コマンドラインコンパイラのWindows$_{TM}$アプリケーションへの編入

大　西　孝　臣*

Inclusion of command-line compiler within Windows$_{TM}$ applications

Takaomi OHNISHI

Abstruct

An approach to inclusion of an MS-DOS$_R$ command-line C compiler within a Windows$_{TM}$ application is introduced. The developed application is also useful for including compilers of any other languages.

1．はじめに

情報工学科では、学生実験において、ディジタル信号処理(DSP)の基本的概念について学習させる為のパソコンを用いた数値計算演習を行なっている。現在、演習での一項目として MS-DOS$_R$ の環境上で、MS-DOS$_R$版のC言語のコンパイラを用いて、学生にC言語のプログラムにより波形の時間離散での標本値データファイルを作成させ、教官が用意したMS-DOS$_R$版のFFT (Fast Fourier Transform)アプリケーションを用いて標本値データを解析させている。ところがMS-DOS$_R$版のFFTでは学生へのGUIによる情報提示能力に限界があり、学習効果を上げる為に苦慮していた。後に Microsoft$_R$Windows$_{TM}$版のFFTアプリケーションを開発したが、MS-DOS$_R$版のC言語の開発統合環境とWindows$_{TM}$版のFFTアプリケーションとの間の往来が煩雑で、操作性に問題があった。

そこで本稿では、開発中のDSP学習環境アプリケーションでのプリミティブな技術として、MS-DOS$_R$版のコマンドラインによるC言語コンパイラをWindows$_{TM}$版アプリケーションに編入させる手法を紹介し、Windows$_{TM}$環境におけるMS-DOS$_R$版コンパイラの有効活用を提案する。

2．アプリケーションの概要

本稿で紹介するWindows$_{TM}$のアプリケーションを実行している様子を、図1に示す。

本アプリケーションでは、MDI(Multi Document Interface)と呼ばれる、1つの親ウィンドゥの中に複数の子ウィンドゥが共存出来る構成を用いている。それら子ウィンドゥは2種類存在し、図1の例では、親ウィンドゥの中に各種の子ウィンドゥが1つずつ、計2つの子ウィンドゥが存在している。

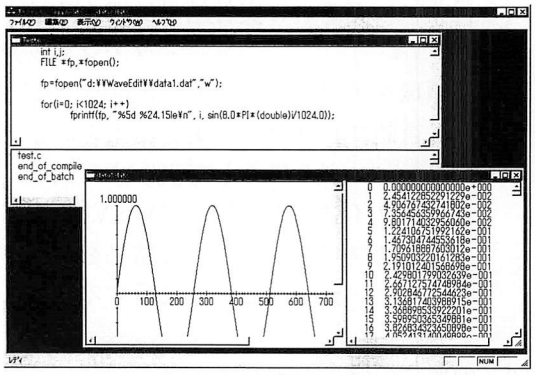

図1　アプリケーション実行時の様子

更に、各種のウィンドゥがそれぞれ分割ウィンドゥとなっており、縦2分割あるいは横2分割され、各子ウィンドゥが2つの領域を持っている。それら各領域は互いに異なる画像情報を提供している。

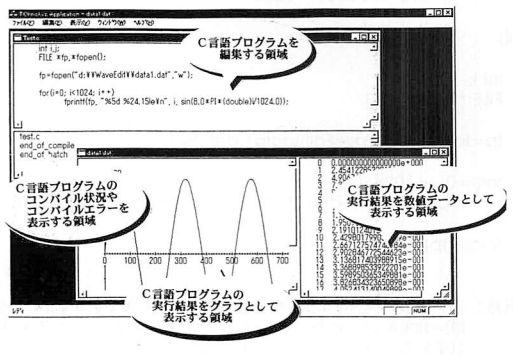

図2　アプリケーション内の各領域の説明

＊　助　手　情報工学科

各子ウィンドゥが持つ各種領域についての説明図を図2に示す。

各種の子ウィンドゥ、及び各子ウィンドゥが持つ各種領域は、次に示す役割を持っている。

1)子ウィンドゥタイプ1："CPROGTYPE"
《C言語プログラムに関する子ウィンドゥ》
・領域タイプ1："CCProgView"
〈C言語プログラムを編集する領域〉
・領域タイプ2："CCResultView"
〈C言語プログラムのコンパイル状況やコンパイルエラーを表示する領域〉

