

# スクールCOBOL2002

## 操作入門書

－(株)日立製作所－

(2005年8月作成)

# 目 次

1. COBOL2002起動から実行まで	----- 3
2. テストデバッガの使用方法	----- 35
3. 関連資料	----- 69
(a) ファイルの入出力処理	----- 70
(b) テストデータの作成方法	----- 81
(c) 用紙の節約方法	----- 92
(d) 印刷書式の設定方法	----- 98
(e) エディタ設定方法	---- 103
(f) コンパイルリストの入手方法	--- 117
(g) オンラインマニュアルの使用方法	--- 127
(h) 登録集原文の指定方法	--- 135
(i) サブプログラムの追加方法	--- 143
(j) 索引ファイルを新規作成する方法	--- 147
(k) 既存のプロジェクトマスタファイルの開き方	150

# 1. COBOL2002起動から実行まで

—初めてCOBOL2002を使う方のために—

# 目 次

1. はじめに
2. COBOL2002の起動
3. プロジェクトマスタファイルの作成
4. プロジェクトの作成
5. COBOLソースプログラムの編集
6. コンパイル(実行用ファイルの生成)
7. 実行
8. プロジェクトの追加
9. 終わりに

# 1. はじめに

これから、COBOL2002の使い方を順を追って説明していきますが、以下の注意事項をまずお読みください。

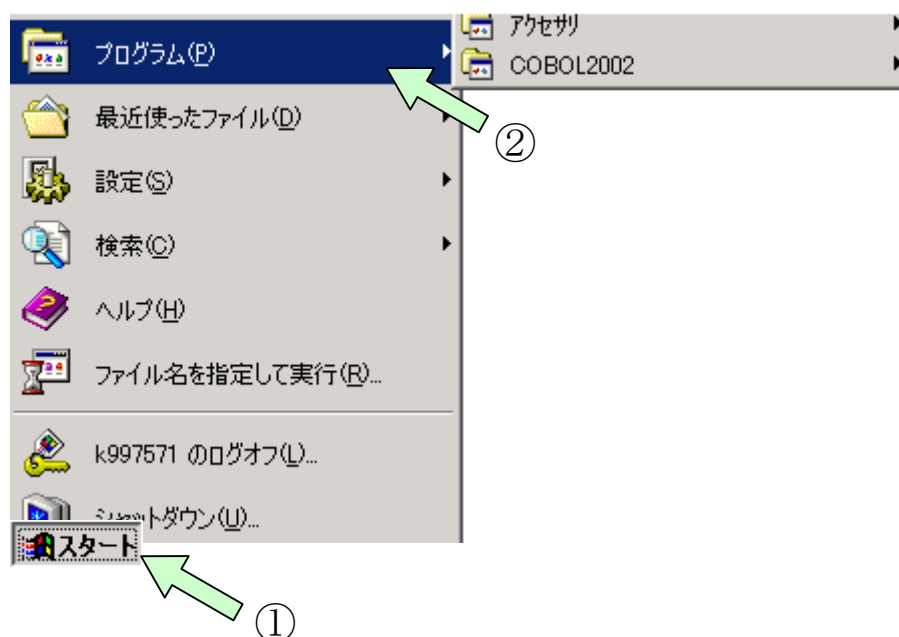
## 注意事項

- (1) COBOLのソースプログラムやでき上がる実行形式ファイルは特定のフォルダの下に作ります。後で削除等しやすいように、練習用のフォルダを作成ください。本説明書では、Cドライブの下にtempという名称のフォルダを作成し、その下にsample01というフォルダを作成するものとして、説明を進めて行きます。

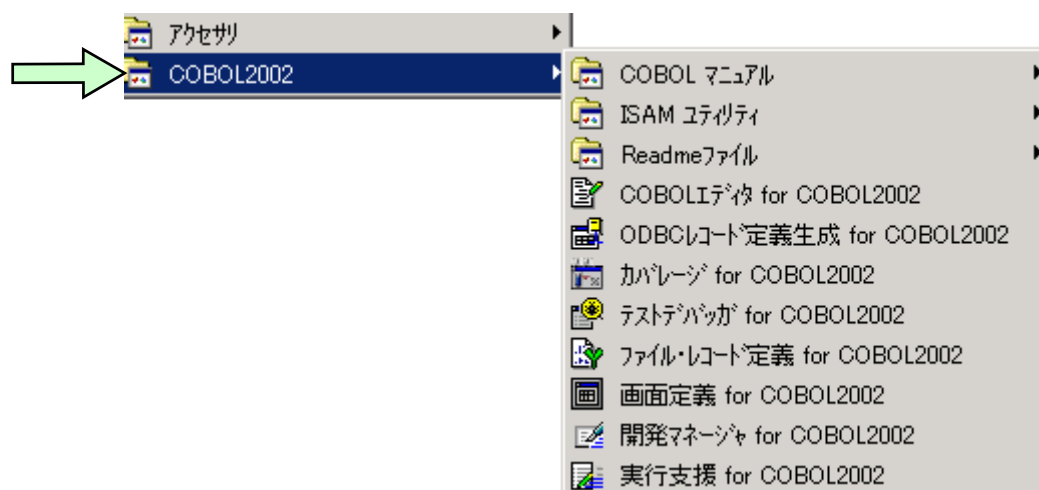
## 2. COBOL2002の起動

COBOL2002を起動するには、スタートボタンから行う方法と、COBOL2002の各種ツールをアイコン化しておいて、そのアイコンを(ダブル)クリックする方法があります。ここでは、スタートボタンから行う方法を説明します。

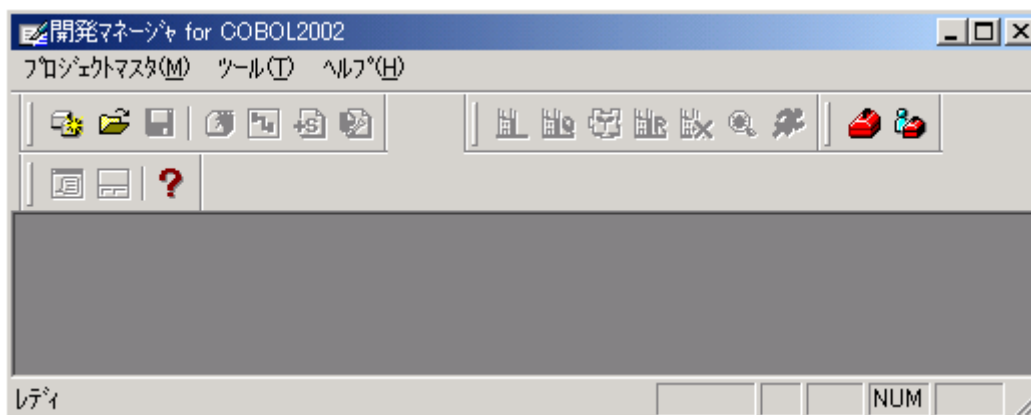
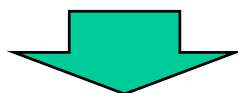
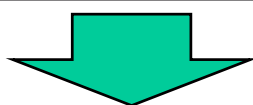
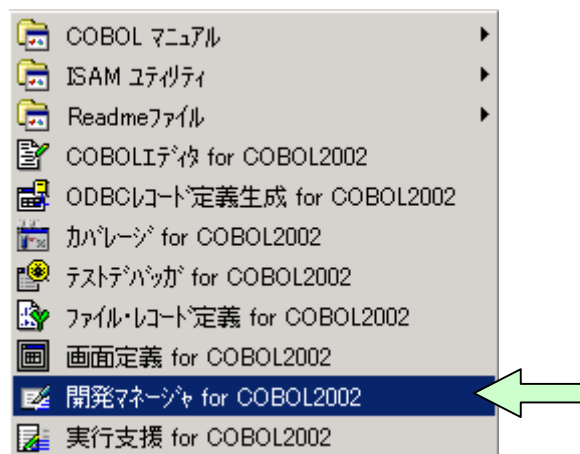
[手順 1] スタートボタンを押し、「プログラム(P)」の所にマウスポインタを移動します。すると起動できるプログラムの一覧が出てきます。



[手順 2] プログラムの一覧の中からCOBOL2002の所にマウスポインタを移動します。するとCOBOL2002の中の使用できるツール一覧が表示されます。



〔手順3〕 COBOL2002の中の使用できるツールの中から、COBOL2002開発マネージャを選びクリックします。すると、COBOL2002開発マネージャが起動されます。



すぐにこの画面が表示されます。



#### 〔用語解説〕 COBOL2002開発マネージャ

日立COBOL2002を用いてコーディングからコンパイル、テスト、実行等の操作をコントロールするツールです。基本操作ではこのツールを使用します。

### 3. プロジェクトマスタファイルの作成

COBOL2002を起動したあと、まず、ソースプログラムの作成からコンパイル・実行までの一連の作業を行うための環境を整えます。

ソースプログラムの作成から実行までを行うために必要な各種リソースやコンパイラオプション等の管理を行うファイルをプロジェクトマスタファイルと言います。最初に「デフォルトオプションの設定」をしてからプロジェクトマスタファイルを作成します。

#### [デフォルトオプションについて]

コンパイラに対するオプション情報をコンパイラオプションと言います。コンパイラオプションは、「プロジェクトの設定」メニューから設定することができますが、プログラムを作成するたびに設定しなければなりません。全てのプログラムに共通のコンパイラオプションは、デフォルトオプションで設定しておくと便利です。デフォルトオプションを設定しておく、新規にプログラムを作成するとき、コンパイラオプションの初期値(設定済の扱い)として有効になります。

#### [手順1] デフォルトオプションの設定

基本操作では、次のコンパイラオプションを設定します。

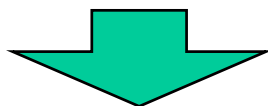
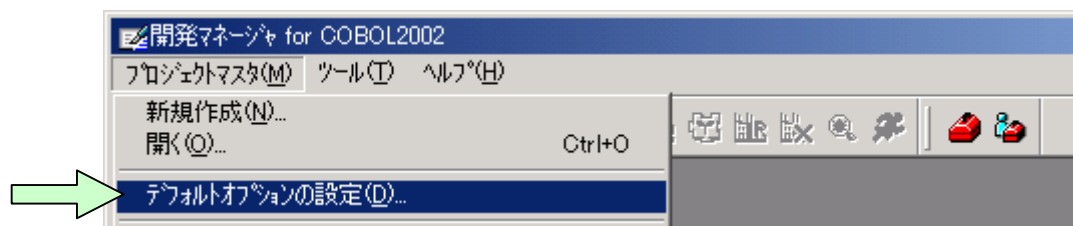
##### ①コンパイルリストの出力

コンパイルした結果のリストが出力されます。

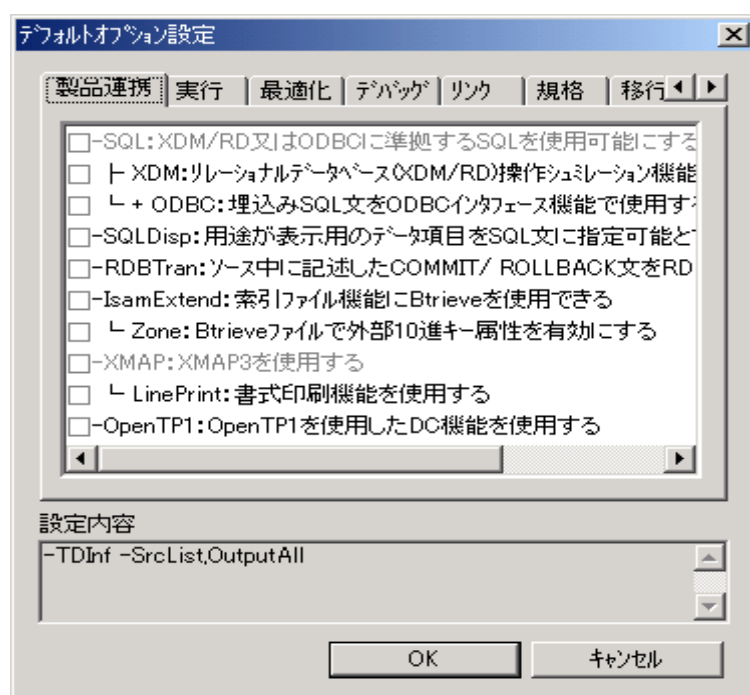
##### ②デバッグ情報の出力

テストデバッガを使用してデバッグするために必要な情報が出力されます。

開発マネージャの画面から、プロジェクトマスタファイルを開いていない状態で設定します。メニューバーの「プロジェクトマスタ(M)」－「デフォルトオプションの設定(D)」をクリックしてください。「デフォルトオプションの設定」画面が表示されます。







## 〔手順2〕 該当するコンパイラオプションの設定

コンパイラオプションは、カテゴリ別に分かれて表示されます。該当するカテゴリのタブをクリックしてからコンパイラオプションをチェックします。

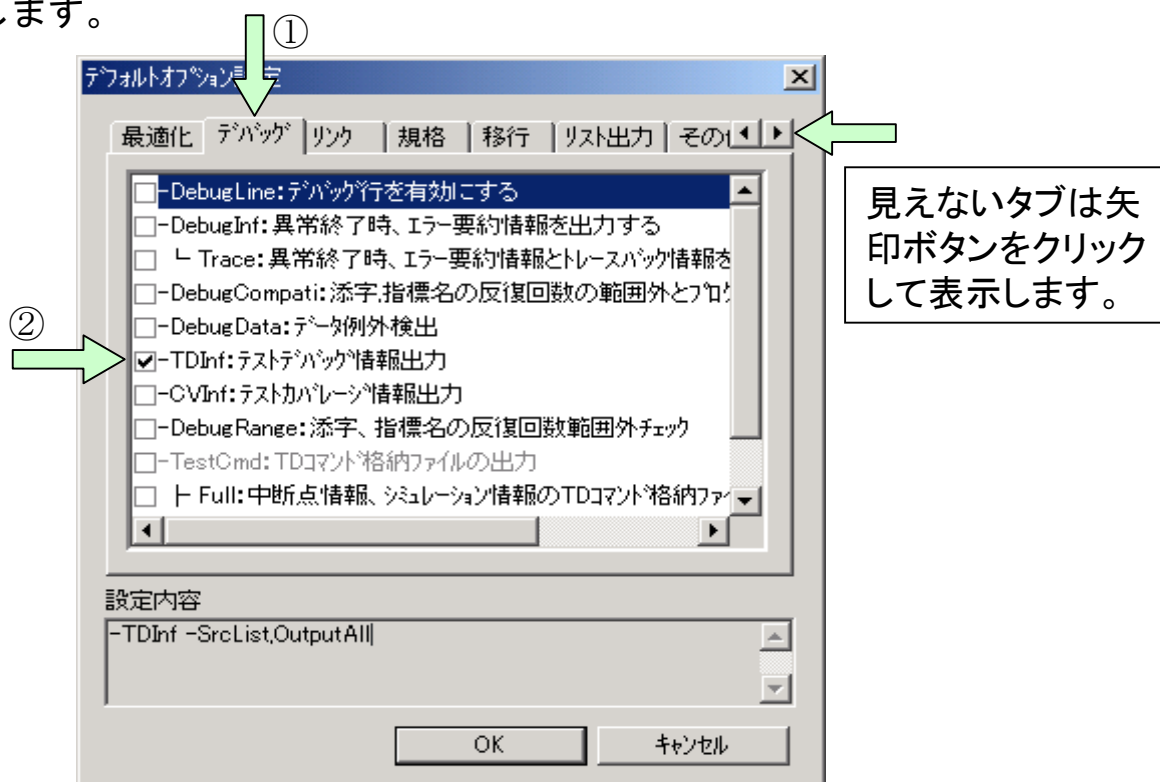
①「デバッグ」タブをクリック（下記画面参照）し、次のオプションをチェックする。

「デバッグ」タブ： ☐ -TDInf: テストデバッグ 情報出力

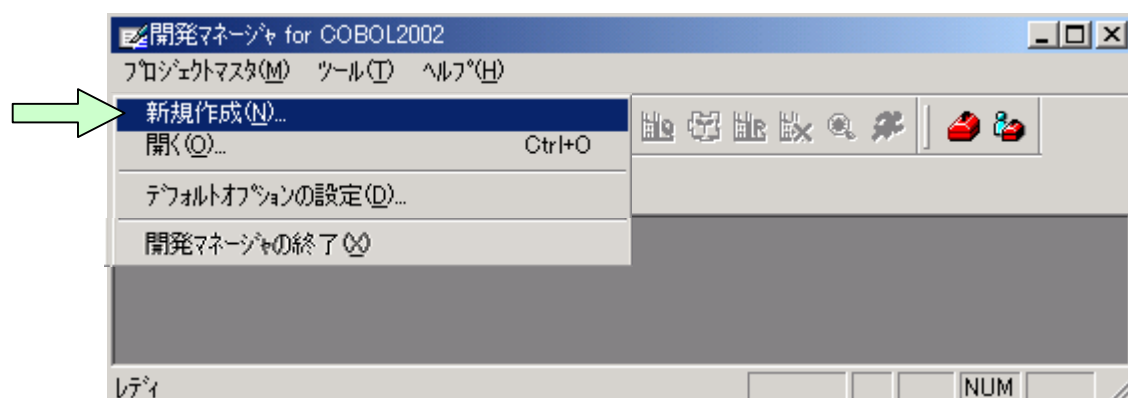
②右矢印（▶）をクリックして「リスト出力」タブを表示し、次のオプションをチェックする。

「リスト出力」タブ： ☐ - OutputAll: 全てのソースをリストに展開する

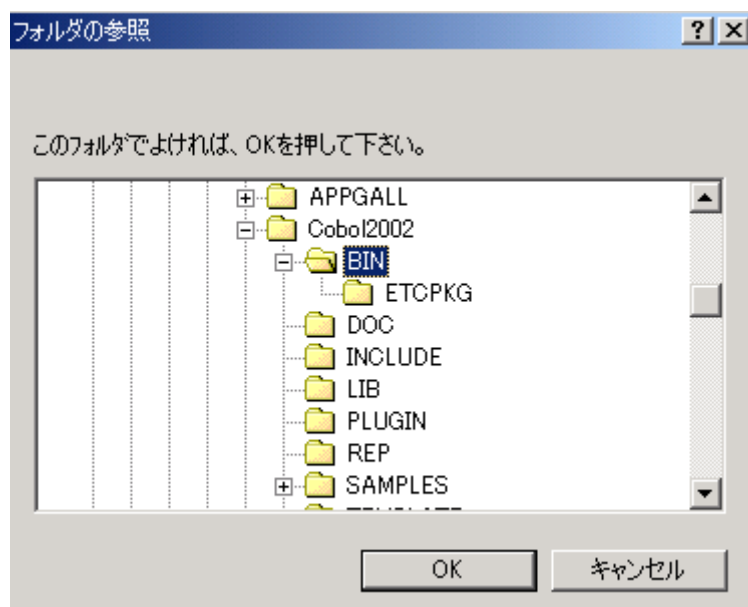
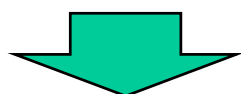
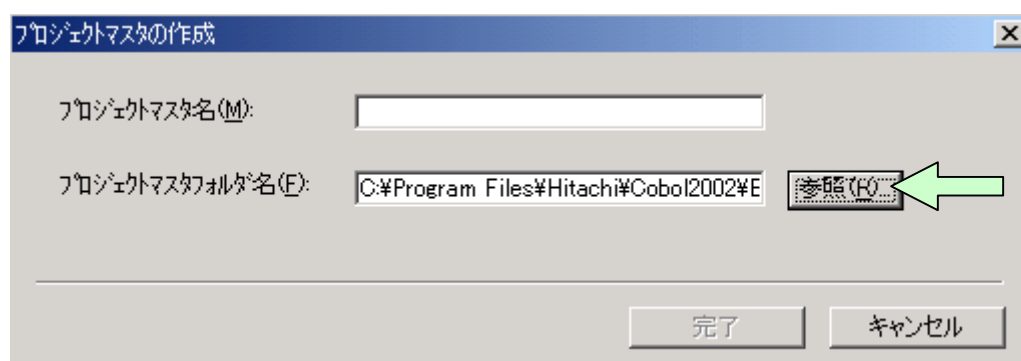
二つのオプションのチェックが完了したら、「Enter」キーを押すか「OK」ボタンをクリックします。



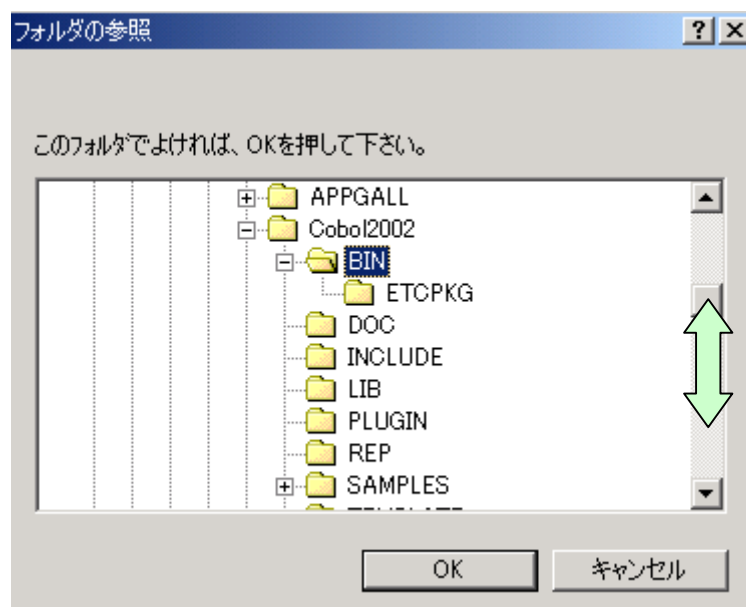
- [手順3] プロジェクトマスタファイルを新規作成します。  
開発マネージャのメニューバーの「プロジェクトマスタ(M)」-「新規作成(N)」をクリックしてください。すると新規作成画面が表示されます。



- [手順4] 新規作成画面の「参照(R)」ボタンを押します。  
すると、「フォルダの参照」画面が表示されます。

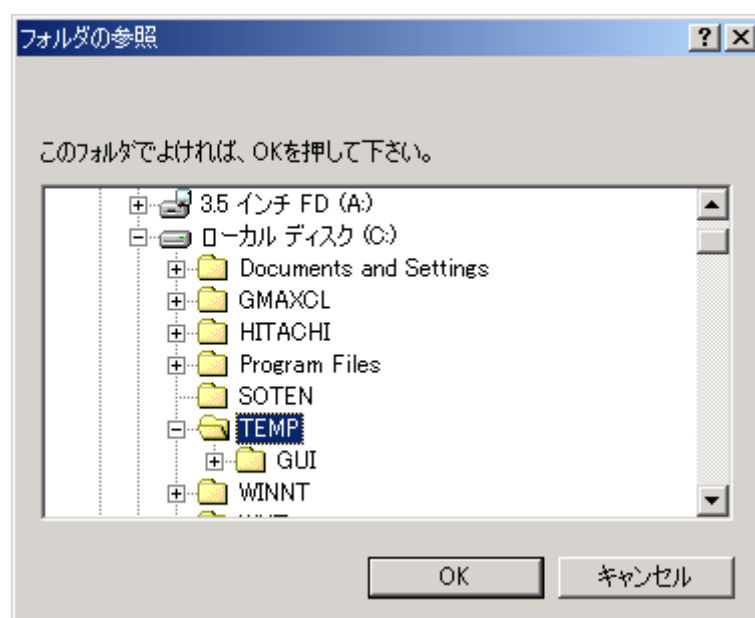


- [手順5] 「フォルダの参照」画面で、右側のスクロールバーを使って、Cドライブのtempフォルダを探します。

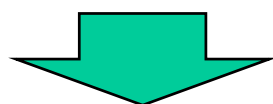
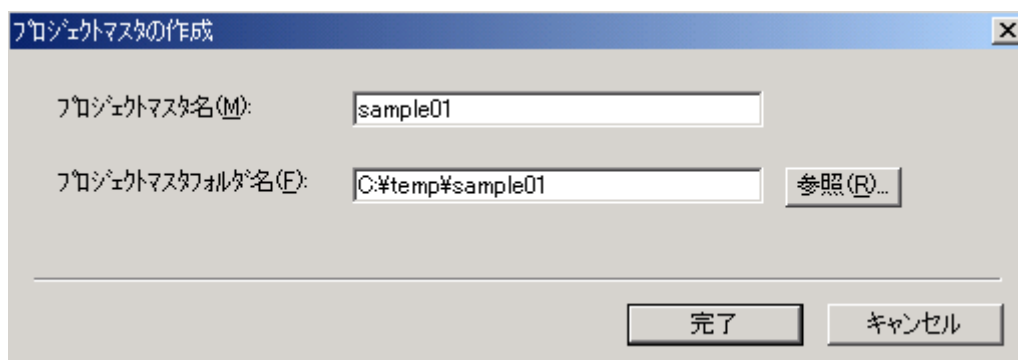


上下にスクロールし、  
表示位置を調整する。

- [手順6] 「tempフォルダ」を選び「Enter」キーを押すかまたは「OK」ボタンをクリックします。



[手順7] プロジェクトマスタ名に「sample01」と入力し、「Enter」キーを押すかまたは「完了」ボタンをクリックしてください。



「Enter」キーを押す。

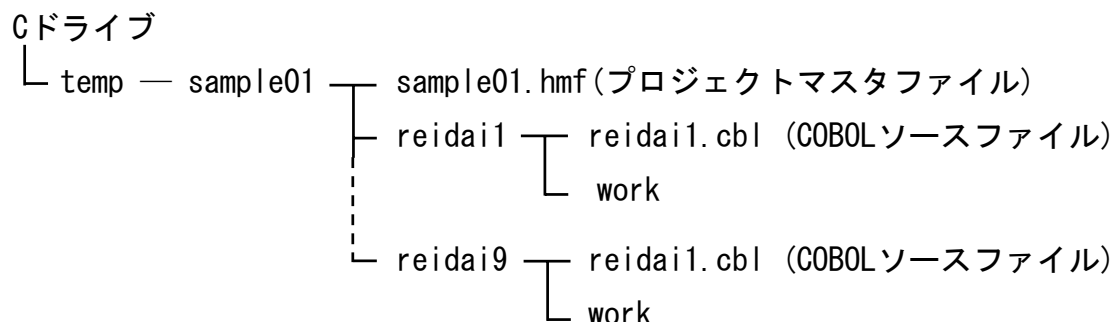
[手順8] 次の画面が表示されるので、「Enter」キーを押すかまたは「OK」ボタンをクリックしてください。



「Enter」キーを押す。

## 4. プロジェクトの作成

プロジェクトマスタは、プロジェクトという単位でリソースを管理します。基本操作では、Cドライブのtempフォルダの下にsample01というプロジェクトマスタを作成し、その中にreidai1というプロジェクト(プロジェクト名はソースファイル名と同じにします)を作成します。このときのフォルダ構成は次のようになります。



※ : sample01の下に複数のプロジェクトを定義できます。基本操作では、reidai1の作成とreidai2のプロジェクトの追加だけを説明します。

### [概要] プロジェクトの作成

プロジェクトは、次の操作で作成します。

- ・ 入力要求には「プログラム名」を入力し、あとは全て「Enter」キーを押すだけで作成できます。

詳細の手順を、次に示します。

- [手順 1] プロジェクトマスタファイルの設定に続いて次の画面が表示されます。プロジェクト名「reidai1」を入力して「Enter」キーを押してください。(「OK」ボタンをクリックしてもよい。)



「reidai1」と入力し、「Enter」キーを押す。

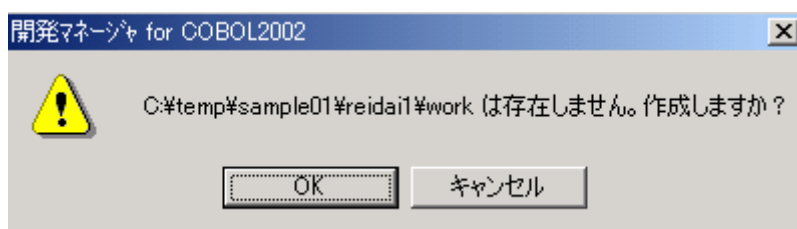
### [ワンポイントアドバイス]

プログラム名は3回入力します。毎回 入力するとスペルミスをすることがあるので、カット＆ペーストで貼り付けるとよいでしょう。

[手順2] 次の画面が順に出ますので、続けて「Enter」キーを押してください。  
（「OK」ボタンをクリックしてもよい。）

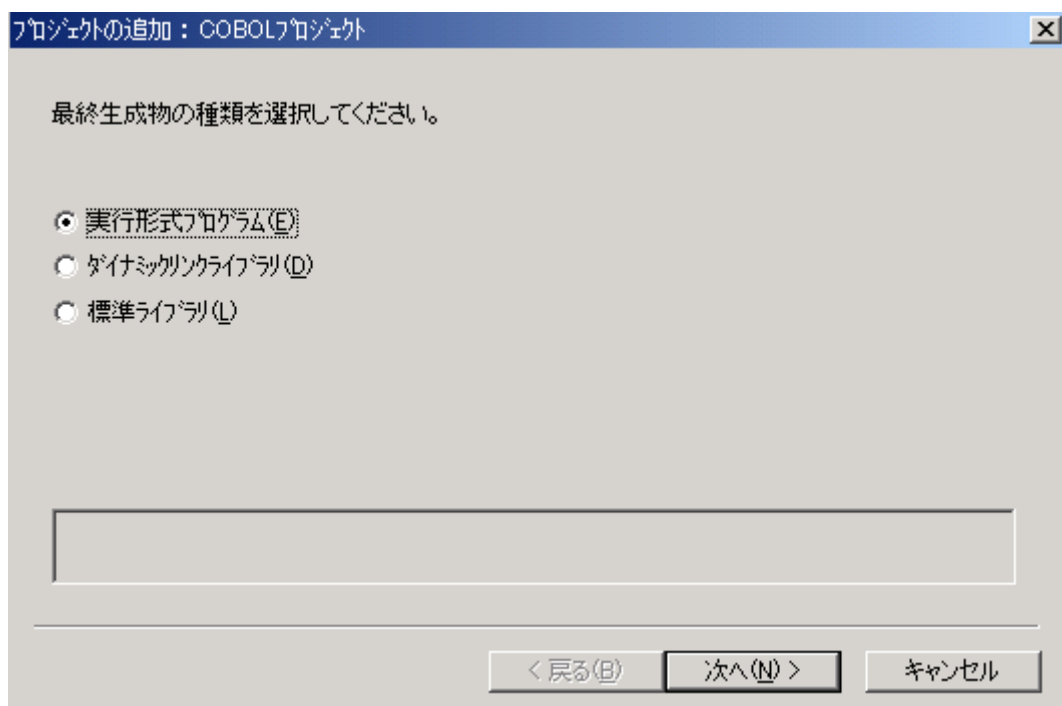


「Enter」キーを押す。



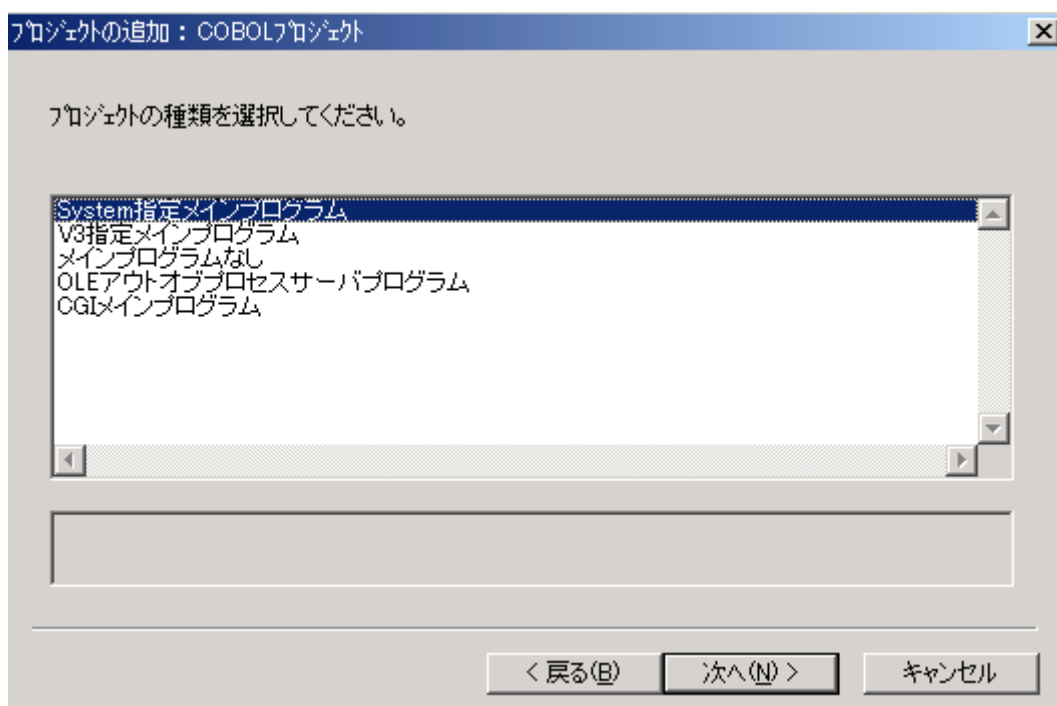
「Enter」キーを押す。

[手順3] プロジェクトの追加画面が表示されます。デフォルトの「実行形式プログラム(E)」を指定すればよいので、この画面も「Enter」キーを押してください。（「次へ(N)」ボタンをクリックしてもよい。）



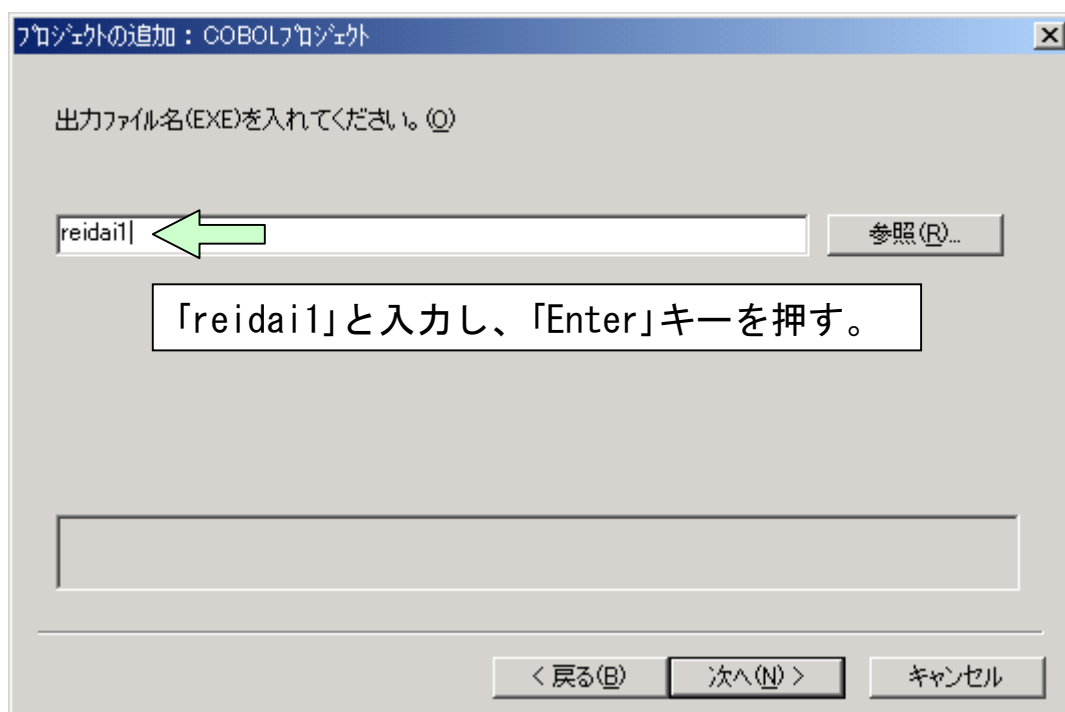
「Enter」キーを押す。

- [手順4] プロジェクトの種類もデフォルトの「System指定メインプログラム」でよいので、「Enter」キーを押してください。  
(「次へ(N)」ボタンを押してもよい。)

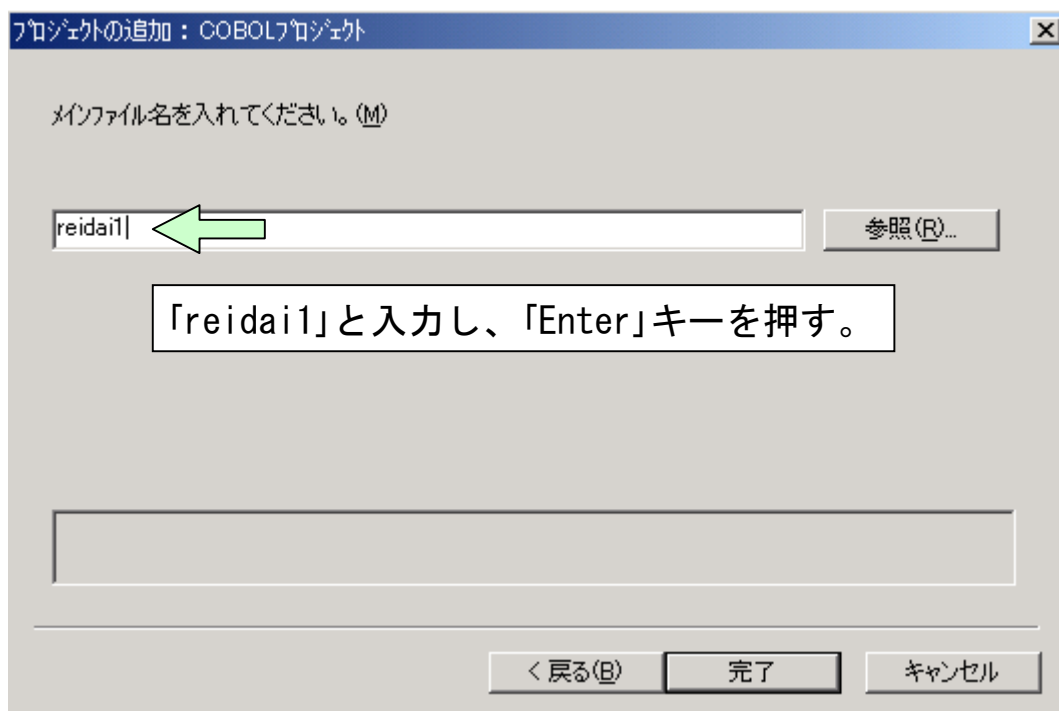


「Enter」キーを押す。

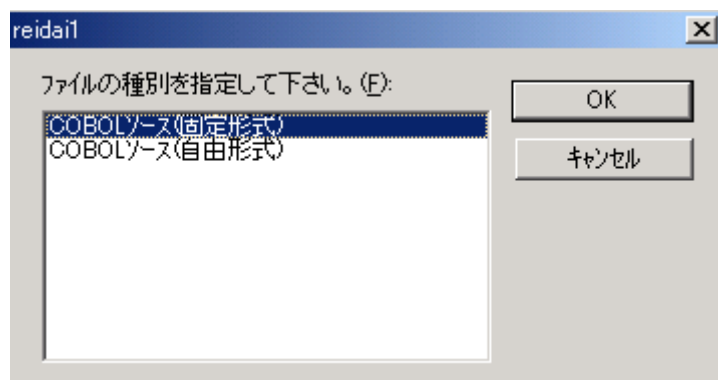
- [手順5] 出力ファイル名に「プログラム名」(プロジェクト名と同じ名称)を入力して「Enter」キーを押してください。(「次へ(N)」ボタンを押してもよい。)



- [手順6] メインファイル名も「プログラム名」(プロジェクト名と同じ名称)を入力します。続いて「Enter」キーを押してください。  
(「完了」ボタンをクリックしてもよい。)



- [手順7] ファイルの種別もデフォルトの「COBOLソース(固定形式)」でよいので、「Enter」キーを押してください。  
(「OK」ボタンをクリックしてもよい。)



「Enter」キーを押す。

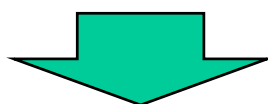
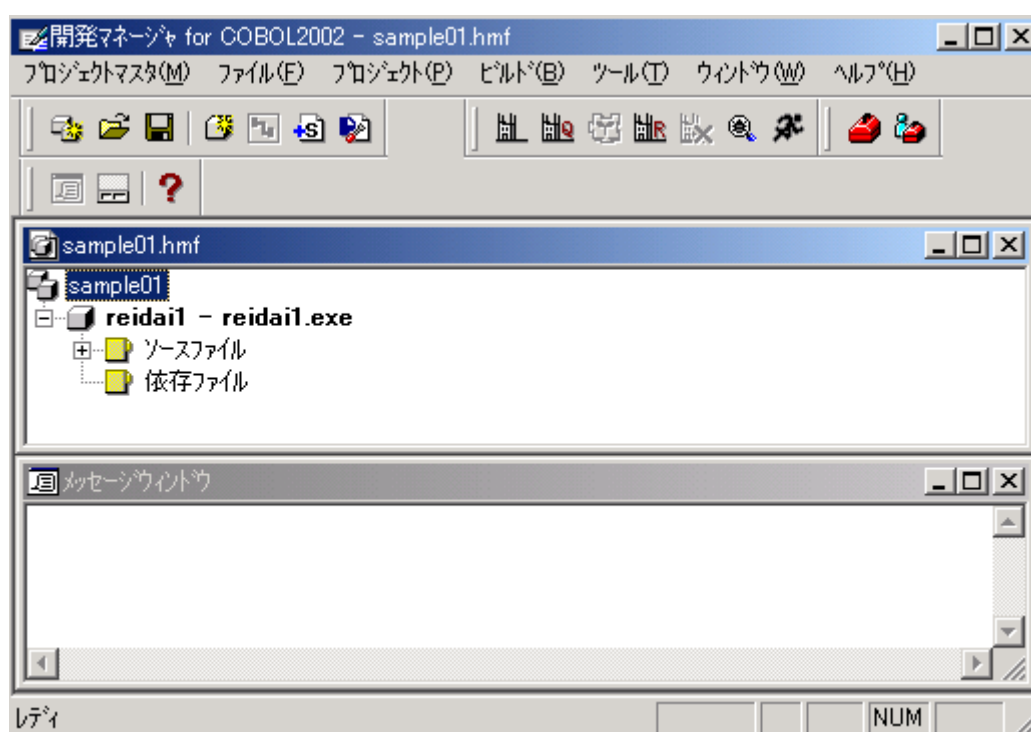
これでプロジェクトの設定が完了です。プログラム名の入力が3回、あとは「Enter」キーを押していく(8回)だけでできました。



〔手順 8〕 開発マネージャの画面に戻り、実行ファイルとソースファイルフォルダ、依存ファイルフォルダ、メッセージウィンドウが表示されます。

### 〔ワンポイントアドバイス〕

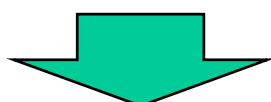
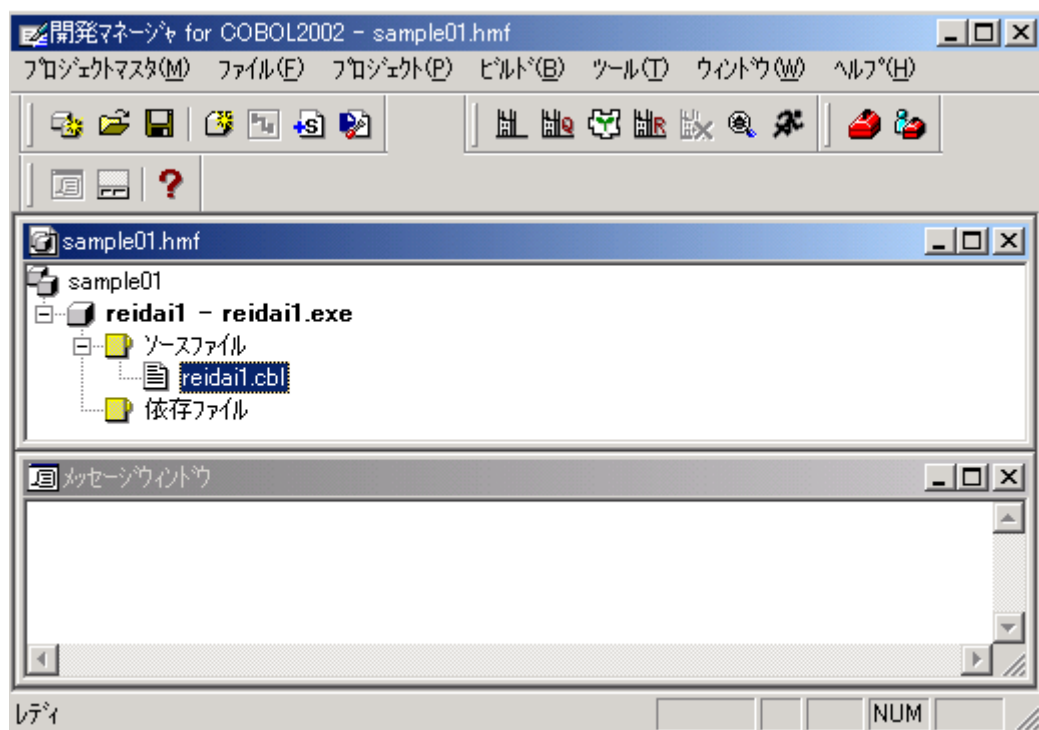
- ①ここで、一度保存しておくといよいでしょう。作業終了時に開発マネージャを閉じるとき、保存するか否かを聞いてきますが、うっかり「いいえ」をクリックしてしまうとせっかく設定してきた内容が失われます。作業終了時にも保存する必要がありますが、ここで保存しておけば全てを失うことは避けられます。
- ②メッセージウィンドウが表示されない場合は、開発マネージャのメニューバーの「ウインドウ(W)」をクリックし、プルダウンメニューの中の「並べて表示(T)」をクリックしてください。実行ファイルとメッセージウィンドウが並べて表示されます。他に「重ねて表示(C)」もできます。




