多くのほかの関数、例えば new(), malloc(), wxString::c_str() にも言える。これは、ステップインをしても、これらの関数をステップオーバーするようにデバッガに指示を与える方法である。

ステップインしたくない関数ごとに、'NoStepInto' エントリを作成する。注意：Visual Studio は、関数名を指定するために正規表現を使用する。正規表現は、? や * などのワイルドカードよりも有用である。正規表現に関する詳しい情報は、ユーズネット (usenet) や wx のドキュメントを参照のこと。

注意：エスケープ文字として '\' を使用しなければならない。もしも wxString::c_str* をステップオーバーしたい場合には、wxString\:\:c_str.* としなければならない。\\:\\: と書くべきか、\:\: と書くべきか、ニュースグループや他の例などで、多くの混乱がある。私の観点では、\: で充分（正規表現として正しい）であり、他の表現は、C 文字列からニュースグループにやってきた表現である（My personal view is, \: is enough (as is correct with regexp) and the other expression came from pasting a C string to a newgroup post.）。

MS Visual Studio 6 を使用している場合には、autoexp.dat に以下を追加する。

```
[ExecutionControl]
wxString\:\:c_str.*=NoStepInto
operator new=NoStepInto
```

plugin::pdf::PDFParser=HASH(0x535b10).NET なので、途中、翻訳を省略
If you are using Visual Studio .NET 2003 or 2002, modify the registry.

1 . Add the Key NativeDE\StepOver under HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\7.1 (or 7.0 if you are using .NET 2002)

2. For every function you want to skip, add a new key/string value under the StepOver key.

3. The keys must be numerical. Think of them as linenumbers like in the old Basic days. Due to a bug in the Debugger, these numbers will get evaluated in reverse order.

4. The string contains the regular expression I have described above.

```
Example:
HKCU\Software\Microsoft\VisualStudio\7.1\NativeDE\StepOver\10 =wxString\:\:c_str.*
```

これにより、F11 を押したときや、" ステップイン " を使用したとき、これらの関数に対してデバッガがステップインすることを阻止できる。他の有用な表現は以下である。：

wxString Constructor	wxString\:\:wxString
wxString Destructor	wxString\:\:~wxString

これを使用してできることは、もっとたくさんある。microsoft.public.vsnet.debugging にあるこの記事は、これをコーディングした人によって書かれ、デバッガソースの断片を載せている。

あるいは (or)、google で NoStepInto を検索のこと。

plugin::pdf::PDFParser=HASH(0x535b10) 僕の環境（WinXP SP1, Ver.6.0 SP5 のはず）で試したら、下記の記述でうまくいきました

```
[ExecutionControl]
wxString::*=NoStepInto
```