2)子ウィンドゥタイプ2："WAVEEDITTYPE"
《C言語プログラムの実行結果に関する子ウィンドゥ》
・領域タイプ1："CWaveGraphView"
〈C言語プログラムの実行結果をグラフとして表示する領域〉
・領域タイプ2："CWaveDataView"
〈C言語プログラムの実行結果を数値データとして表示する領域〉

図2の例での学生が編集すべきC言語のプログラムtest.cは次の通りである。但し前提として、学生がC言語プログラムを格納するディレクトリを、学生が操作してもよいファイル群を置くディレクトリとして d:¥junks¥ と定めた。

test1.cは波形データを計算し、データファイルdata1.datを、本稿でのアプリケーション自身の諸ファイルの存在場所であり、基本的に学生が操作しないファイル群を置くディレクトリである d:¥WaveEdit¥ に格納する。

```
// d:\junks\test.c：アプリケーション内で実行するC言語のプログラム例
#include <math.h>
#include <stdio.h>

#define PI 3.141592653589793238

main()
{
    int i;
    FILE *fp,*fopen();

    fp=fopen("d:\\WaveEdit\\data1.dat","w");

    for(i=0; i<1024; i++)
        fprintf(fp, "%5d %24.15le\n", i, sin(8.0*PI*(double)i/1024.0));

    fclose(fp);
}

// (例題) 半開半閉の区間 [0,1) の定義域を持つ独立変数 t に関する関数
//      f(t)=sin( 8πt ) について、t の定義域全体に渡って等間隔に
//      1024 点分のサンプリングを行なったとした場合を仮定し、
//      サンプリング値を倍精度にて計算し、計算した数値データ列を
//      データファイル "d:\WaveEdit\data1.dat" として出力する。
```

3．アプリケーションの開発

3-1 Visual C++TMとMFCからのクラス派生

本稿でのアプリケーションは、MicrosoftR VisualC++TM Ver.4.2eを用いて開発したWindowsTMアプリケーションである。VisualC++TMでは、WindowsTMアプリケーションの持つ機能をMFC(MicrosoftR Foundation Class)と呼ばれるクラスライブラリで提供しており、アプリケーションの開発者は、MFCライブラリから開発するアプリケーションの為のクラスを派生させて利用出来る。但し、WindowsTMにて動作するアプリケーションとして成立する為の条件として、図3に示す様に、アプリケーションクラス、ドキュメントクラス、フレームウィンドゥクラス、ビュークラスをMFCライブラリの基本クラスから派生させ、CDocTemplateクラスのインスタンスを派生させたアプリケーションクラス内で発生させ、ドキュメントテンプレートを構築しなければならない（ドキュメントテンプレートの具体的な機能については3-7節で述べる）。

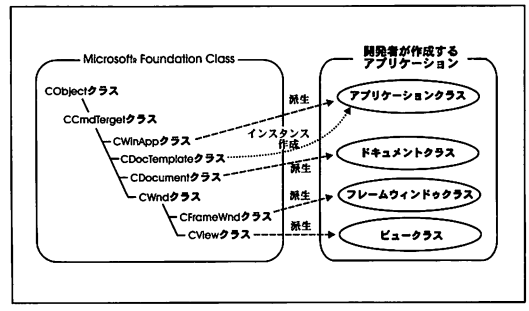

図3　MFCからアプリケーションへのクラス派生

本稿でのアプリケーションは、マルチドキュメント・マルチビューという、やや特殊な構成をしている為、各種クラスのMFCライブラリからの派生状況を次節以降にて述べる。

3-2 アプリケーションクラスの派生

アプリケーションクラスは、アプリケーションで唯一のクラスで、アプリケーション内の他のクラスやリソース（アプリケーションでのGUIの実体や、計算機の資源等を指す）を総轄する。本稿では図4に示す様に、CWaveEditAppクラスという名称でCWinAppクラスから派生させた。

又、アプリケーションクラスではドキュメントテンプレートを構築する。本稿でのアプリケー

ションはMDIである為、CMultiDocTemplateクラスのインスタンスを、2つの子ウィンドゥの種類（CPROGTYPEとWAVEEDITTYPE）に対応する形で発生させた（ドキュメントテンプレートの具体的な機能については3-7節で述べる）。