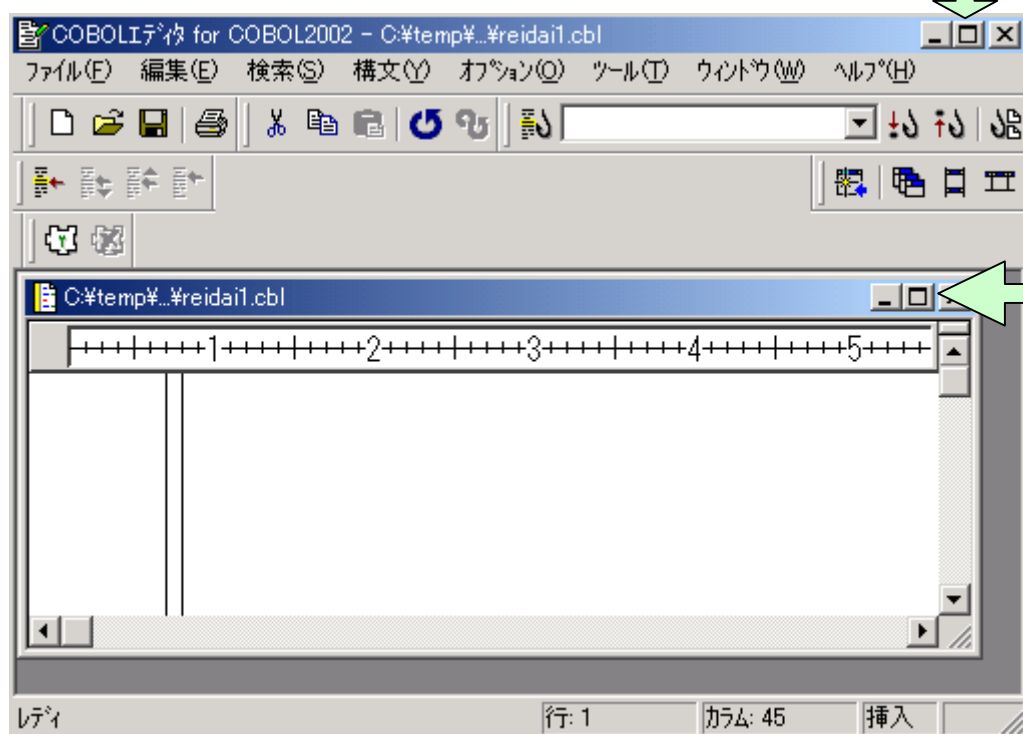
## 5. COBOLソースプログラムの編集

開発マネージャで、環境を作りました。この後は実際にCOBOLソースプログラムをコーディングしていきます。

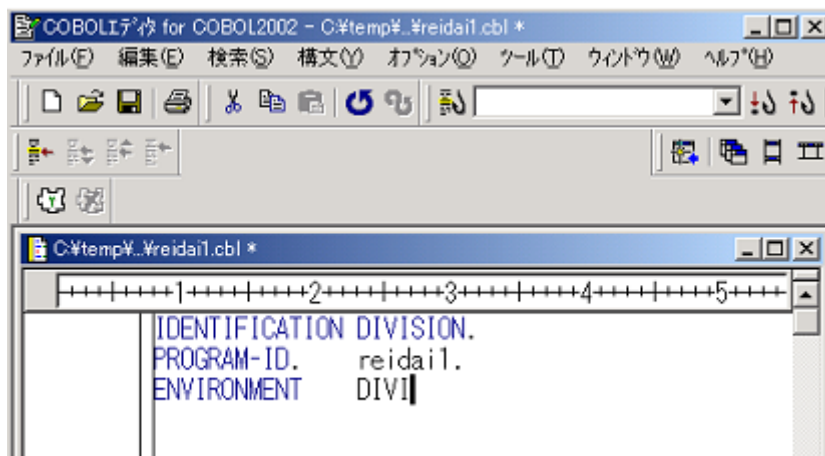
[手順 1] 開発マネージャに表示されている、ソースファイルフォルダ配下にある.cblファイルをダブルクリックしてください。すると、自動的にCOBOL2002の専用エディタが起動されます。



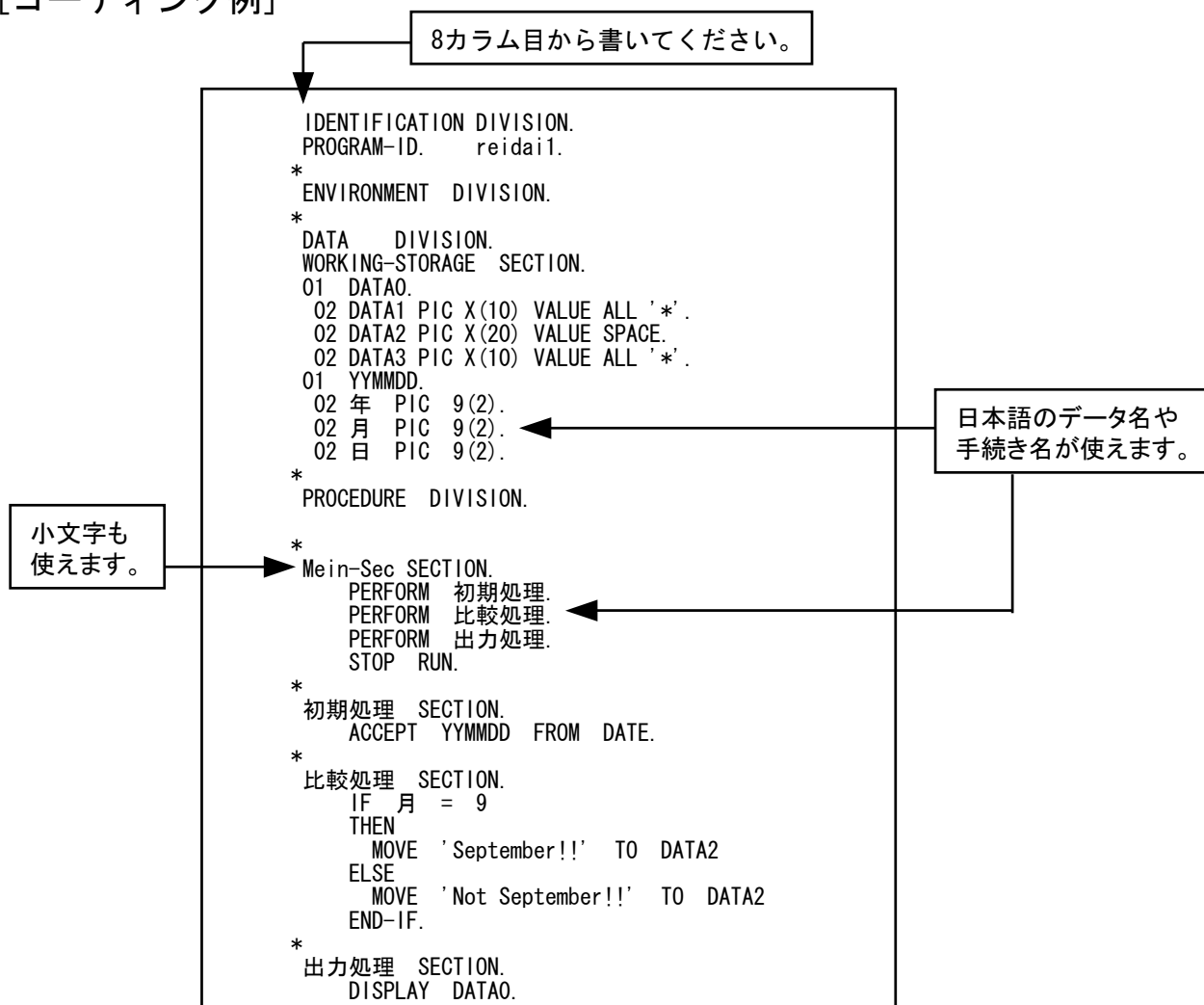
COBOLエディタのウィンドウの大きさは右上の  ボタン等で調整してください。



[手順2] エディタを使用してCOBOLソースを編集(作成)します。練習用に以下のようなプログラムをコーディングしてみてください。

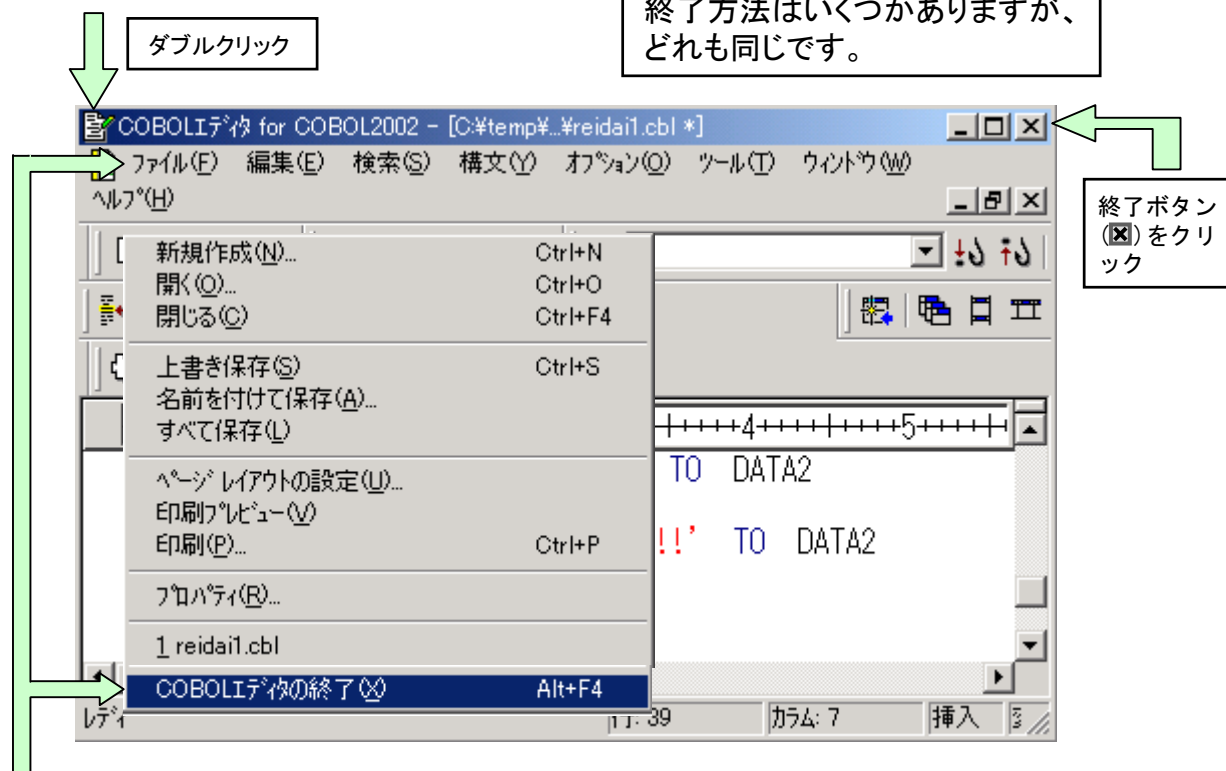


## [コーディング例]



COBOL専用エディタは、予約語・定数等の色分け表示、キーワード補完、構文テンプレート、構文チェック等のCOBOLの文法に対応した各種機能を用意しています。

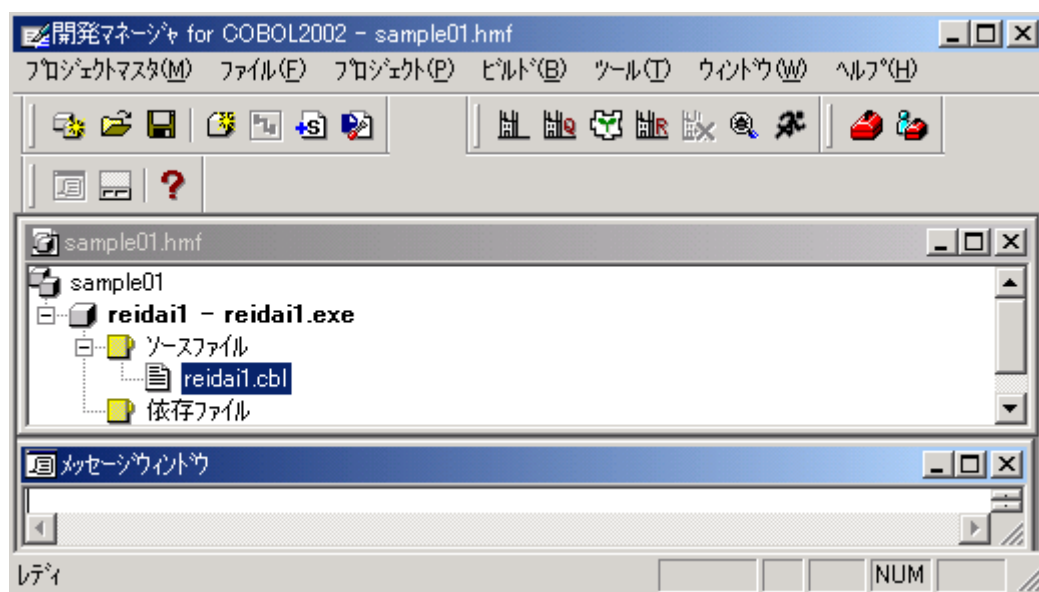
[手順3] コーディングが終了したら、終了ボタンを押してエディタを終了してください。すると、保存するかどうかの応答が返ってきますので、「はい」を選択して保存してください。




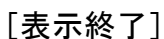
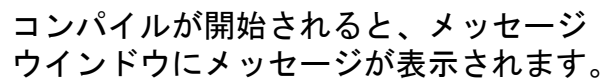
「ファイル(F)」をクリックして、プルダウンメニューから「COBOL2002の終了」を選択



開発マネージャの画面に戻ります。

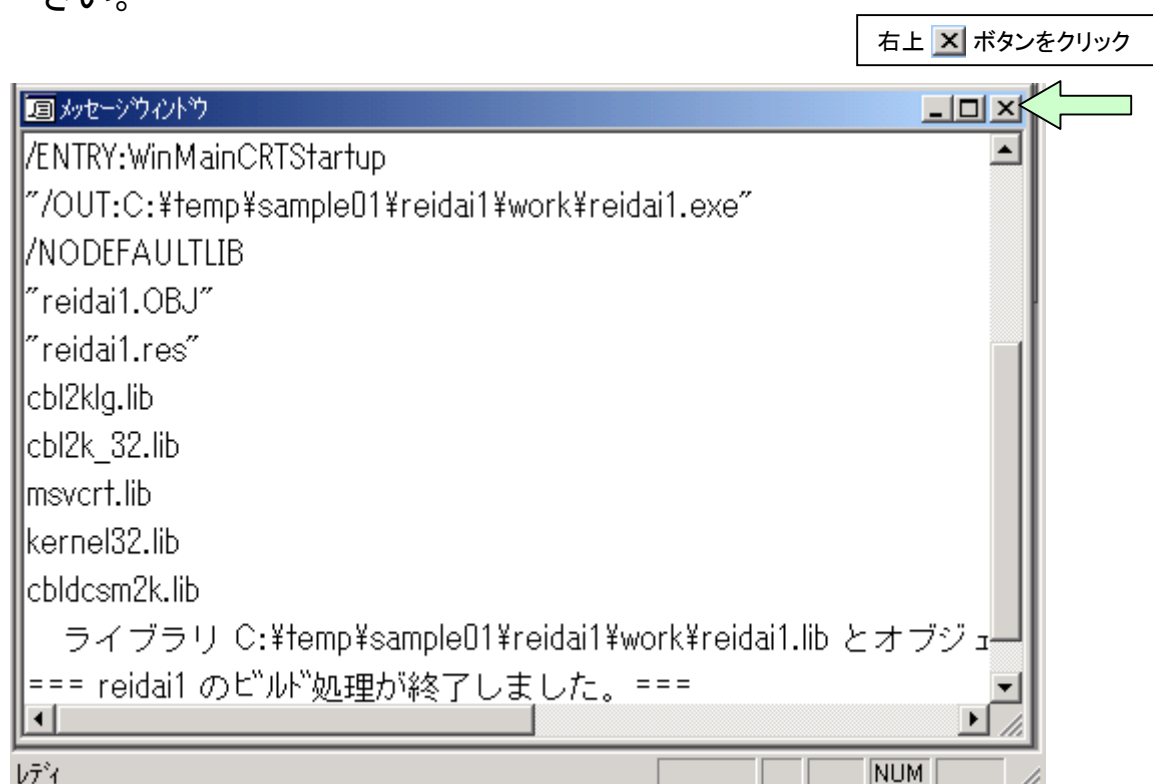


[手順 1] 開発マネージャの上の方にあるビルドボタン(  )をクリックします。

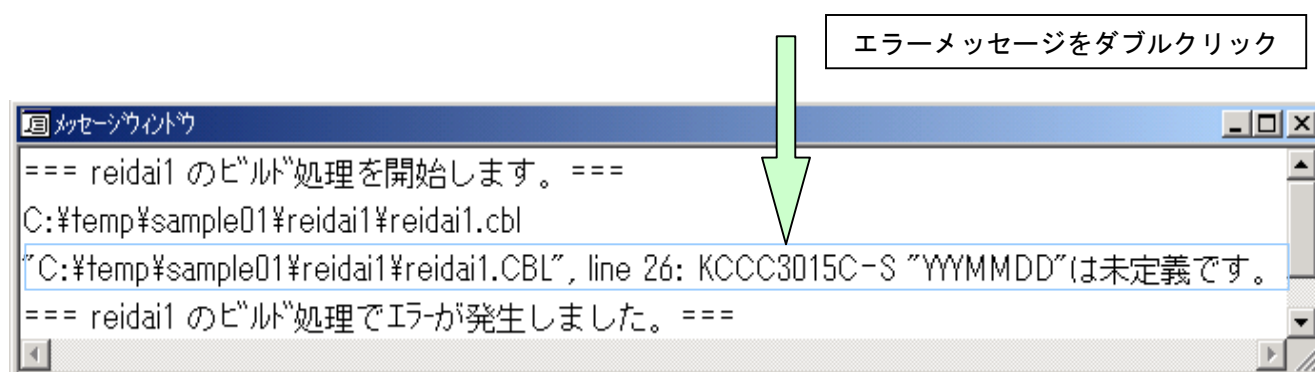


コンパイルとリンケージを一括して行うことを、「ビルド」といいます。

- [手順2] ビルドが終了したら、メッセージウインドウを閉じてください。これで、コンパイルは終わりです。メッセージウインドウに「KCCCXXXX」のエラーメッセージが出力されたときは、手順3以降を参照してください。

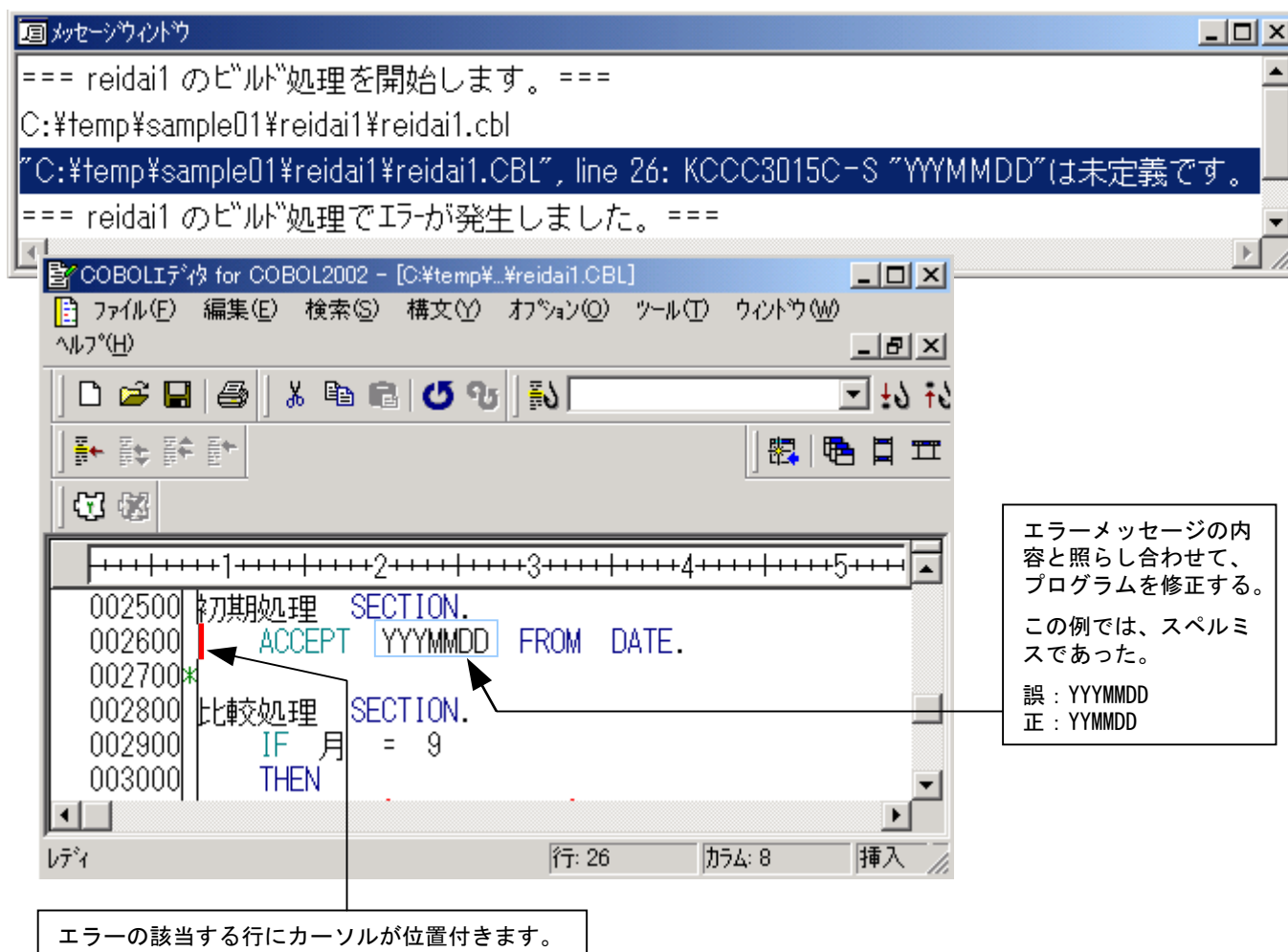


- [手順3] エラーがなくビルドが終了していたら、次の章の「実行」に進んでください。ここからは、コンパイルエラーが出たときのソース修正の方法を説明します。  
メッセージウインドウ中に表示されているエラーメッセージをダブルクリックしてください。すると、エディタが自動的に起動されます。



エラーメッセージが見にくい場合は、スクロールバーを使ってスクロールするか、メッセージウインドウの大きさを調整することで、見やすいようにしてください。

[手順4] エディタが自動起動し、エラーに該当する行の先頭にカーソルが位置付きます。先のエラーメッセージの内容と照らし合わせて、エラーを修正してください。




[手順5] エラーの修正が終わったら、エディタとメッセージウインドウを閉じて手順1に戻り、コンパイルからやり直してください。  
なお、エラーが複数ある場合は、メッセージウインドウ上のエラーメッセージを次々にダブルクリックすれば、エディタの該当位置に位置付きます。また、一つのエラーのために複数のエラーが派生することや、一つのエラーに隠れて他のエラーが検知できない場合もありますので、ご注意ください。

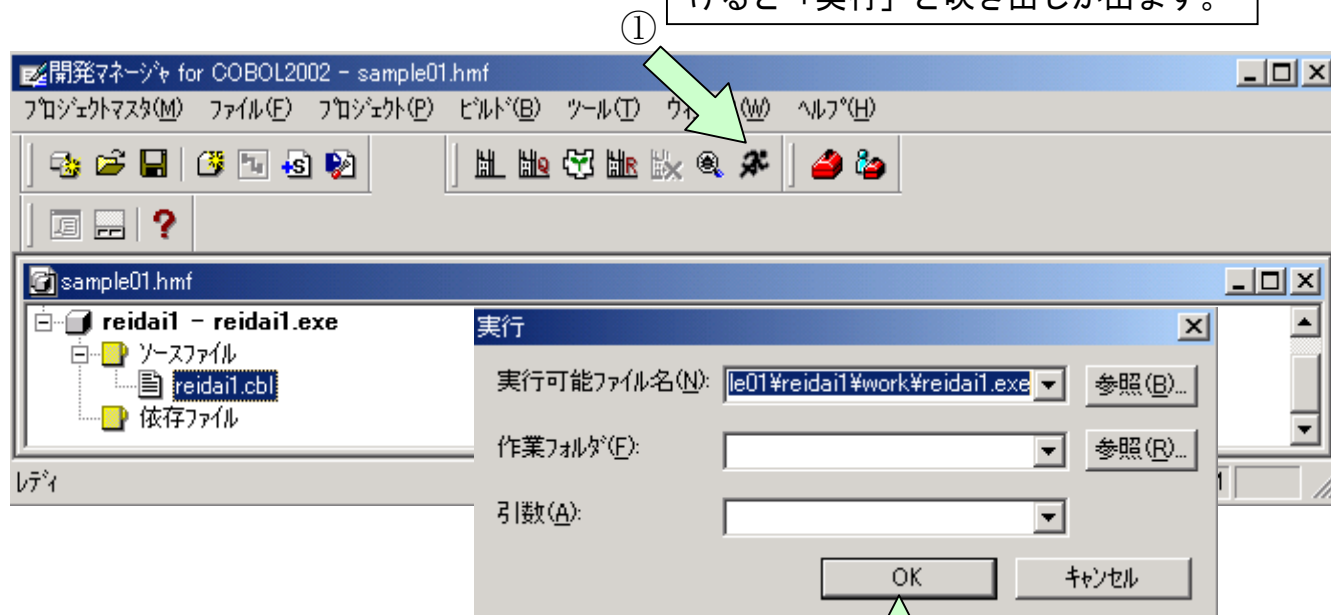


## 7. 実行

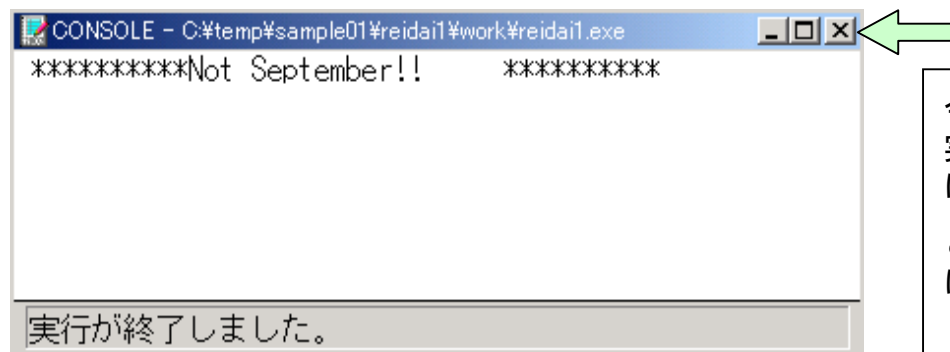
次は実行です。今回の例題では特に入出力ファイルを用いていないので、簡単に実行できます。入出力ファイルがある場合は、ファイルの割り当てを行う必要があります。ファイルの割り当てについての説明は、「後述の3. 関連資料」の「(a) ファイルの入出力処理」を参照してください。

[手順 1] 開発マネージャの上の方にある実行ボタン (  ) をクリックします。このボタンを「実行ボタン」と呼びます。実行ボタンをクリックすると、実行確認画面が出ます。この画面の「OK」ボタンをクリックしてください。


このボタンにマウスポインタを位置づけると「実行」と吹き出しが出ます。



[CONSOLE画面]



今回の例題のDISPLAY文の実行結果は、CONSOLE画面に表示されます。

この画面を終了させるには、 ボタンをクリックします。

### [ワンポイントアドバイス]

CONSOLE画面を閉じないうちは、実行ファイルは起動されたままの状態になっています。このままにしておくと、再度コンパイルしたときエラーになります。実行結果を確認したら、必ずCONSOLE画面を閉じるようにしましょう。



## 8. プロジェクトの追加

プログラムの作成から実行までの操作を一通り説明しました。ここでは、新たにプロジェクトを登録する方法と注意事項について説明します。


最初にも述べましたが、一つのプロジェクトマスタファイルの中に複数のプロジェクトを登録できます。もちろん、プロジェクト毎にプロジェクトマスタファイルを作成してもかまいません。この場合は、これまで説明した手順にしたがって作業してください。

プロジェクトを登録するときの二つの形態を以下の図に示します。

### <二つの形態>

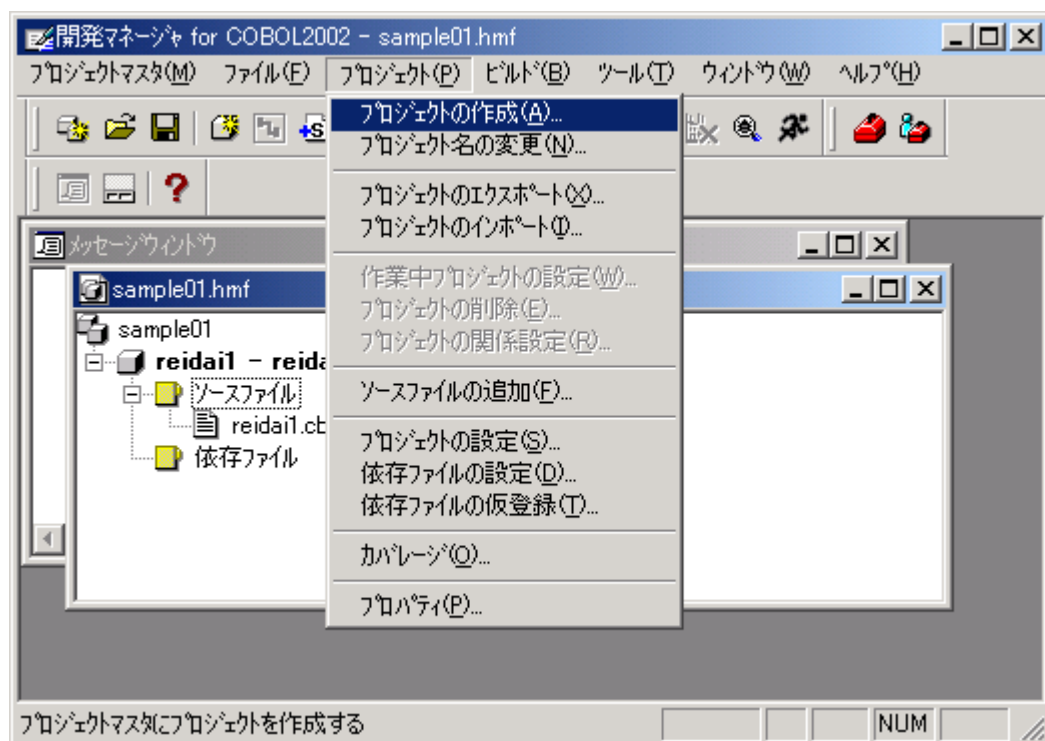
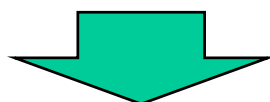
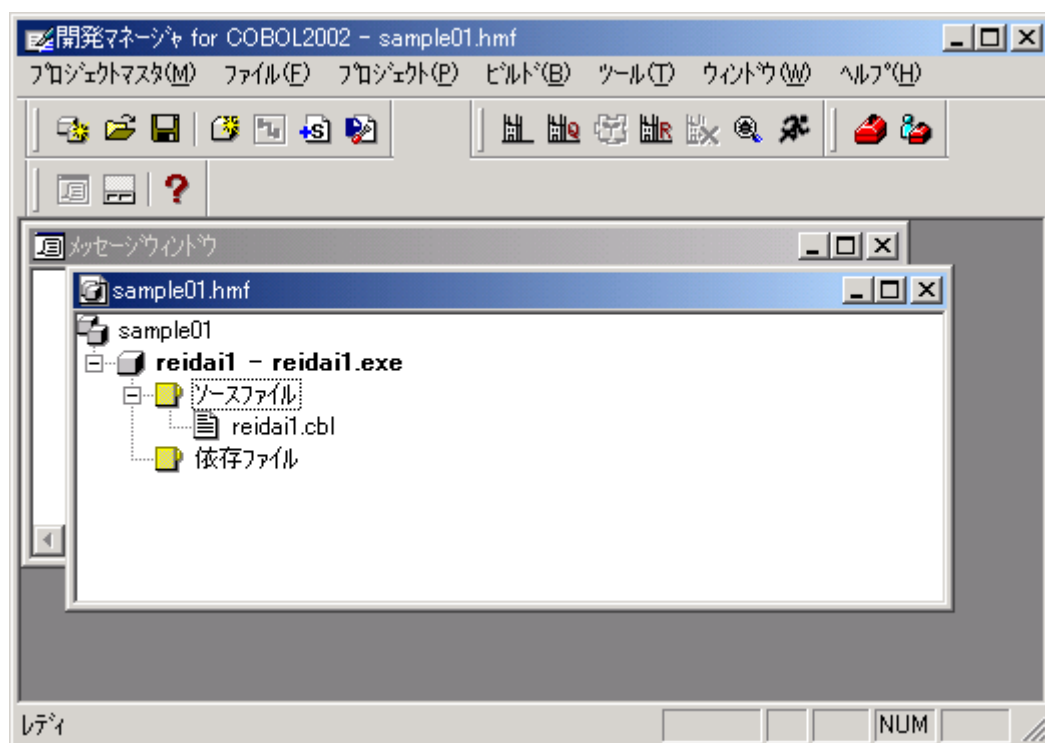
① プロジェクトマスタファイル 1 — プロジェクト 1

プロジェクトマスタファイル 2 — プロジェクト 2

② プロジェクトマスタファイル 1 —  プロジェクト 1  
プロジェクト 2  
プロジェクト 3  
プロジェクト n

それでは、既にプロジェクトを作成したプロジェクトマスタファイルに新たなプロジェクトを追加する手順を示します。

[手順 1] 新たなプロジェクト「reidai2」を追加します。開発マネージャの画面から「プロジェクト(P)」-「プロジェクトの作成」の順にクリックします。



- [手順2] 続いて、「プロジェクトの作成」画面が表示されます。  
 これは、「4. プロジェクトの作成」の手順1の画面と同じです。  
 もうおわかりですね。ここからは、これまで説明したプロジェクトの  
 作成手順にしたがってください。おさらいの意味で画面の遷移を以下  
 に示します。



プロジェクト名を入力する。



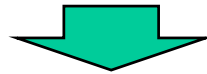
「Enter」キーを押す。



「Enter」キーを押す。



「Enter」キーを押す。



プロジェクトの追加 : COBOLプロジェクト

最終生成物の種類を選択してください。

☒ 実行形式プログラム(E)  
☐ ダイナミックリンクライブラリ(D)  
☐ 標準ライブラリ(L)



「Enter」キーを押す。

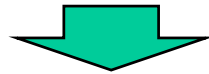
プロジェクトの追加 : COBOLプロジェクト

プロジェクトの種類を選択してください。

System指定メインプログラム  
 V3指定メインプログラム  
 メインプログラムなし  
 OLEアウトオブプロセスサーバプログラム  
 CGIメインプログラム  
 GUIプログラム



「Enter」キーを押す。



プロジェクトの追加：COBOLプロジェクト

出力ファイル名(EXE)を入れてください。(O)

reidai2 参照(R)...

「reidai2」と入力し、「Enter」キーを押す。

< 戻る(B) 次へ(N) > キャンセル



プロジェクトの追加：COBOLプロジェクト

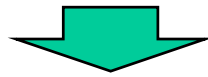
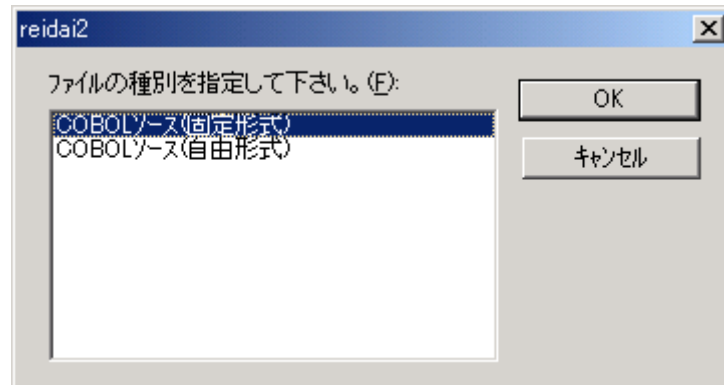
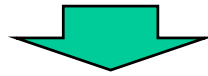
メインファイル名を入れてください。(M)

reidai2 参照(R)...

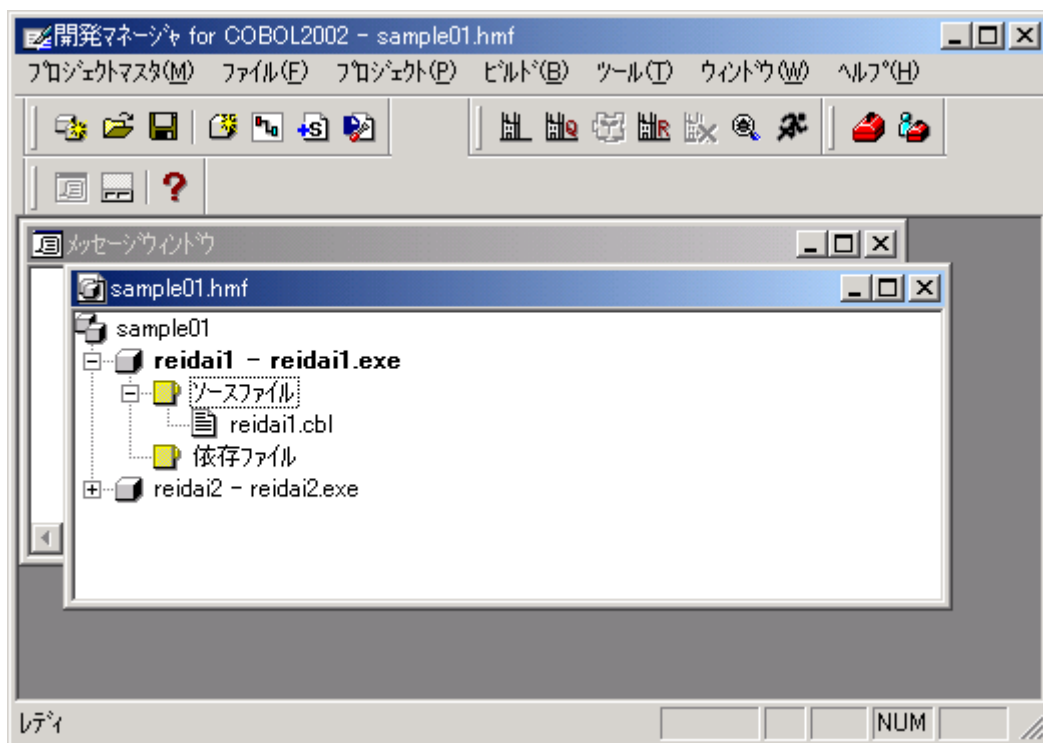
「reidai2」と入力し、「Enter」キーを押す。

< 戻る(B) 完了 キャンセル





「Enter」キーを押す。



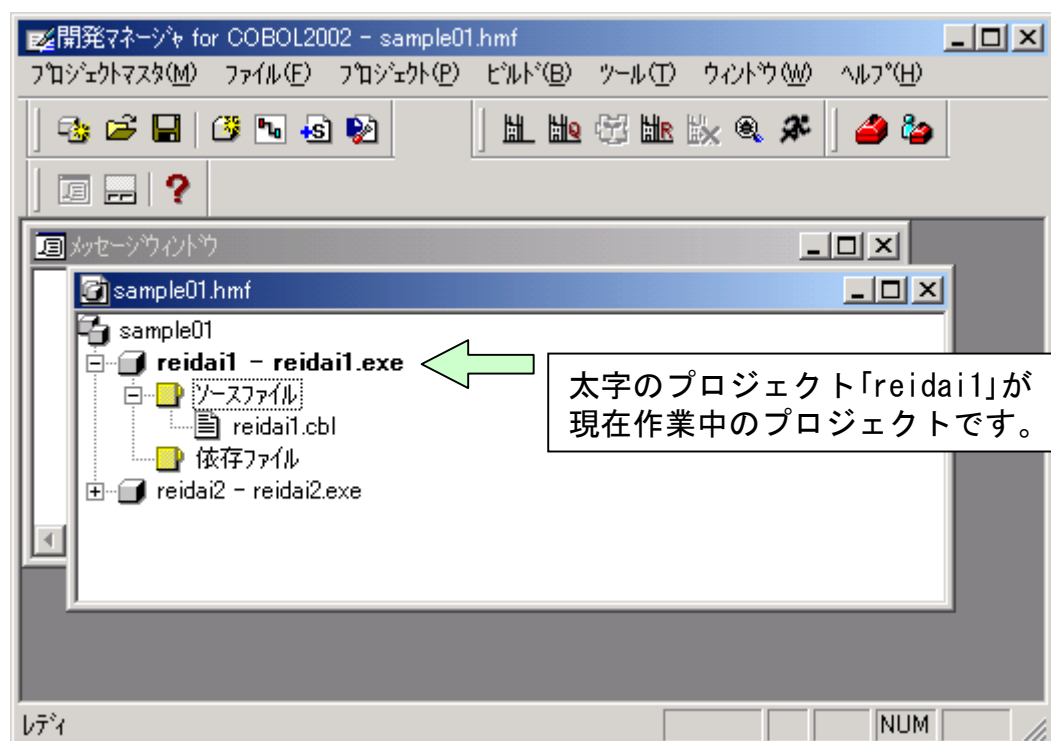
プロジェクトの追加が完了しました。

続いて、複数プロジェクト環境で作業をするときに必要な「作業対象プロジェクトの選択」と「プロジェクトの削除」について説明します。

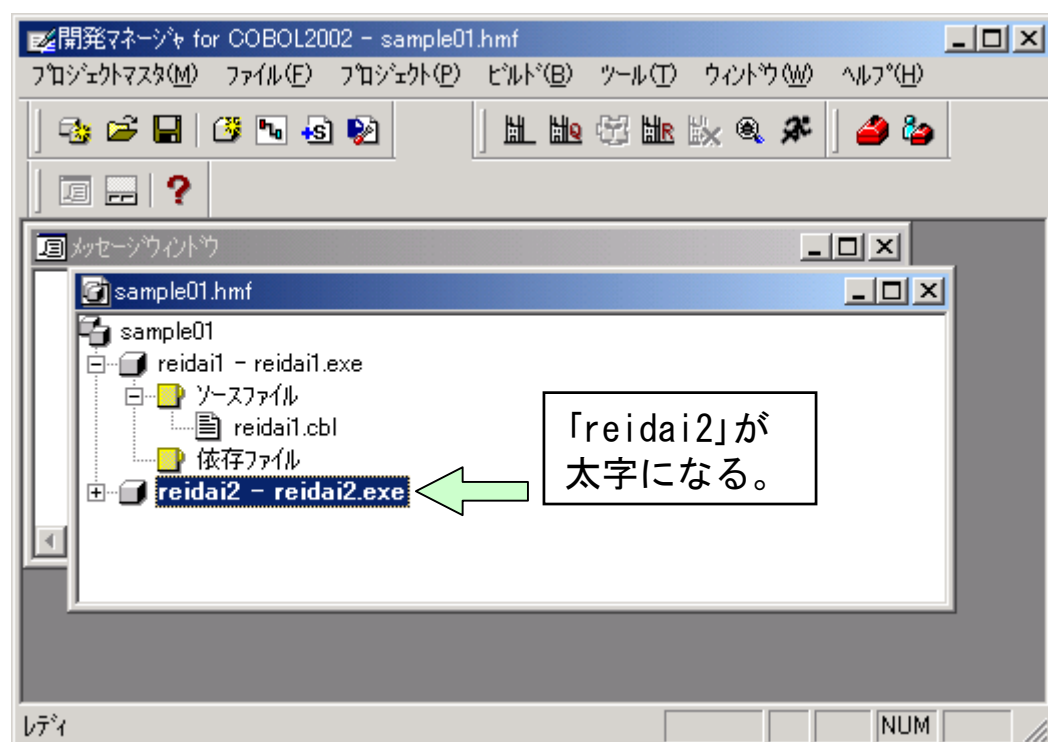
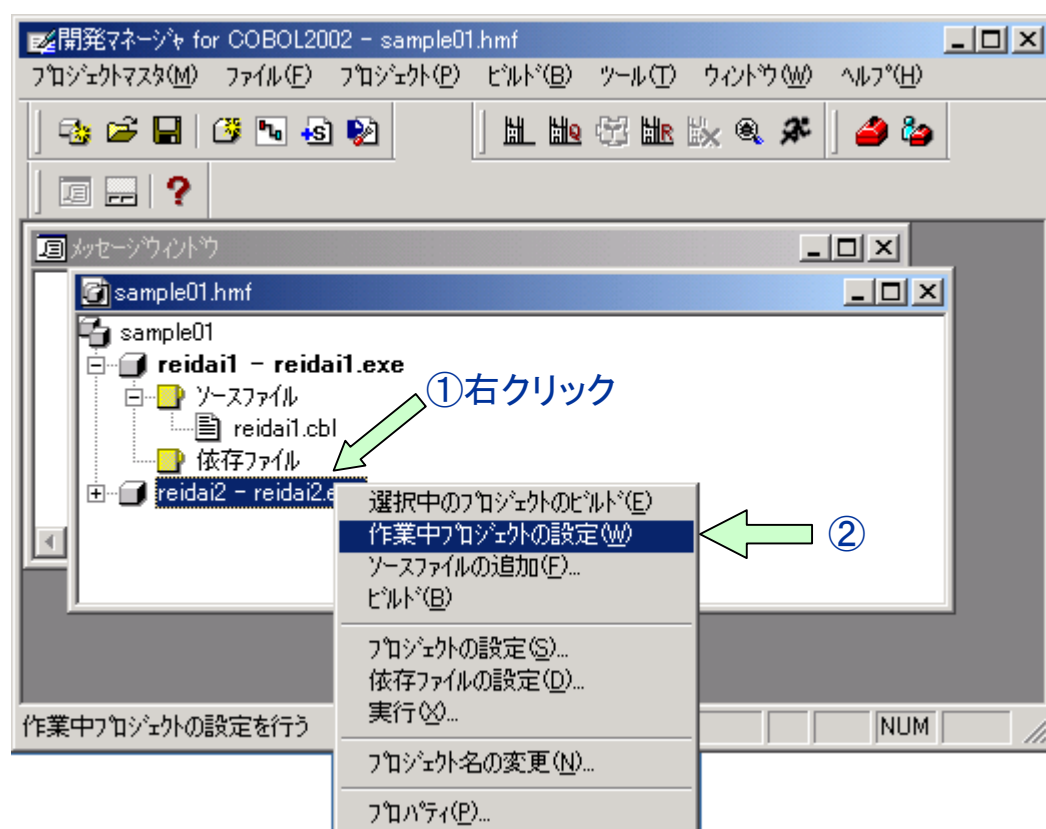
### 作業対象プロジェクトの選択

プロジェクトが複数ある場合、作業を開始するときに、どのプロジェクトの作業をするかを明示的に指定する必要があります。

開発マネージャの画面で、一つだけ太字で表示されているプロジェクトがあります。これが、現在作業中のプロジェクトです。



プロジェクト「reidai2」の作業をしたい場合、「reidai2」をクリックします。次に右クリックして表示されるプルダウンメニューの中から「作業中プロジェクトの設定(W)」を選択します。



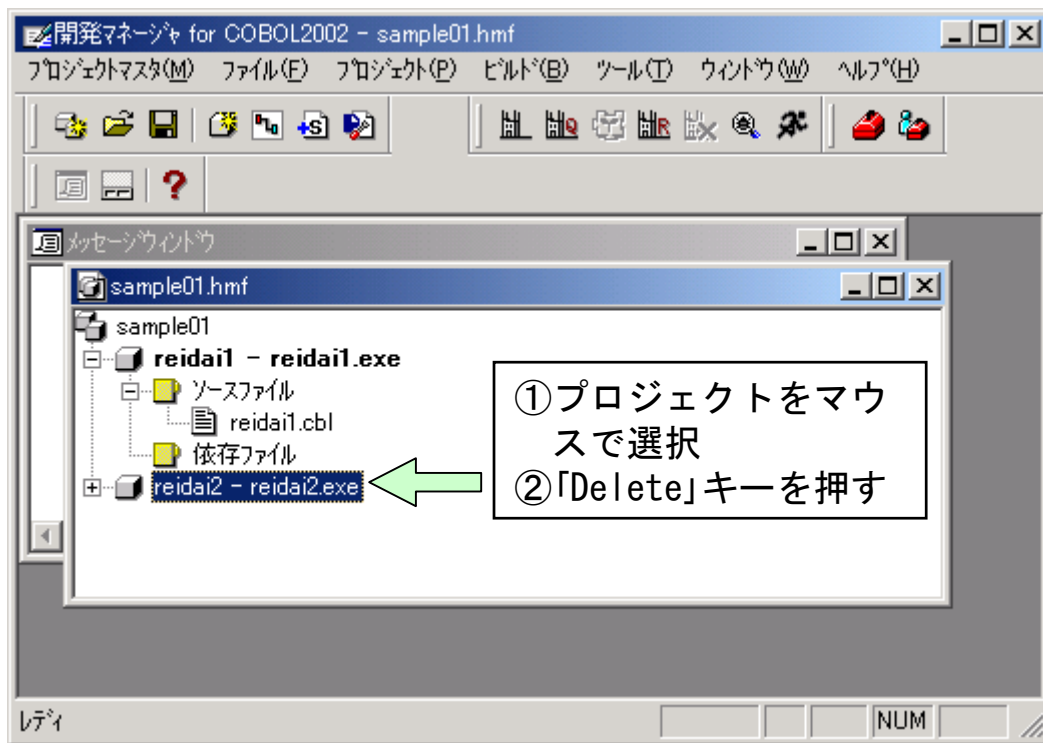
Reidai2に対して、ソース編集やビルド、実行をしてみてください。  
 なお、ソースの編集は、該当ソースファイルをダブルクリックして編集するため、プロジェクトを選択しなくても編集作業が可能です。しかし、ビルド等をするときには、作業中のプロジェクトを設定しておかなければなりません。



## プロジェクトの削除

プロジェクトを削除したい場合は、次のように操作します。

削除したいプロジェクトをマウスで選択(クリック)し、「Delete」キーを押します。



## 9. 終わりに

以上で、プロジェクトの作成、及びプログラムの作成から、コンパイル、エラー修正、実行までの一連の操作の説明は終わりです。

いろいろなプログラムを試してみてください。

テストデバッグ機能をお使いになりたい場合は後述の「2. テストデバッガの使用方法」をご覧ください。また、順ファイル等のファイルを使用するプログラムの実行については、後述の「3. 関連資料」の「(a) ファイルの入出力処理」を、エディタの細かい設定方法については後述の「3. 関連資料」の「(e) エディタ設定方法」をご覧ください。

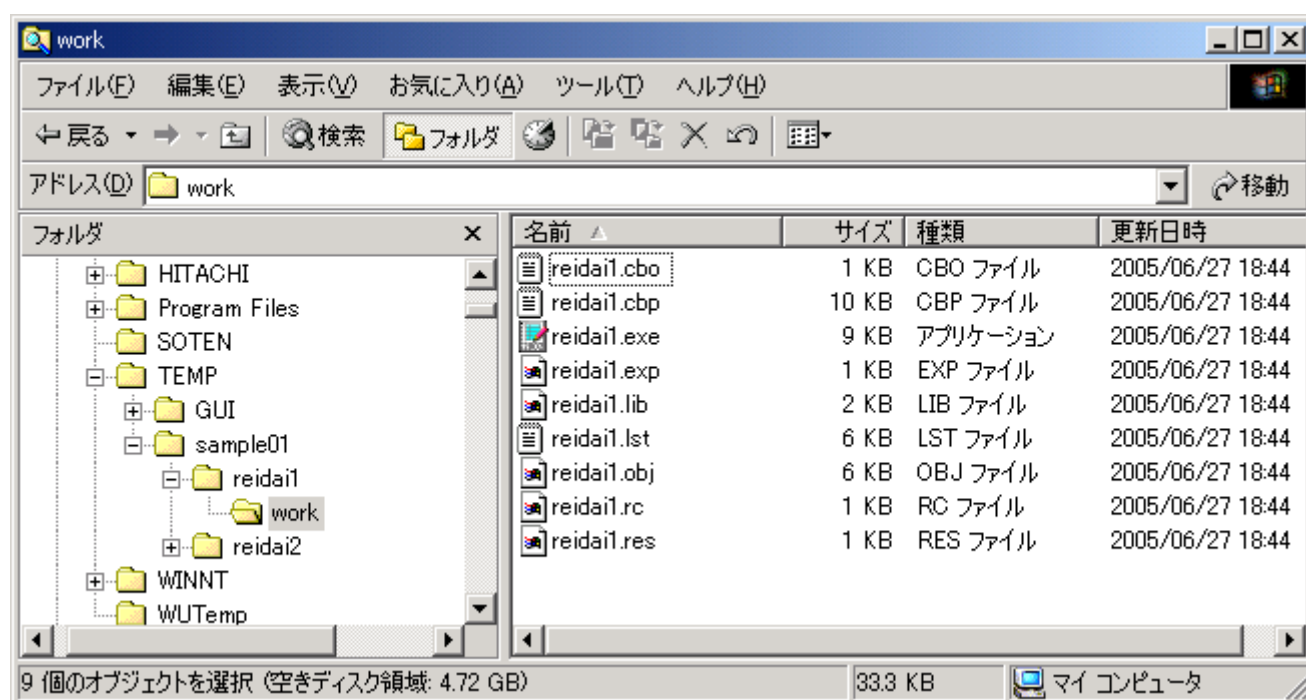
最後に、reidai1のプログラムのコンパイルによって生成されたファイルを示しますので、エクスプローラで参照してみてください。コンパイラの生成物は、「reidai1」フォルダの下の「work」フォルダに格納されています。

**Reidai1.exe** : 実行可能ファイルです。「7. 実行」で実行したのはこのファイルです。

**Reidai1.lst** : コンパイルリストです。

**Reidai1.cbp** : デバッグ情報を格納したファイルです。

**その他** : 各種管理情報を持つファイルです。



## 2. テストデバッグの使用方法

－COBOL2002専用テストデバッグツールの使い方－

# 一 目 次

1. はじめに
2. コンパイル時のオプション
3. テストデバッガの起動と終了
4. ソースの表示方法と中断点の設定
5. データの内容表示と代入
6. データのトレース
7. エディタとの連携
8. デバッガの色などの変更
9. カバレッジ情報の蓄積と表示
10. カウント情報の表示
11. 終わりに

# 1. はじめに

この章では、COBOL2002専用のテストデバッグツールの使用方法について説明します。

既に「1. COBOL2002の起動から実行まで」を読み終わっているものとして、説明を行います。

例題プログラムは1章で作成したプログラム(reidai1)を使用します。

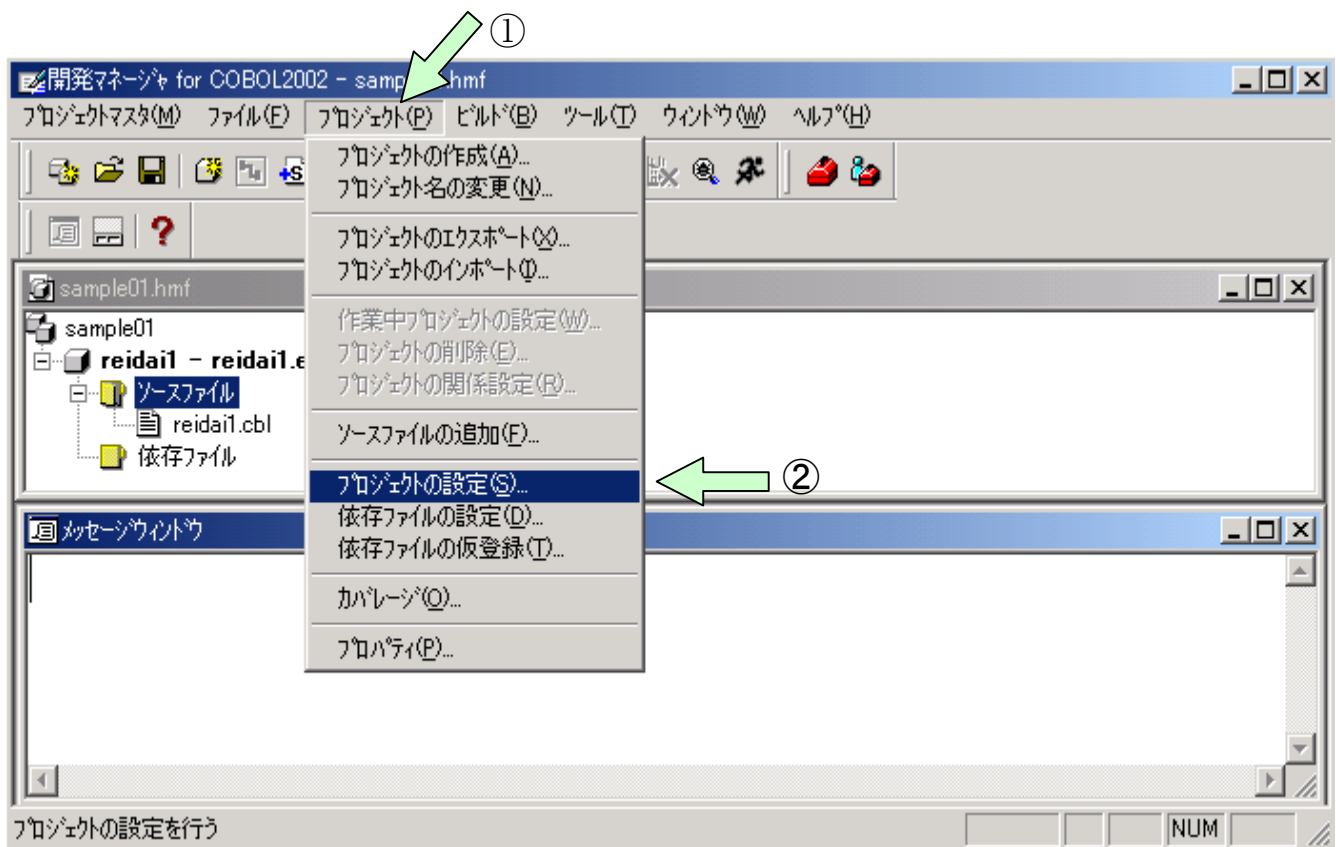
ここでは、COBOL2002専用のテストデバッグツールの機能のうち、特に知っておいて頂きたい基本的な機能について説明します。COBOL教育では、この基本的な操作を理解すれば十分と考えますが、更に詳細を知りたいという場合には、マニュアル「COBOL2002操作ガイド」、  
「COBOL2002ユーザズガイド」を参照ください。

## 2. コンパイル時のオプション

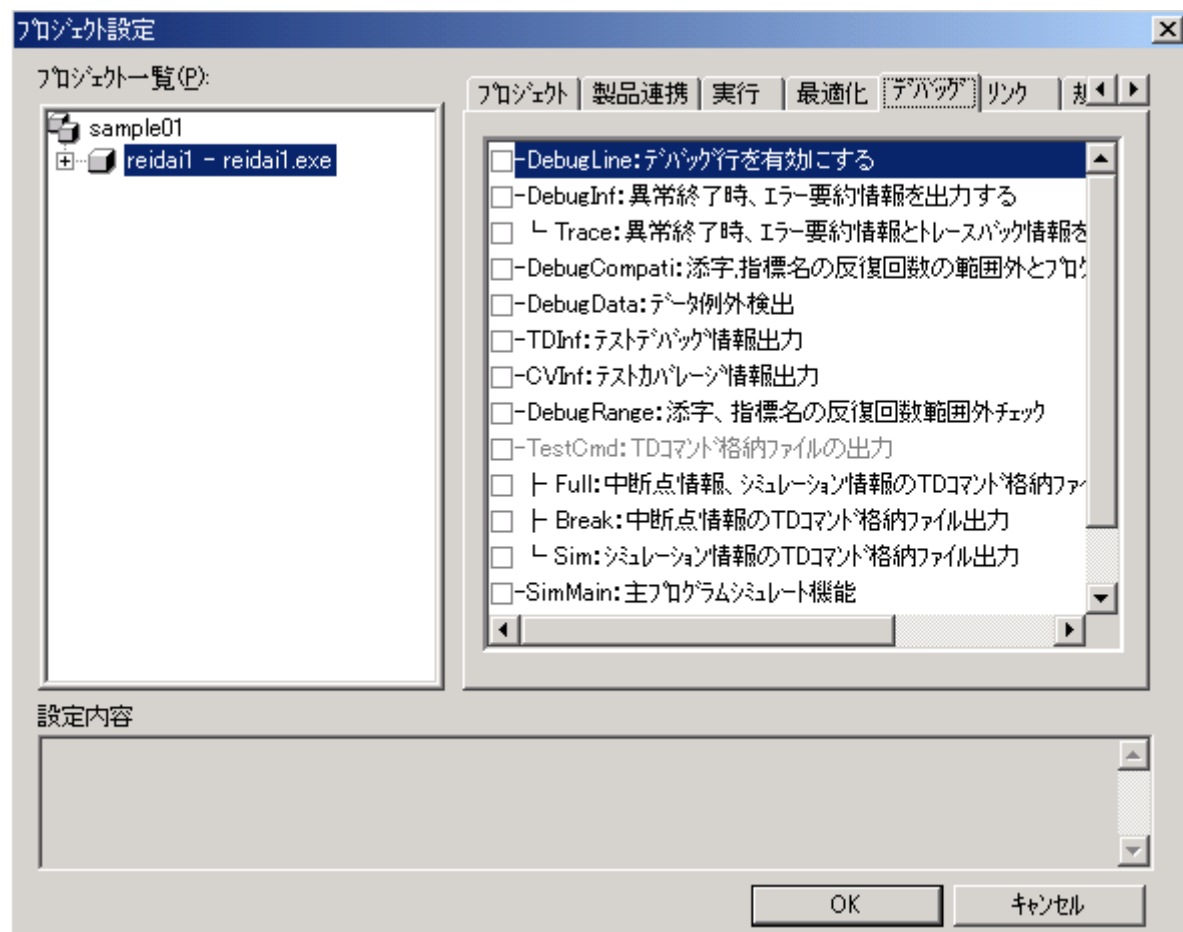
テストデバッグツールを使用するためには、COBOLプログラムのコンパイル時に「コンパイラオプション」を指定する必要があります。

1章で説明したように、デフォルトオプションの設定で「-TDInf」を指定してあれば、ここでオプションを指定する必要はありません。ここでは、デフォルトオプションを設定していないものとして手順を示します。

〔手順 1〕 開発マネージャのメニューバーの「プロジェクト(P)」をクリックし、プルダウンメニューの「プロジェクトの設定(S)」をクリックします。  
すると、コンパイラオプションの一覧が出力されます。



オプションの一覧が出力されます。  
-TDInfをチェックして「OK」ボタンをクリックします。

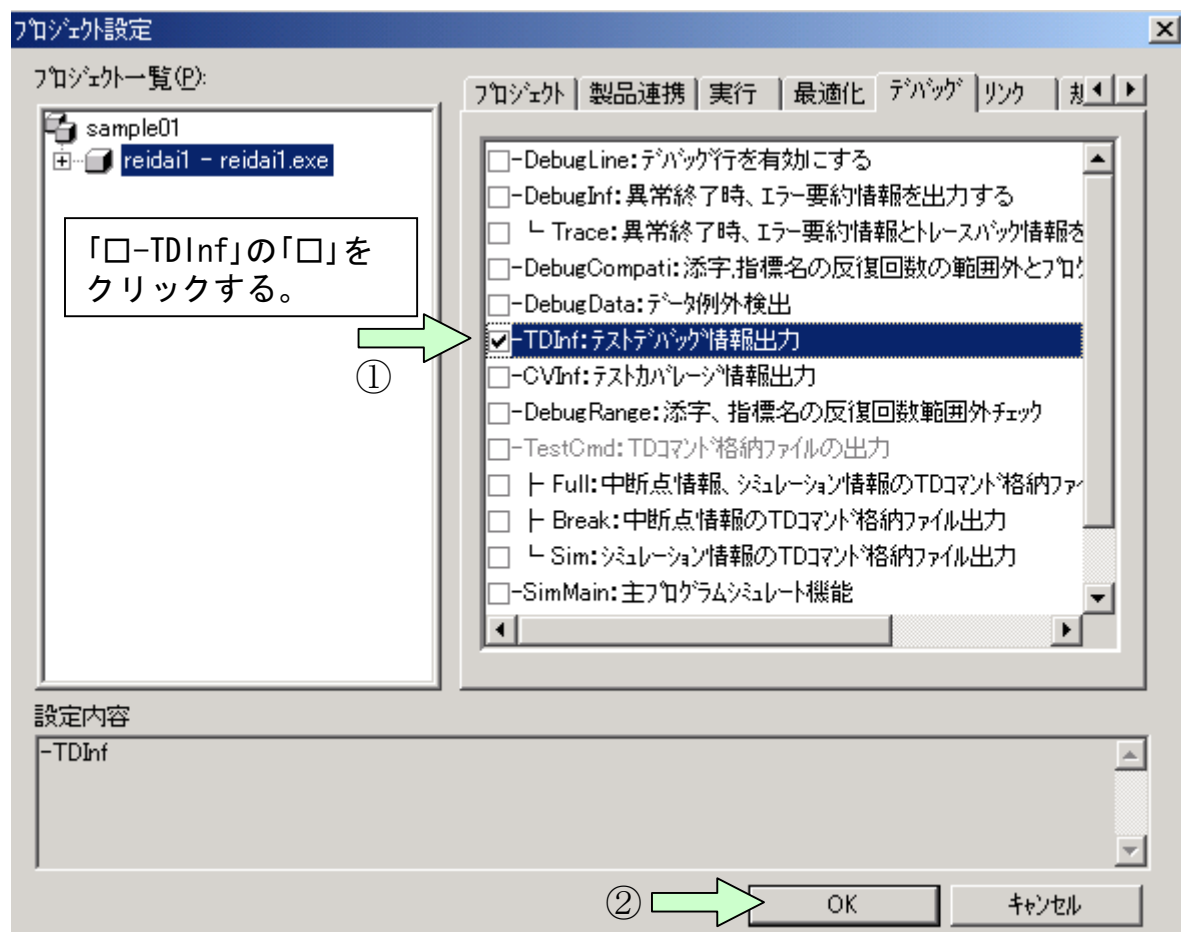


### [用語解説] コンパイラオプション

コンパイラオプションは、コンパイラがオプションでサポートしている機能を使うときに指定します。背反する仕様を使い分けるために用意されているオプション等もあります。

デバッグが完了すると、そのプログラムにはデバッグ情報は不要になります。デバッグ情報の出力がオプション機能になっているのは、完成したプログラムに余分な情報を持たなくてもよいよう配慮している意味もあります。

- [手順2] コンパイラオプションの一覧の中から「デバッグ」タブの「-TDInf」の「□」をクリックして「レ」印をつけます。  
「レ」印が設定されたことを確認して、「OK」ボタンをクリックします。

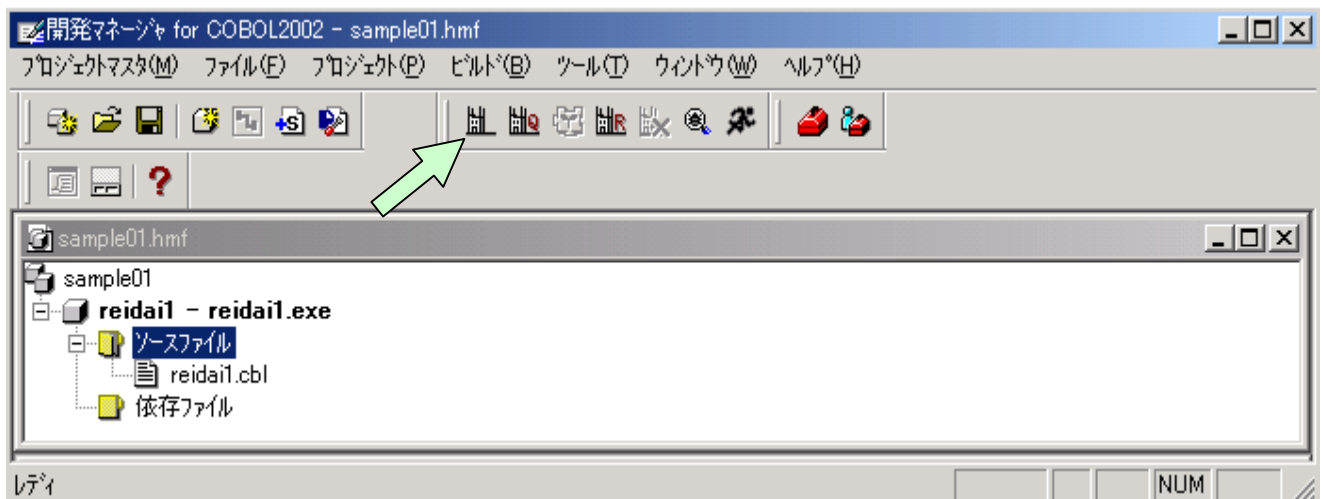


#### [ワンポイントアドバイス]

オプション設定の画面を見てわかるように、多数のコンパイラオプションがあります。オプションの意味はマニュアルを参照し、必要のないオプションは指定しないようにしましょう。



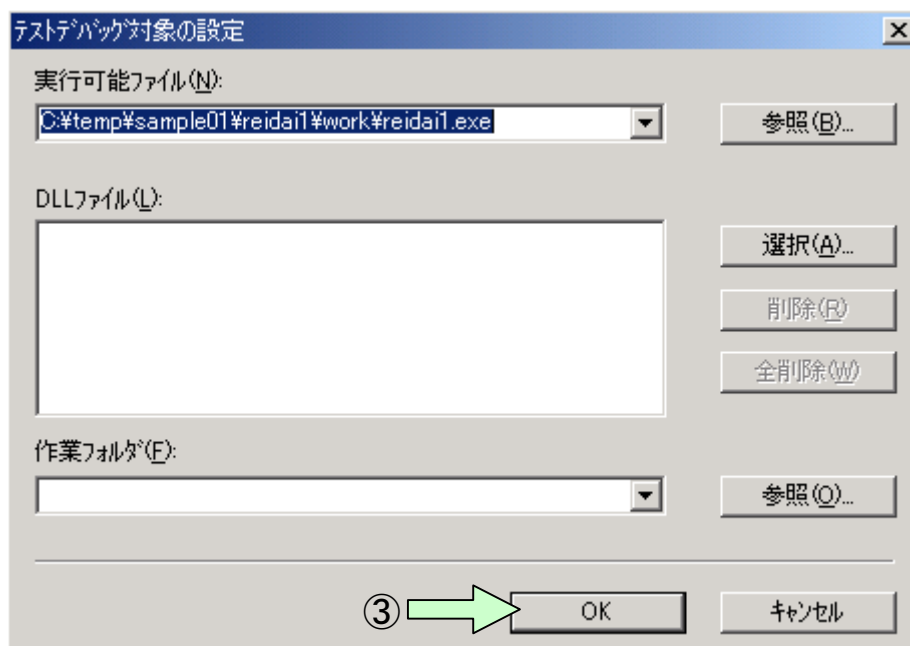
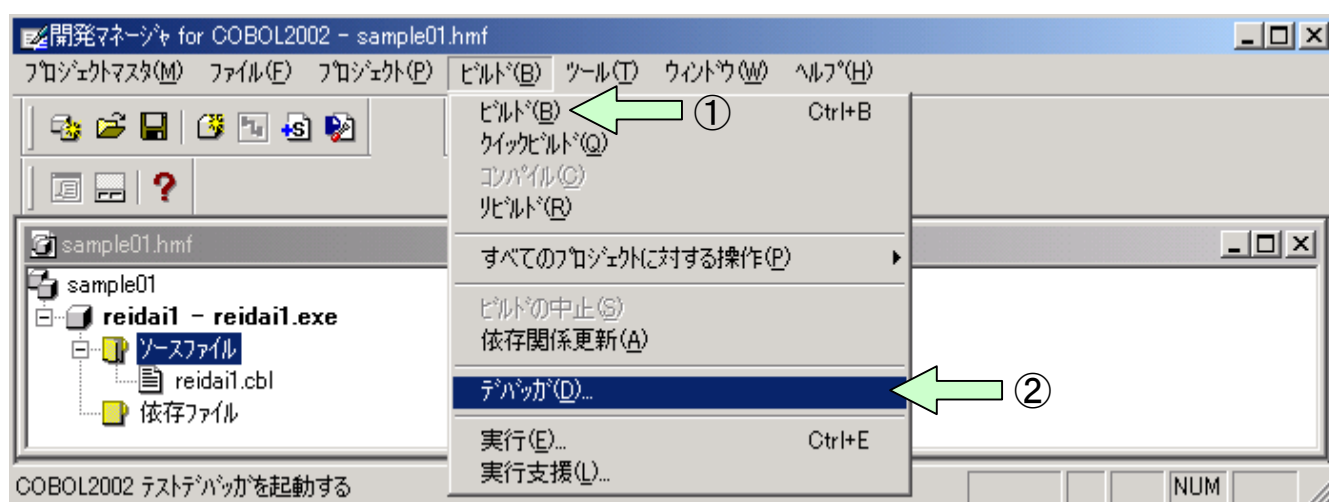
[手順3] 開発マネージャ画面に戻りますので、改めてコンパイルを行ってください。



### 3. テストデバッガの起動と終了

-TDInfオプションを指定して、コンパイル&リンケージが終わると、テストデバッグツールを使用することができます。（「実行」ボタンをクリックすれば、実行だけすることもできます。）

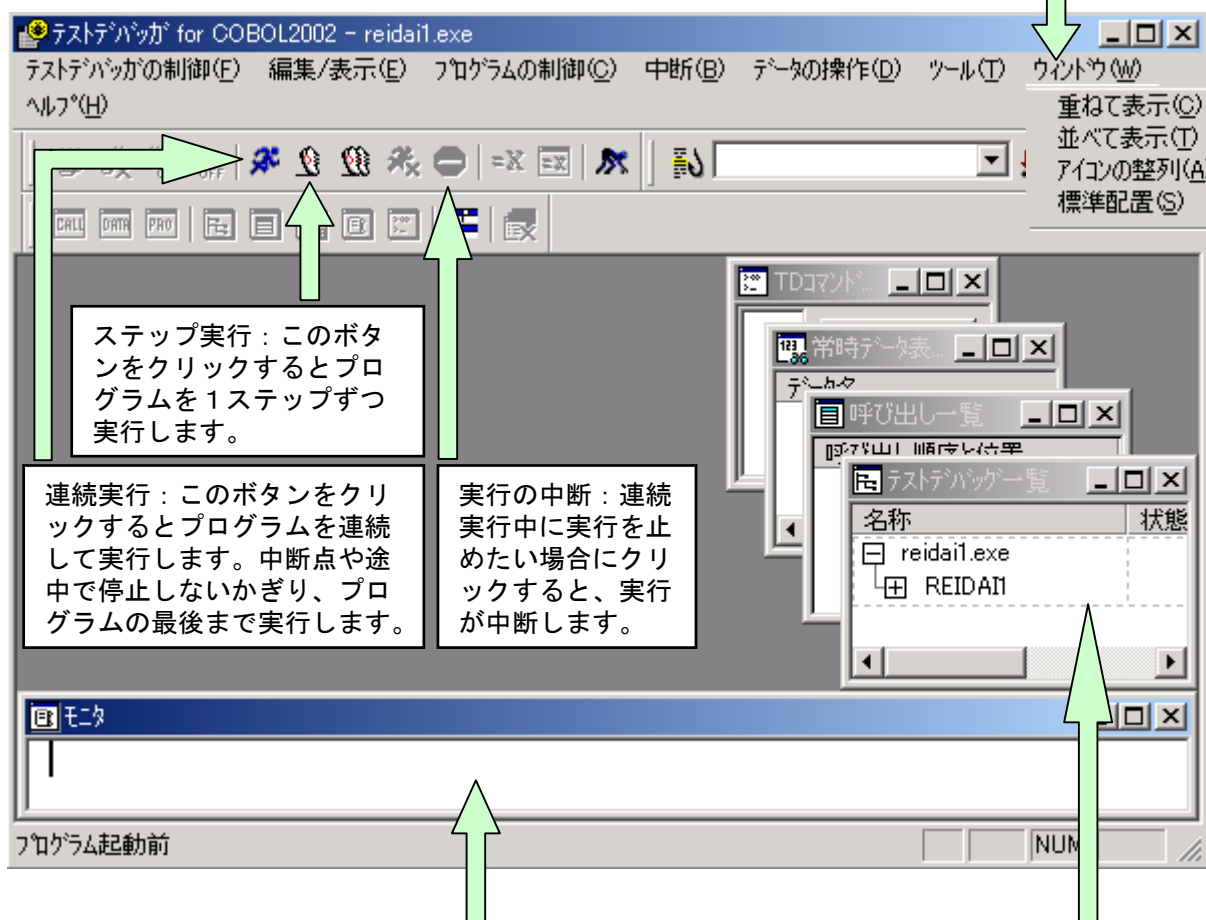
[手順 1] 開発マネージャのメニューバーの「ビルド (B)」をクリックし、プルダウンメニューの中の「デバグ (D)」をクリックします。すると、デバッグするプログラムの確認画面が出ますので、「OK」ボタンをクリックします。



〔手順2〕 以下のような画面が出力されます。主な各部の機能は以下の説明の通りです。基本的には速度調整して実行すれば、デバッグを行っていきます。次の章以降は、デバッグの基本的な操作方を説明します。

画面はTDコマンド画面、常時データ表示画面、呼び出し一覧画面、テストデバッガー一覧画面、モニタ画面とマルチ表示になっています。

下記画面は「標準配置」ですが、ウインドウタブから「重ねて表示」、「並べて表示」が選択できます。



ステップ実行：このボタンをクリックするとプログラムを1ステップずつ実行します。

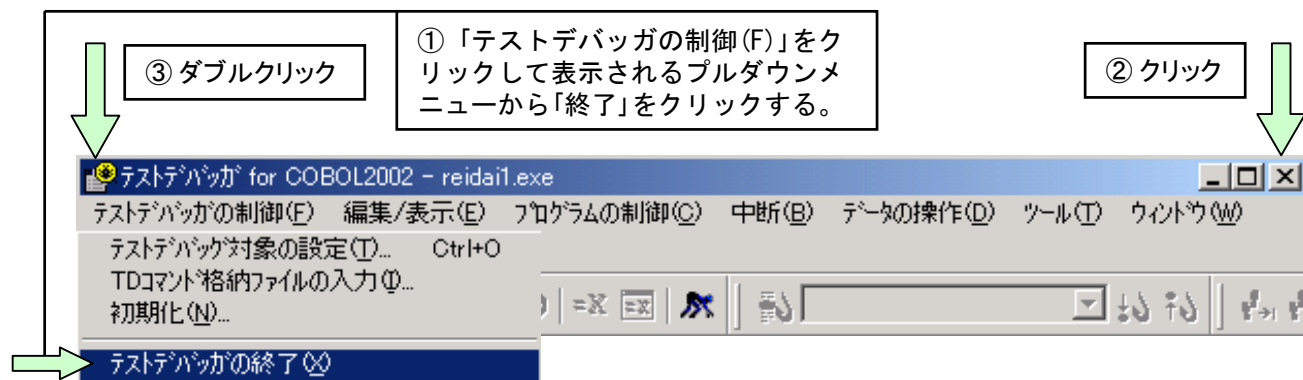
連続実行：このボタンをクリックするとプログラムを連続して実行します。中断点や途中で停止しないかぎり、プログラムの最後まで実行します。

実行の中断：連続実行中に実行を止めたい場合にクリックすると、実行が中断します。

モニタ画面：実行の状態や、各種情報を表示します。

一覧画面：デバッグするプログラムの一覧が表示されます。

〔手順3〕 デバッグの方法を説明する前に終了方法を説明しておきます。終了方法は、Windowsの基本操作に沿った3通りの方法があります。

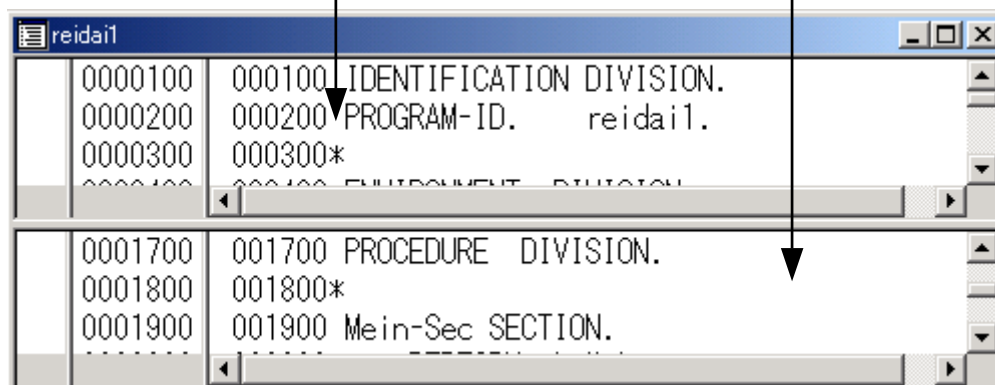
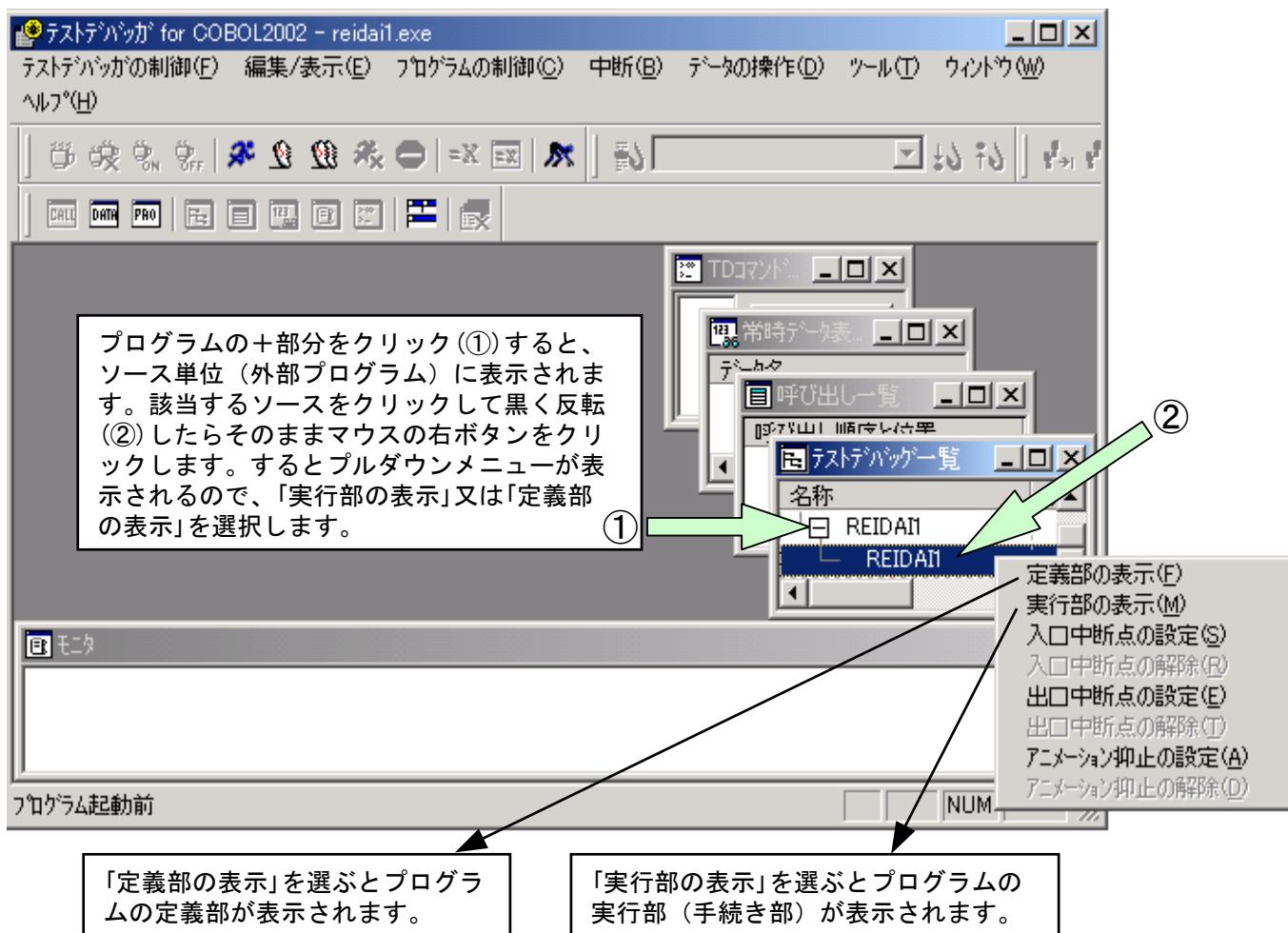


## 4. ソースの表示方法と中断点の設定

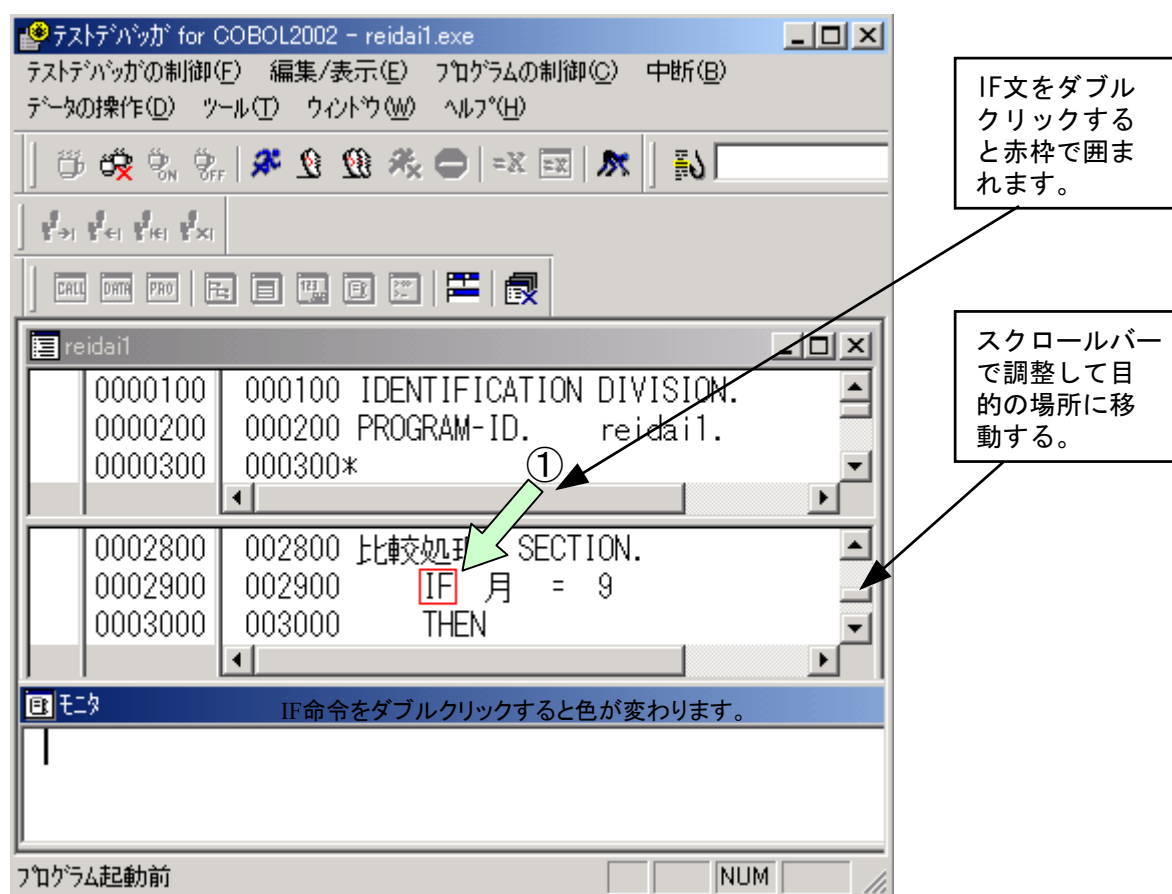
さて、いよいよプログラムのテストデバッグの開始ですが、まず、プログラムの実行部と定義部の表示方法から説明します。

実行部に関しては先に表示しておかなくても、プログラムを実行すれば自動的に表示されます。ただし、事前に中断点等を設定したい場合には、予め表示しておく必要があります。

[手順 1] テストデバッグ画面の一覧ウィンドウの中のプログラム名をクリックし、そのままマウスの右ボタンをクリックすると「実行部」と「定義部」のプルダウンメニューが表示されます。これらのどちらかをクリックすると選択したものがそれぞれの画面に表示されます。



[手順2] 中断点を設定します。実行画面のスクロールバーを調整して中断したい文に位置付けます。この例では2900行目のIF文の所まで移動しています。ここで、文字列の「IF」をダブルクリックすると、赤色の枠で囲まれます。これで、中断点を設定できました。この文に制御が渡ってきた時点で（この文を実行する直前で）止まります。

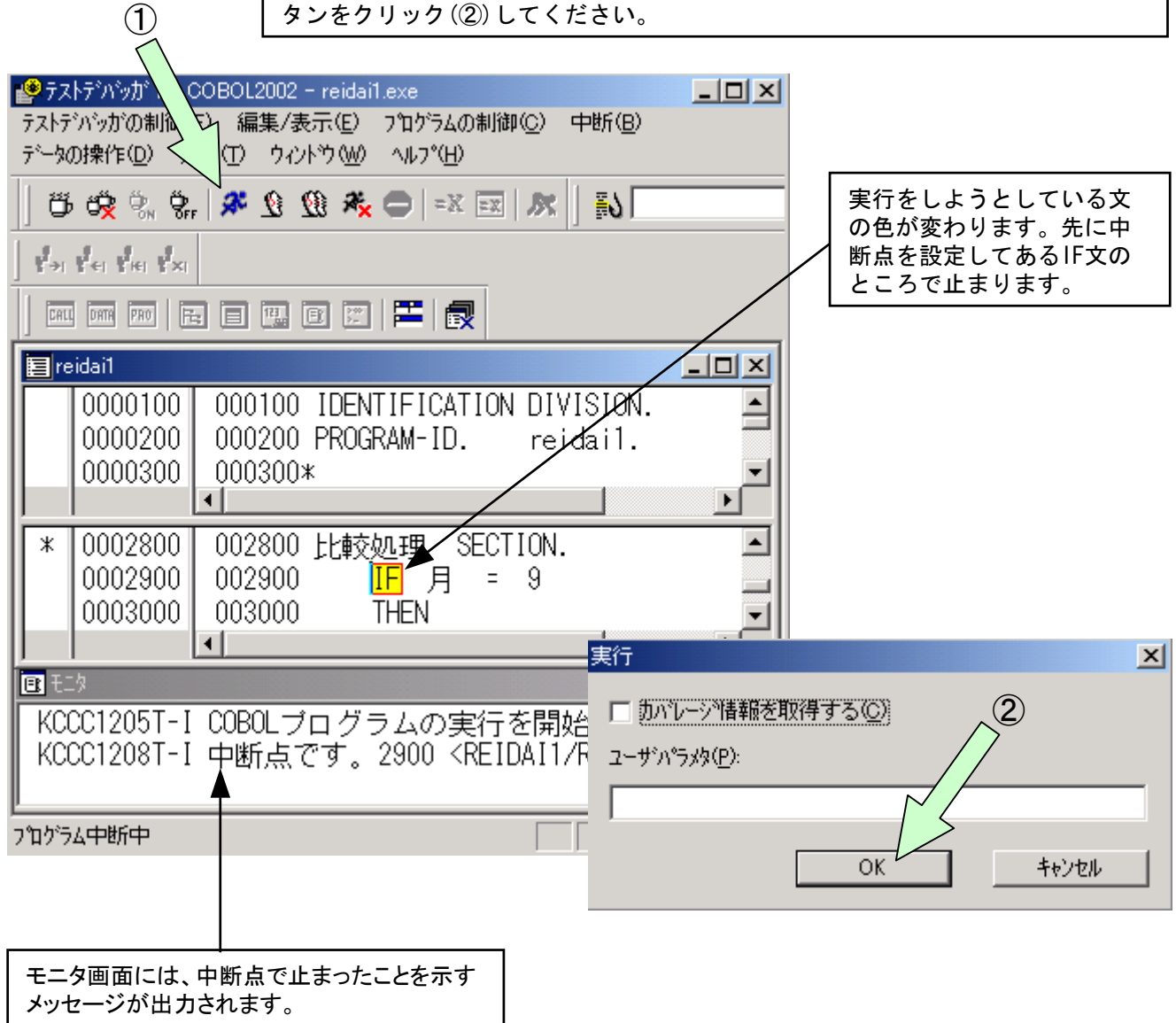


### [ワンポイントアドバイス]

このように文やタグ名称をダブルクリックすると、赤色に変わって中断点が設定されます。ここでは、ダブルクリックをして設定しましたが、文等を黒く反転させそのままマウスの右ボタンをクリックするとプルダウンメニューが表示されます。ここで、「中断点の設定/解除」を選んでもできます。なお、設定を解除する場合はもう一度ダブルクリックして、色を戻せば解除されます。

[手順3] 中断点はいくつ設けてもかまいませんが、必要最小限にした方がよいと思います。中断点の設定ができれば、「連続実行」ボタンをクリックして、実行してみましょう。

「連続実行」ボタンをクリック(①)すると「実行」画面が表示されますので、「OK」ボタンをクリック(②)してください。



①

実行しようとしている文の色が変わります。先に中断点を設定してあるIF文のところで止まります。

実行

☐ 効パレーン情報を取得する(C)

ユーザパラメ(P):