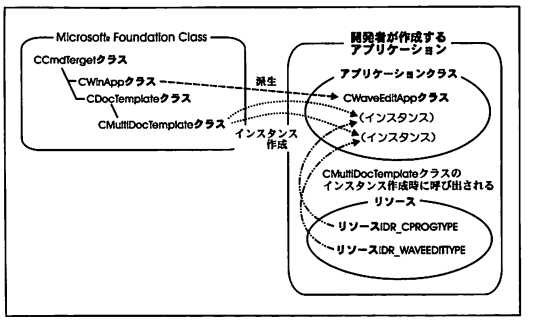

図4　アプリケーションクラスの派生

3-3 ドキュメントクラスの派生

ドキュメントクラスは、アプリケーションが持つデータを管理し、記憶装置（HDDやFDD等）とのファイルのやり取りを担う。本稿では図5に示す様に、MDIの2つの子ウィンドゥの種類（CPROGTYPEとWAVEEDITTYPE）に対応する形で派生させた。

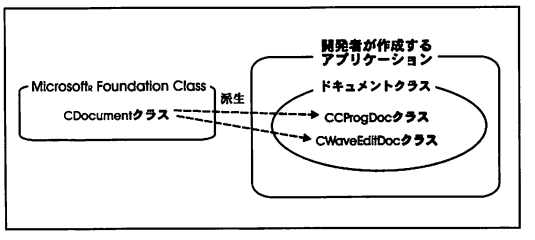

図5　ドキュメントクラスの派生

3-4 フレームウィンドウクラスとビュークラス

フレームウィンドウクラスとビュークラスは、アプリケーションとユーザとの間のGUIを担う。本稿でのアプリケーションの実行画面における各クラスの対応は、図6に示す通りである。

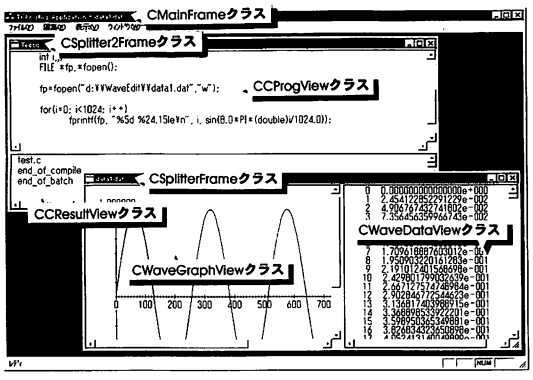

図6　実行画面におけるフレームウィンドゥクラス・ビュークラスの対応

3-5 フレームウィンドゥクラスの派生

本稿でのアプリケーションのフレームウィンドゥクラスは、図7に示す様に、MDIの親ウィンドゥのフレームをCMDIFrameWndクラスから派生させ、子ウィンドゥCPROGTYPEのフレームクラスをCSplitter2Frameクラスという名称で、子ウィンドゥWAVEEDITTYPEのフレームクラスをCSplitterFrameクラスという名称で、それぞれCMDIChildWndクラスから派生させた。なお、2つの子ウィンドゥは共に分割ウインドウである為、実際には、MFCのCSplitterWndクラスのインスタンスが、各子ウィンドゥの派生フレームウィンドウクラス内で起動され、メソッドを利用する形になる。

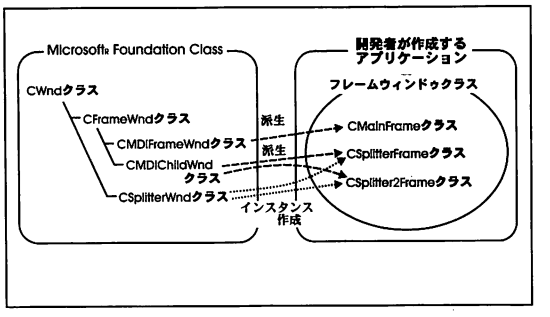

図7　フレームクラスの派生

3-6 ビュークラスの派生

本稿でのアプリケーションのビュークラスは、図8に示す様に、CCProgViewクラスはテキストファイルを編集出来るエディタ機能を提供するCEditViewクラスから派生させ、その他の派生ビュークラスはスクロールの機能があるグラフィック画面を提供するCScrollViewクラスから派生させた。

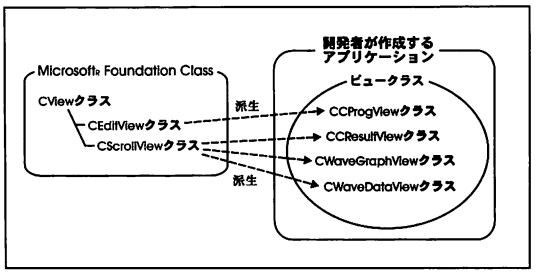

図8　ビュークラスの派生

3-7 ドキュメントテンプレートと ドキュメント＝ビュー構造

3-2節にても述べたが、アプリケーションクラスCWaveEditAppでは、MDIの2種類の子ウィンドゥCPROGTYPEとWAVEEDITTYPEに対応する形でそれぞれのドキュメントテンプレートを構築する。

ドキュメントテンプレートは、アプリケーションで使用する各ドキュメントクラス毎に、ドキュメントクラス・フレームウィンドゥクラス・ビュークラスの関連性を次の様に定義し、関連付けられた各クラスはCWaveEditAppクラス内で、図9に示す通りに機能する。

1)CPROGTYPEテンプレート

・ドキュメントクラス：　CProgDoc
・フレームウィンドウクラス：　CSplitter2Frame
・ビュークラス：　CCProgView,CCResultView

2)WAVEEDITTYPEテンプレート

・ドキュメントクラス：　CWaveEditDoc
・フレームウィンドウクラス：　CSplitterFrame
・ビュークラス：　CWaveGraphView,CWaveDataView

図9でのドキュメントテンプレートにおけるフレームウィンドゥクラスとビュークラスとの関連性については、図6においても紹介した通り、実行画面においてのビューという画像情報を提供する領域と、そのビューを囲むフレームとの関連を表している。

もう一方のドキュメントクラスとビュークラスとの関連性を最も簡単に説明すれば、アプリケーションが持つデータを担当するクラスと、アプリケーションが出力する画像情報を担当するクラスを分けるという事であるが、アプリケーションの開発者はドキュメントテンプレートを用いて、1つのドキュメントクラスに対して複数のビュークラスを関連付ける事（すなわちマルチビュー構成）が出来、例えばCWaveEditDocクラスに対するCWaveGraphViewクラスとCWaveDataViewクラスの様に、同じ数値データ列に対するグラフ表示と数値データの列挙表示といった多面的な情報提示を可能にしている。この考え方が“ドキュメント＝ビュー構造”と呼ばれる、VisualC＋＋TMを用いたWindowsTMアプリケーション開発での重要な概念である。