OK キャンセル

モニタ

KCCC1205T-I COBOLプログラムの実行を開始

KCCC1208T-I 中断点です。2900 <REIDAIL/R

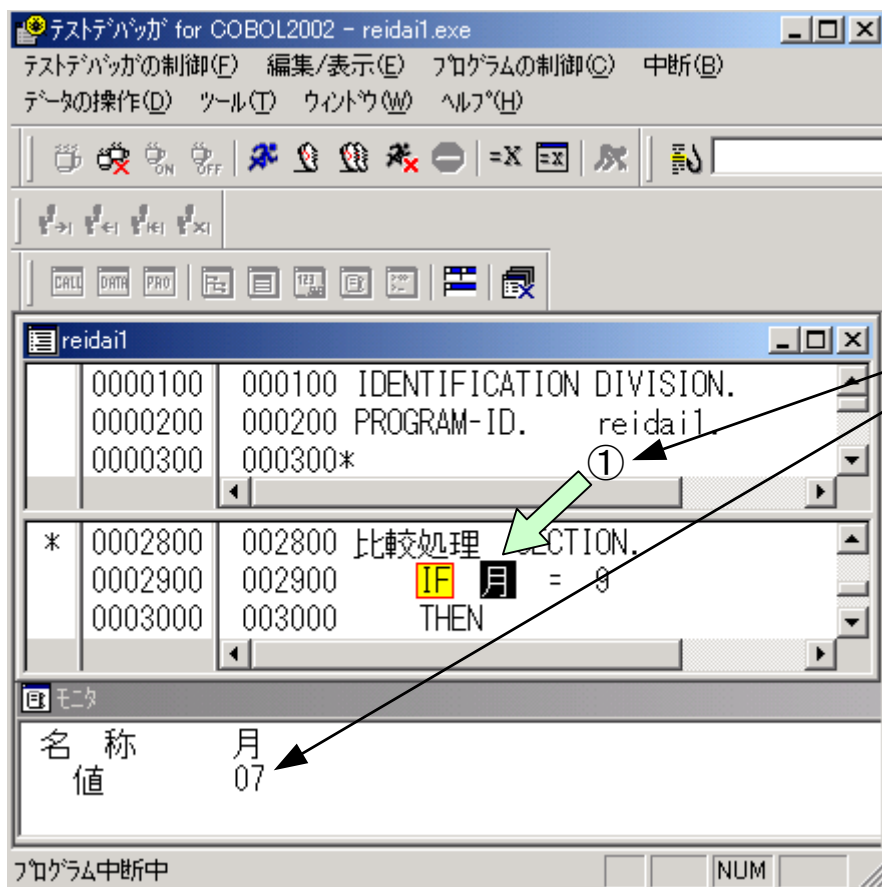
プログラム中断中

モニタ画面には、中断点で止まったことを示すメッセージが出力されます。

## 5. データの内容表示と代入

データの内容を表示したり、データに値を設定することができます。

[手順 1] データ名称をダブルクリックすると、モニタ画面にデータの内容が表示されます。



ここでは、データ名「月」をダブルクリックしてみました。

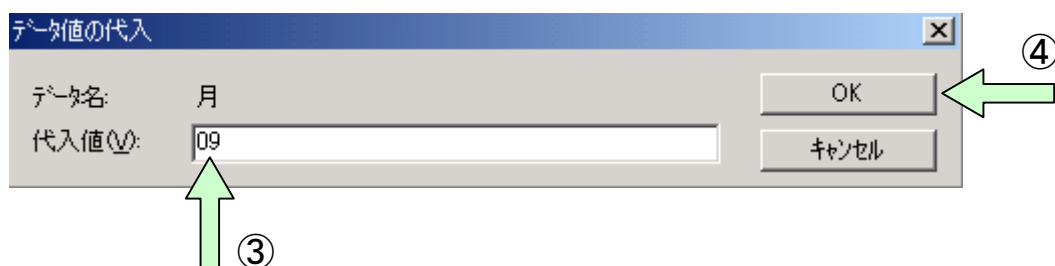
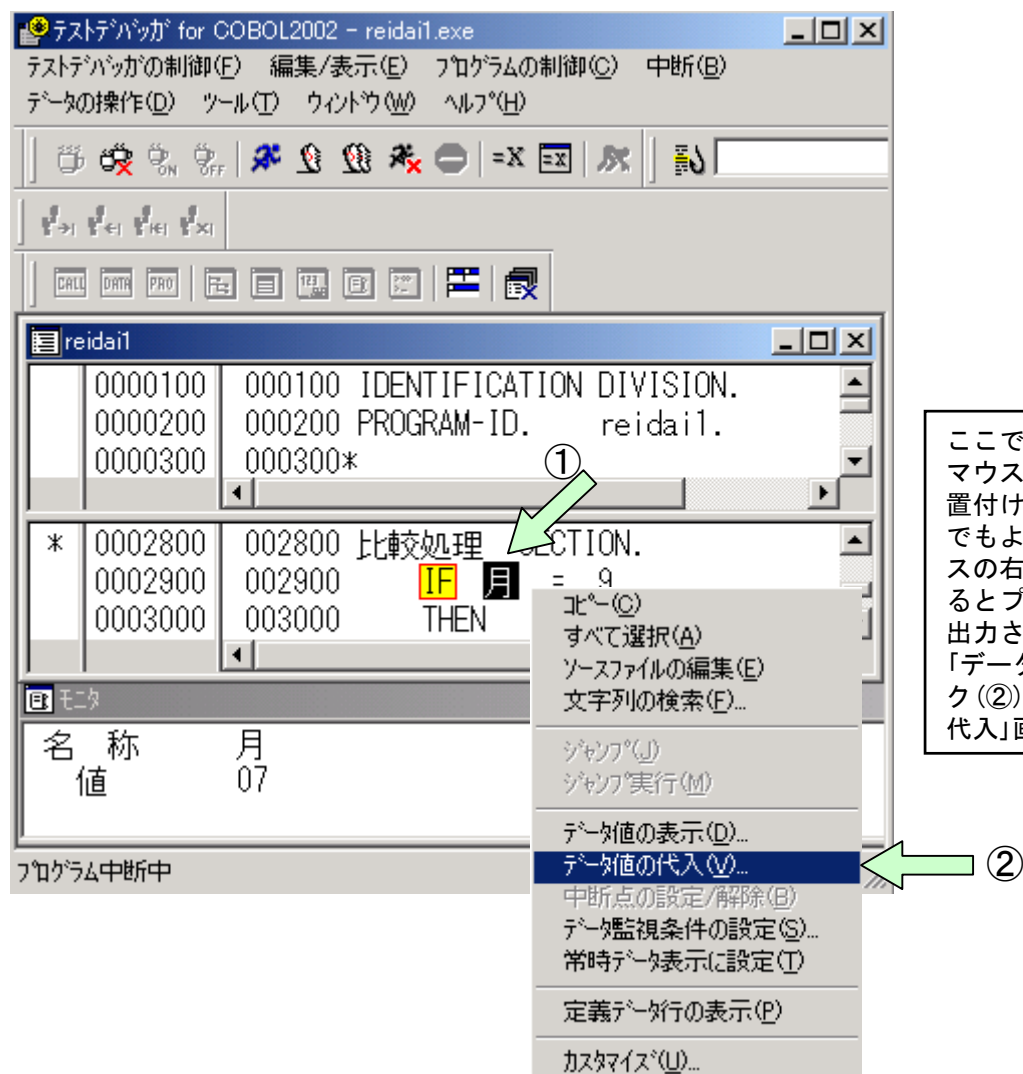
その時のデータの内容がモニタ画面に表示されます。

### [ワンポイントアドバイス]

このようにデータ名をダブルクリックすると、データの内容が表示されます。ここでは、ダブルクリックをして表示しましたが、データ名をクリックし、マウスの右ボタンをクリックして表示されるプルダウンメニューから「データ値の表示」を選んでもかまいません。



[手順 2] データ名に値を設定するには、データ名称をクリックしてそのままマウスの右ボタンをクリックすると、プルダウンメニューが出ますので、その中の「代入」を選択してください。

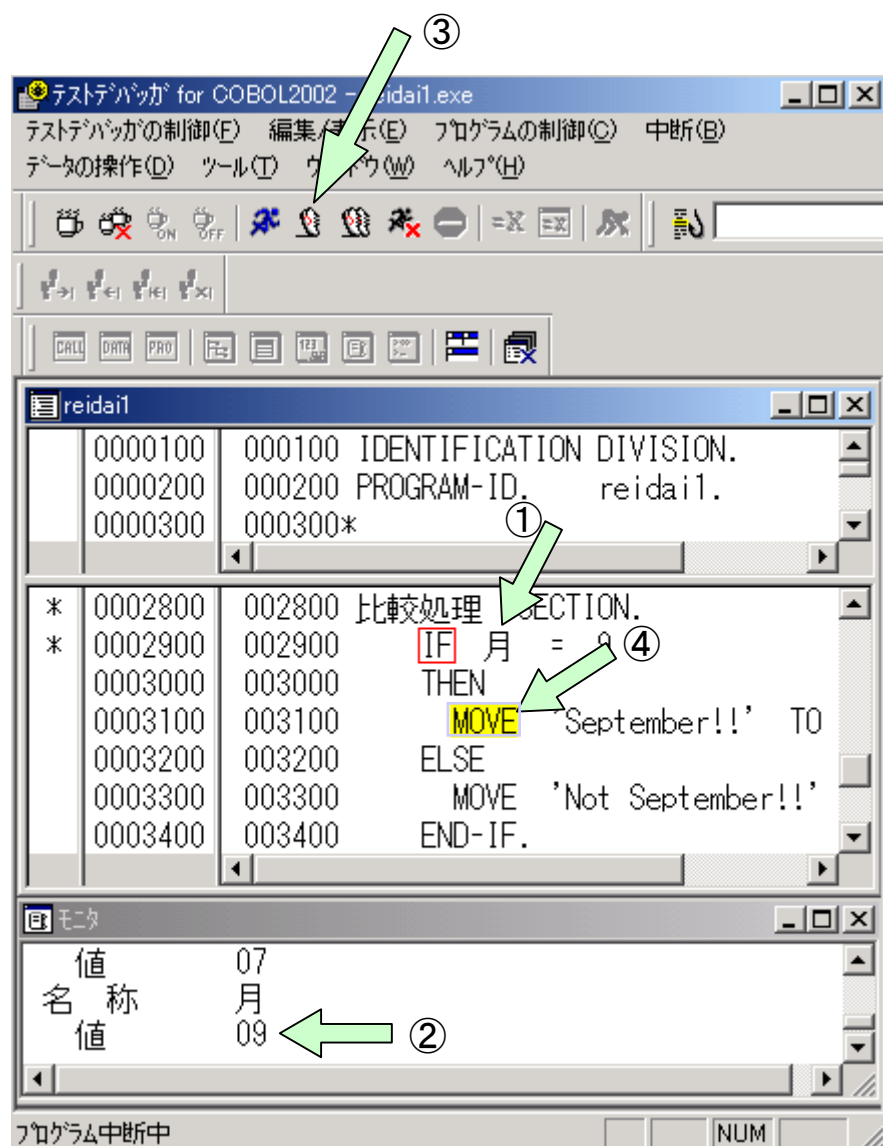


代入したい値を入れます。例えば数字項目である「月」に対して値「09」を設定し、「OK」ボタンをクリックします。

データ名が英数字項目の場合は、値を引用符(')で囲んで指定します。



[手順3] データ名に正しく設定されたかチェックして、1ステップだけ実行してみます。（この操作は確認のためにやっているだけです）



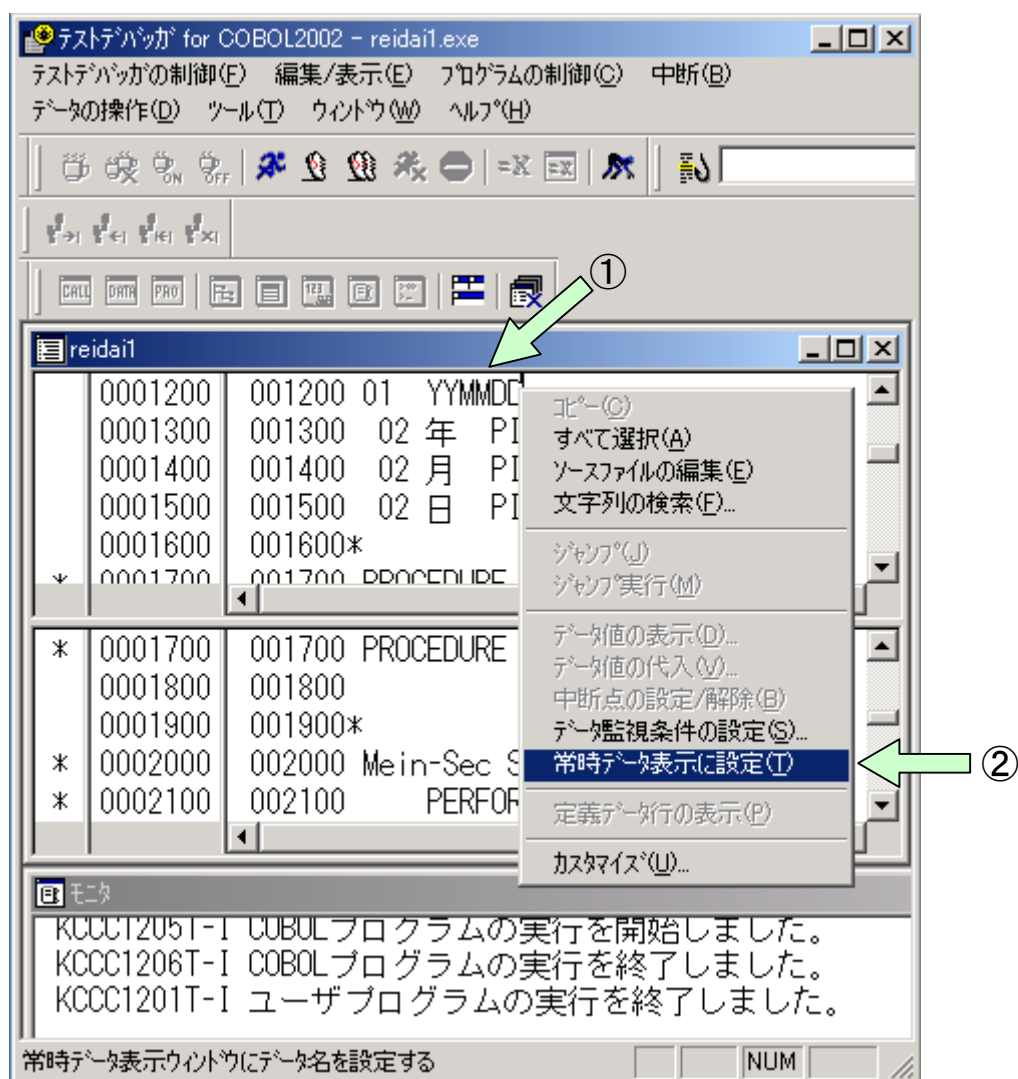
データ名「月」をダブルクリック (①) して、表示してみます。  
値が「09」である (②) ことを確認します。

1ステップだけ実行してみます (③)。  
実行状態を示す色が1ステップだけ、移動します (④)。

## 6. データのトレース

次は、データの状態を常に表示する方法を説明します。

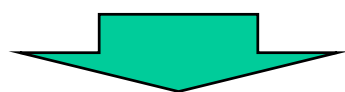
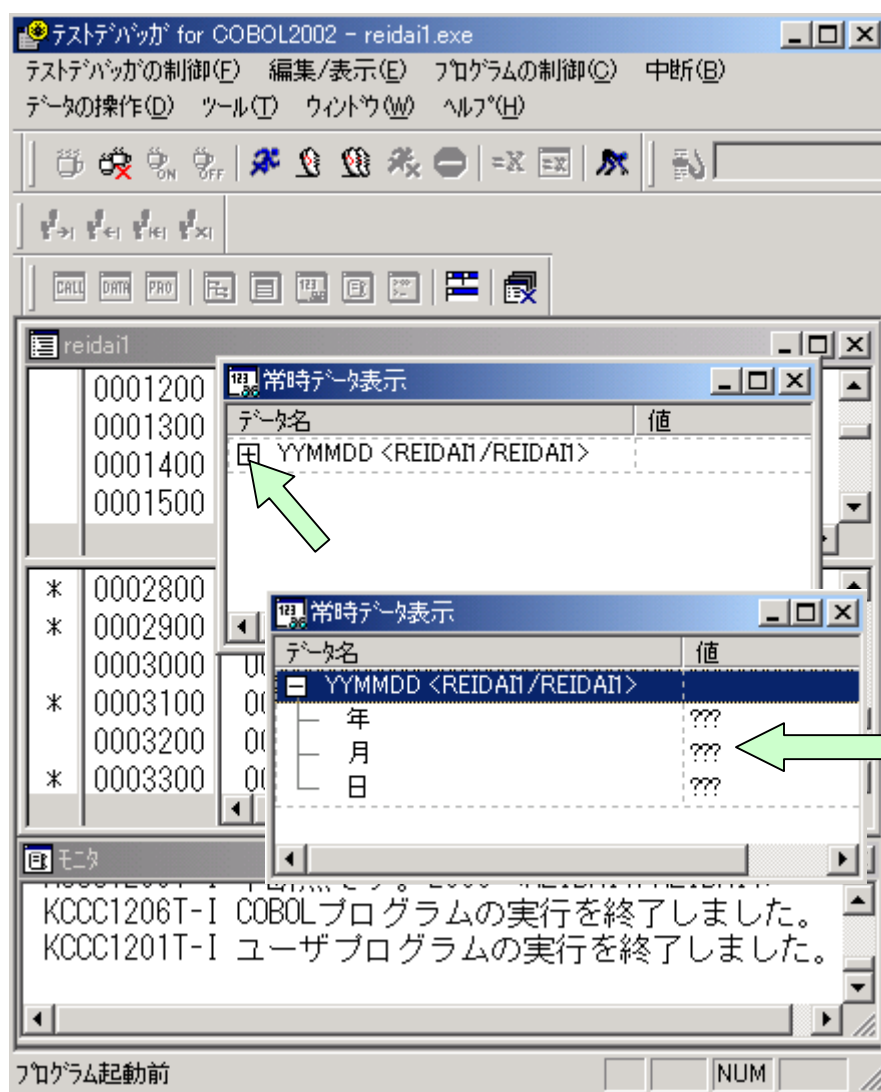
〔手順 1〕 定義画面を適当にスクロールして、表示したいデータ名を見つけます。表示するデータ名をクリックして、そのままマウスの右ボタンをクリックして表示されるプルダウンメニューから「常時データ表示に設定」をクリックします。この例では、データ名「YYMMDD」を常時データ表示します。



### 〔ワンポイントアドバイス〕

常時表示を行うデータ名はいくつ指定してもかまいませんが、必要最小限にとどめた方がよいと思います。

[手順 2] 常時表示画面が表示されてその中に先に指定したデータ名称が表示されます。データ名の+の箇所をクリックすると、下位項目まで表示されます。この常時表示画面のデータ名の右にデータ名の値が変化した場合に値が表示されます。

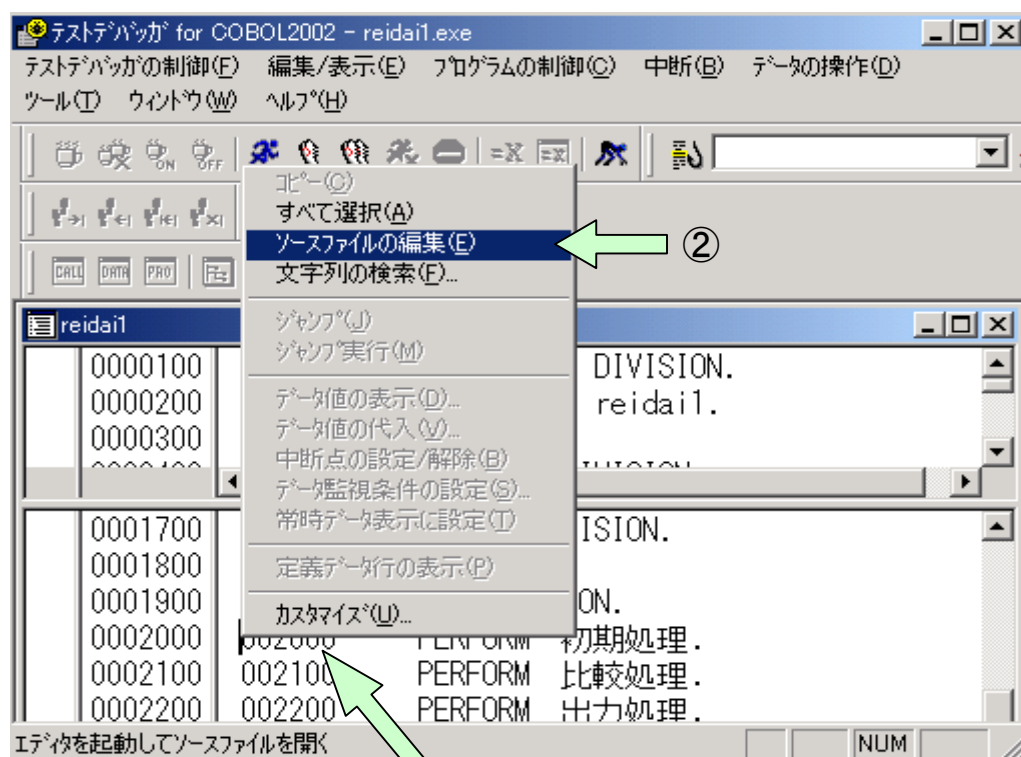


実行ボタンをクリックして実行すると、常時表示画面のデータ名称の値がこのように変化します。  
プログラム中で値が変化しないデータ名は何も変わりません。

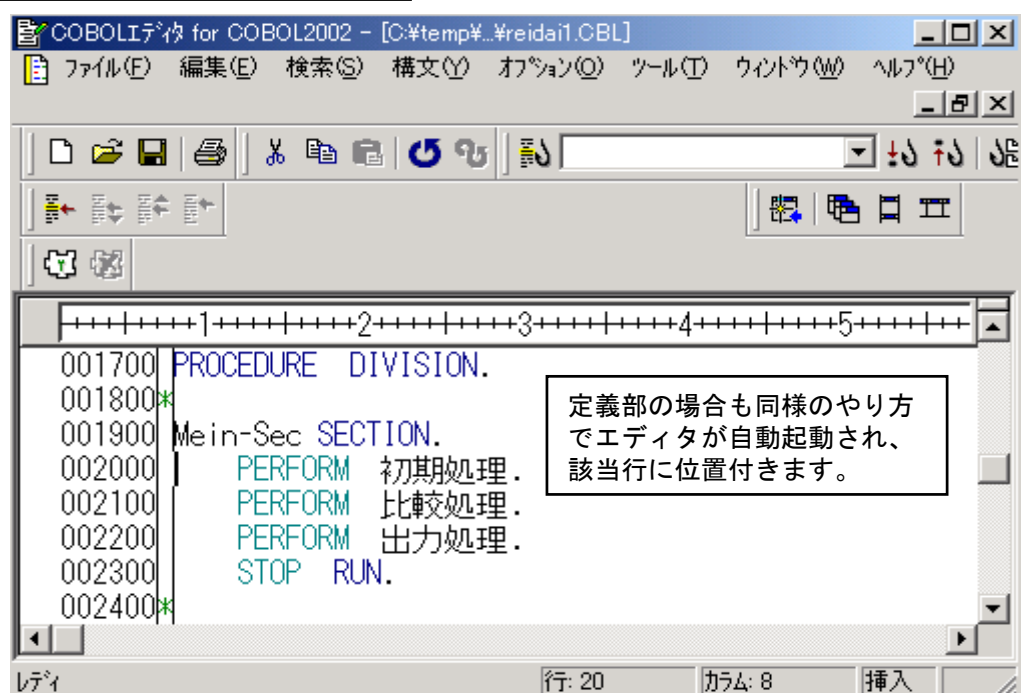
## 7. エディタとの連携

デバッグ中にプログラムを修正したくなった場合は、以下のやり方で簡単にエディタを自動起動できますので、そこで修正してください。

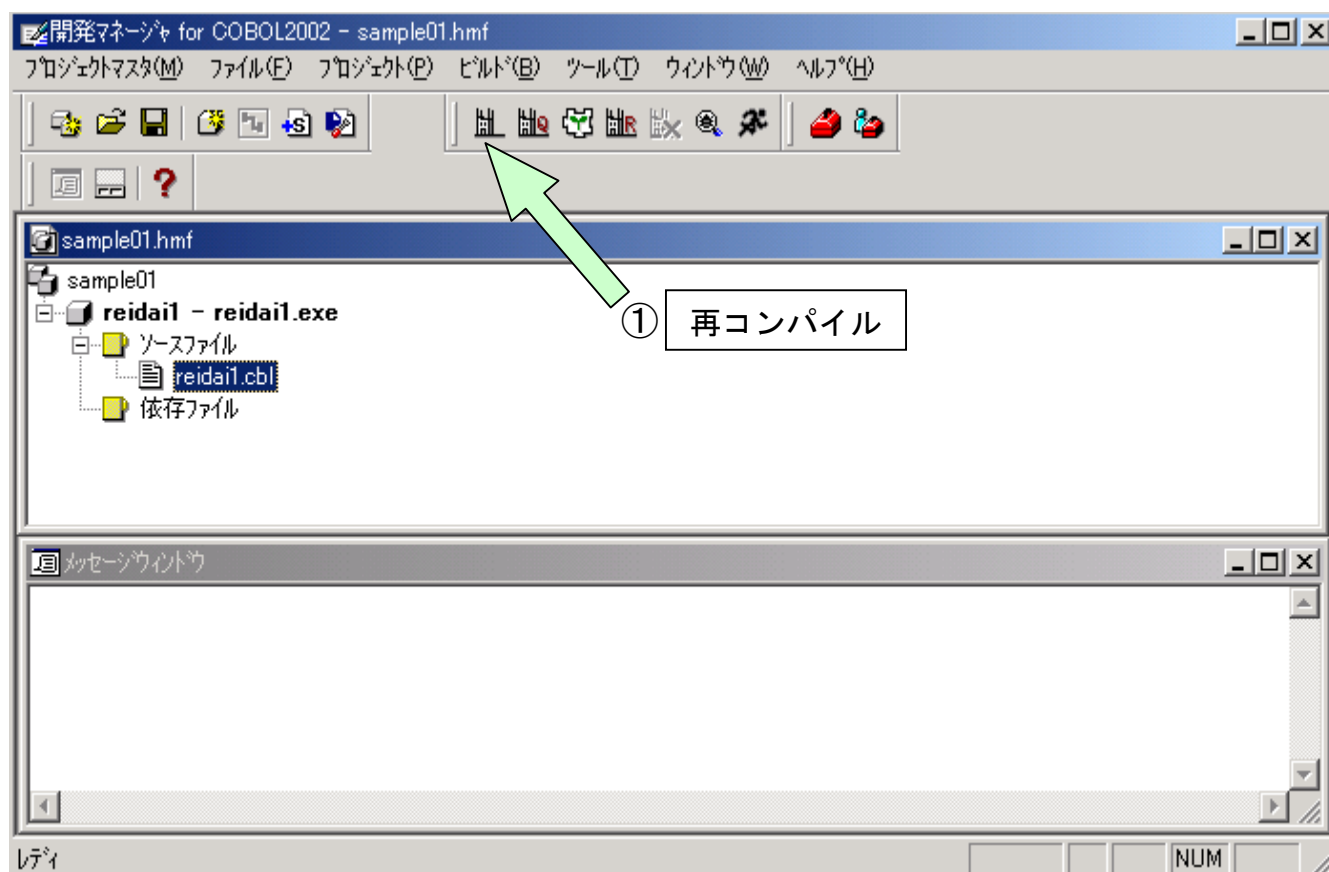
[手順 1] デバッグ画面中の実行画面・定義画面の該当する行にカーソルを位置付け、右クリックをするとプルダウンメニューが出ます。  
そこで、「ソースファイルの編集(E)」を選択してください。エディタが起動され、該当行にカーソルが位置付きます。



該当行にカーソルを位置付ける。①



[手順2] 修正が完了したら、「上書き保存」してエディタを終了します。  
デバッグ画面に戻りますが、デバッグは一旦終了してください。というのは、エディタで修正した部分は現在のデバッグ情報には反映されていないので、改めてコンパイルし直す必要があるからです。したがって、デバッグを終了して開発マネージャに戻り、再コンパイルしてから再度デバッグをしてください。



## 8. デバッガの色などの変更

ここでは、デバッガの画面の配色と、フォントの変更方法を説明します。

[手順 1] デバッグ画面のメニューバーの「ツール(T)」をクリックするとプルダウンメニューが出ます。この中の「カスタマイズ(U)」をクリックするとカスタマイズ画面が出ますので、そこで、色やフォントの設定を行ってください。

テストデバッガ for COBOL2002 - reidai1.exe

テストデバッガの制御(E) 編集/表示(E) プログラムの制御(C) 中断(B) データの操作(D) ツール(T) ウィンドウ(W) ヘルプ(H)

オプション(O)... カスタマイズ(U)...

reidai1

テストデバッガのカスタマイズを行う

カスタマイズ

画面の色とフォント TDコマンド入力 ツールバー

カテゴリ(C): ソーステキストウィンドウ

配色(L)

背景色  
行番号領域の文字色  
ソーステキスト表示域の文字色  
現在位置を表す強調色  
バックトレース位置を表す強調色  
中断点の枠の色  
実行履歴マークの色

色を設定(O)

色を設定

基本色(B):

作成した色(C):

色の作成(D) >>

OK キャンセル

標準に戻す(D)

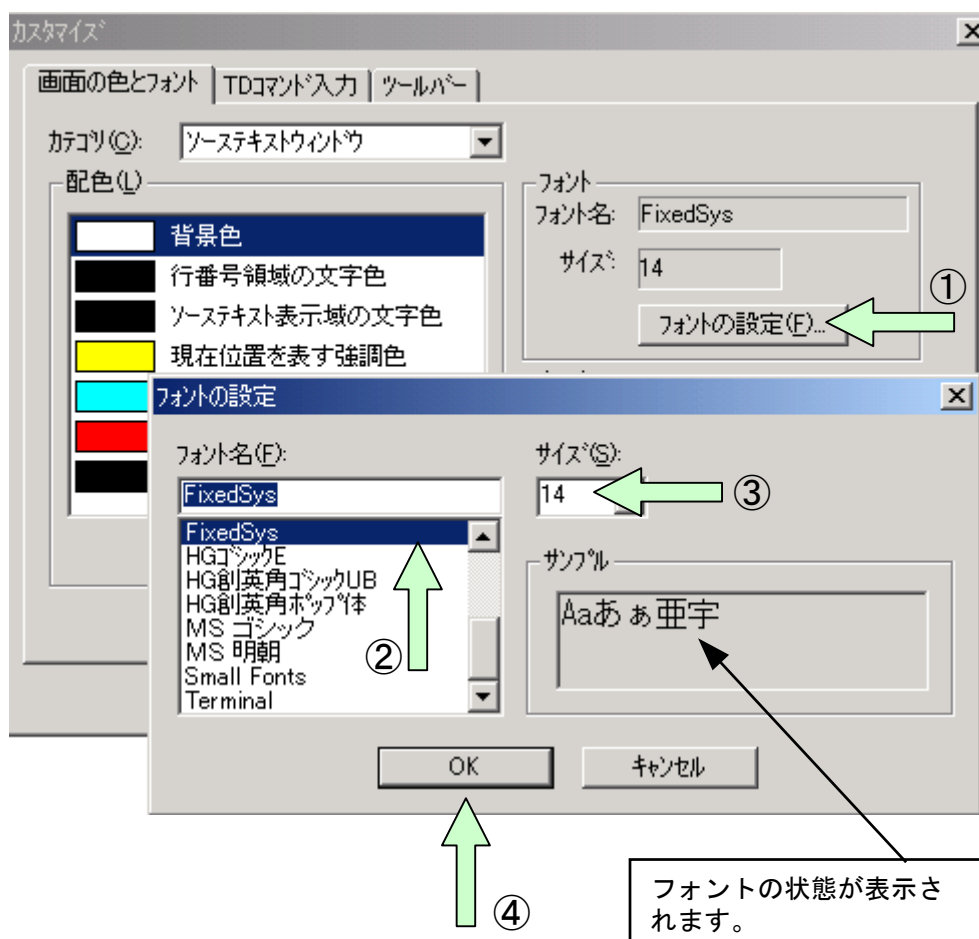
OK キャンセル

▼をクリックして、変更したい項目を選びます。

「色の設定」ボタンをクリックすると選択できる色の一覧がでますので、そこで色を選んでください。

色の設定が終わったら「OK」ボタンをクリックしてください。

フォントの種類やサイズを選んで変更してください。  
修正が終わったら「OK」ボタンをクリックしてください。





## 9. カバレッジ情報の蓄積と表示

カバレッジとは、テスト進捗状況を定量的に把握する機能です。カバレッジ情報を採取するには、コンパイラオプション-CVInfを指定してコンパイルします。カバレッジ情報には、次の3種類の指標があります。

- ①C0メジャー：実行した文の割合を示します。  

$$C0メジャー = (\text{実行が済んだ文の数}) / (\text{実行文の数}) \times 100 (\%)$$
- ②C1メジャー：分岐する個所で、実行した分岐先の割合を示します。  

$$C1メジャー = (\text{実行が済んだ分岐先の数}) / (\text{分岐先の数}) \times 100 (\%)$$
- ③S1メジャー：実行した呼び出し文 (CALL文や INVOKE文) の割合を示します。  

$$S1メジャー = (\text{実行が済んだ呼び出し文の数}) / (\text{呼び出し文の数}) \times 100 (\%)$$

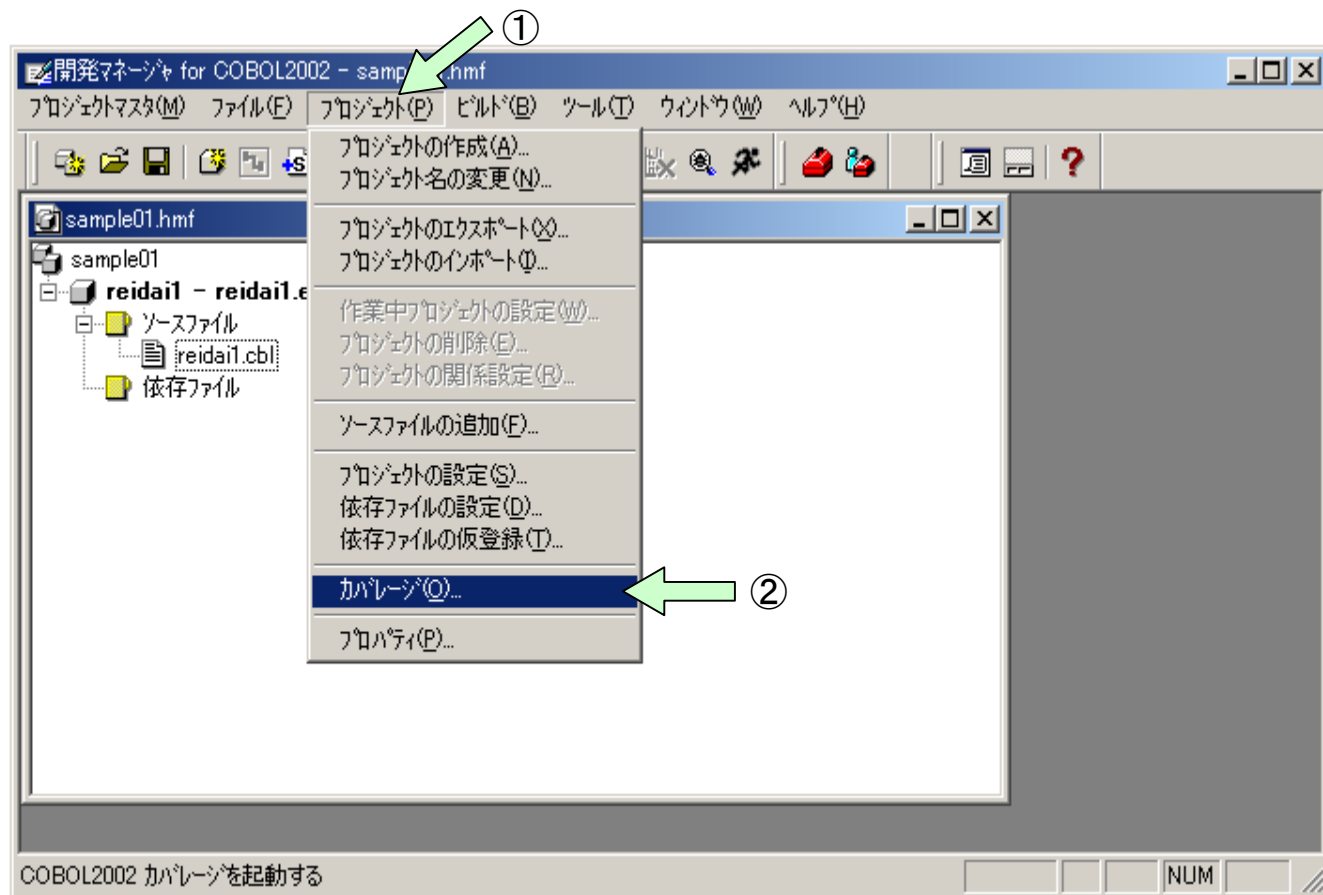
これらの情報は、プログラム情報ファイルに蓄積されます。プログラム情報ファイルは、実行可能ファイルと同じフォルダに作成されます。

### [ワンポイントアドバイス]

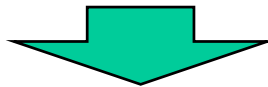
プログラム情報ファイルを実行可能ファイルとは別のフォルダに作成したい場合は、コンパイル時の環境変数CBLPIDIRでフォルダを指定します。コンパイル時の環境変数は、コンパイラオプションを指定するときと同様に「プロジェクトの設定(S)」をクリックし、コンパイラオプション一覧の中から「環境変数」タブをクリックして設定します。

### [手順1] カバレッジ情報の蓄積

開発マネージャのメニューバーの「プロジェクト(P)」をクリックし、プルダウンメニューの中の「カバレッジ(O)」をクリックします。すると、カバレッジのウインドウが開かれます。





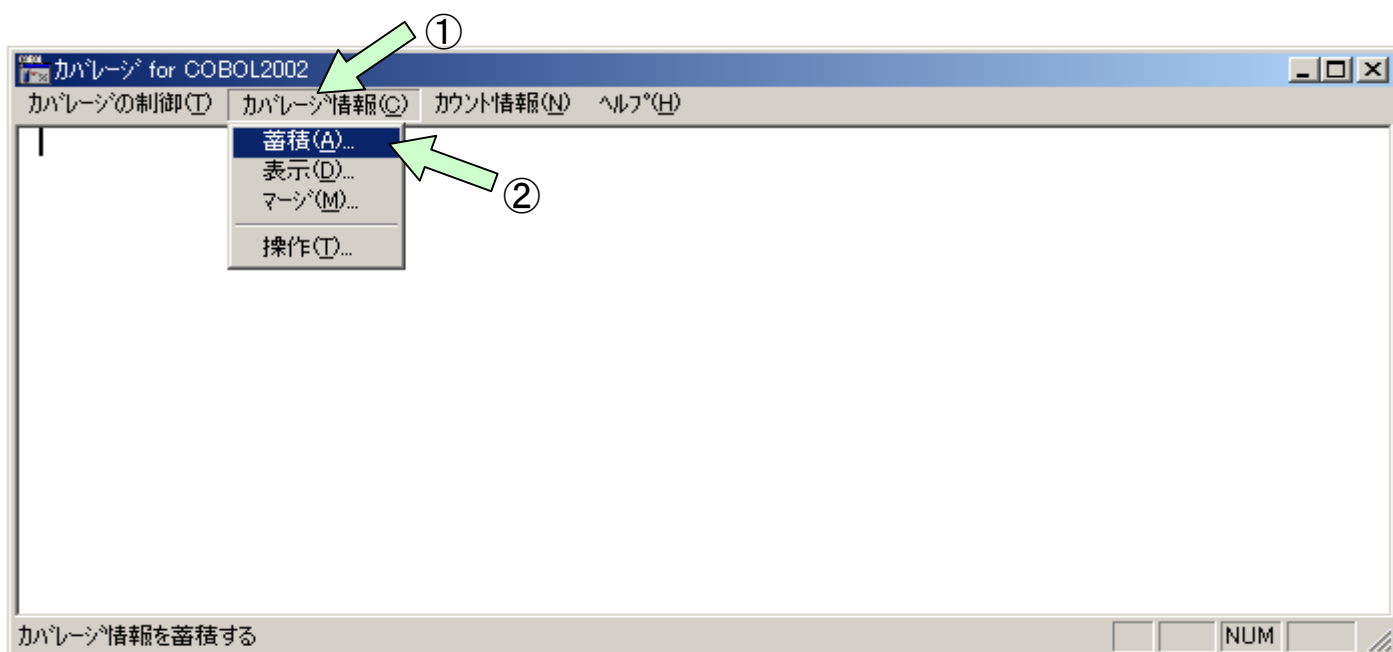


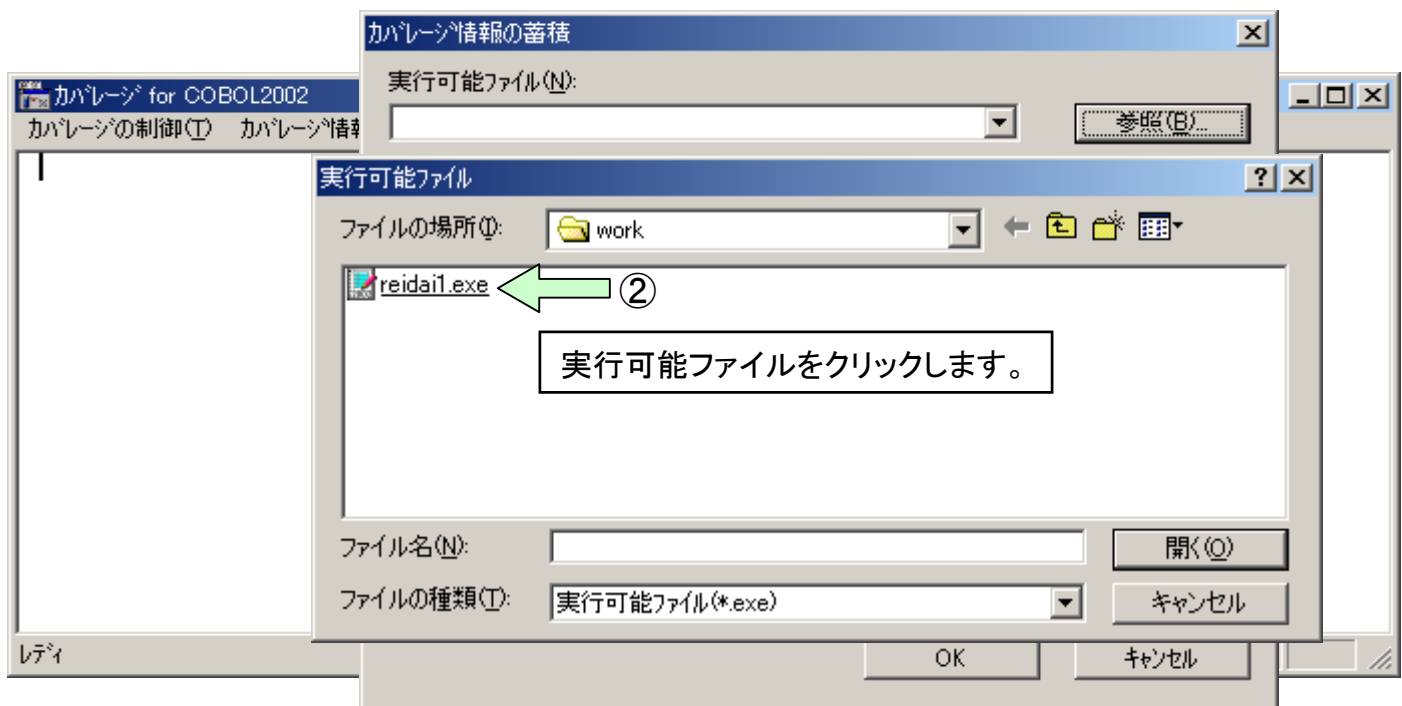
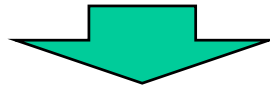
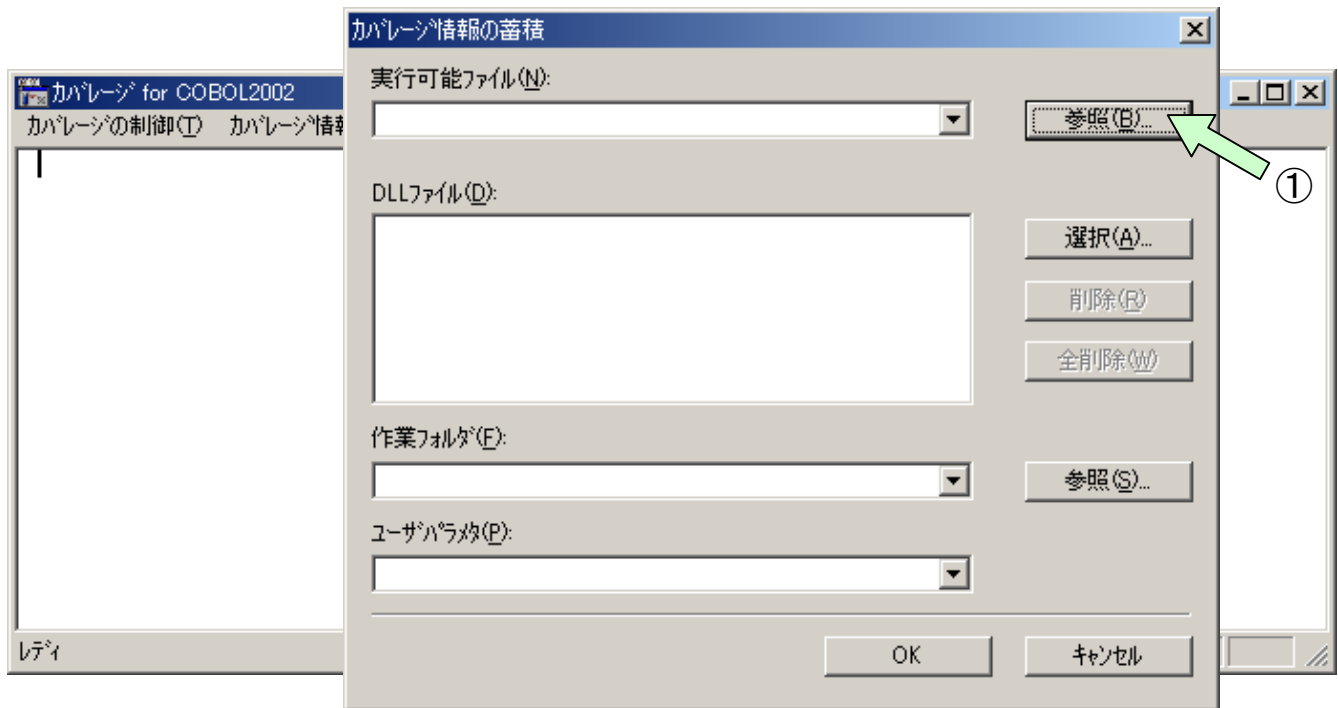
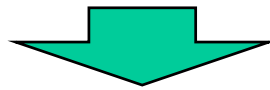
## [手順 2] カバレッジ情報の蓄積