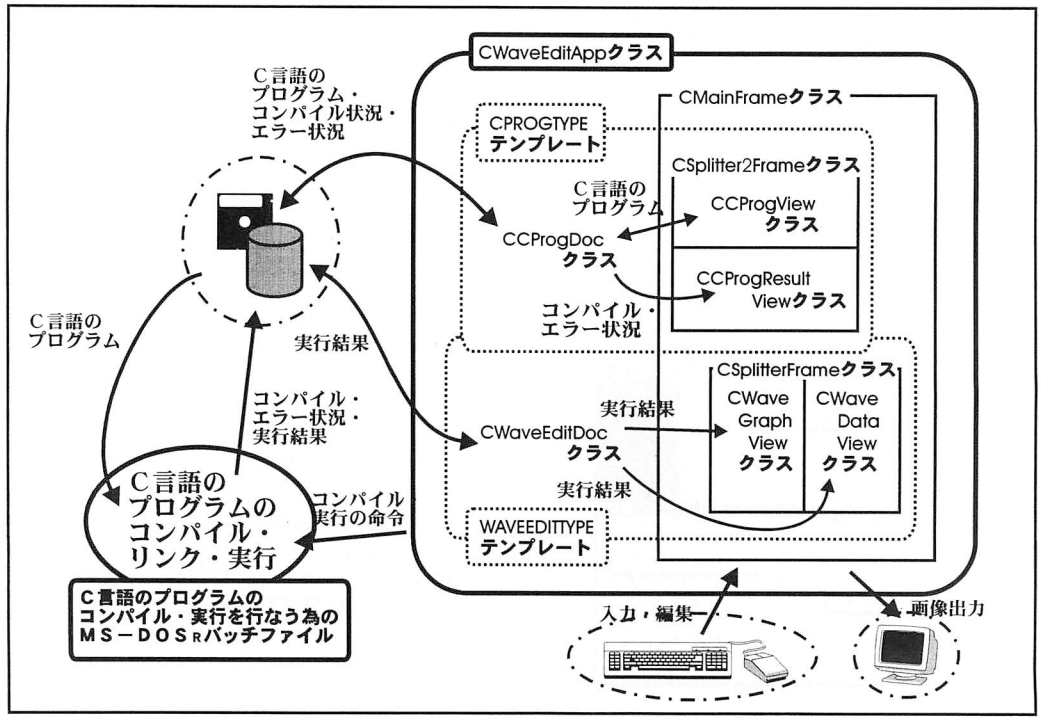

図9　アプリケーション内のドキュメント＝ビュー構造とファイル授受の様子

3-8 アプリケーション内での「C言語プログラムのコンパイル・実行」の実現

図9に示す様に、本稿でのアプリケーションにて学生が作成したC言語のプログラムtest.cをコンパイルし実行させる方法として、アプリケーションクラスCWaveEditApp（正確には図10の通りに、CWaveEditAppクラスのメソッドOnRun()が起動させた、MDIの親ウィンドゥのフレームクラスCMainFrameのメソッドOnRun()）が、次頁に示すバッチファイルb.bat（ディレクトリ d:¥junks¥ にて格納されているとする）を、Windows$_{TM}$の「MS-DOS$_R$プロンプト」にて起動させている。なお、b.batが起動している間、MS-DOS$_R$プロンプトのウィンドゥはアイコンへと縮小させているので、学生はMS-DOS$_R$プロンプトウィンドゥの存在を意識しない。

バッチファイルb.batでは、VisualC++$_{TM}$に付属しているMS-DOS$_R$のコマンドラインで使用出来る、C/C++コンパイラ「CLコマンド」を用いたが、コマンドラインで使用出来るコンパイラなら、b.batを編集する事により、C言語以外のコンパイラを含め、本稿でのアプリケーション環境での広い範囲の応用が可能であると考える。

図10に、C言語プログラムのコンパイル・実行の具体的手順を示す。

学生がC言語プログラムの“実行の命令”（マウスによるメニューの選択）を本アプリケーションに与えると、CWaveEditAppクラスのメソッドOnRun()が“実行の命令”というイベントに対するハンドリングとして起動し、図10に示す一連の手続きを行なう。CMainFrameクラスのメソッドOnRun()がb.batを起動させる事による標準出力（コンパイラが出力するエラー表示等）は、メソッドOnRun()が「コンパイル結果ファイル」を発生させてリダイレクトする。但し、b.batのプロセスよりもOnRun()のプロセスの方が先行してしまうので、コンパイル結果ファイルのファイル名は、イベントが発生する度毎にユニークな名称にし、b.batがコンパイル結果ファイルを解放するのをOnRun()は待たなければな

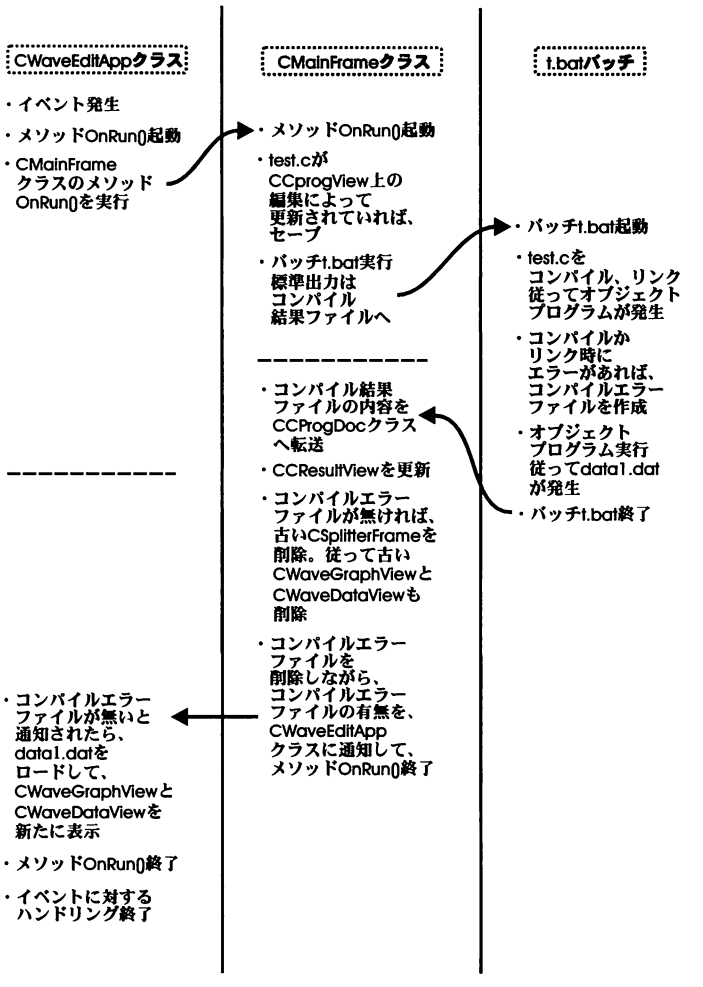

図10　アプリケーションでのC言語プログラムのコンパイル・実行の手順

らない。

```
: d:\junks\t.bat : C言語のプログラム"d:\junks\test.c"の
:                 コンパイル、リンク、実行を行なうバッチファイル

@echo off

:C言語プログラムのコンパイル、リンク
cl d:\junks\test.c /Fod:\junks\test.obj /Fed:\junks\test.exe /nologo

:コンパイル・リンク時にエラーがあれば、
:ファイル"d:\WaveEdit\error.txt"を作成する。
:コンパイル・リンク時にエラーがなければ、
:ファイル"d:\WaveEdit\error.txt"を削除する。
if errorlevel 3 echo error3 > d:\WaveEdit\error.txt
if errorlevel 3 goto label
if errorlevel 2 echo error2 > d:\WaveEdit\error.txt
if errorlevel 2 goto label
if errorlevel 1 echo error1 > d:\WaveEdit\error.txt
if errorlevel 1 goto label
del d:\WaveEdit\error.txt

label:

echo end_of_compile

:C言語プログラムの実行
d:\junks\test.exe

echo end_of_batch
```

又、C言語プログラムのコンパイルにおけるエラーが無かった場合、各派生クラスが持つメンバメソッドの関係により、CMainFrameクラスにて古いCWaveGraphViewとCWaveDataViewを破棄し、CWaveEditAppクラスにてtest1.cが作成したデータファイルdata1.datをロードして、新たなCWaveGraphViewとCWaveDataViewを構築する（次図参照）。

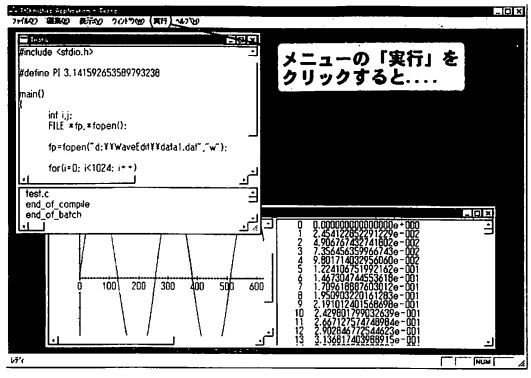

図11　メニュー選択による実行命令

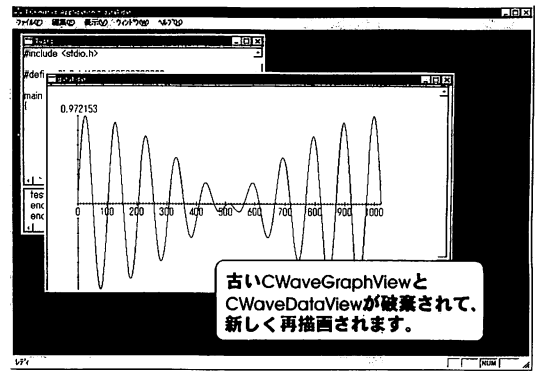

図12　ビューの再構築の様子

4．おわりに

本稿では、MS-DOS$_R$版のコマンドラインによるC言語コンパイラをWindows$_{TM}$アプリケーションに編入させる手法を提案した。本稿で開発したアプリケーションでは、バッチファイルによる設定を編集する事によりC言語以外コンパイラでの応用も可能であり、Windows$_{TM}$環境において既存のMS-DOS$_R$版コンパイラを有効に活用出来る。

参考文献

1) D. J. Kruglinski著、榊　正憲・梅原　系　共訳、Inside Visual C++ Version4、アスキー、1996

（平成9年11月28日受理）