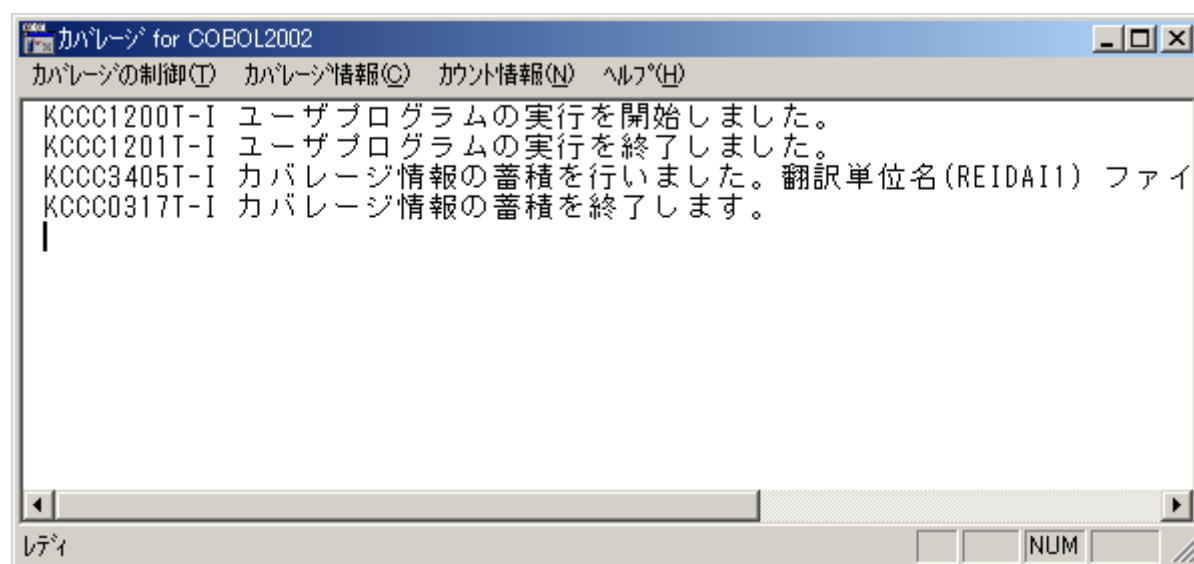
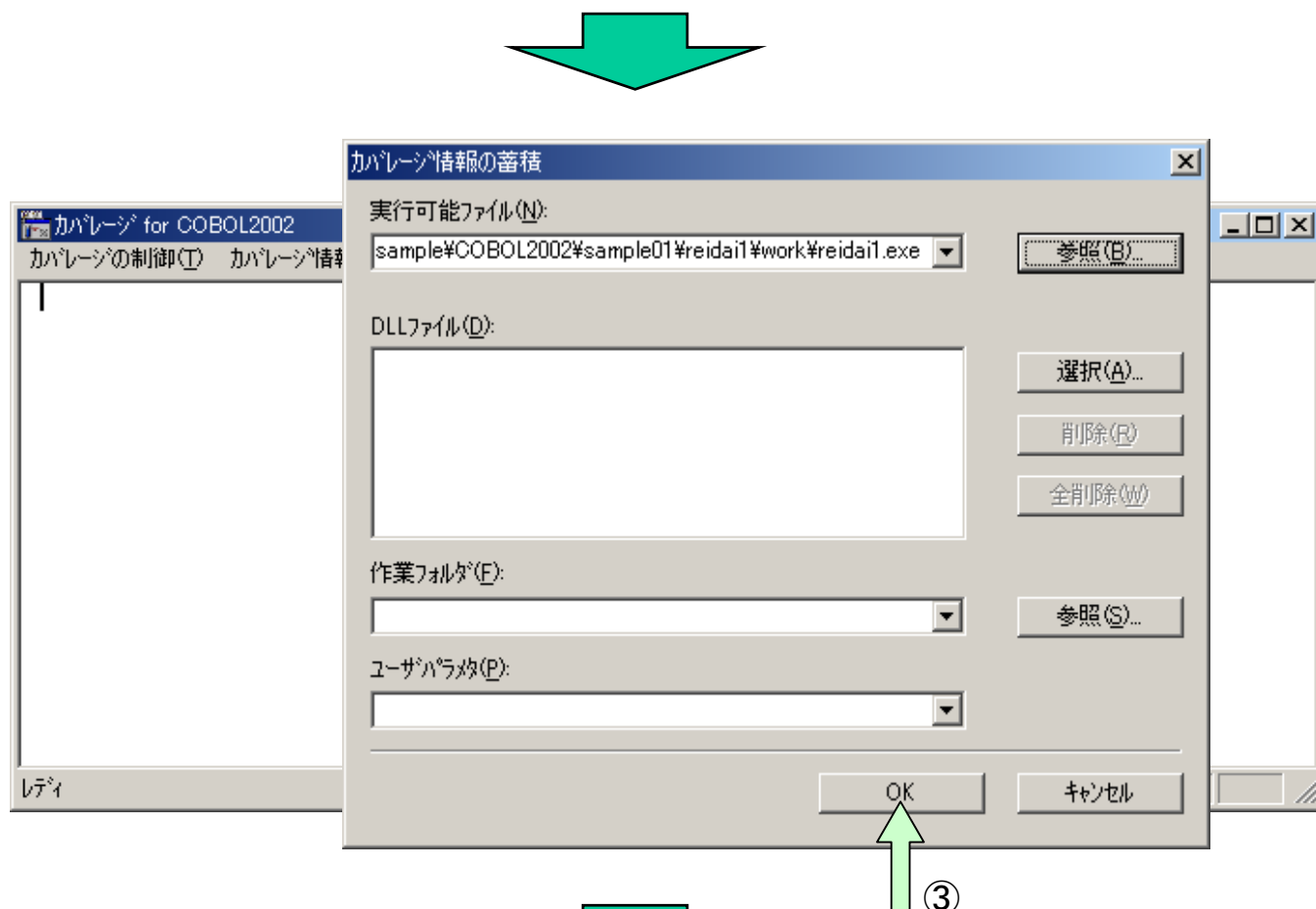
カバレッジウインドウのメニューバーの「カバレッジ情報(C)」をクリックし、プルダウンメニューの「蓄積(A)」をクリックすると、「カバレッジ情報の蓄積」画面が表示されます。「実行可能ファイル(N)」の参照ボタンで実行可能ファイルを指定します。「OK」ボタンをクリックすると、プログラムが実行されカバレッジ情報が蓄積されます。

### [ワンポイントアドバイス]

デバッガからの実行でもカバレッジ情報を蓄積することができます。

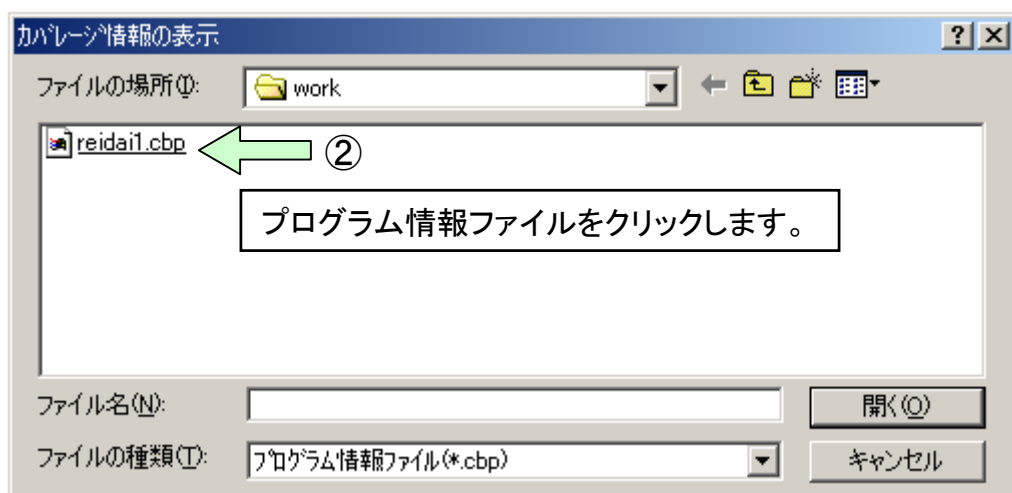
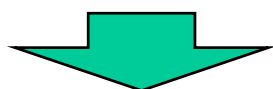
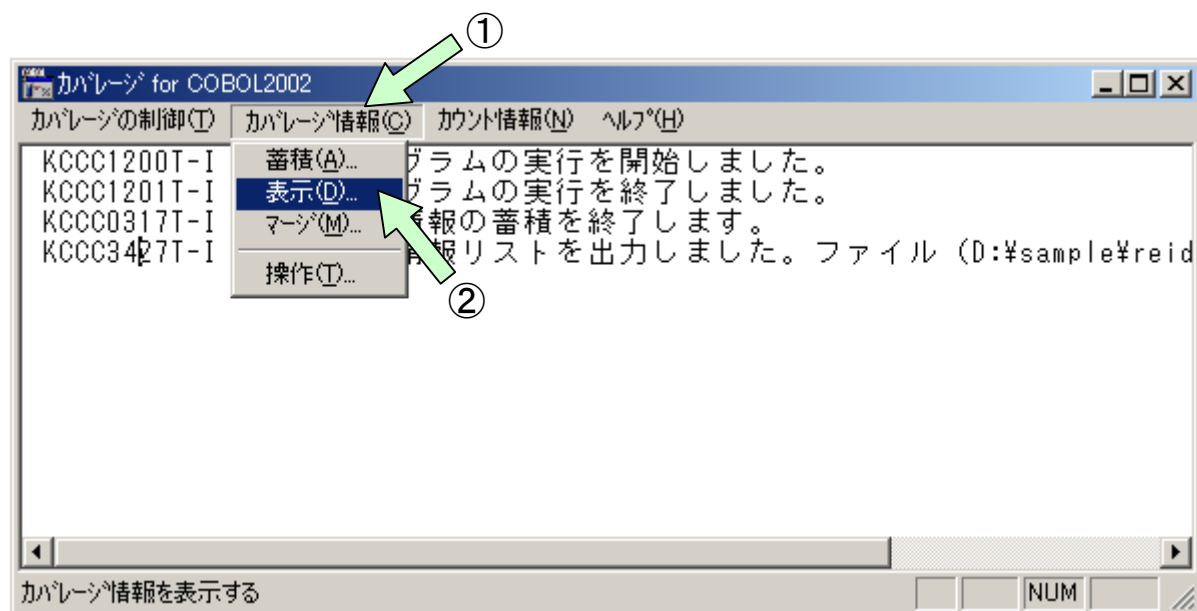






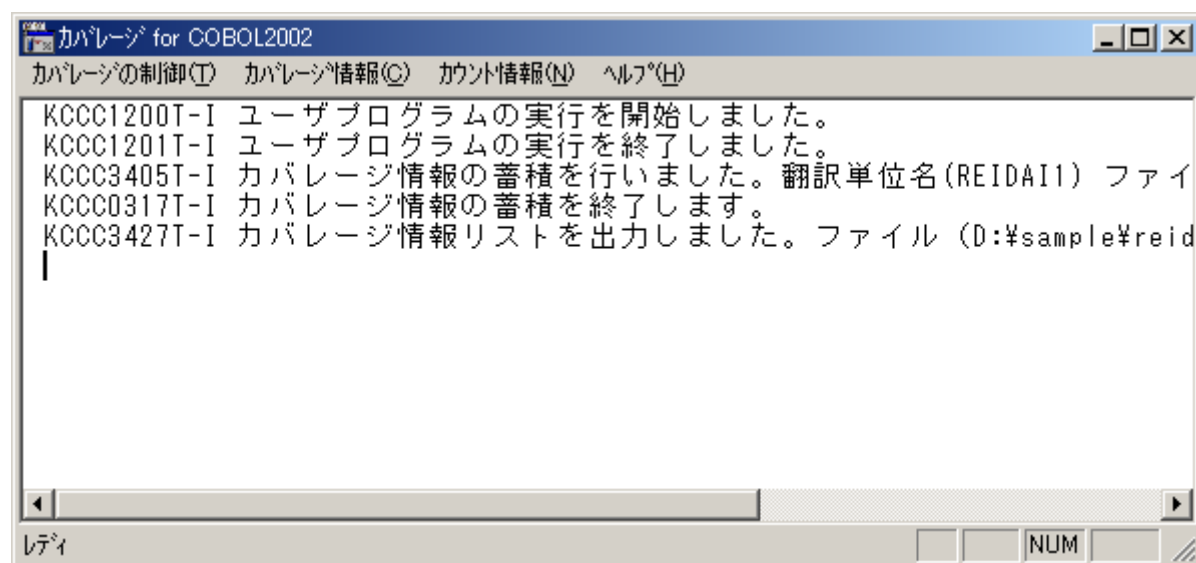
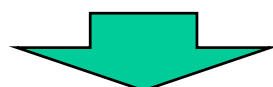
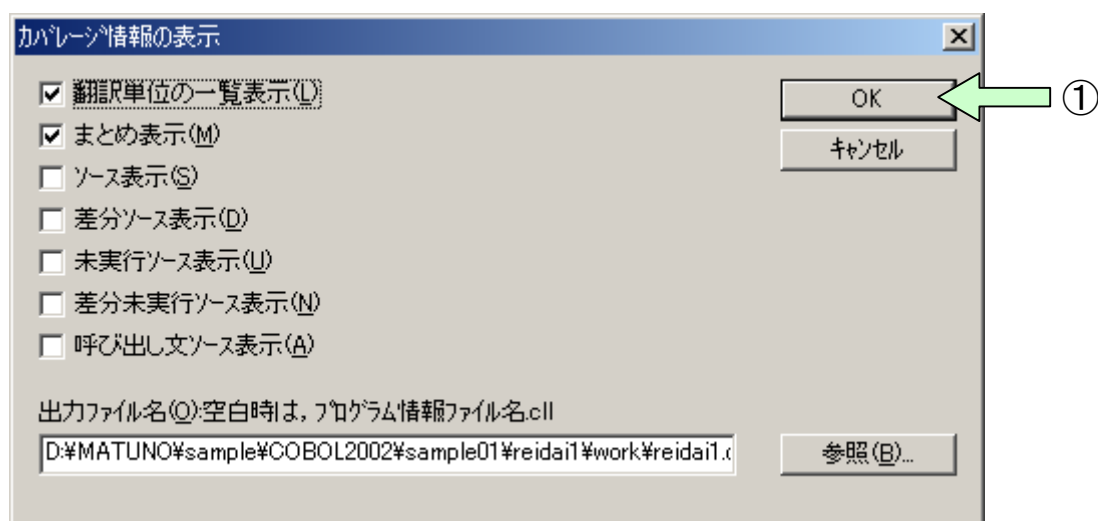
### 〔手順3〕 カバレージ情報の表示

カバレージウインドウのメニューバーの「カバレージ情報(C)」をクリックし、プルダウンメニューの「表示(D)」をクリックします。すると、「カバレージ情報の表示」画面が表示されるので、プログラム情報ファイル(.cbp)を指定します。



## [手順 4] カバレッジ情報の表示

「カバレッジ情報の表示」画面の中の表示したい項目をクリックします。ここでは、「翻訳単位の一覧表示」と「まとめ表示」をクリックし、「OK」ボタンをクリックします。

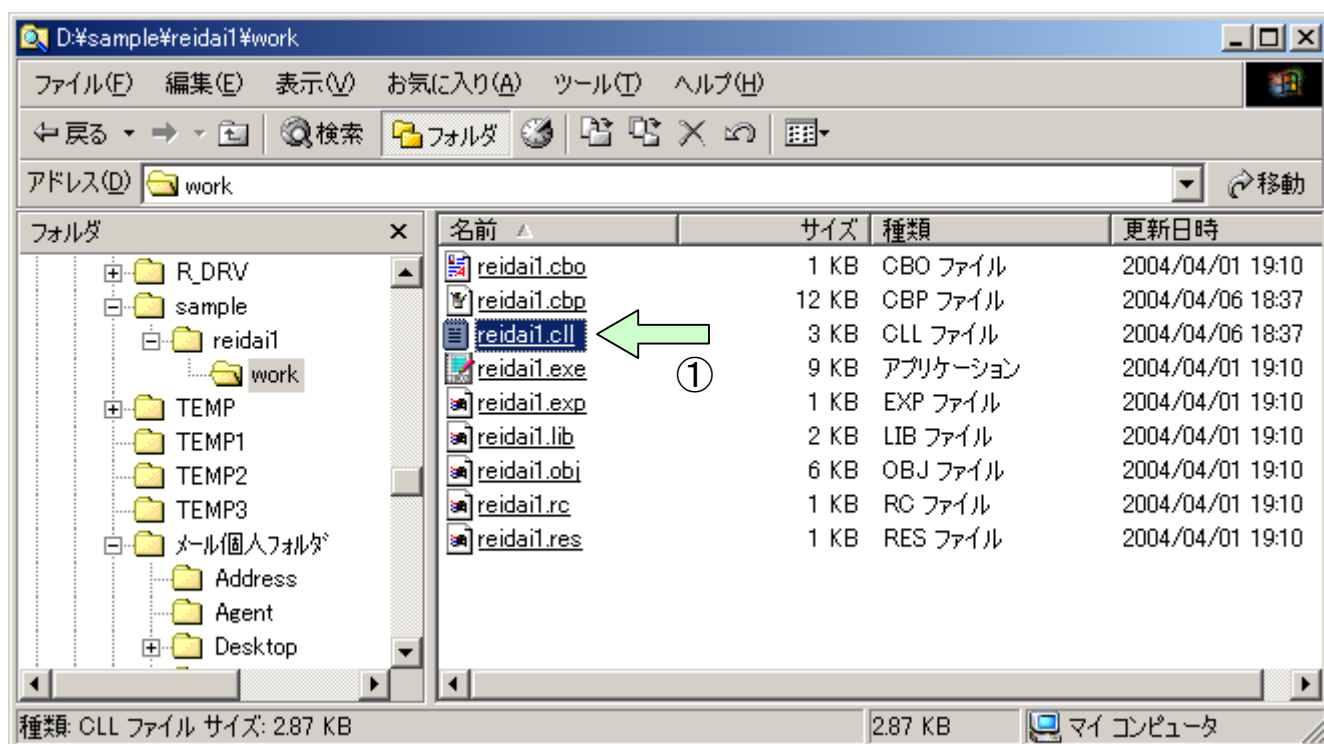


## [手順5] カバレッジ情報の表示

実行可能ファイルと同じフォルダに.cllという拡張子のファイルが生成されています。このファイルをCOBOLエディタやメモ帳で開いて、カバレッジ情報を見ることができます。カバレッジ情報の表示例を次ページに示します。

### [ワンポイントアドバイス]

同じ条件で複数回実行してもテスト回数は1回として扱われます。実行ルートが異なるテストをする度にカバレッジ情報は蓄積されます。



[カバレッジ情報の表示例]

「まとめ情報」の例を次に示します。

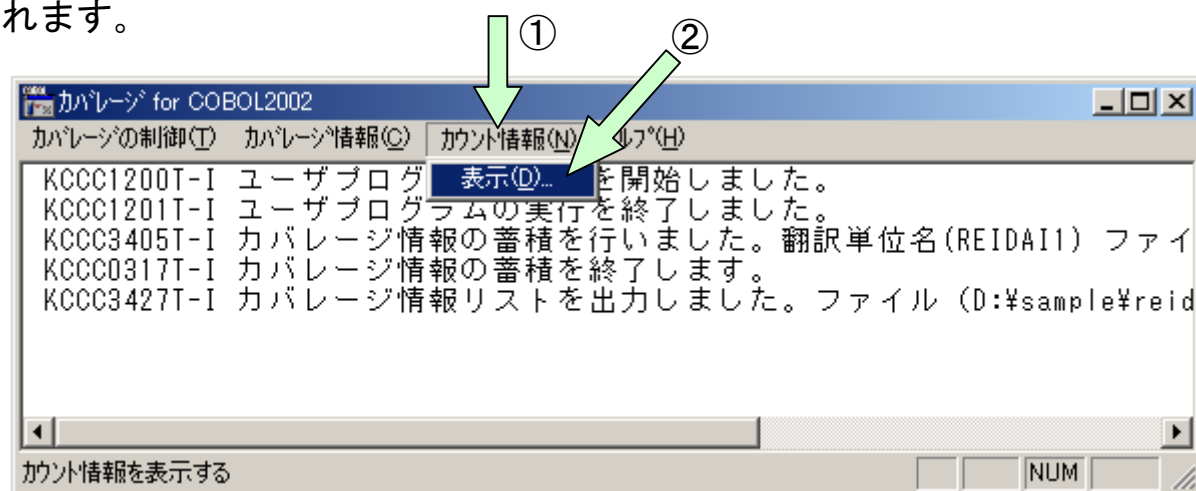
*****													
* カバレッジ情報 *													
*****													
COBOL2002 (X) 01-01 2004-04-06 18:37:15													
-----													
プログラム名 : REIDA11													
コンパイル日時: 2004-04-01 19:10:39 変更回数 : 0													
テスト日時 : 2004-04-06 18:37:03 テスト回数 : 1													
*****													
* <C0> <差分C0> <C1> <差分C1> <S1> <差分S1> *													
* 対象総数 9 0 2 0 0 0 *													
* 実行済数 8 0 1 0 0 0 *													
* 未実行数 1 0 1 0 0 0 *													
* カバレッジ率 88.8% 0.0% 50.0% 0.0% 0.0% *													
*****													
*****													
* カバレッジ情報 *													
*****													
COBOL2002 (X) 01-01 2004-04-06 18:37:15													
*****													
* -----C0-----* -----差分C0-----* -----C1-----* -----差分C1-----*													
番号	名 称	種 別	対象総数	実行済数	C0	対象総数	差分C0	実行済数	C1	対象総数	差分C1	実行済数	差分C1
1	REIDA11	P	9	8	88.8%	0	0	0.0%	1	50.0%	0	0	0.0%
*****													
* 合計 9 8 88.8% 0 0 0.0% 2 1 50.0% 0 0 0.0%*													
*****													

## 10. カウント情報の表示

カウント情報は、プログラム中の文の実行回数を示します。

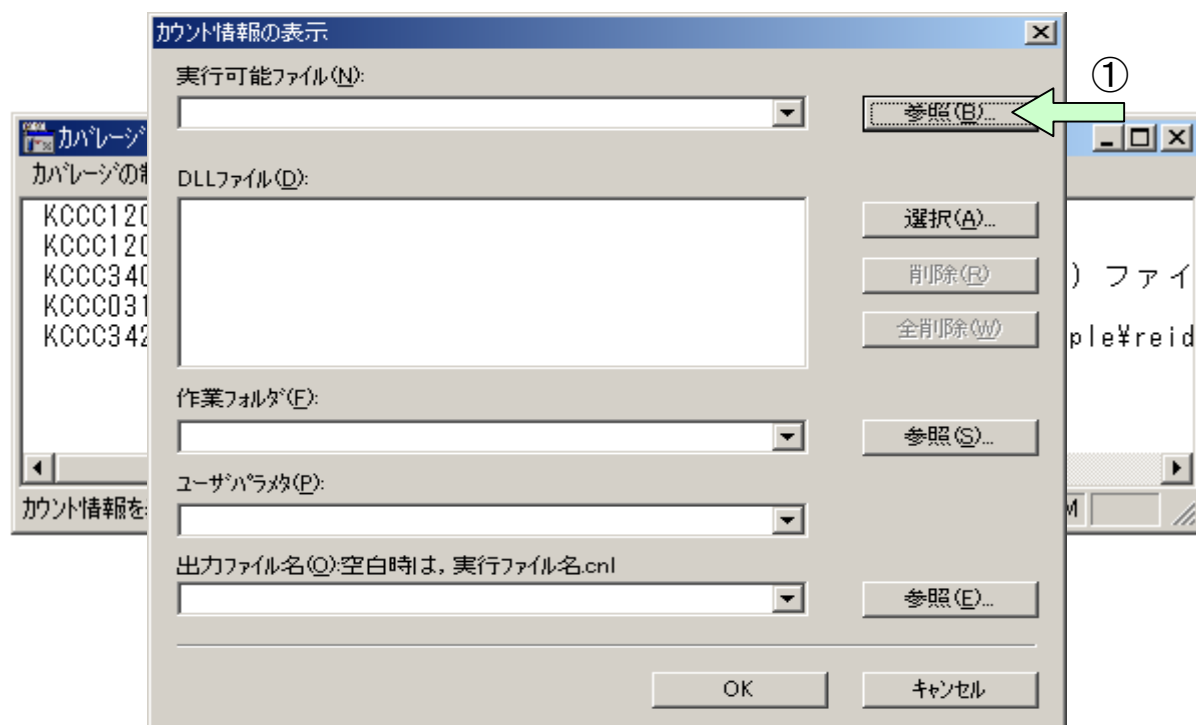
### [手順 1] カウント情報の表示

カバレッジウインドウのメニューバーの「カウント情報(N)」をクリックし、プルダウンメニューの「表示(D)」をクリックすると、「カウント情報の表示」画面が表示されます。

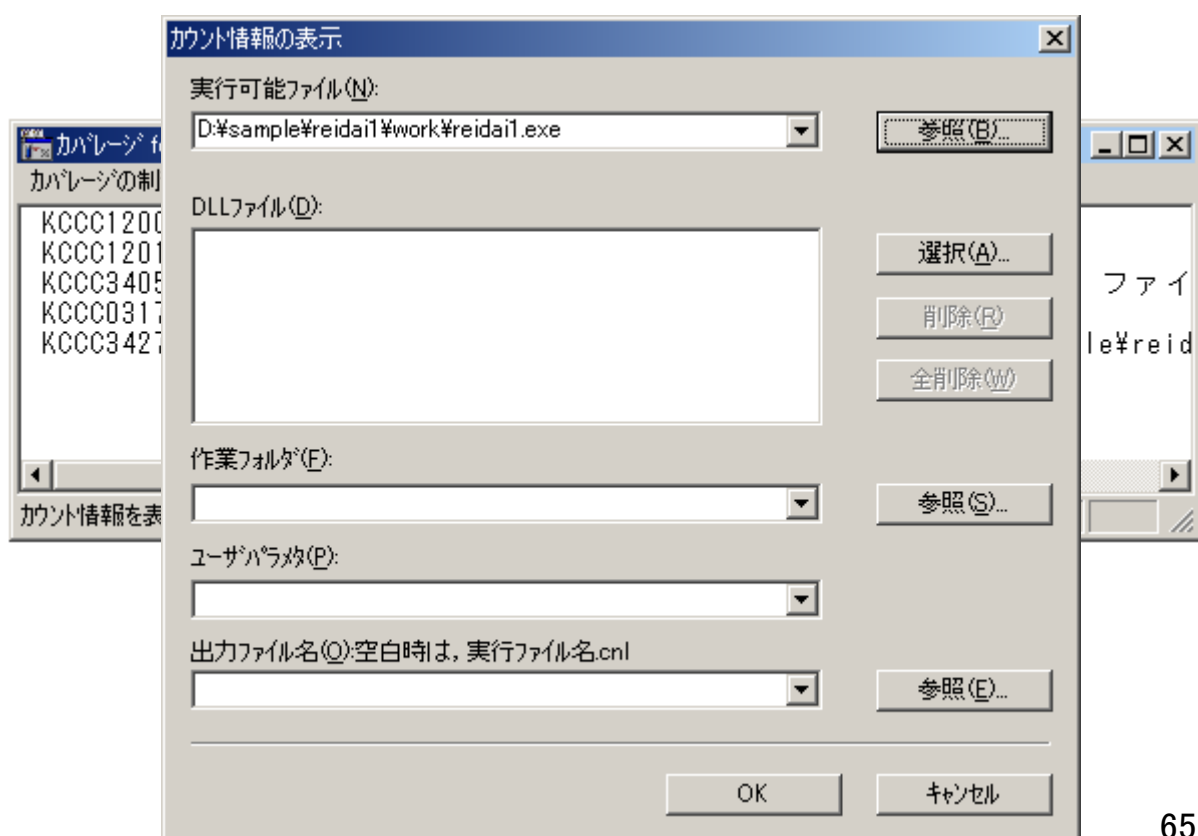
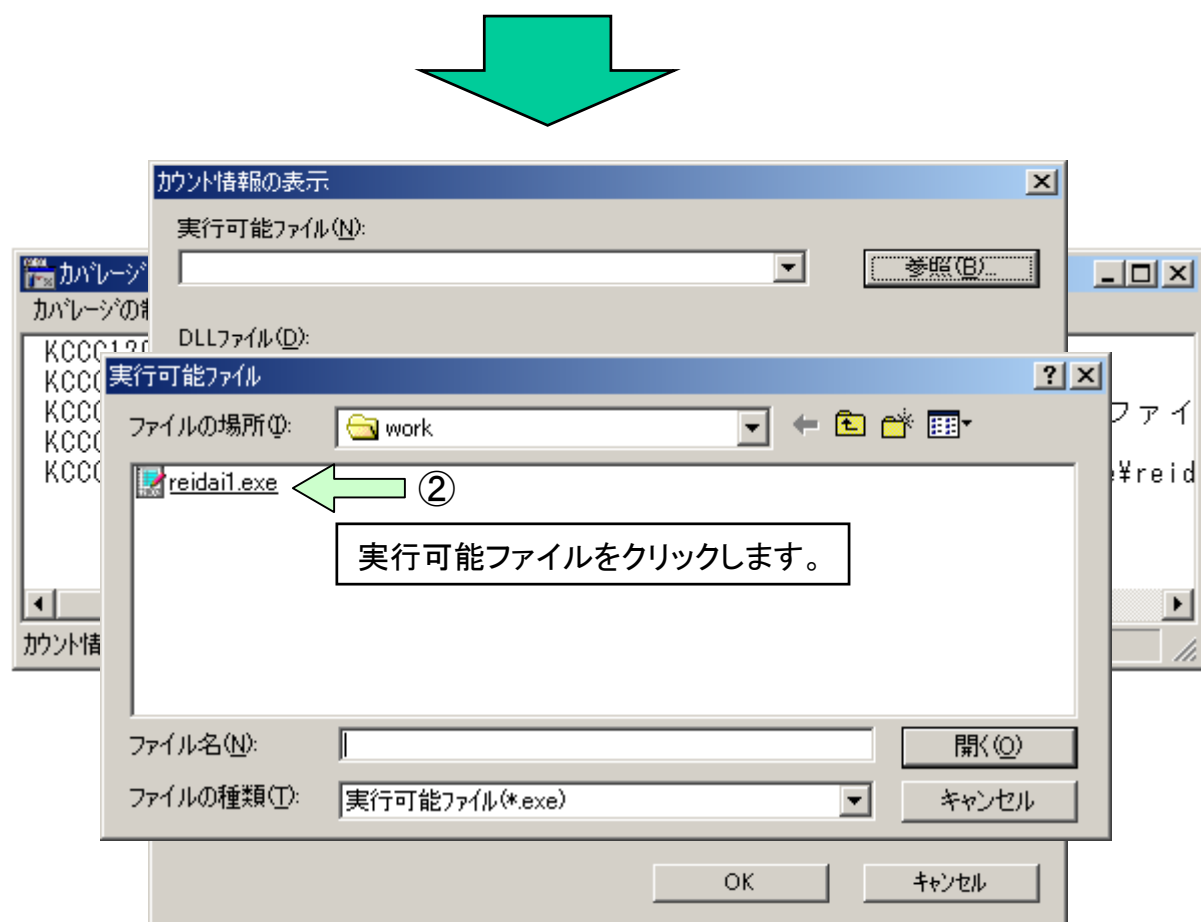


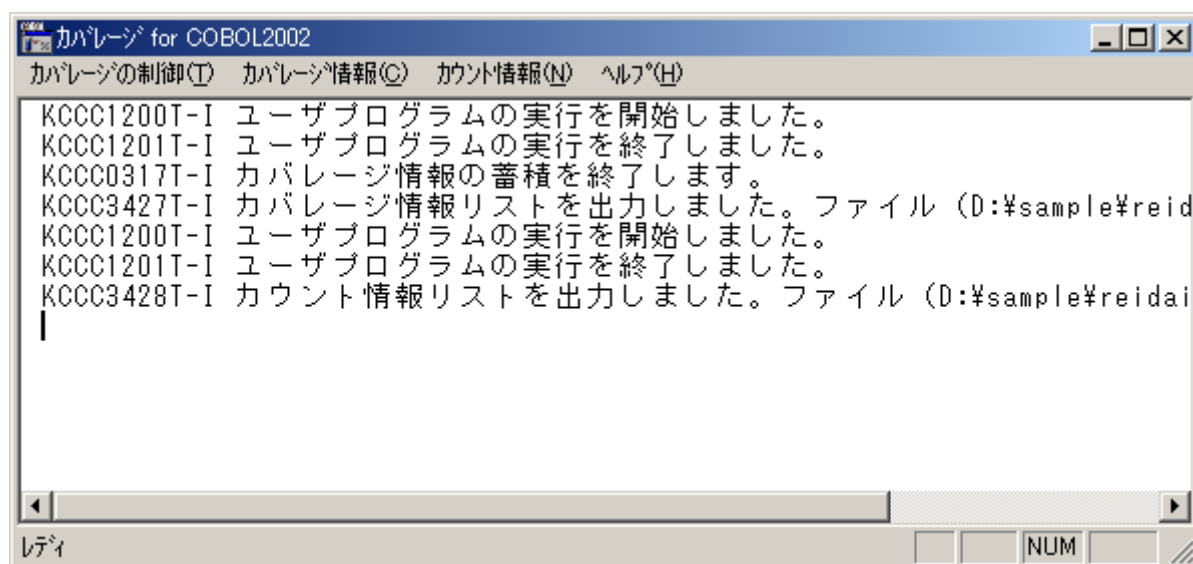
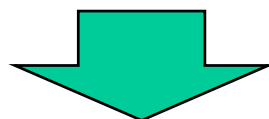
### [手順 2] カウント情報の表示

「カウント情報の表示」画面で、「実行可能ファイル(N)」の参照ボタンをクリックし実行可能ファイルを指定します。



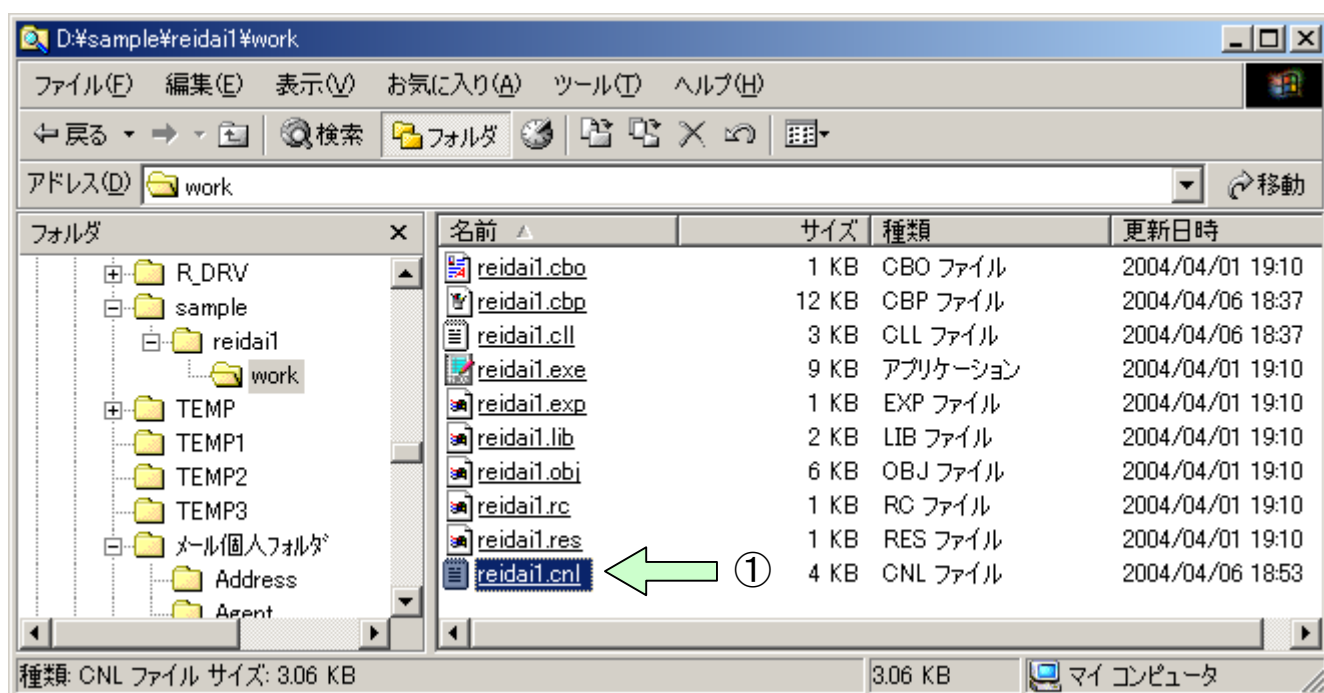






### 〔手順3〕 カウント情報の表示

実行可能ファイルと同じフォルダに、.cniという拡張子のファイルが生成されています。このファイルをCOBOLエディタやメモ帳で開いて、カウント情報を見ることができます。



# [カウント情報の表示例]

\*\*\*\*\*

\*                    カウント情報                    \*

COBOL2002 (X)    01-01                    \*\*\*\*\*                    2004-04-06 18:53:00

-----

プログラム名    : REIDA11

コンパイル日時: 2004-04-01 19:10:39

実行日時        : 2004-04-06 18:53:00

-----

プログラム名    : REIDA11

実行回数        -----

0001700 PROCEDURE DIVISION.            aa

0001701

0001900 Mein-Sec SECTION.

1    0002000        PERFORM    初期処理.

1    0002100        PERFORM    比較処理.

1    0002200        PERFORM    出力処理.

1    0002300        STOP    RUN.

0002500 初期処理 SECTION.

1    0002600        ACCEPT YYMMDD FROM DATE.

0002800 比較処理 SECTION.

1    0002900        IF    月    =    9

0003000        THEN

0    0003100        MOVE    'September!!' TO DATA2

0003200        ELSE

1    0003300        MOVE    'Not September!!' TO DATA2

0003400        END-IF.

0003600 出力処理 SECTION.

1    0003700        DISPLAY DATA0.

0003800

0003900

## 11. 終わりに

テストデバッグツールにおける基本的な使用方法是以上の説明で終わります。一通りのデバッグを行う場合には、今までの説明の機能だけで十分であると思います。

しかし、テストデバッグツール自身には、その他の機能も備わっていますので、それらをお知りになりたい方は、マニュアル「COBOL2002操作ガイド」を参照ください。

## 3. 関連資料

－よく使われる機能の操作方法－

## (a)ファイルの入出力処理

ー順ファイル等を使ったプログラムの実行ー

## － 目 次 －

1. はじめに
2. コーディング上の指定
3. 順ファイルの使用方法
4. プリンタへの出力方法
5. 索引ファイルの使用方法
6. 終わりに

# 1. はじめに

本説明書では、簡単なプログラム(ファイル等を使わないプログラム)の作成からコンパイル、実行までの使用方法是既に理解しているものとして、説明を進めます。

説明は、既にコンパイル&リンケージが終わり、実行ファイルが生成されているところから始めます。フラグ消しなどの実行ファイルの生成までは入門編他を参照ください。

通常、出力系で指定したファイルの実体が存在しない場合は、COBOL2002が、自動的にファイルを割り当てて生成します。入力・更新等で指定したファイルの実体がない場合は、実行時にエラーとなります。

なお、ここでは、ファイル等を用いたCOBOLプログラムを実行する上での基本的な使い方を説明します。その他の詳細な使用方法については、マニュアル「COBOL2002 操作ガイド」または、「COBOL2002 ユーザーズガイド」を参照ください。



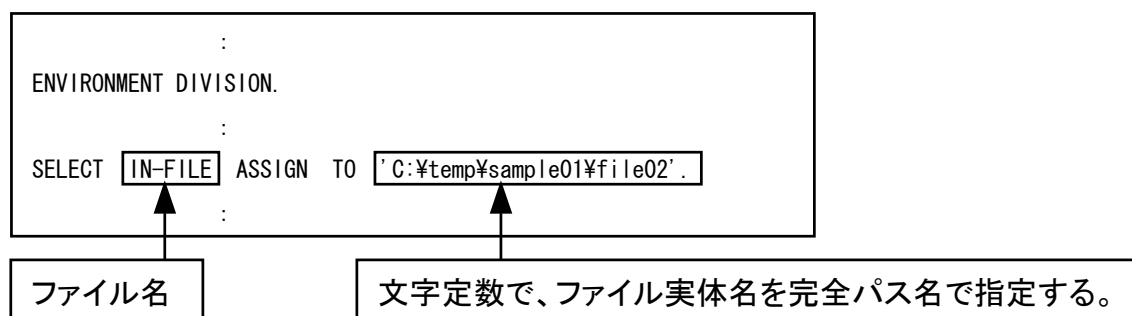
## 2. コーディング上の指定

まず、ソースプログラム上のファイルの指定方法を説明します。  
ファイルの割り当ては、プリンタに直接出力する場合などの特別な場合を除いて、主に2つの方法があります。

- ① ソースプログラム中に、直接「ファイル実体名」を指定する方式
- ② 外部装置名を指定して、プログラムを実行するときに実行時環境変数でファイル実体と結びつける方式

[ソースプログラム中に直接「ファイル実体名」を指定する方法]

ASSIGN句に、文字定数でファイル実体名(完全パス名)を指定します。パス名を省略すると、実行可能ファイルがあるパスが仮定されます(実行可能ファイルと同じフォルダにファイルがある場合はパス名を省略できます)。

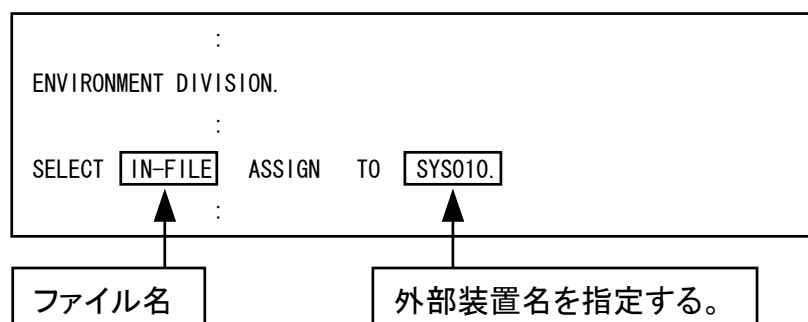


ファイル名を間違えるとプログラム実行時に、エラーとなります。(コンパイル時にはチェックされないので、誤った名称であってもエラーにはなりません。)

この指定方法では、3章以降で説明していくファイルの割り当て方法は必要ありません。

[外部装置名を指定する方法]

ASSIGN句に「SYS010」などの外部装置名を指定します。



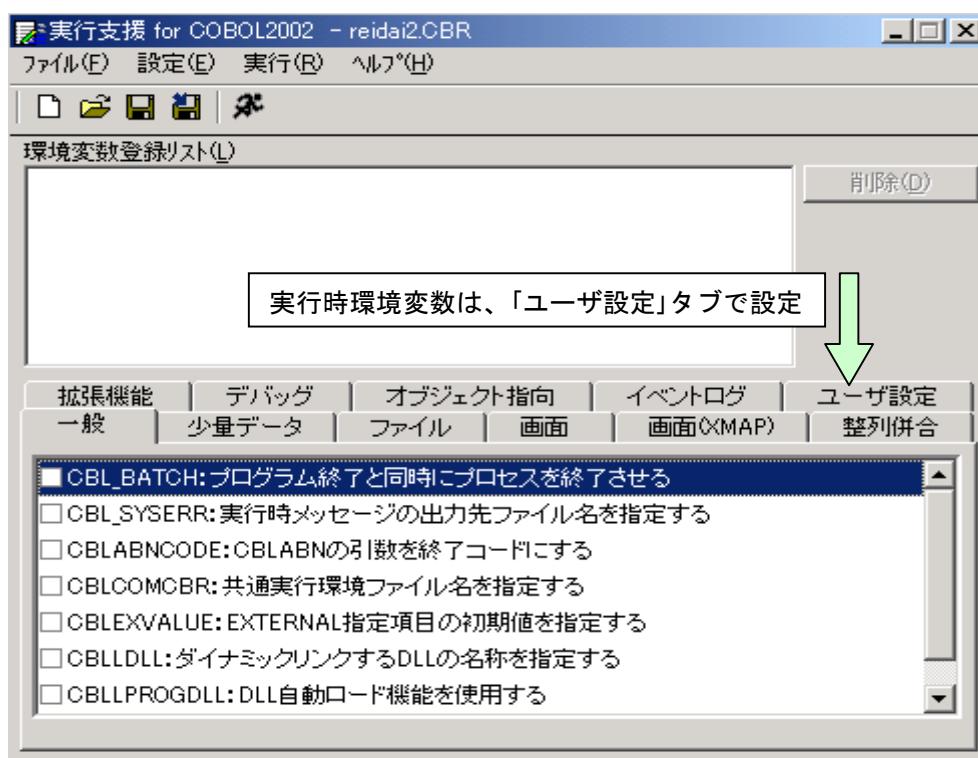
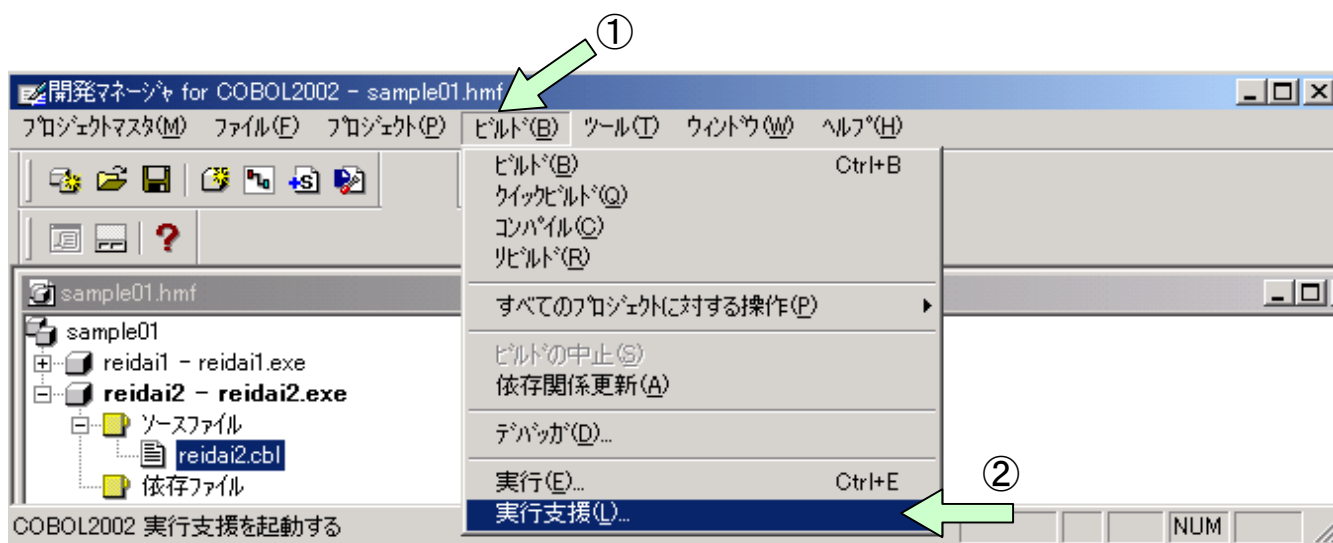
外部装置名は英数字の名称で「SYSXXX」などが一般的に使用されます。

この指定方法では、3章以降で説明するファイルの割り当て方法によりファイル実体と結びつけます。

### 3. 順ファイルの使用方法

COBOLプログラム中に、外部装置名「SYS010」を指定したものとして説明します。また、コンパイル&リンケージが終わった状態から操作方法を説明します。

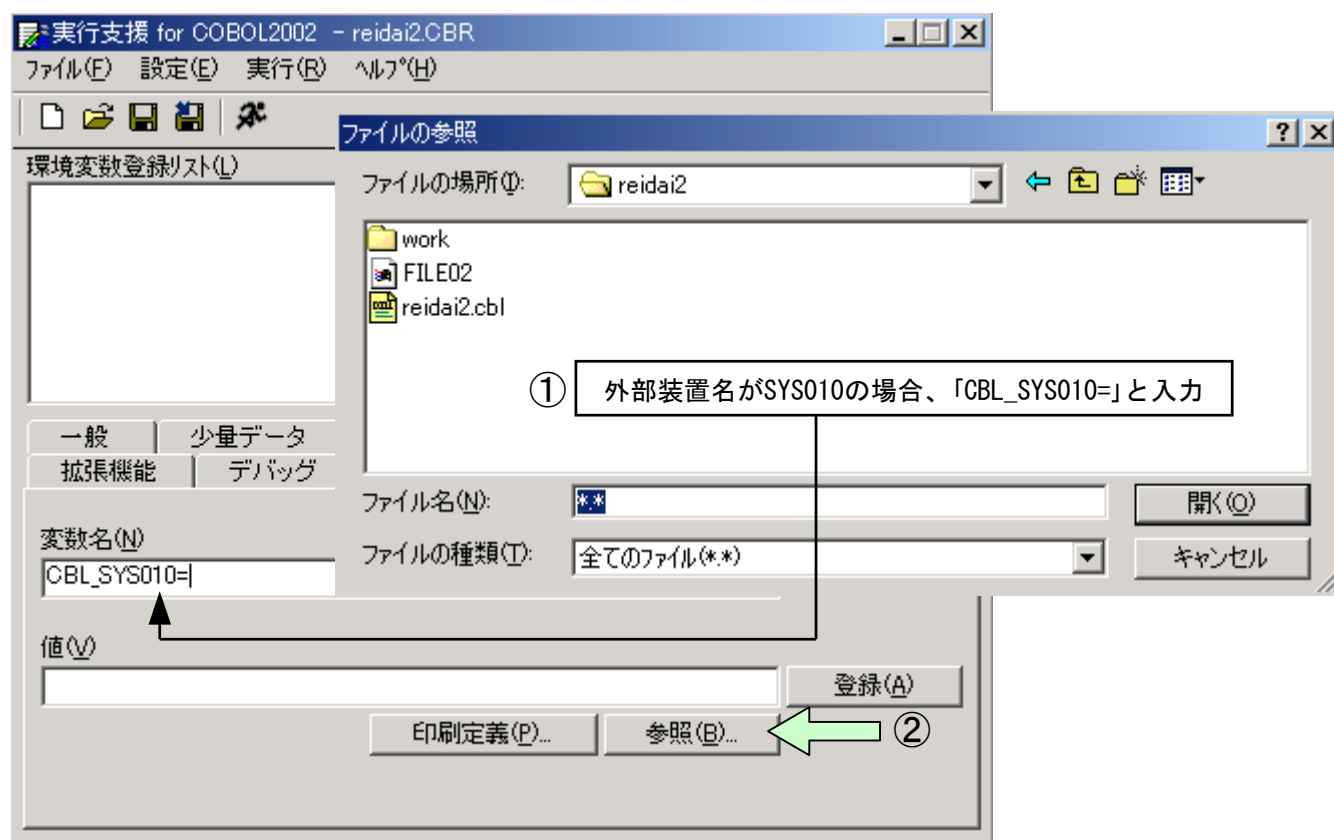
[手順1] 開発マネージャのメニューバーの「ビルド(B)」をクリックし、プルダウンメニューの中から「実行支援(L)」をクリックします。



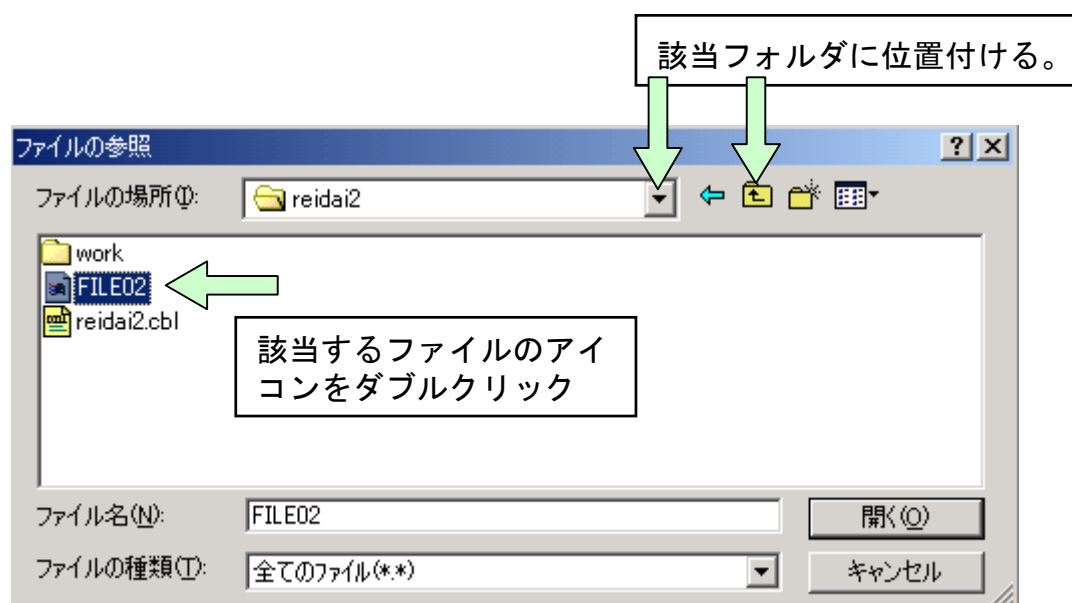
#### [用語解説] COBOL2002実行支援

COBOL2002実行支援とは、プログラムの実行に必要な環境設定をするツールで、ファイルの割り当てやプリンタに対する印刷書式の設定などを行います。

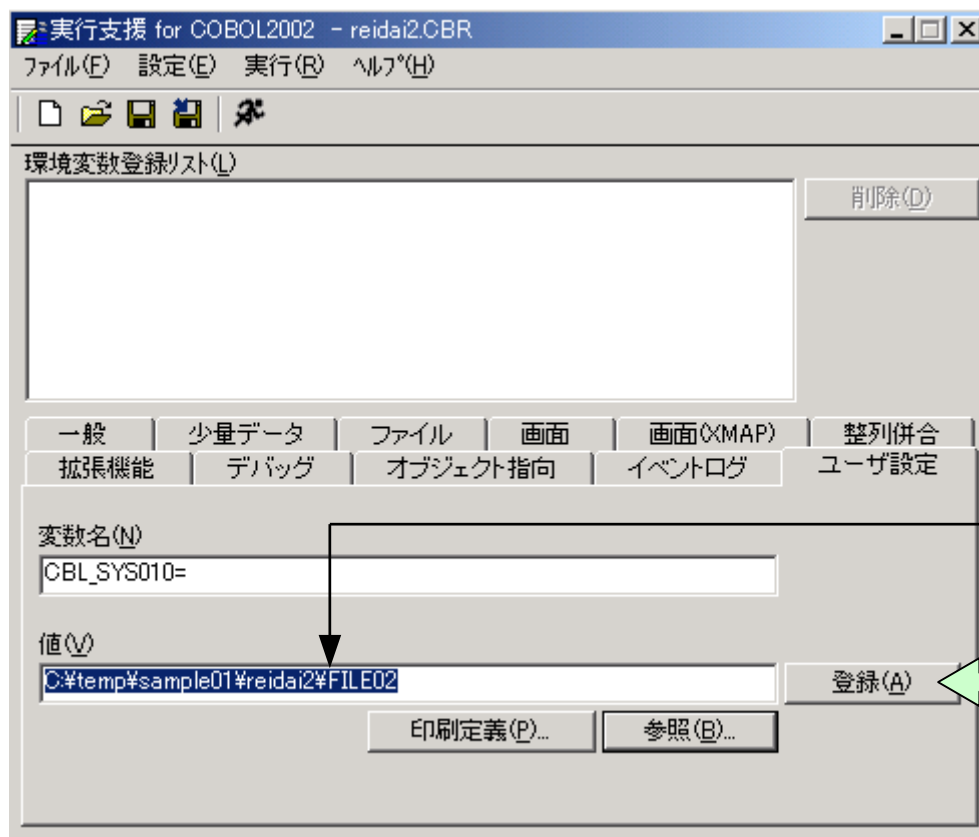
[手順 2] 実行支援画面の中の「ユーザ設定」タブをクリックし、変数名の入力エリアに「CBL\_外部装置名=」と半角で入力します。その後、参照ボタンをクリックします。すると、ファイルの参照画面が出ます。



[手順 3] 必要であればフォルダの表示を調整して該当するファイルを探し、ファイルのアイコンをダブルクリックします。



[手順 4] COBOL2002実行支援の画面に戻り、ファイルが完全パス指定で表示されます。ここで、登録ボタンをクリックください。



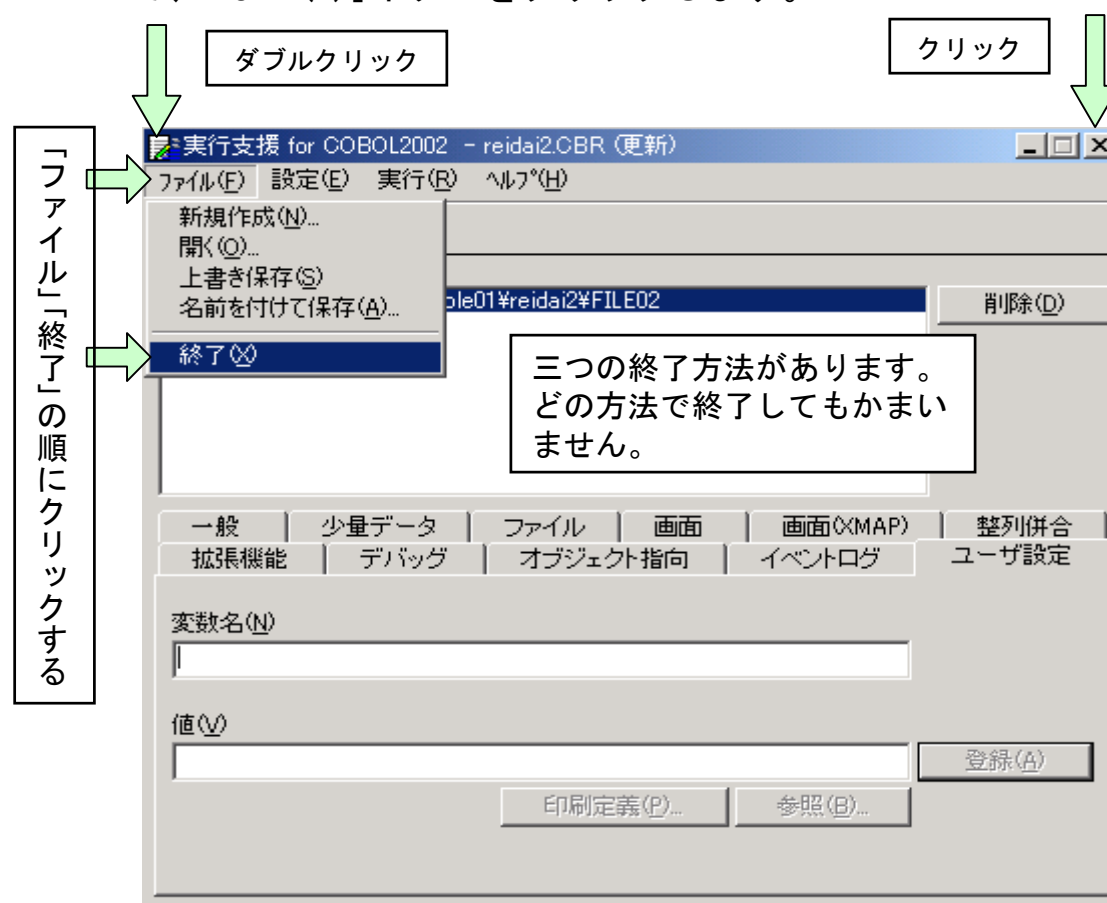
登録ボタンをクリックすると環境変数登録リスト上に登録されます



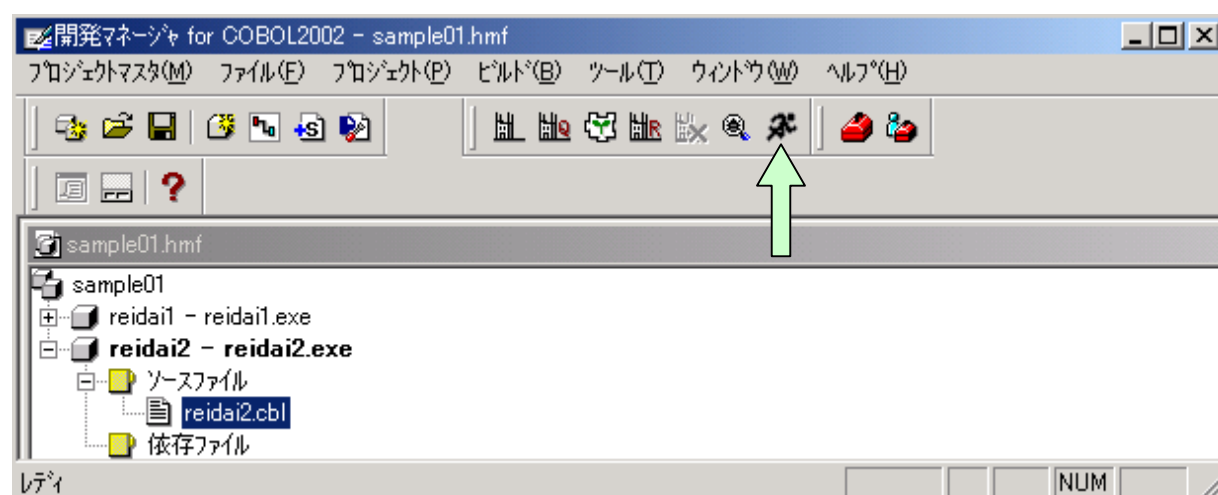
### [ワンポイントアドバイス]

- ①ファイル実体名が拡張子付きの場合は、拡張子も含めて指定してください。  
 〈例〉CBL\_SYS010=C:\temp\sample01\reidai2\FILE02.dat
- ②本例題ではファイル実体名は完全パス名で指定しましたが、実行可能ファイル(.exe)と同じフォルダの中にファイルの実体があるときは、最後のファイル名だけですみます。  
 〈例〉CBL\_SYS010=FILE02
- ③新規に作成するファイル(OPEN OUTPUTのファイル)の場合、ファイル実体が存在しなくてもかまいません。この場合、参照ボタンで該当フォルダに位置付けて、ファイル名の欄にファイル名を入力し、「開く(O)」ボタンをクリックします。実行可能ファイル(.exe)と同じフォルダに作成する場合は、ファイル名だけを入力します。

[手順5] COBOL2002実行支援を終了します。「保存しますか?」と聞いてくるので、「はい(Y)」ボタンをクリックします。



[手順6] 開発マネージャに戻って、「実行」ボタンを押します。

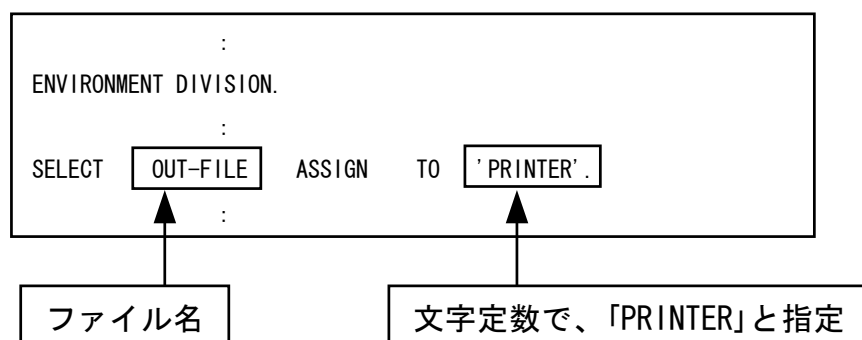


## 4. プリンタへの出力方法

COBOLプログラムから印刷データをプリンタに出力する方法を説明します。

[印刷データを直接プリンタに出力する方法]

2章で説明したように、ソースプログラム中で指定することができます。次に示すように、外部装置名の所に文字定数で「PRINTER」と指定します。



[印刷データを一旦ファイル上に出力する方法]

プリンタに直接出力するのではなく、一旦ディスク上に出力する方式も考えられます。この場合は、順ファイルに出力するのと同じやり方でできます。順ファイルの出力内容が正しいことを確認したら、COBOLエディタ等を使ってディスク上のファイルを開いて印刷してください。

## 5. 索引ファイルの使用方法

索引ファイルの割り当ては、順ファイルと同様に次のどちらかで行います。

- ・ ソースプログラム中のASSIGN句の外部装置名に文字定数でファイル実体名を指定する。
- ・ 外部装置名に「SYSXXX」を指定しておき、COBOL2002実行支援を起動して、「CBL\_SYSXXX=ファイル実体名」を指定する。

順ファイルと異なるのは、次の点です。

- ・ 順ファイルは単独のファイルですが、索引ファイルは3種類のファイルで構成されます。副キーは、最大98個まで指定可能です。
  - ①キー定義ファイル(.kdf)：キーとデータの対応を示す。
  - ②キーファイル(主キー：.k01, 副キー：.k02～.k99)
  - ③データファイル(.drf)
- ・ 順ファイルでは拡張子を含めてファイル実体名を指定しましたが、索引ファイルの場合拡張子は指定しません。

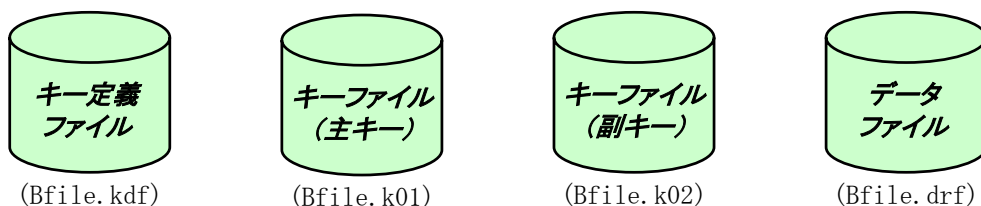
＜COBOLソースの例＞

```
FD B-FILE ASSIGN TO 'C:¥temp¥sample01¥Bfile'
```

拡張子は指定しません。

索引ファイル(副キーが1個の場合)の例

次の4つのファイルが生成されます。



### 【ワンポイントアドバイス】

順ファイルは任意の拡張子を付けることができます。また、拡張子がなくともかまいません。拡張子を付ける場合は、ファイルの割り当て時に拡張子を含めて指定します。索引ファイルは、拡張子が決められています。ファイルの割り当てをする際は、上記の例のように拡張子を除いて指定します。

## 6. 終わりに

ファイルを使用したプログラムの実行においては、ソースに直接ファイルの実体名を記述する方法と、実行支援でファイル実体と結びつける方法があることを説明しました。また、プリンタに直接出力する方法も説明しました。

しかし、プログラムを実行するにあたっては、更に詳細な設定が必要な場合もあります。

不明な点がありましたら、マニュアル「COBOL2002 操作ガイド」または「COBOL2002 ユーザーズガイド」をご参照ください。



## (b)テスト データの作成方法

ー テストデータを容易に作成するためにー

## 一 目 次

1. はじめに
2. COBOLエディタの起動方法
3. データの入力方法
4. COBOLエディタの終了
5. 改行コードの表示方法

# 1. はじめに

本説明書では、テキストファイルのデータの作成方法を説明します。

テキストファイルと順ファイルはほとんど同じ形式をしています。改行コード付きのデータをテキストファイルと呼んでいます。改行コードまでを1レコードとして扱います。テキストファイルのデータはCOBOLエディタやメモ帳で作成できます。

順ファイルとテキストファイルは、ソースプログラム上の定義が次のように異なります。順ファイルの場合、「ORGANIZATION」句は省略できます。

## 順ファイル指定方法

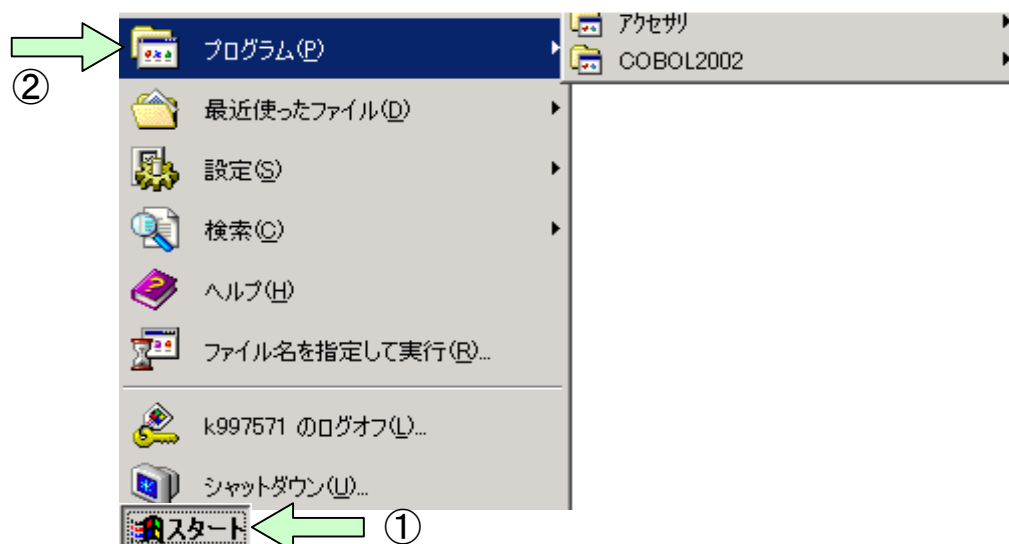
```
SELECT   ファイル名  ASSIGN TO  SYS001  
          ORGANIZATION IS SEQUENTIAL.
```

## テキストファイル指定方法

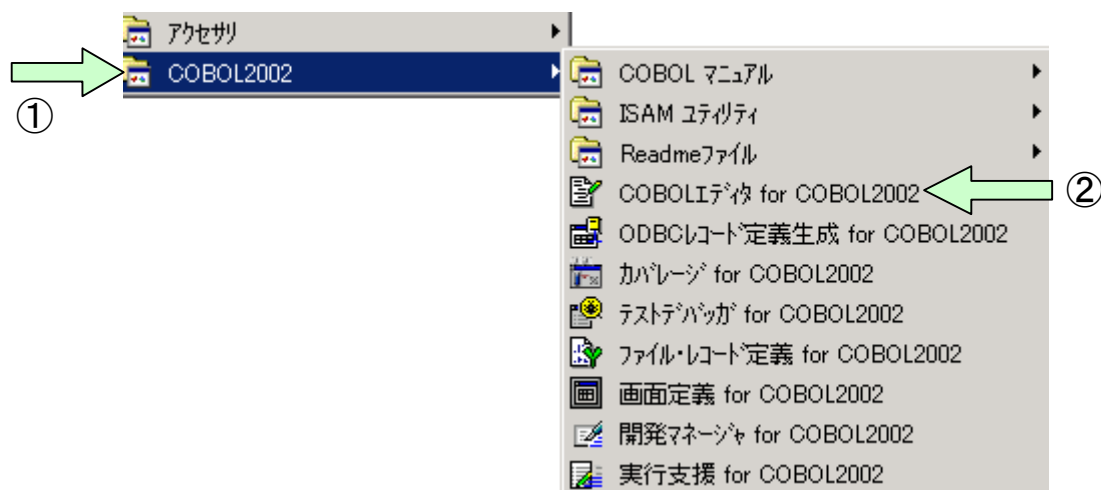
```
SELECT   ファイル名  ASSIGN TO  SYS001  
          ORGANIZATION IS LINE SEQUENTIAL.
```

## 2. COBOLエディタの起動方法

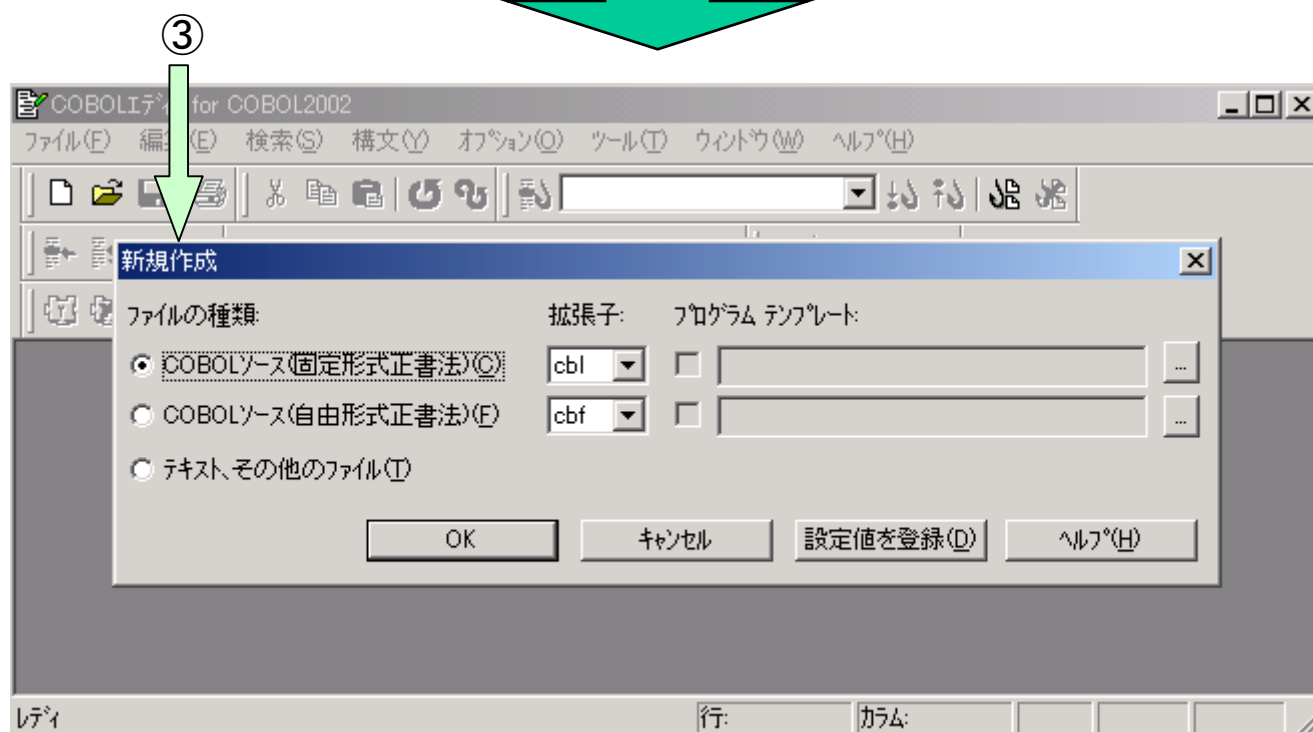
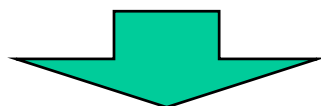
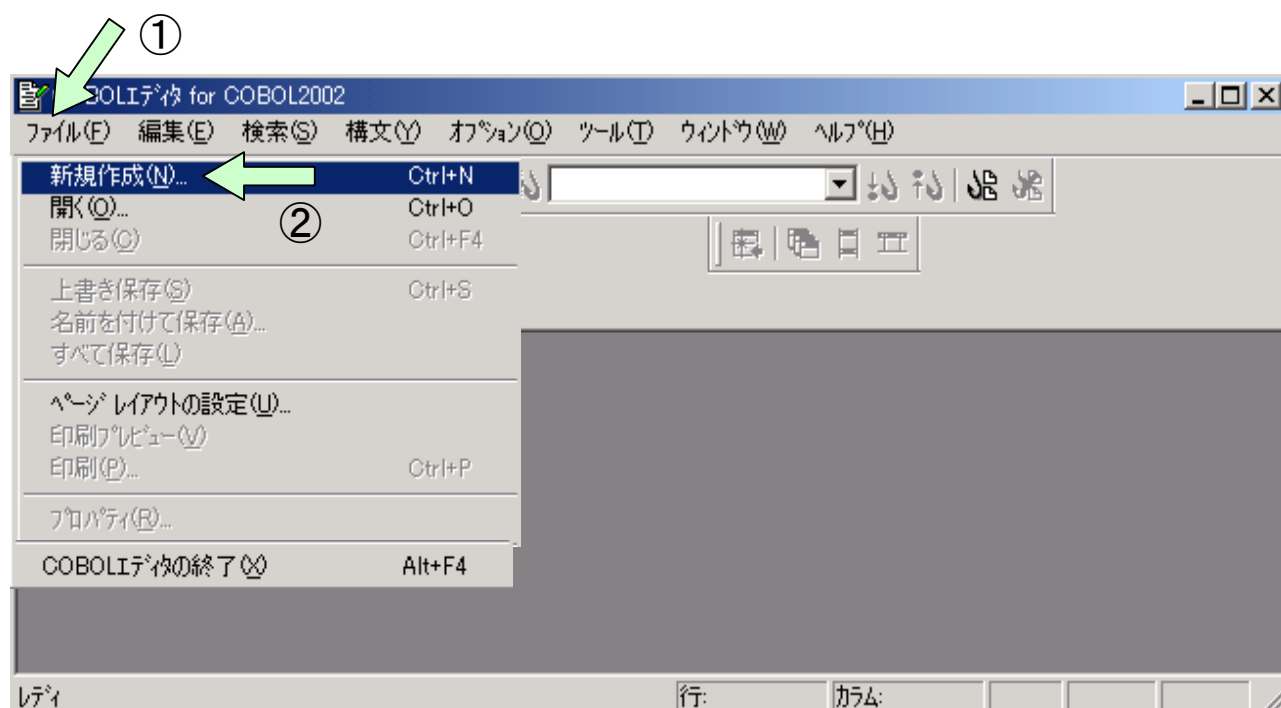
[手順 1] スタートボタンを押し (①)、「プログラム」の所にマウスポインタを移動します (②)。すると起動できるプログラムの一覧が出てきます。



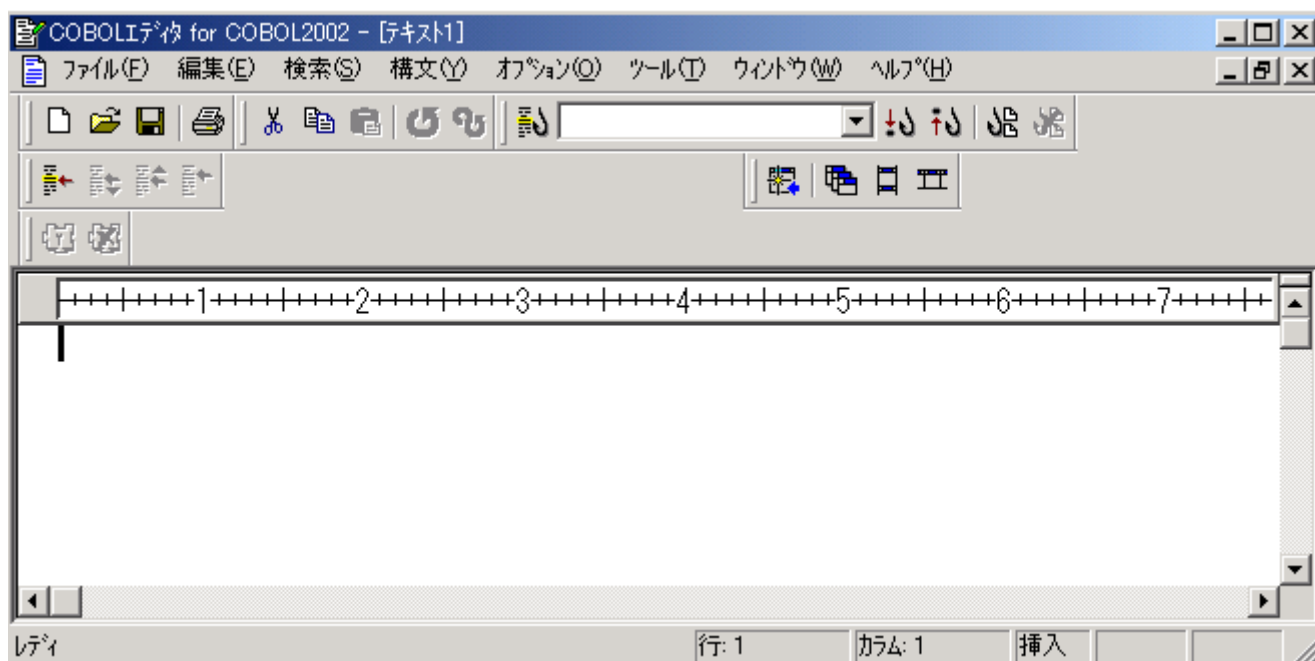
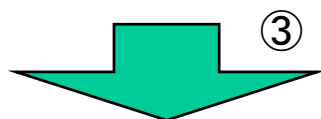
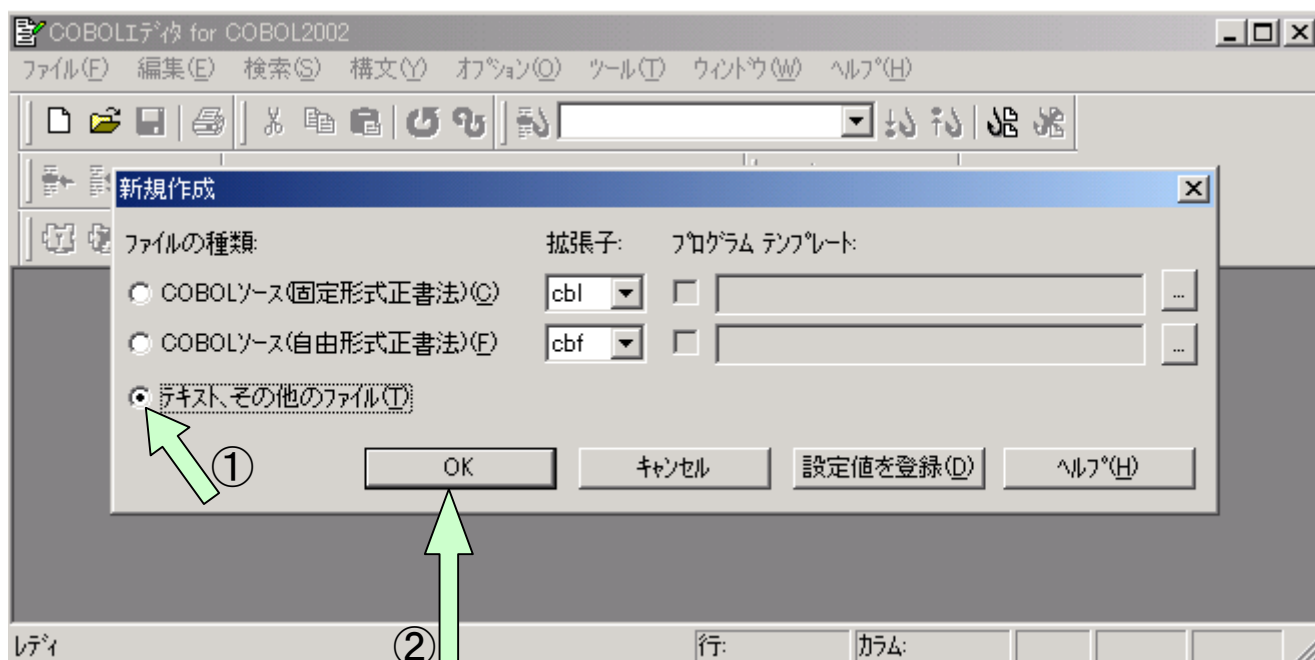
[手順 2] プログラムの一覧の中からCOBOL2002の所にマウスポインタを移動します (①)。メニューから「COBOLエディタ」を選択します (②)。



[手順3] 起動されたCOBOLエディタ画面より「ファイル(F)」-「新規作成(N)」の順に選択します(①-②)。すると「新規作成画面」が表示されます(③)。



〔手順 4〕 ファイルの種類を「テキスト、その他のファイル(T)」にして①  
「OK」ボタンを押す②と、新規にテキストファイルが表示され  
ます③。

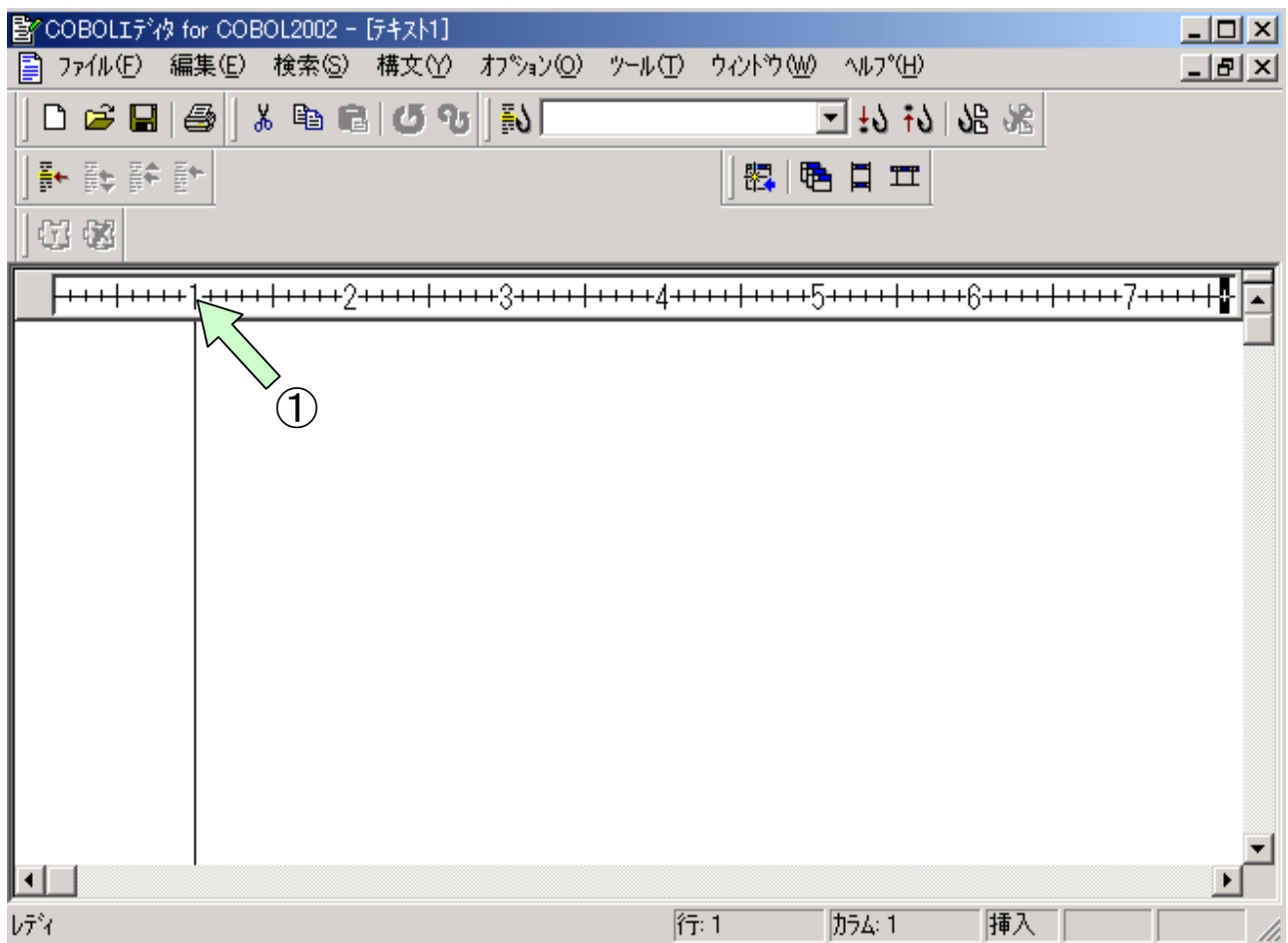


### 3. データの入力方法

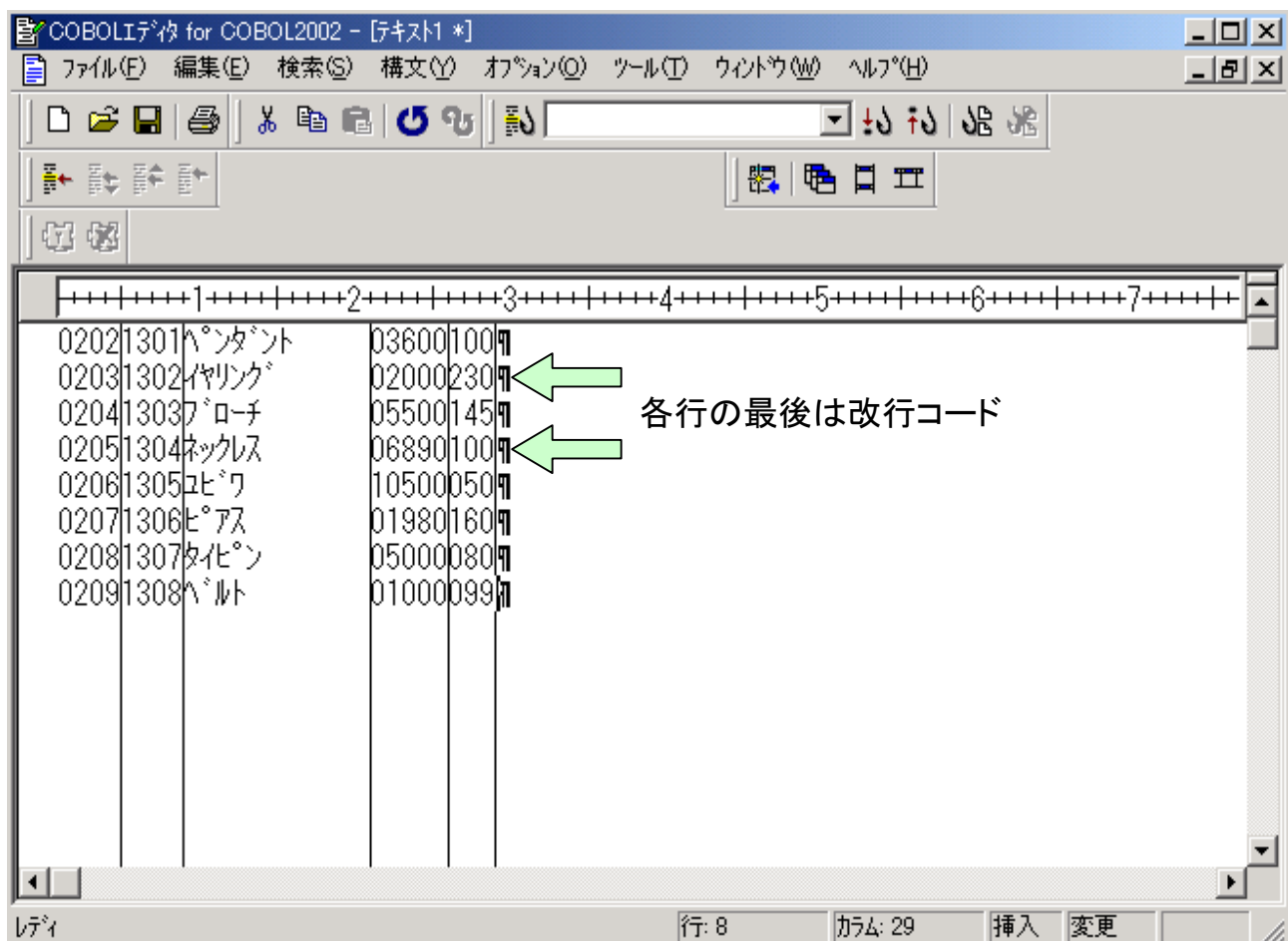
[手順 1] エディタ画面上のカラム目盛りの任意の位置をクリックすると補助線が引かれます (①)。

※ 補助線を消去するには、補助線のカラム目盛りの位置をダブルクリックします。

※ 補助線は保存されません。



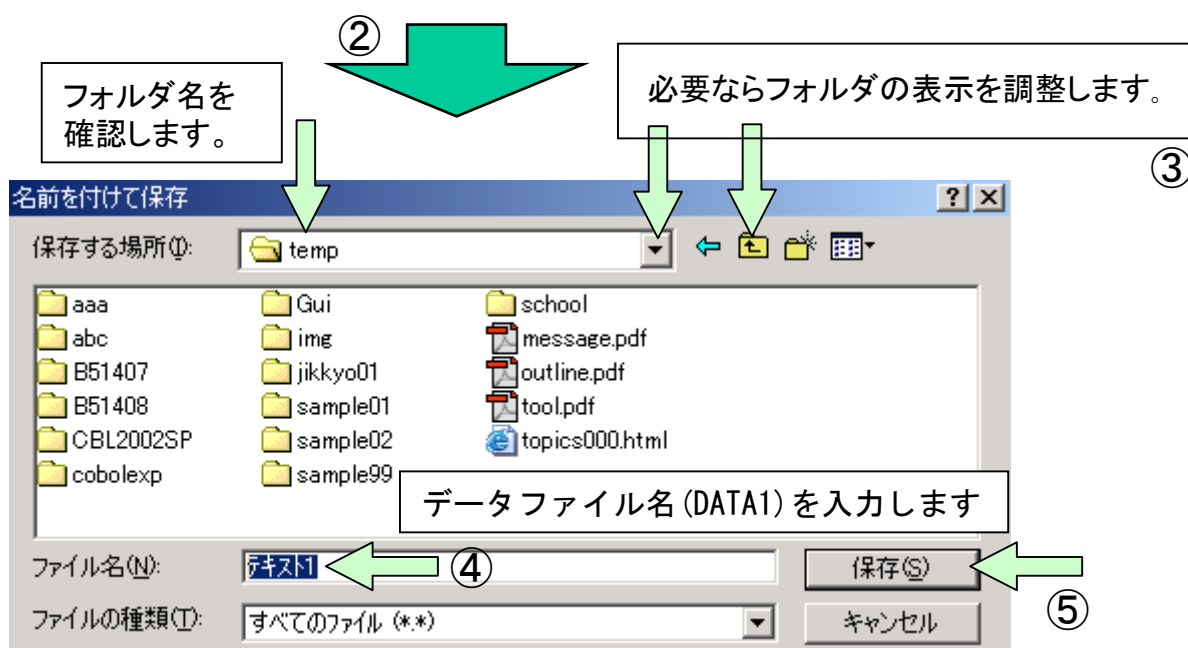
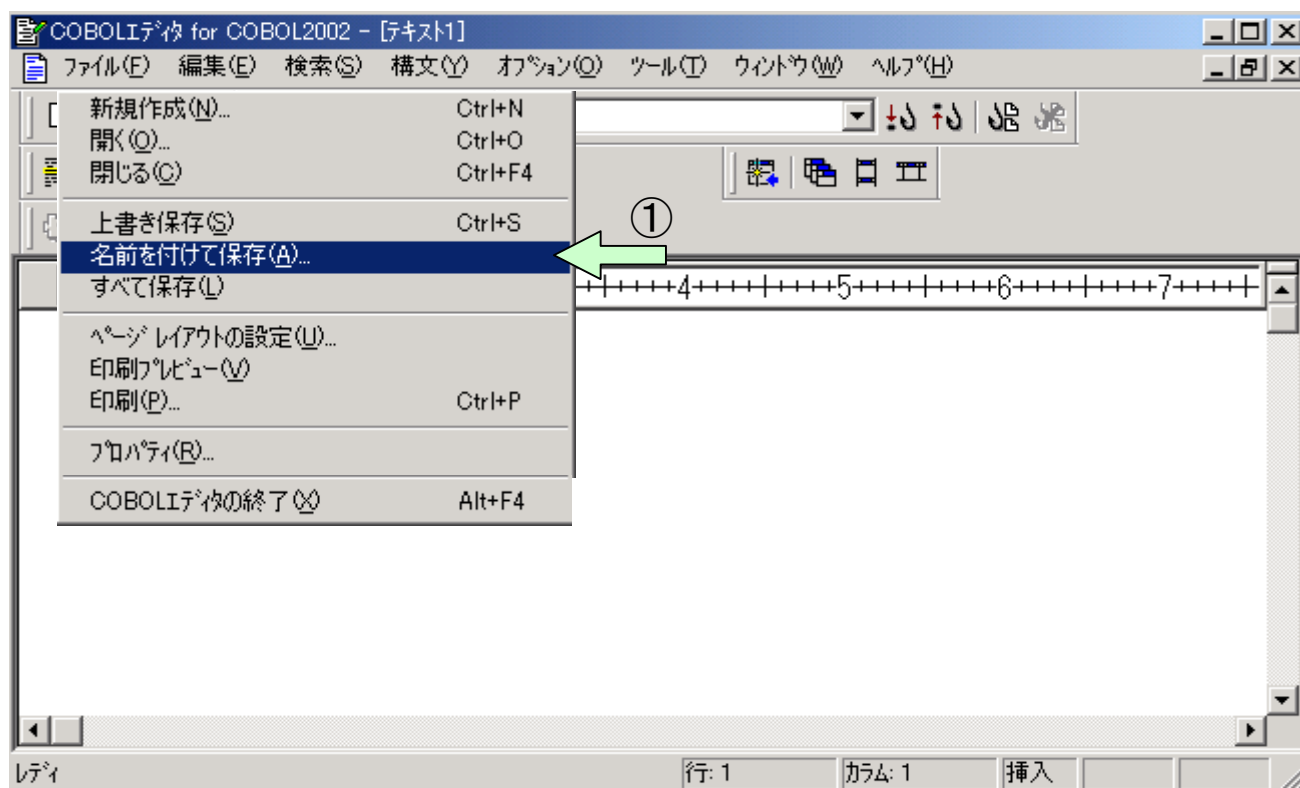
〔手順2〕 データをキーインします。各データの最後は改行します。





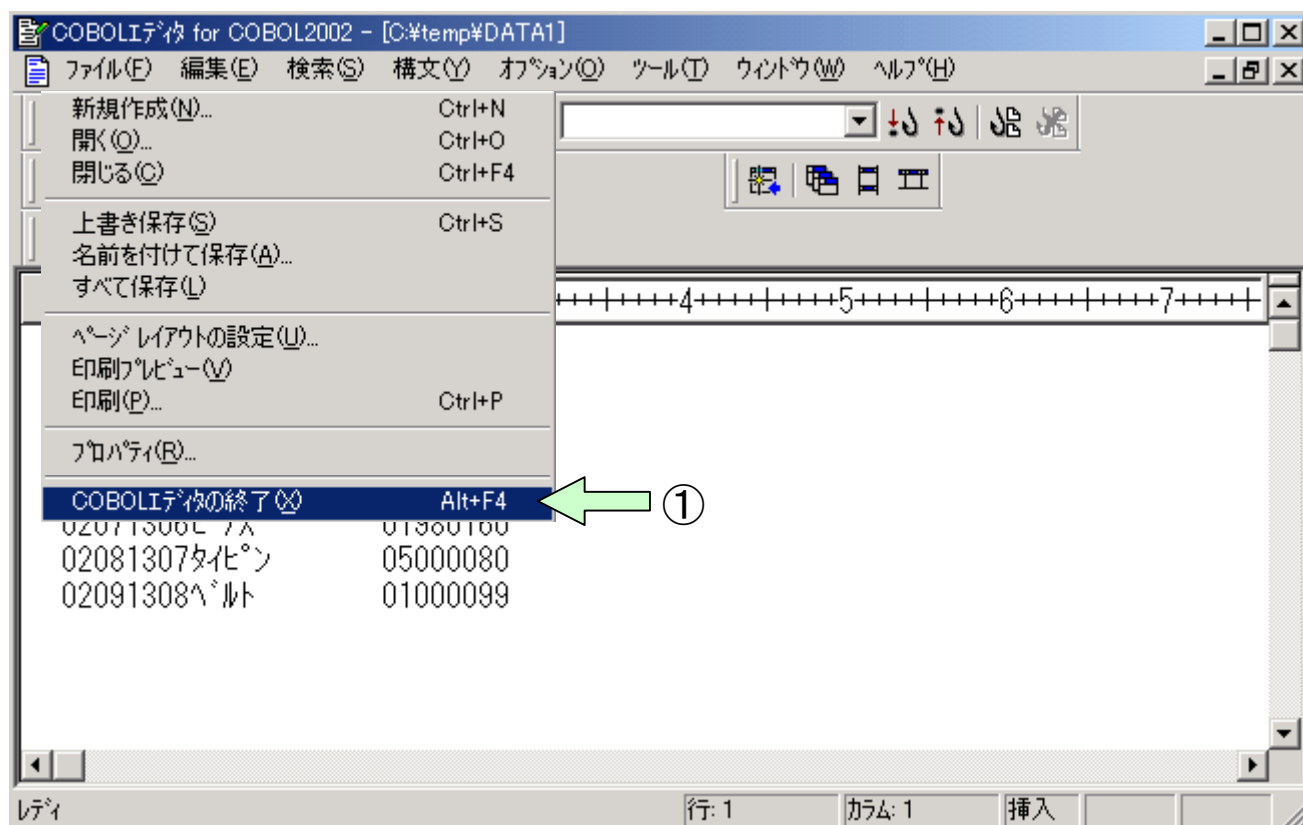
[手順3] 名前を付けてデータをセーブします。

- ・「ファイル(F)」をクリックして、プルダウンメニューから「名前を付けて保存(A)」をクリックします(①)。
- すると、「名前を付けて保存画面」が表示されます(②)。
- ・必要ならフォルダの表示を調整して、格納したいフォルダに位置づけます(③)。
- ・フォルダに位置付いたらデータファイル名を指定します(④)。
- ・「保存(A)」ボタンをクリックします(⑤)。



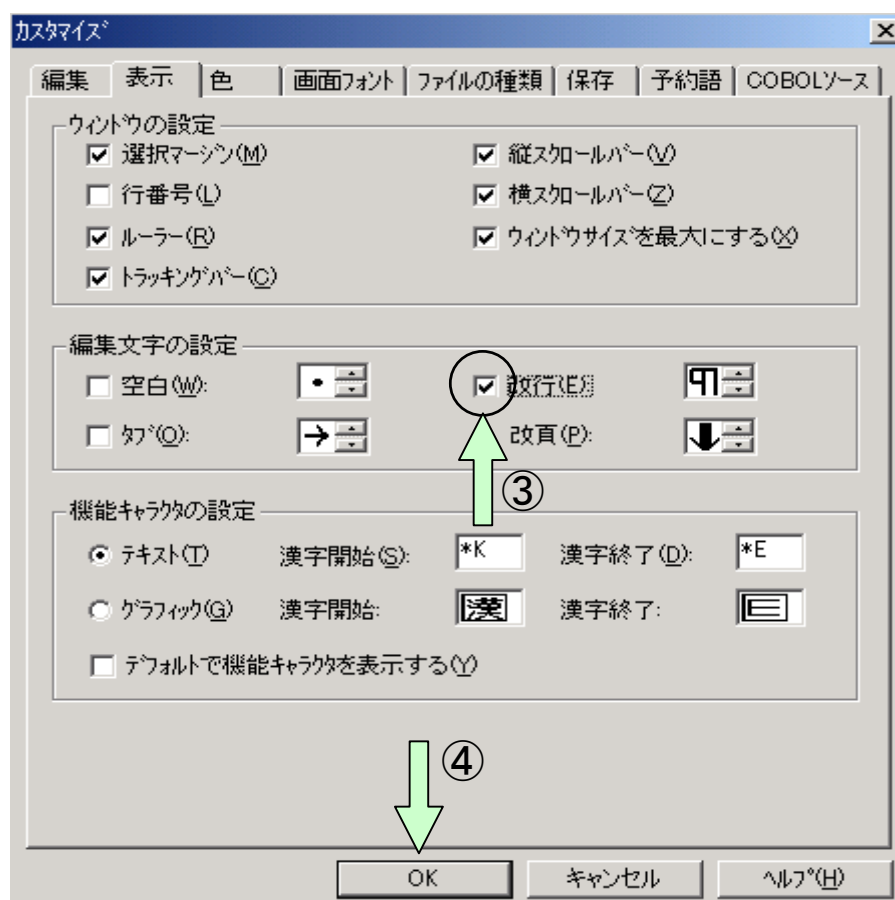
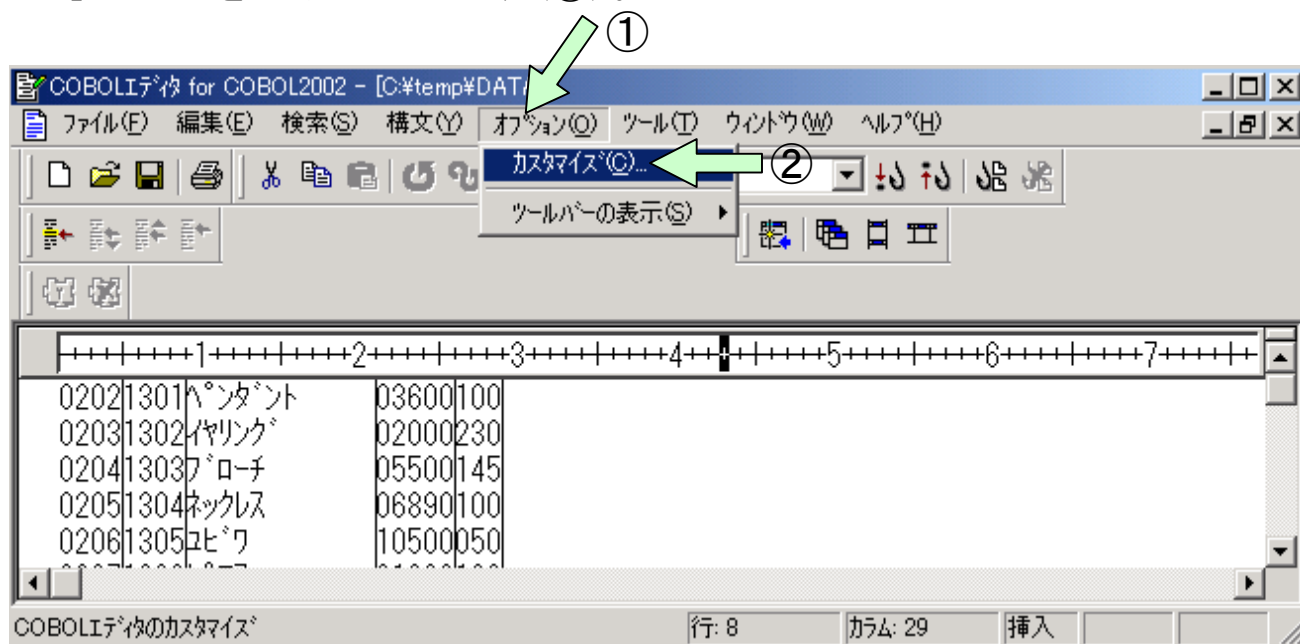
## 4. COBOLエディタの終了

「ファイル(F)」をクリックして、プルダウンメニューから「COBOLエディタの終了(X)」をクリックします(①)。



## 5. 改行コードの表示方法

- ・エディタのメニューバーの「オプション(O)」をクリックします(①)。
- ・プルダウンメニューから「カスタマイズ(C)」をクリックします(②)。
- ・「カスタマイズ画面」出たら、「表示」タブ中の「改行」をクリックします(③)。
- ・「OK」ボタンをクリックします(④)。



## (c) 印刷用紙の節約方法

－ 実行結果の確認と印刷方法 －

## 1. はじめに

実行結果は、通常直接プリンタに出力して確認しますが、この方法では用紙の使用量が多くなることがあります。下記のように、一旦ファイルに出力して、出力結果が正しいことを確認してからプリンタに出力する手順にすると、用紙を節約することができます。

- ・ 実行結果をファイルに出力します。
- ・ COBOLエディタを使用して出力結果を確認してから印刷します。

## 2. ソースプログラムの変更

次に示すように任意のフォルダのファイル名を指定します。

(ファイル実体が存在しなくても、指定されたファイルが作成されます)

直接プリンタに出力するときの指定

```
SELECT ファイル名 ASSIGN TO 'PRINTER'.
```

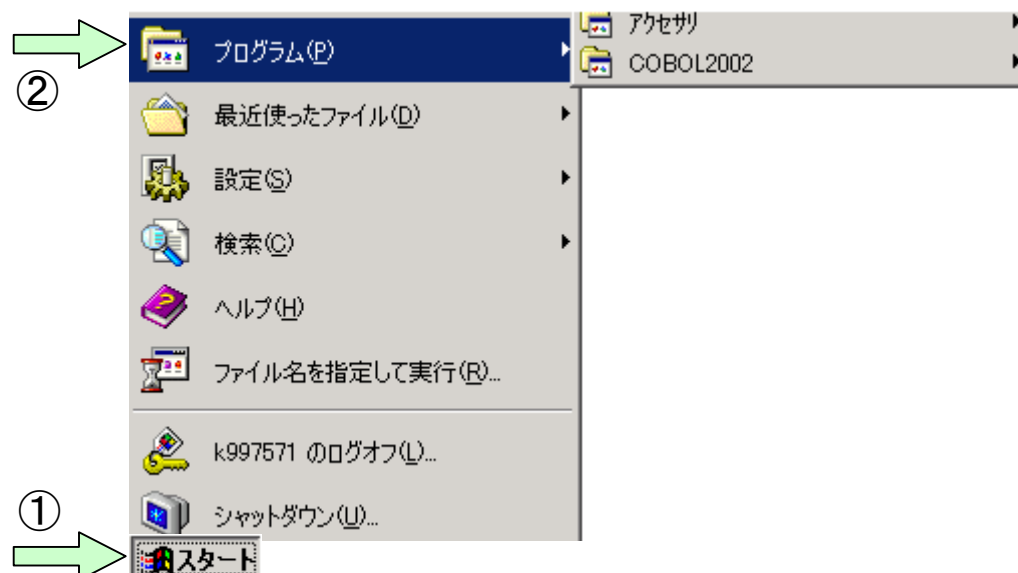


ファイルに出力するときの指定

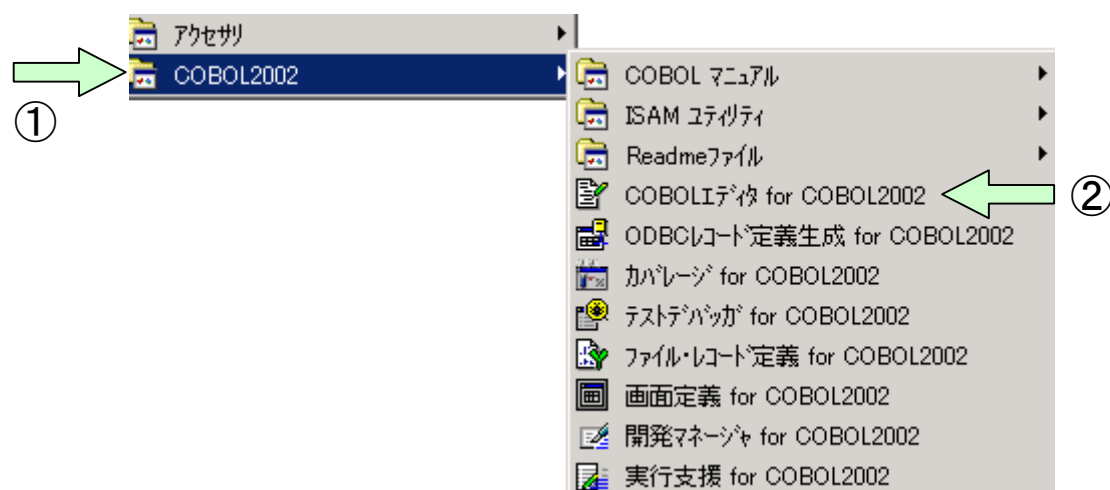
```
SELECT ファイル名 ASSIGN TO 'C:¥Temp¥data¥OUTFILE'.
```

### 3. COBOLエディタの起動と印刷方法

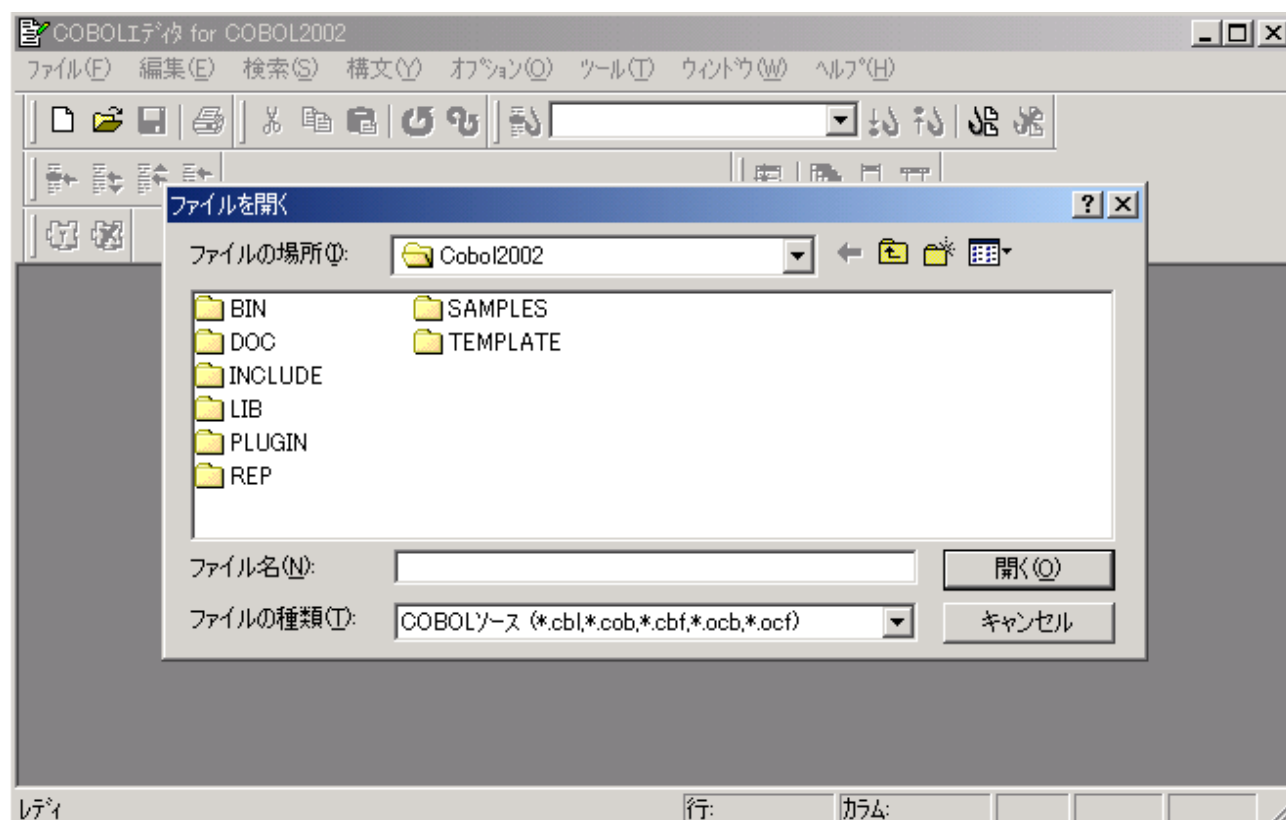
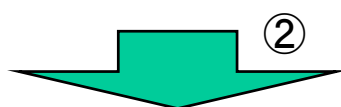
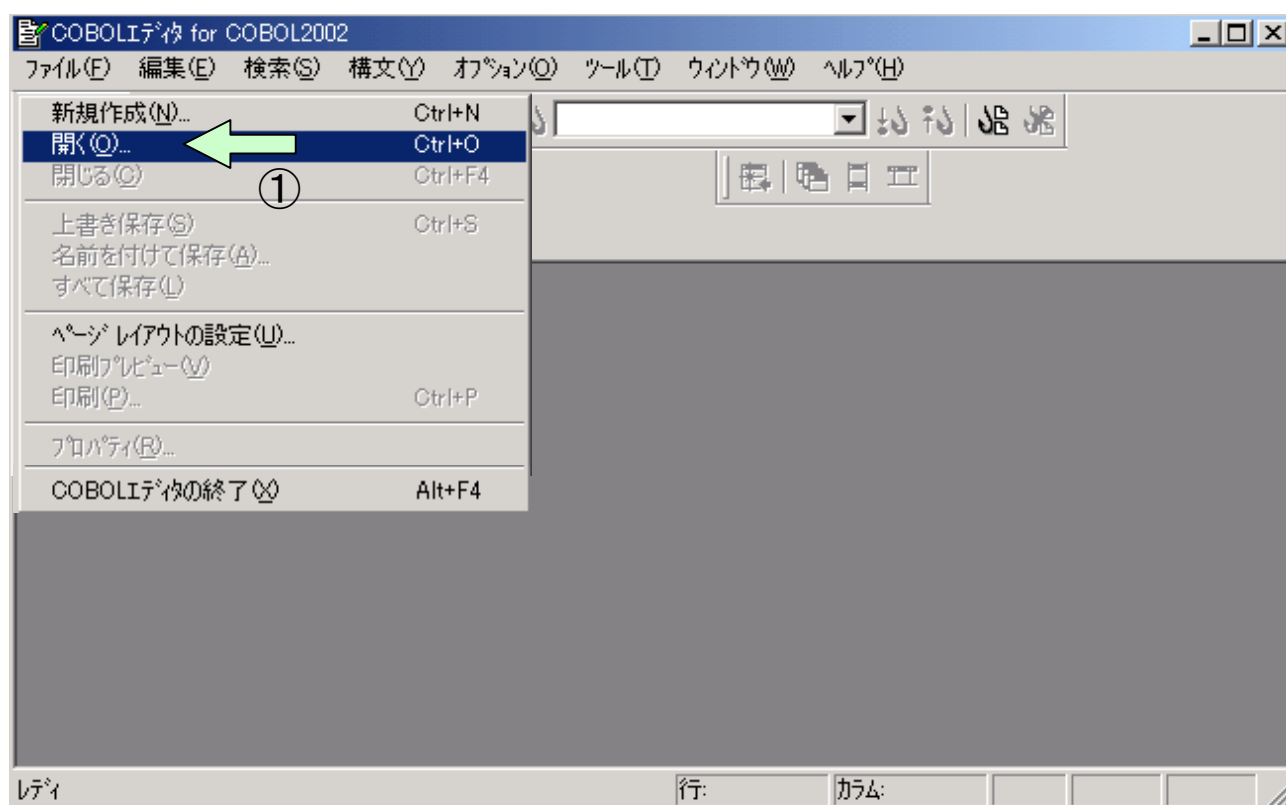
[手順 1] スタートボタンを押し (①)、「プログラム (P)」の所にマウスポインタを移動します (②)。  
すると起動できるプログラムの一覧が表示されます。



[手順 2] プログラムの一覧の中から「COBOL2002」の所にマウスポインタを移動します (①)。プルダウンメニューから「COBOLエディタ」を選択します (②)。



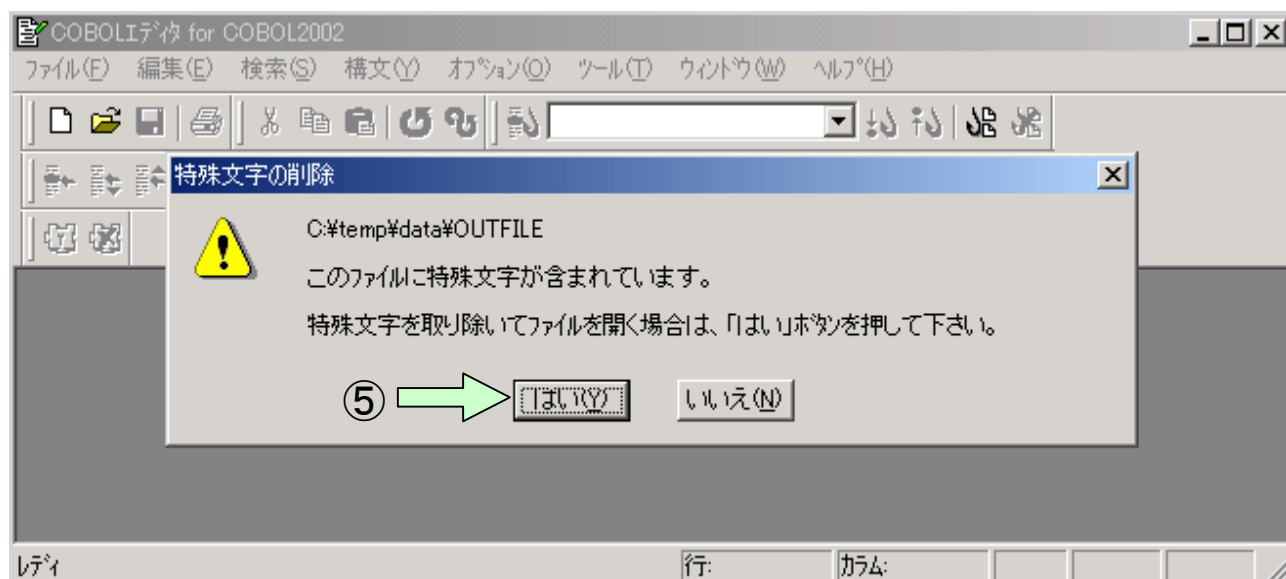
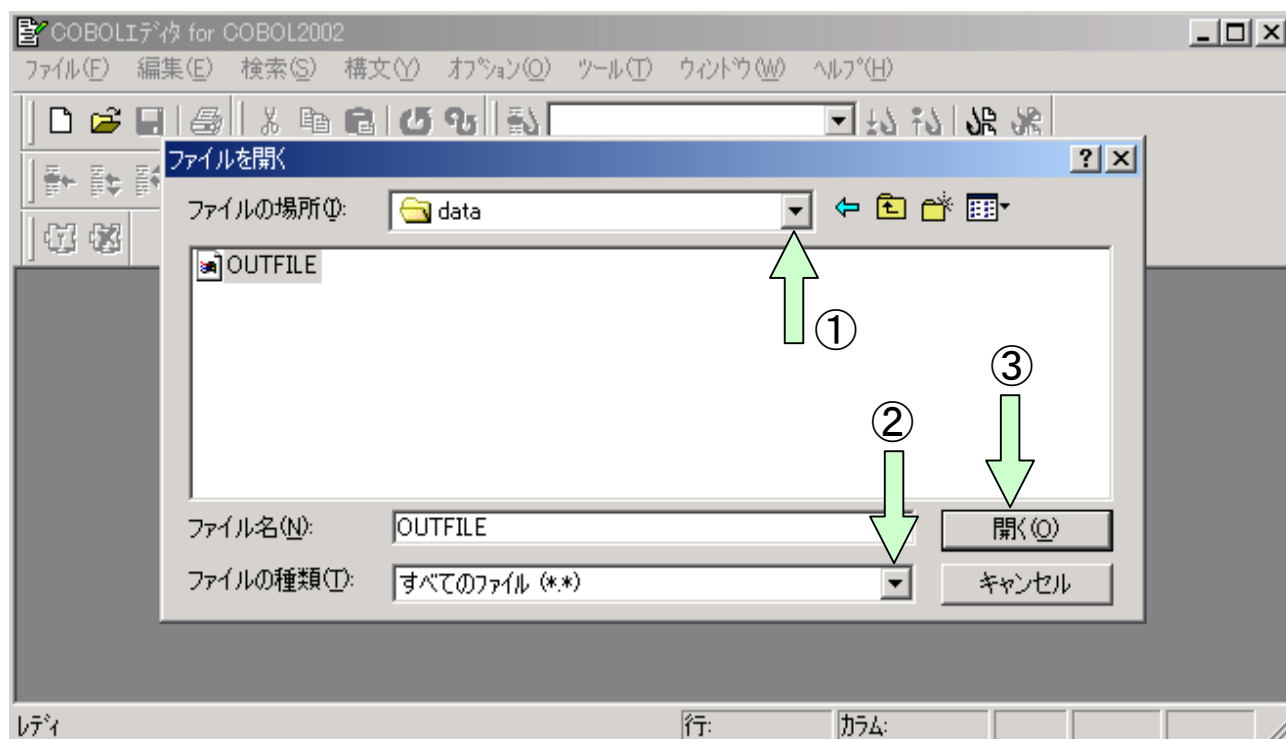
- [手順 3] 出力ファイル(C:\¥temp¥data¥OUTFILE)の表示  
 起動されたCOBOLエディタ画面より「開く(O)」を選択します(①)。  
 すると「ファイルを開く」画面が表示されます(②)。



# [手順 4] 「ファイルを開く」画面から

- ①「ファイルの場所」で該当フォルダ (C:\temp\data) を指定します。
- ②「ファイルの種類」で「すべてのファイル (\*.\*)」を選択し、表示されたファイルの一覧から「OUTFILE」を選択します。
- ③「開く (O)」ボタンをクリックします。
- ④ファイル中に特殊文字が含まれていると、「特殊文字の削除」画面が表示されます。
- ⑤「はい (Y)」をクリックすると、「OUTFILE」が表示されます。

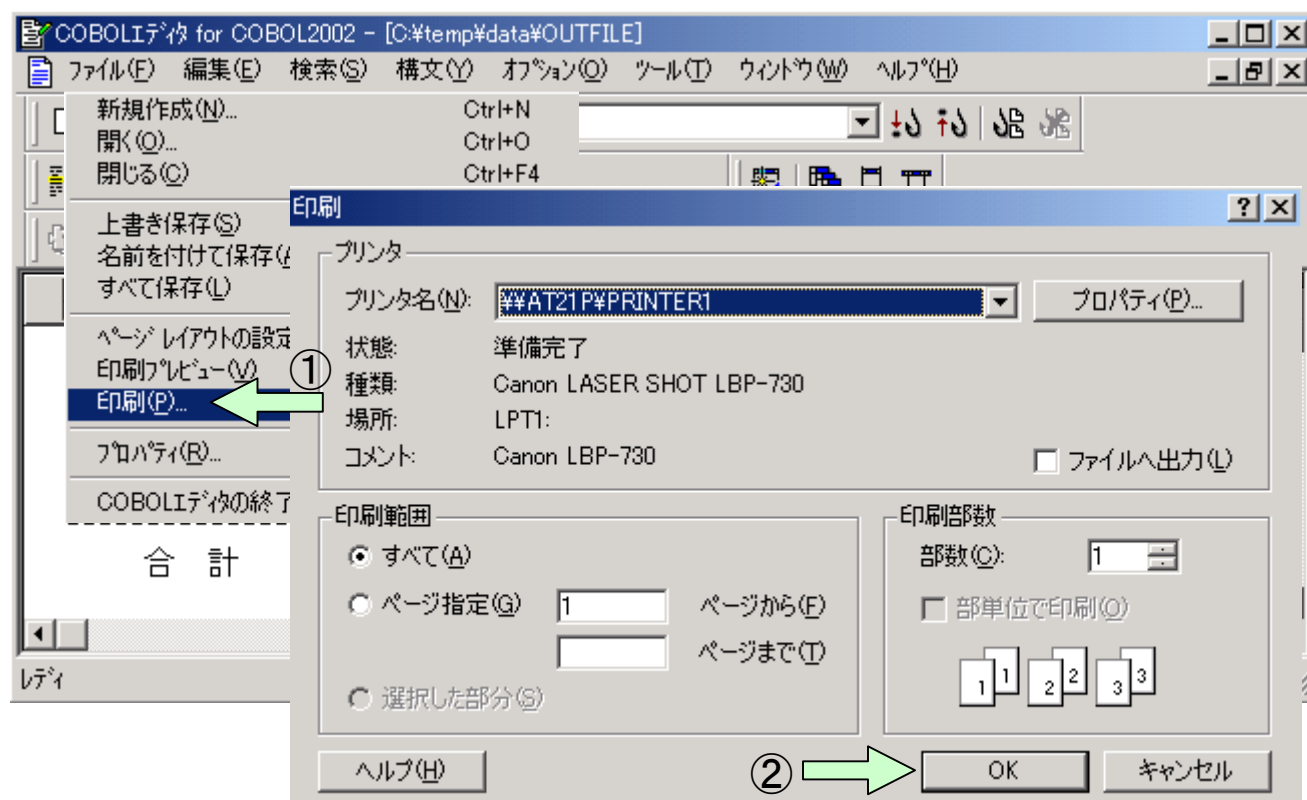
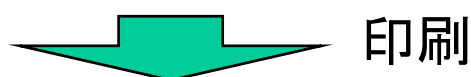
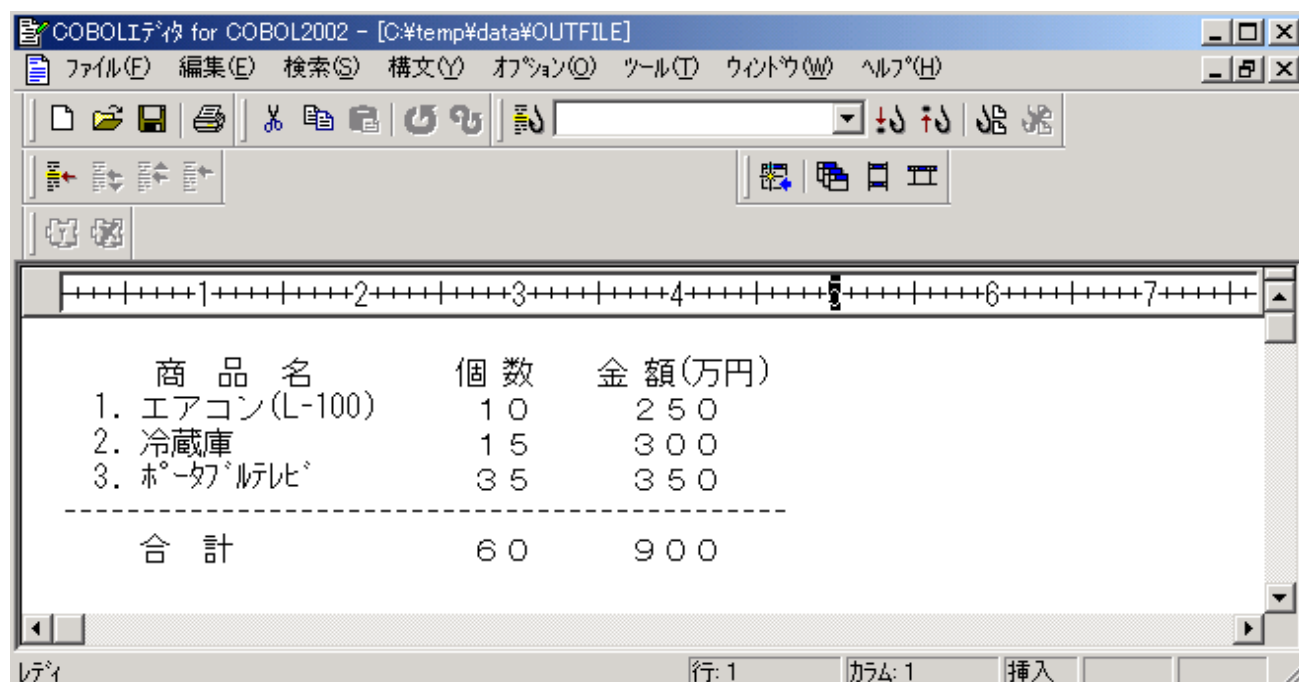
\* 印刷ファイルは改行コード (特殊文字) が含まれるため「特殊文字の削除」画面が表示されますが、ファイルの表示内容には影響ありません。





〔手順5〕 表示された「OUTFILE」の内容が正しい場合は印刷します。

- ① エディタのメニューバーの「ファイル(F)」をクリックし、プルダウンメニューの中から「印刷(O)」を選択すると「印刷」画面が表示されます。
- ② 「印刷」画面の「OK」ボタンをクリックします。



## (d) 印刷書式の設定方法

ーはじめにー

プリンタへ出力するときの印刷書式の設定は、開発マネージャから「実行支援」ツールを起動して設定します。

### [ワンポイントアドバイス]

プリンタ出力時に次のエラーが出力されることがあります。

KCCC3907R-W 文字列の出力がページの右端を超えています。

このエラーの対処としては、次の方法が考えられます。

#### <ソースプログラムの修正>

- ・ プリンタファイルに対するファイル管理記述項に次の指定をする。

**ORGANIZATION IS LINE SEQUENTIAL**

この指定をすると、後部の空白を詰めて印刷します。各行の後部に空白があるときは、空白の分だけ出力データが短くなるので、エラーを回避できることがあります。

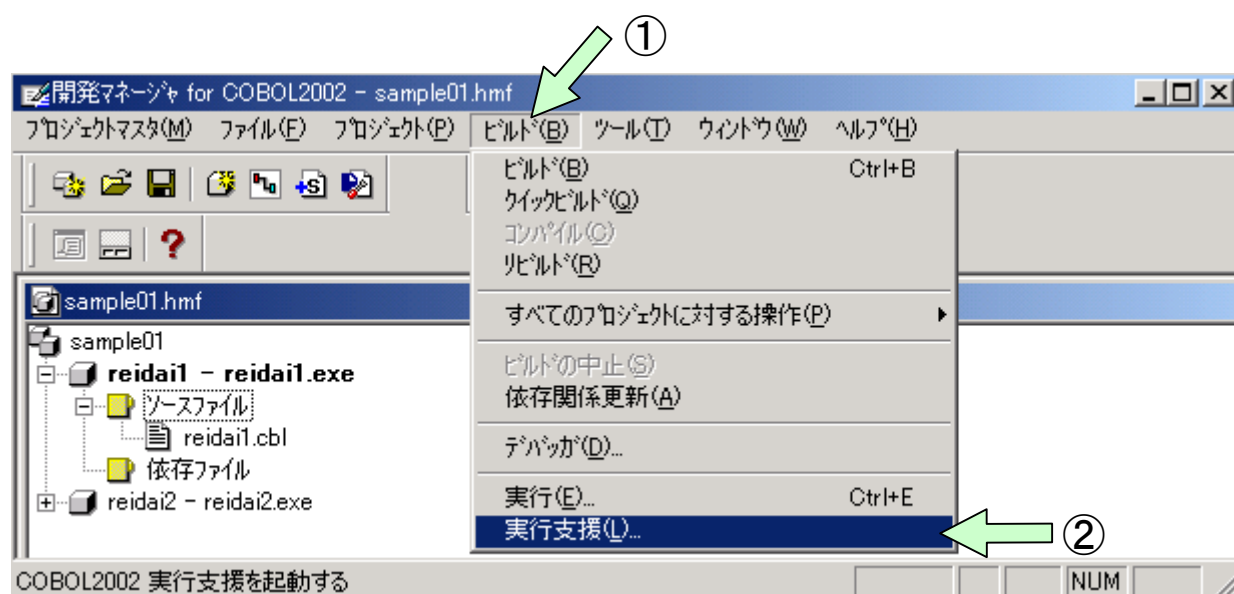
- ・ 出力レコード長を印刷できる範囲に変更する。

#### <実行支援ツールの設定>

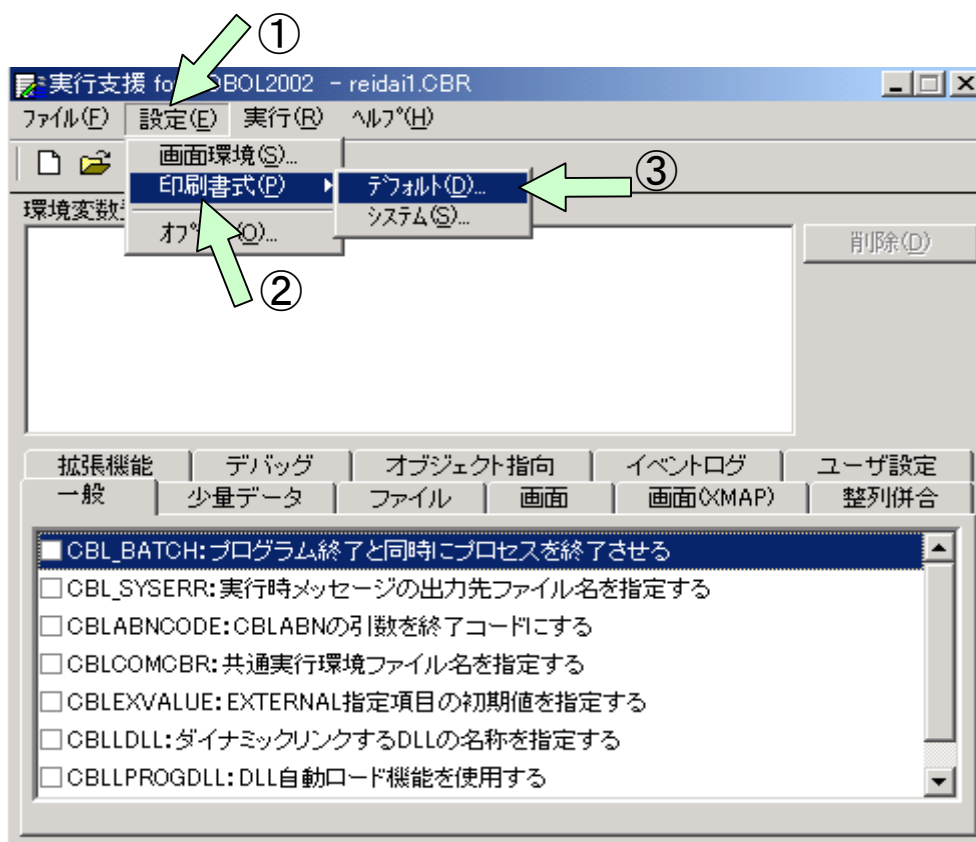
- ・ 用紙の向きを横に変更する。用紙の向きが縦になっていると、1行に印刷できる文字数が少ないためにエラーになりやすいといえます。
- ・ 余白/字間値/文字サイズを調整して1行に収まるようにする。

# 1. 印刷書式の設定

[手順 1] 「開発マネージャ」のメニューバーから「ビルド (B)」をクリックし、プルダウンメニューの中から「実行支援 (L)」を選択します。



[手順 2] 「実行支援」のメニューバーの「設定 (E)」をクリックし、プルダウンメニューの中から「印刷書式 (P)」 - 「デフォルト (D)」の順に選択します。

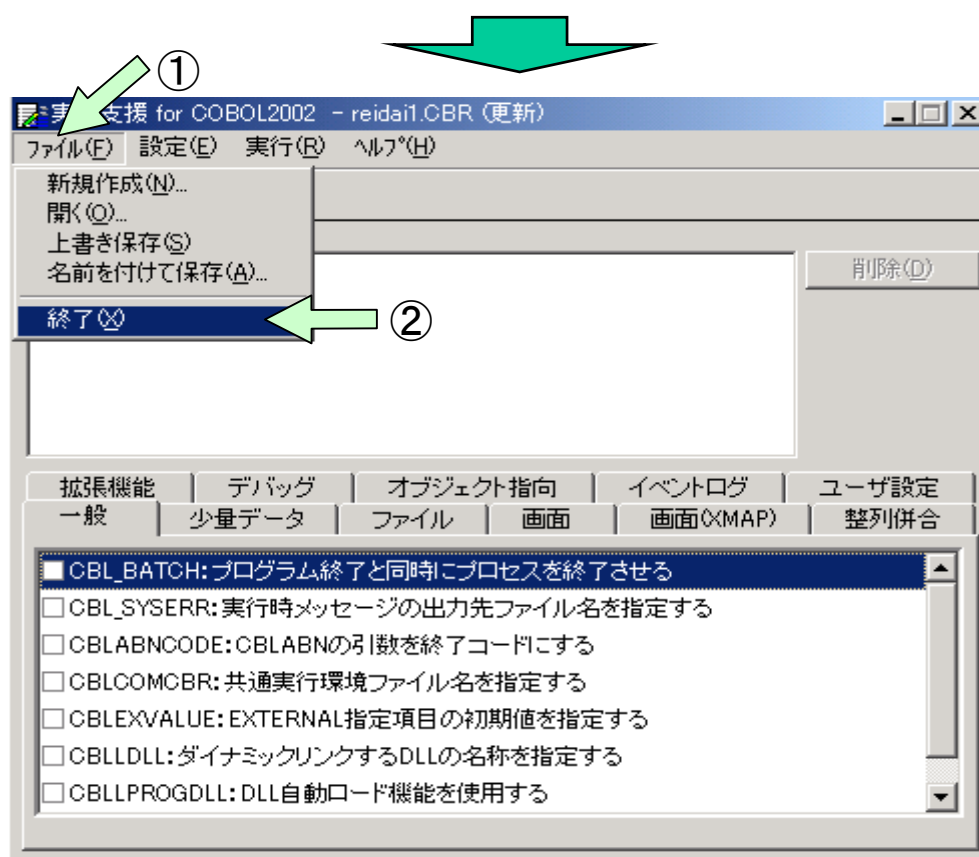
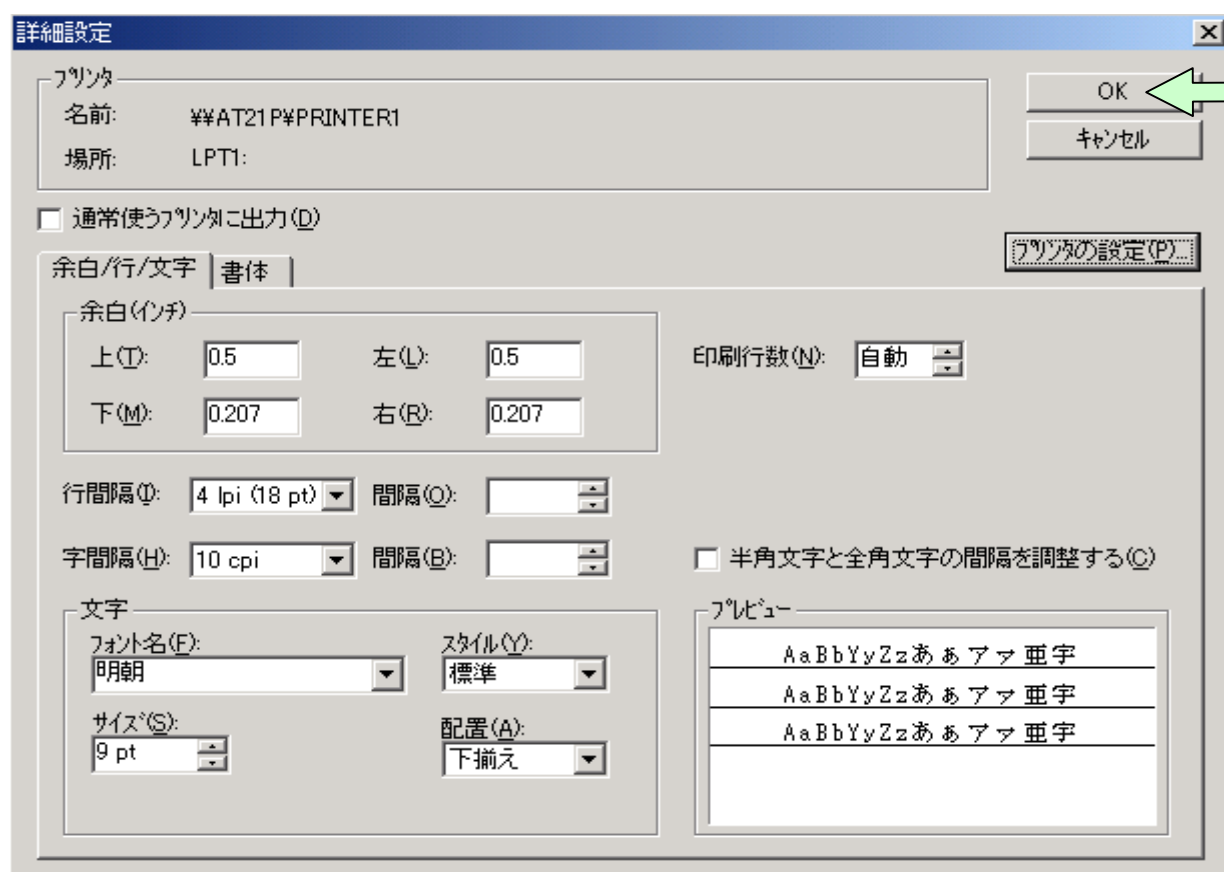


- [手順3] 余白/行間隔/フォント/文字サイズ等の設定をします。  
用紙のサイズや印刷の向きは「プリンタの設定(P)」で設定します。

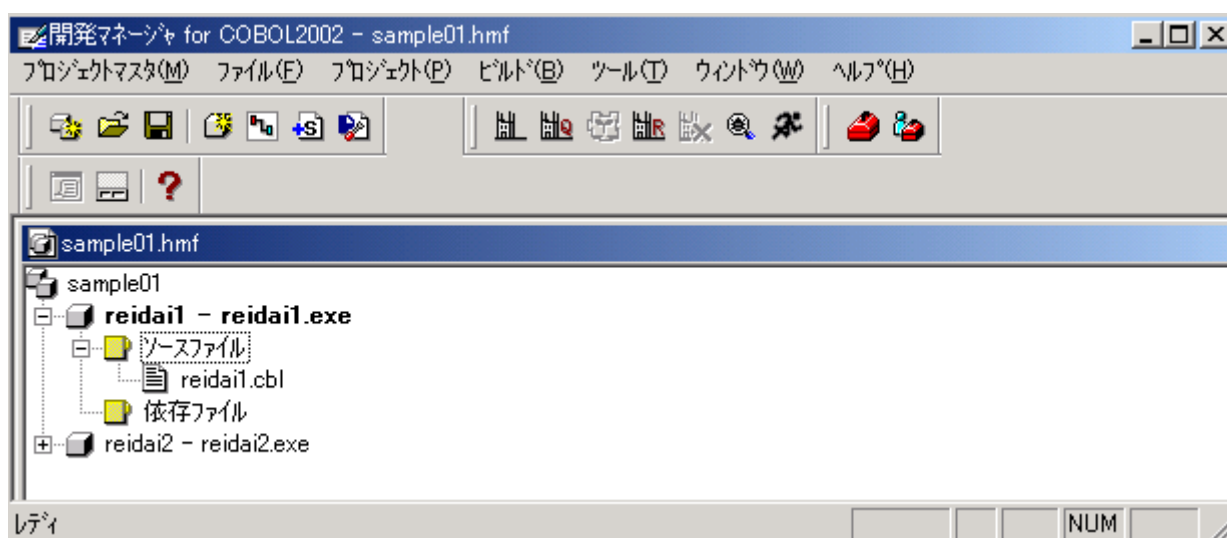


- [手順4] 「プリンタの設定」画面で、用紙サイズのリストボックスから該当する用紙サイズを選択(①)します。印刷の向きは、「縦」、「横」のラジオボタンから選択(②)します。必要な情報を設定したら「OK」ボタンをクリック(③)して終了します。

- [手順5] 「詳細設定」画面に戻ったら、「OK」ボタンをクリックして終了します。  
「実行支援」画面に戻ったら、メニューバーの「ファイル(F)」をクリックし、プルダウンメニューの中から「終了(X)」を選択します。



[手順7] すると以下のダイアログが表示されます。必ず「はい」ボタンをクリックしてください。「はい」ボタンをクリックすると設定した内容が保存され、開発マネージャの画面に戻ります。



## (e)エディタ設定方法

—COBOL専用エディタを使いこなすために—

## － 目 次 －

1. はじめに
2. マーカーとシーケンスの指定
3. 表示のカスタマイズ
4. キーワード補完
5. 構文テンプレート
6. 構文チェック



## 1. はじめに

本説明書では、COBOL2002専用エディタの起動方法を理解しているものとして説明を進めます。

使用する例題プログラムは入門編で用いたプログラムです。

ここでは、COBOL言語でコーディングするときに特に便利な機能について説明します。更に詳細を知りたい場合は、マニュアル「COBOL2002操作ガイド」を参照ください。

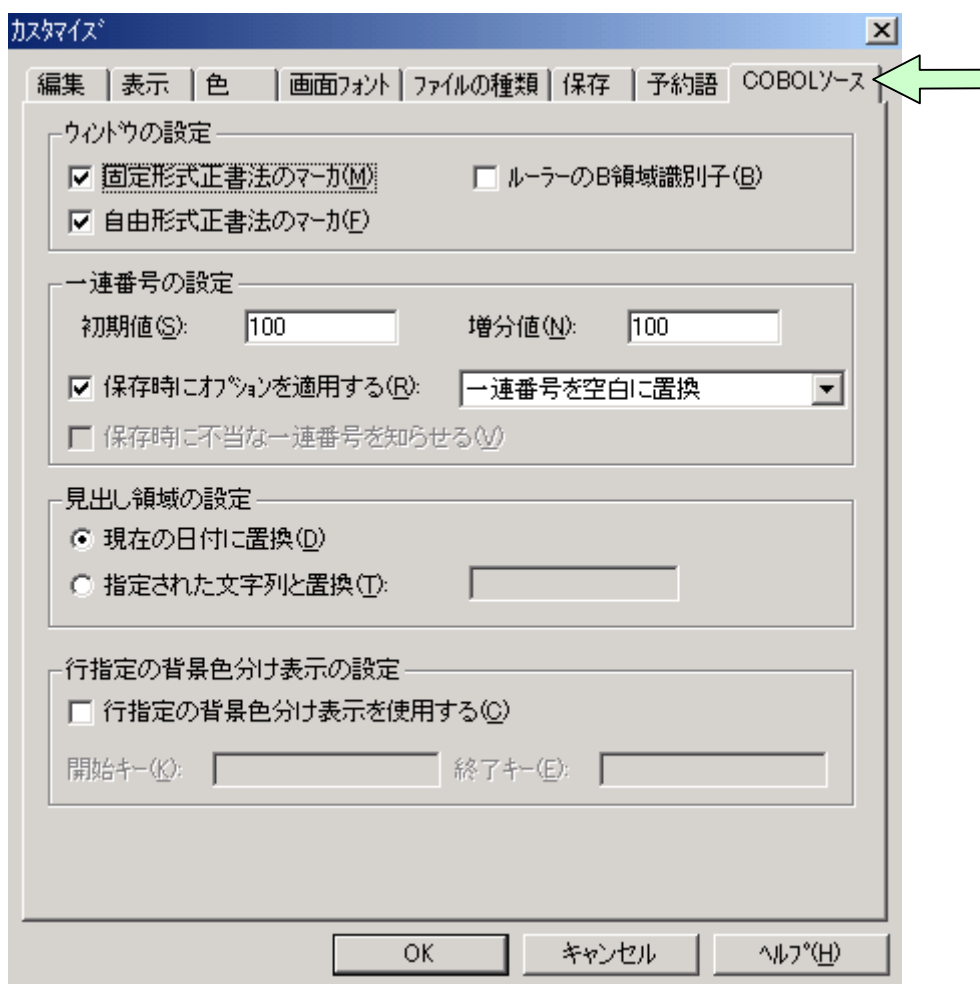
## 2. マーカーとシーケンスの指定

マーカーは、COBOLプログラムのコーディングの際に、インデントーションを揃えるために使用します。

[手順 1] エディタのメニューバーの「オプション(O)」をクリックし、プルダウンメニューの「カスタマイズ(C)」をクリックします。



カスタマイズ画面で  
「COBOLソース」タブ  
を選択します。



[手順 2] 「COBOLソース」画面で必要な設定をします。  
設定したら「OK」ボタンをクリックしてください。

COBOL用の補助線(7, 8及び73カラム目に縦線)を引きます。

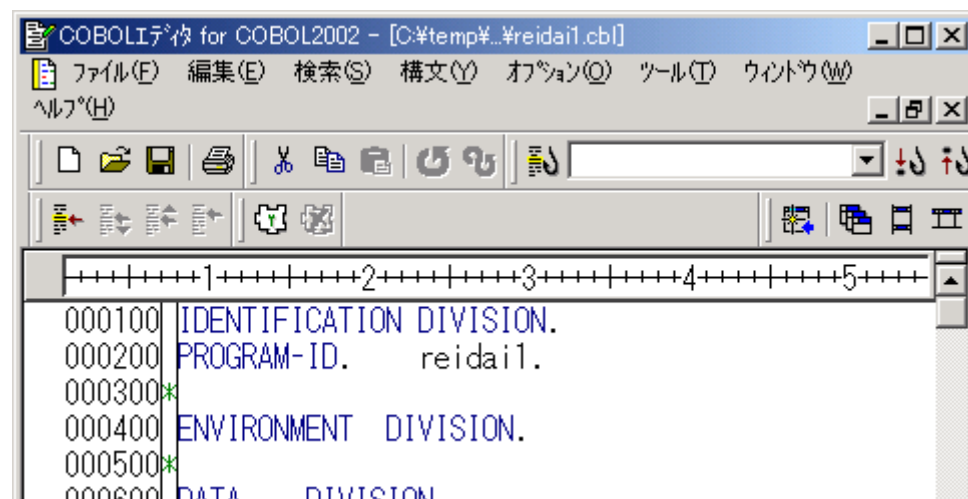
フリー形式のソースの時の指定です。普通は使いません。

シーケンスの初期値と増分値を指定します。

保存時にふりなおす時に指定します。  
ソースが修正されていない場合は、上書き保存しないとふり直しは行われません。

COBOL用のシーケンスを自動的にふり直すかどうかを指定します。

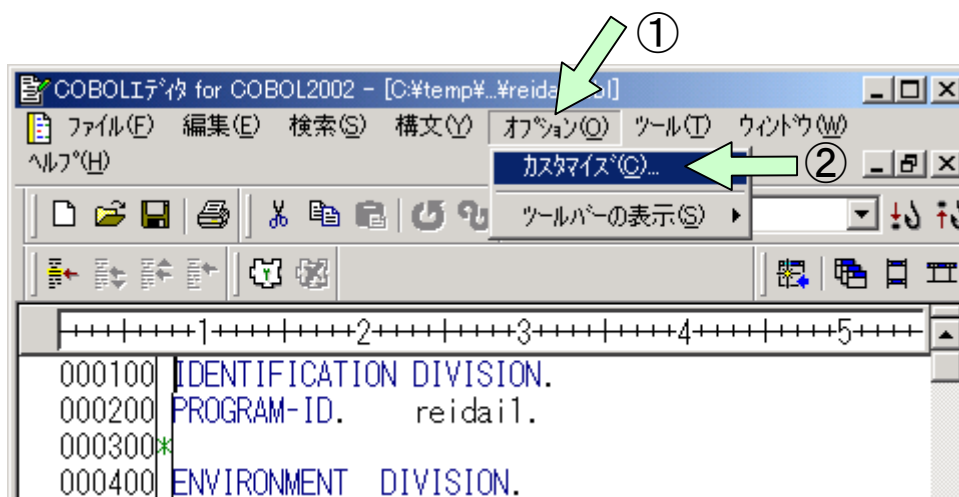
上書き保存すると同時にシーケンスがふり直されます。COBOLエディタの終了を行った場合は、次回当該ソースを開いたときにシーケンスがふり直されています。



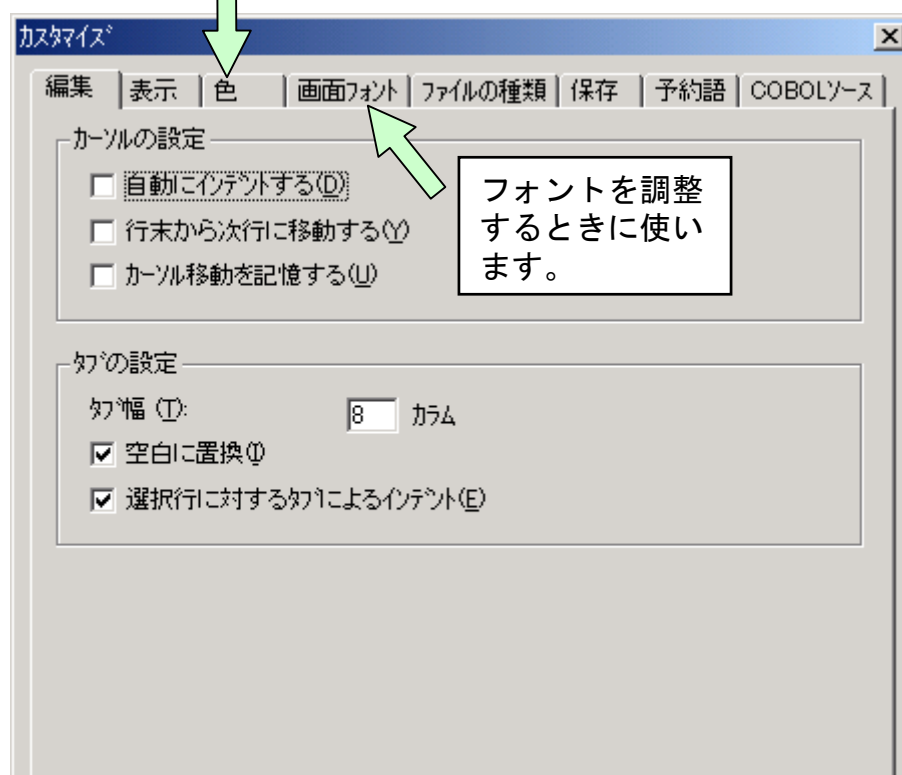
### 3. 表示のカスタマイズ

表示を見易くするための設定です。

[手順 1] エディタのメニューバーの「オプション(O)」をクリックし、プルダウンメニューの「カスタイズ(C)」を選択します。「カスタマイズ」画面が出たら、「色」または「画面フォント」タブをクリックします。



予約語等の色を設定するときに使います。



[手順2] 予約語(動詞やコメントや定数等)の色の指定を行います。

変更したい項目を選んでクリックします。

背景色や文字の色を指定します。

選んだ項目の色が表示されます。

設定が終わったら「OK」ボタンを押して終了します。

### [ワンポイントアドバイス]

予約語が色分け表示されるので、コーディングの際にスペルミスをチェックすることが可能です。見分けが付きやすい色を選ぶと一目でスペルミスがわかります。

[手順2] フォントの指定を行います。

デフォルトのイメージが表示されます。

デフォルトのイメージが表示されます。

フォント、サイズの順に設定します。サンプルの欄に設定したイメージが表示されます。

設定が終わったら「OK」ボタンを押して終了します。

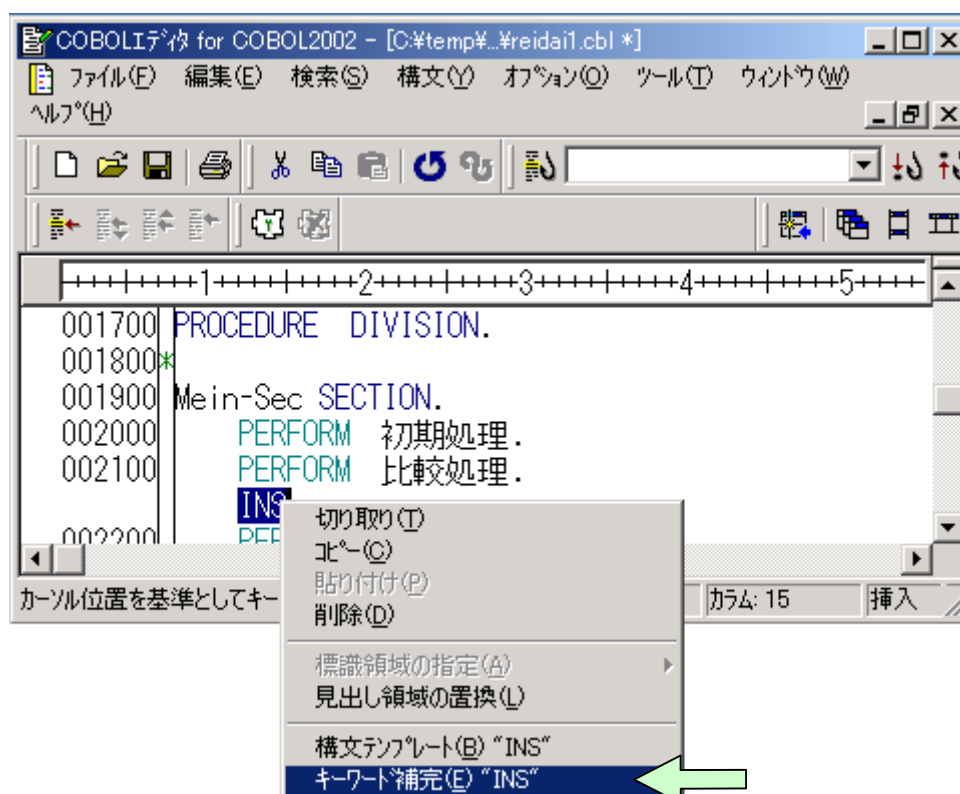
## 4. キーワード補完

本機能は語の綴りを忘れたとき、語の一部を指定してフルスペルに補完する機能です。

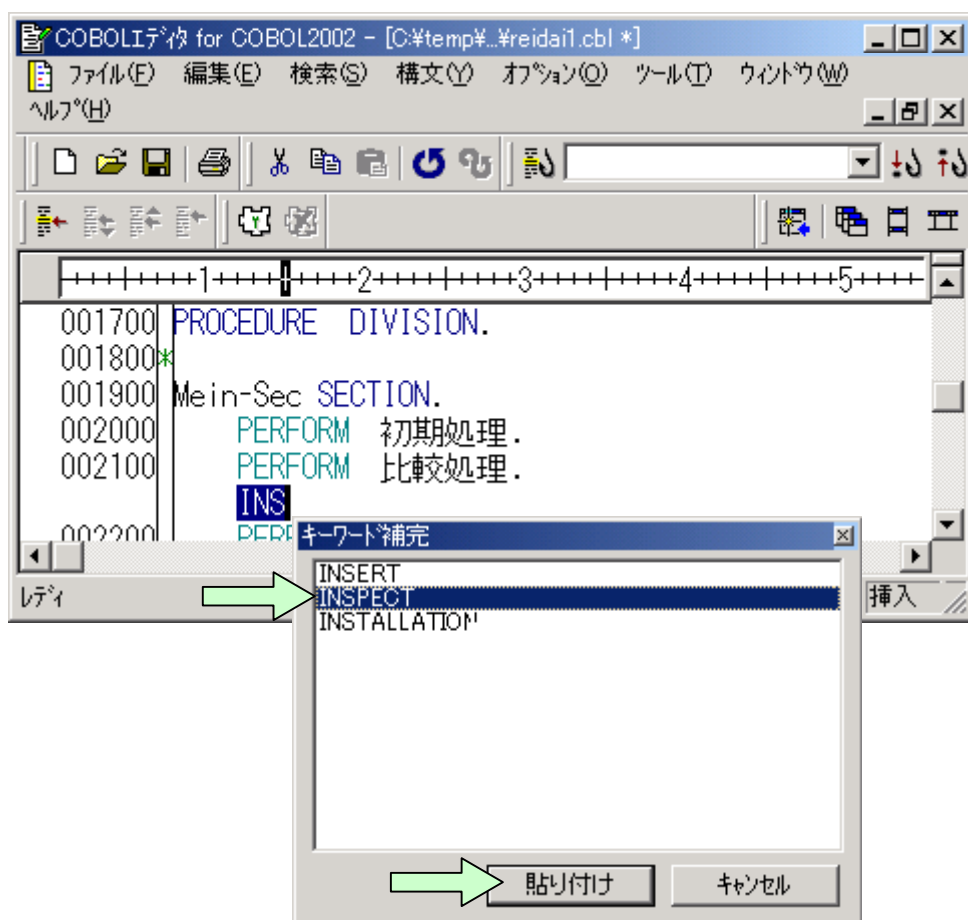
[手順 1] 例えばINSPECT文の「INS」までの綴りしか思い出せない場合、記述したい箇所に「INS」を入力し、「INS」の部分にカーソルを位置付けるかまたはリバース表示させます。



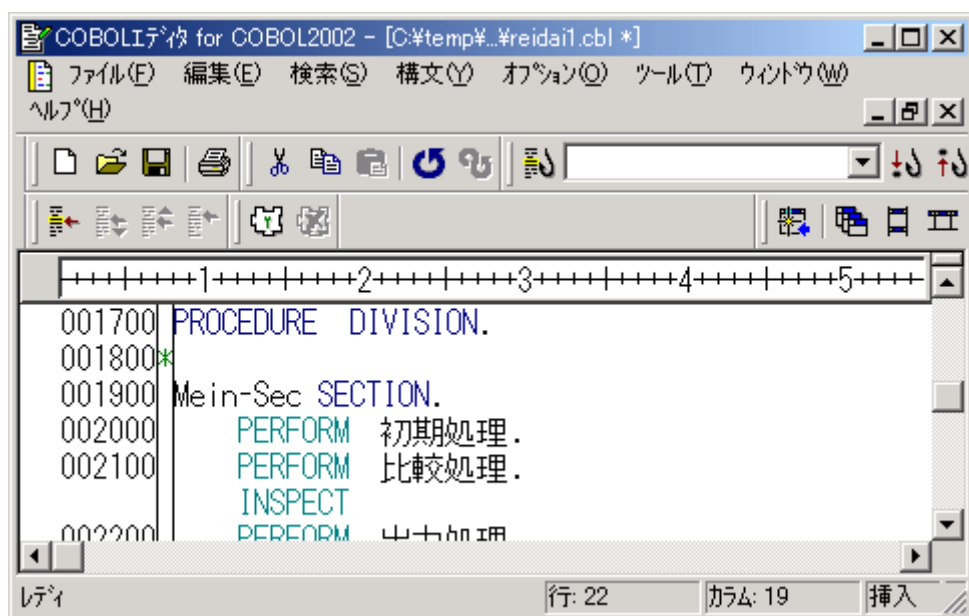
[手順 2] カーソルを位置付けて、あるいはリバース表示させた箇所で右クリックし、「キーワード補完 "INS"」を選択します。



[手順3] キーワード補完の画面中の「INSPECT」を選択し、「貼り付け」をクリックします。



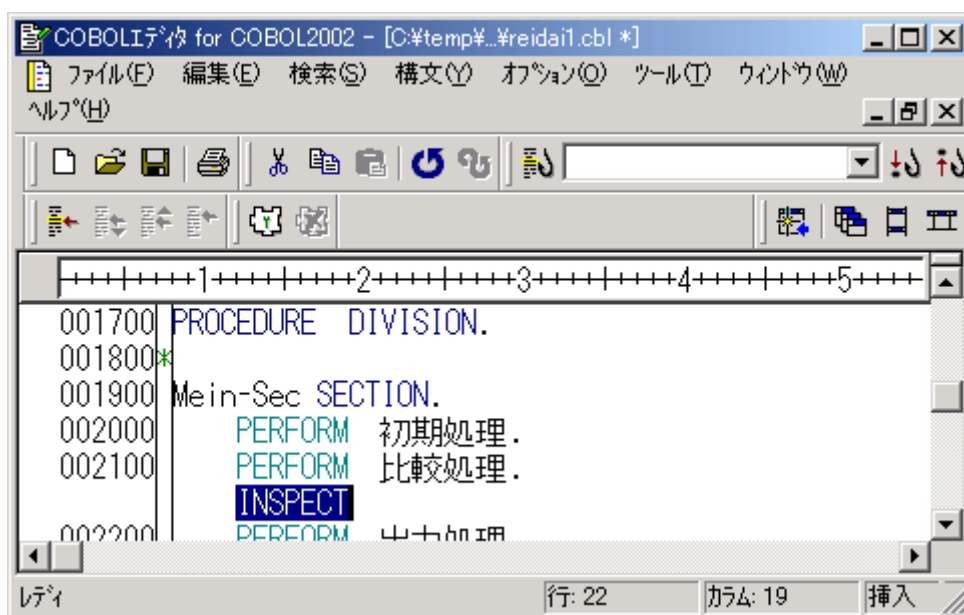
「INS」という文字列が補完されて、「INSPECT」という語が完成します。



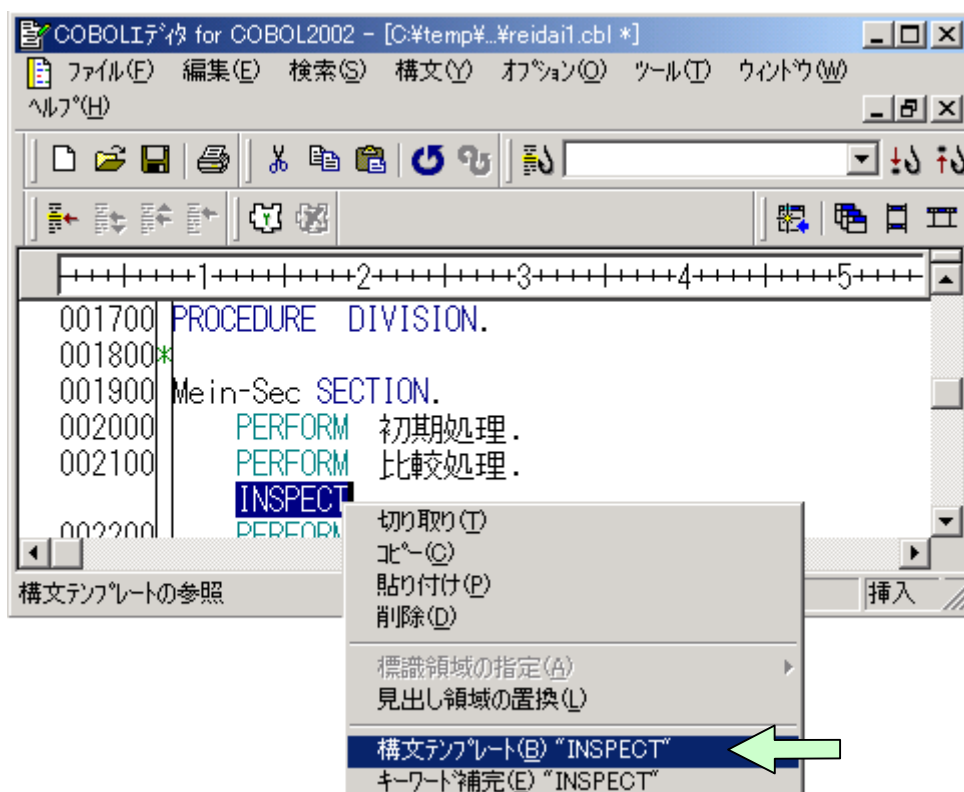
## 5. 構文テンプレート

本機能は文の書き方を知りたいとき、マニュアルを見なくても書き方を参照できる機能です。

[手順 1] 例えばINSPECT文の書き方を知りたいとき、エディタ画面上の記述したい箇所にキーワード「INSPECT」を入力し、その一部にカーソルを位置付けるか、またはリバース表示させます。



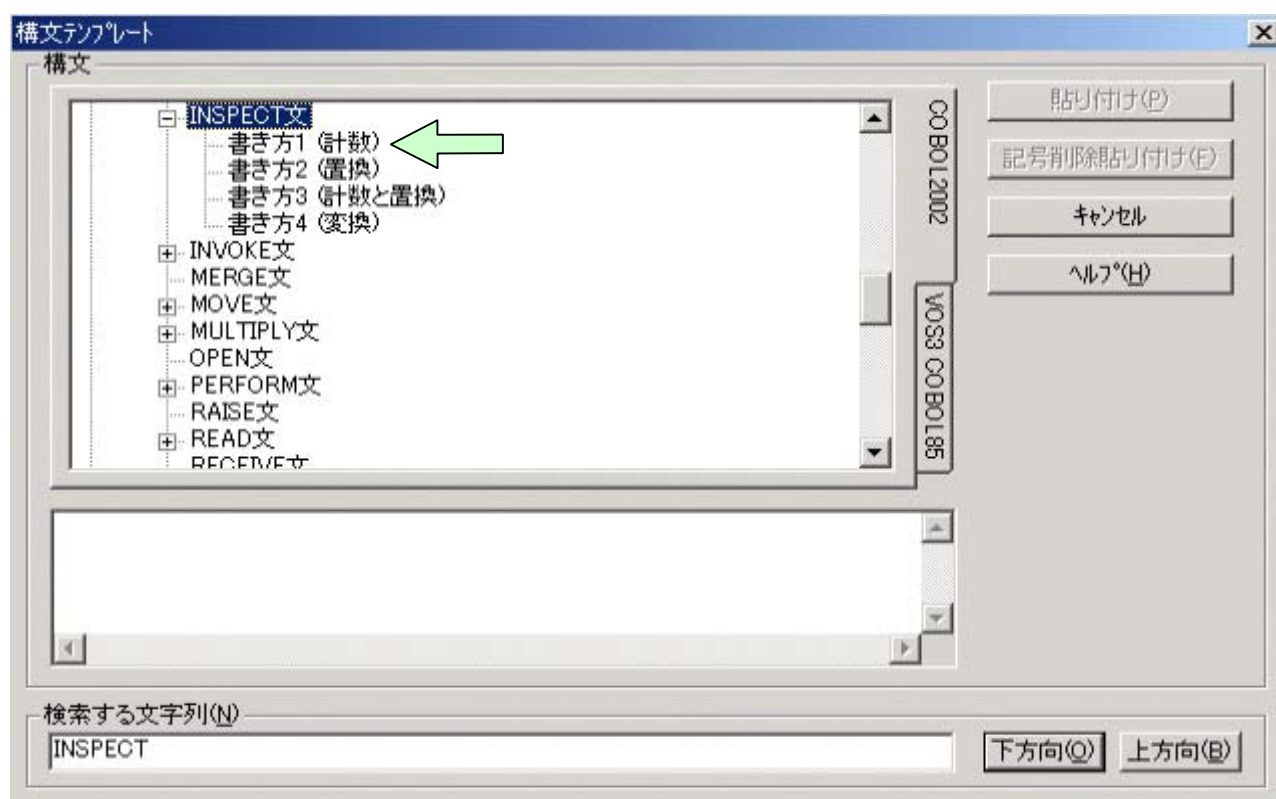
[手順 2] カーソルを位置付けた箇所またはリバース表示させた箇所で右クリックし、「構文テンプレートの“INSPECT”」を選択します。



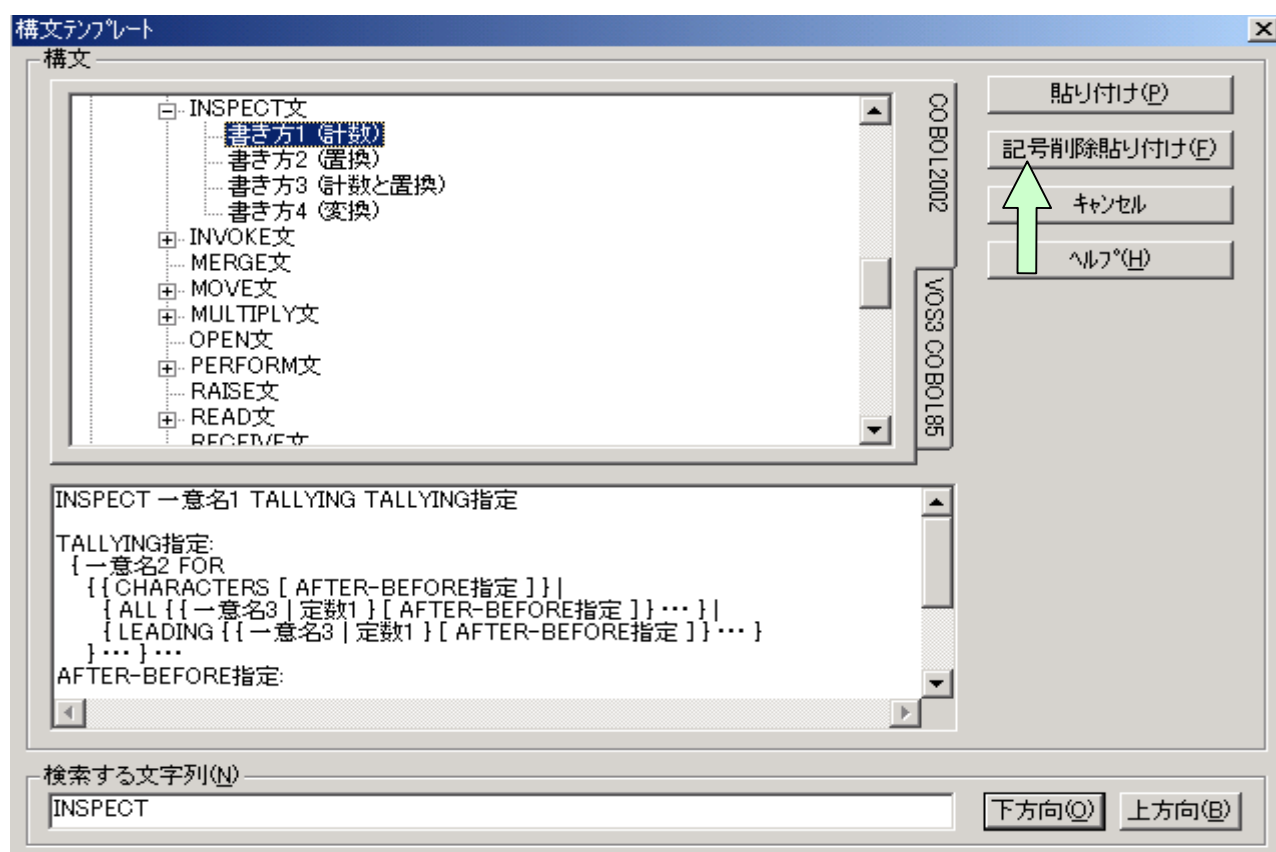


[手順3] 「構文テンプレート」画面が表示されます。

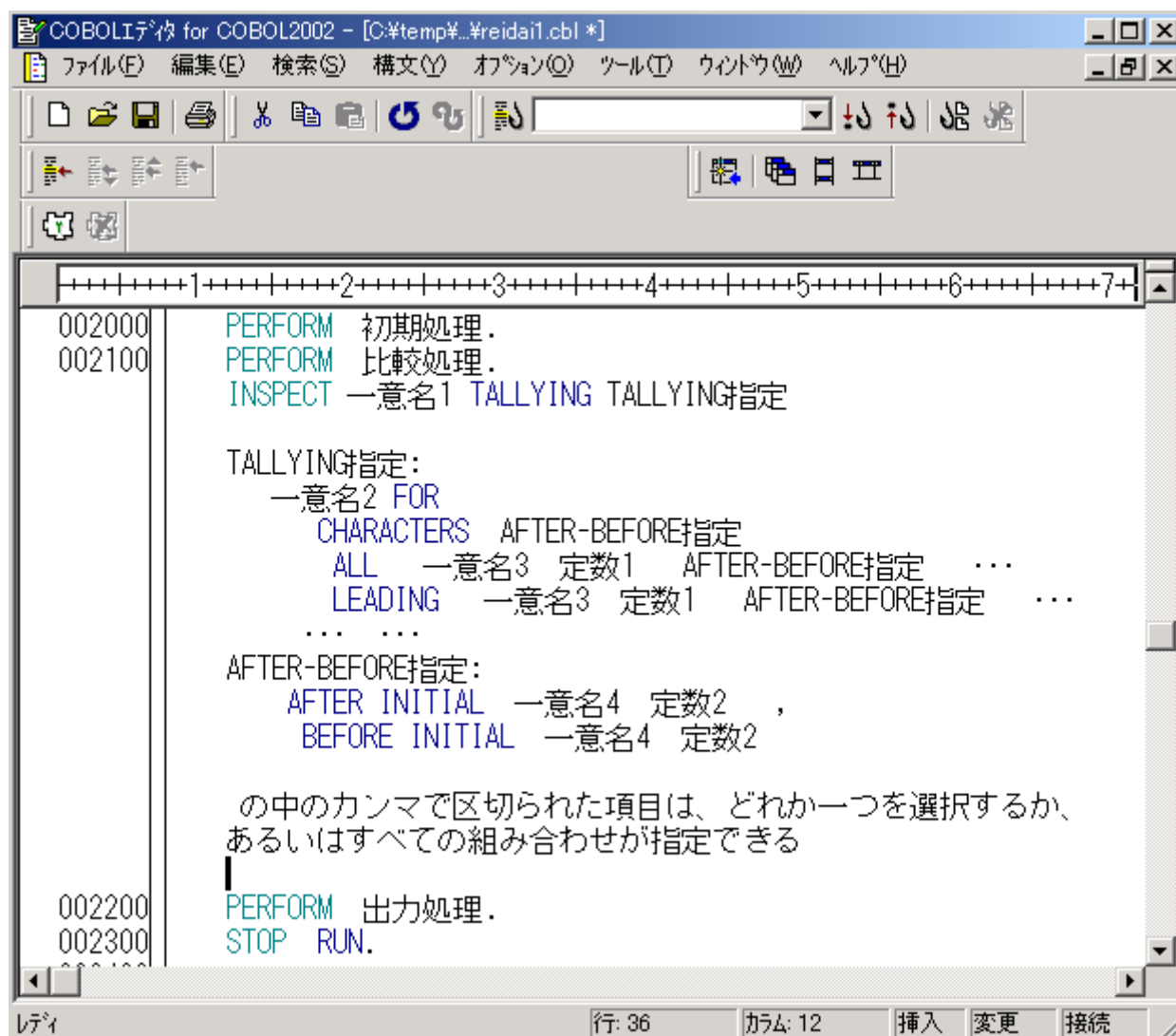
ここで、例えば書き方1 (計数) をクリックすると、プレビュー画面に使用方法が表示されます。



[手順4] プレビュー画面へ使用方法が表示され、「記号削除貼り付け」をクリックすると記号が削除されて該当する箇所へ貼り付けられます。



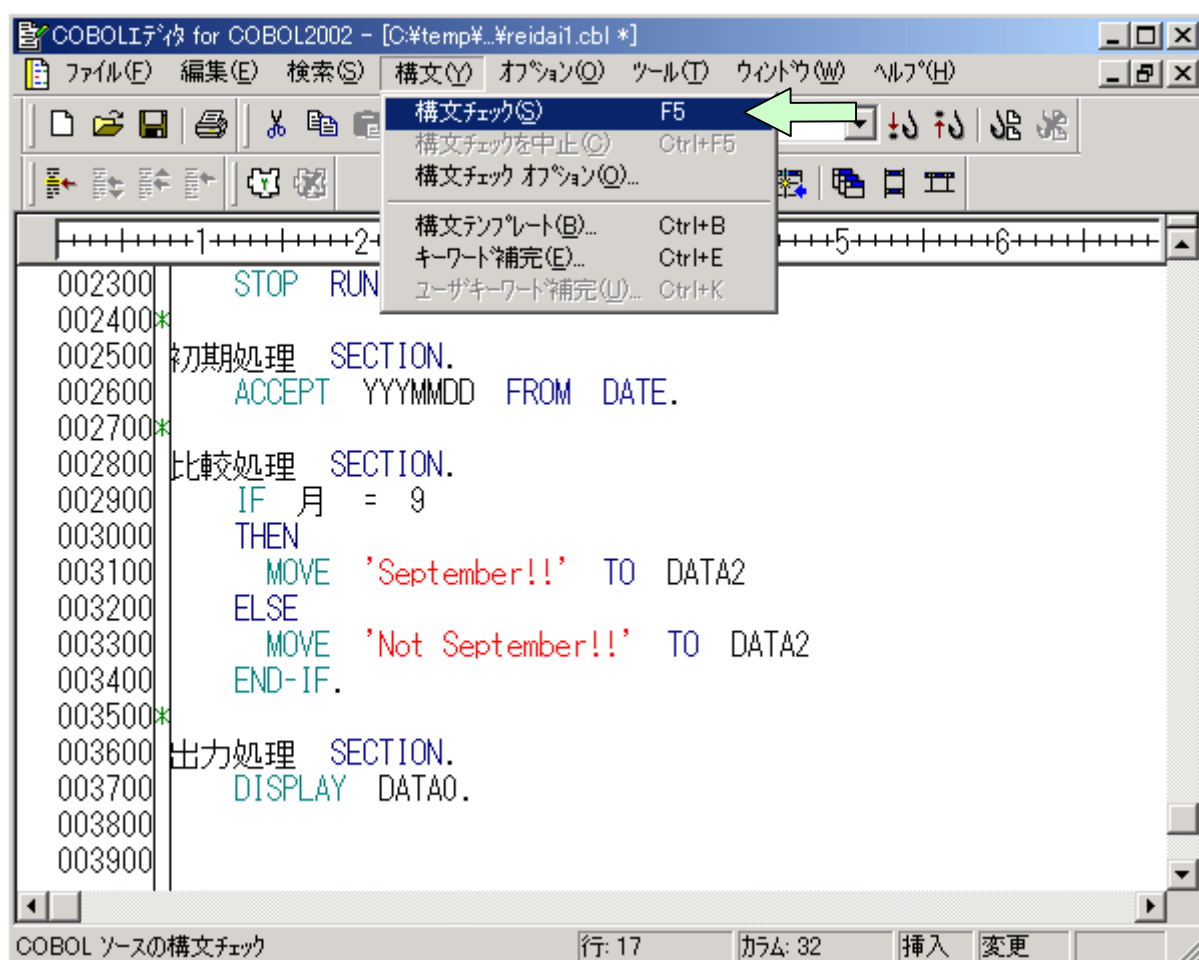
「記号削除貼り付け」の結果を以下に示します。「張り付け」をクリックすると記号も一緒に貼り付けられてしまいます。なお、手順4の画面に表示されている構文を直接書き換えることもできます。書き換えが完了したら、「張り付け」をクリックします。



## 6. 構文チェック

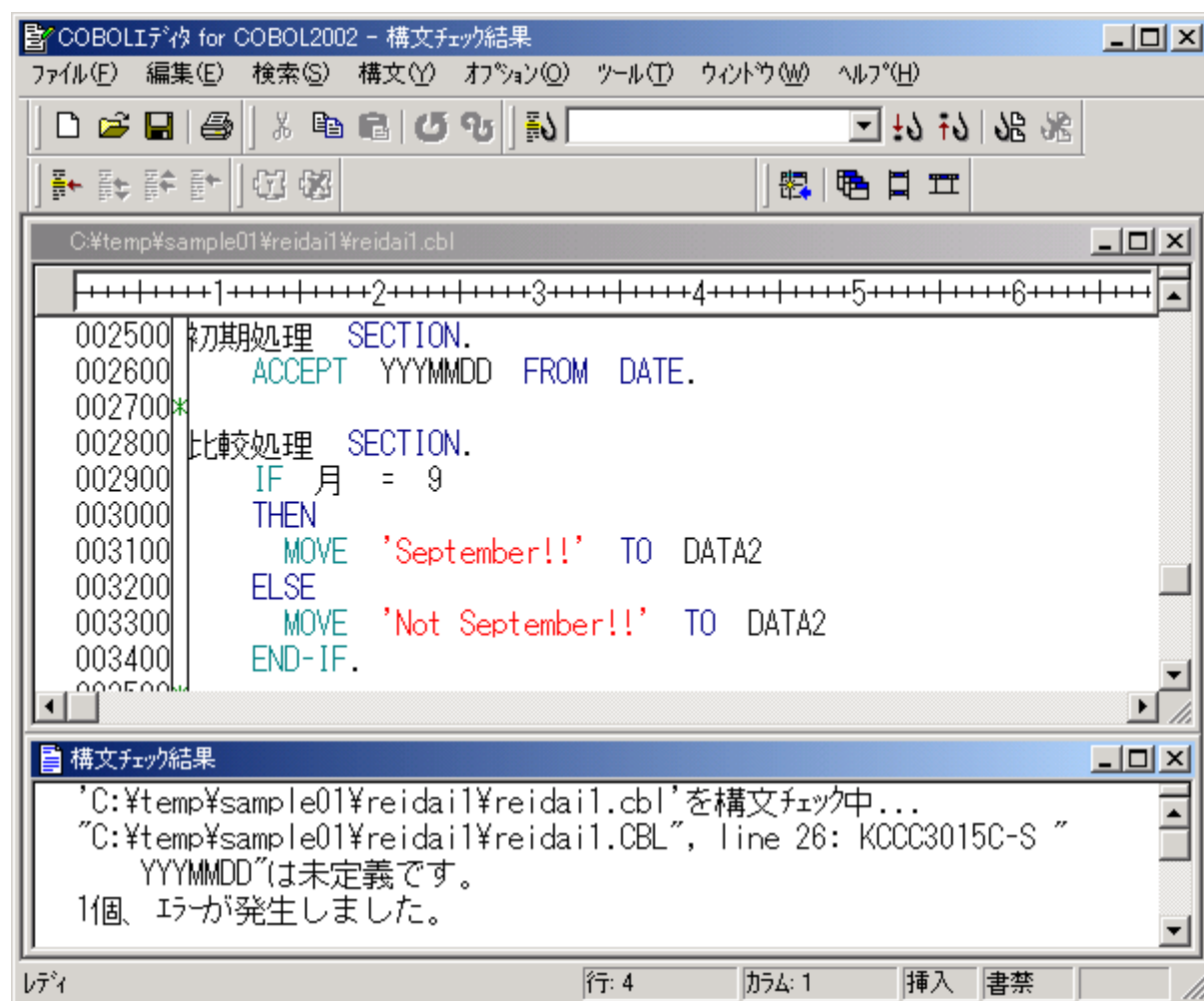
構文チェック機能は、エディタ画面でプログラム編集中に、文法チェックを行う機能です。

[手順 1] エディタのメニューバーの「構文(Y)」をクリックし、プルダウンメニューの「構文チェック(S)」を選択します。



構文チェック結果がCOBOLエディタ画面の下に表示されます。

エラーがある場合は、通常のコンパイルと同様にエラーメッセージが表示され、エラーメッセージ部分をダブルクリックすると、エラーの箇所にカーソルが位置付きます。



## (f)コンパイルリストの入手方法

## ( 1 ) コンパイルリストの種類

コンパイラが出力するリストの種類を次に示します。

### ( a ) 情報リスト

プログラム情報やエラーの総数等のコンパイル時の情報を要約して出力したものです。

### ( b ) 原始プログラムリスト

コンパイル時に入力した原始プログラムのリストです。相互参照情報やコンパイル時にエラーが検出されたときのエラーメッセージなども出力されます。

### ( c ) エラーリスト

コンパイルエラーのエラーレベルやエラーメッセージを出力したものです。

## ( 2 ) リストの出力方法

### ( a ) コンパイラオプションの指定

コンパイルリストを出力するためのオプションを次に示します。これらのコンパイラオプションを指定しない場合、コンパイルリストは出力されません。

-SrcList, NoCopy

COPY文で複写した登録集原文の内容を原始プログラムリスト中に展開しません。

-SrcList, CopySup

SUPPRESS指定のあるCOPY文で複写した登録集原文の内容は原始プログラムリスト中に展開しません。 SUPPRESS指定のないCOPY文の場合はすべて展開します。

-SrcList, CopyAll

COPY文で複写した登録集原文の内容をすべて原始プログラムリスト中に展開します。

-SrcList, OutputAll

COPY文の指定や条件翻訳、LISTING翻訳指令にかかわらず、強制的にすべてのソース原文をコンパイルリスト中に展開します。SUPPRESS指定のあるCOPY文も展開します。

-SrcList, XXXXX, NoFalsePath

条件翻訳結果の無効行はコンパイルリストに出力されません。  
XXXXXには、CopyAll, CopySup, NoCopyのどれかを指定します。

NoCopy, CopySup, CopyAll, およびOutputAllサブオプションは同時には指定できません。同時に指定した場合、最後に指定したオプションが有効になります。

NoFalsePathサブオプションは、その他のオプション(OutputAll以外)と同時に指定する必要があります。

全ての情報を表示したいときは、OutputAllオプションだけを指定します。

#### (b) コンパイルリストの出力先

情報リストと原始プログラムリストは、コンパイルリストファイル(.lst)に出力されます。また、エラーリストは標準エラー出力(stderr)、すなわちCOBOL2002開発マネージャのメッセージウィンドウに表示されます。)

コンパイルリストの出力手順および出力例は、次ページ以降で説明します。

下記プログラムは「reidai1」プログラムのデータ定義部の「DATA0」と「YYMMDD」を登録集原文として別ファイルに登録したものです。DATA0.cblとYYMMDD.cblは、ソースファイルと同じフォルダに作成します。このプログラムをコンパイラオプション「-SrcList, NoCopy」を指定してコンパイルしてみます。

```

000500*
000600 DATA    DIVISION.
000700 WORKING-STORAGE SECTION.
000800 COPY    DATA0.
000900 COPY    YYMMDD    SUPPRESS.
001000*

```

行: 1      カラム: 8      挿入

```

000100 01 DATA0.
000200 02 DATA1 PIC X(10) VALUE ALL '*'.
000300 02 DATA2 PIC X(20) VALUE SPACE.
000400 02 DATA3 PIC X(10) VALUE ALL '*'.

```

行:      カラム:

```

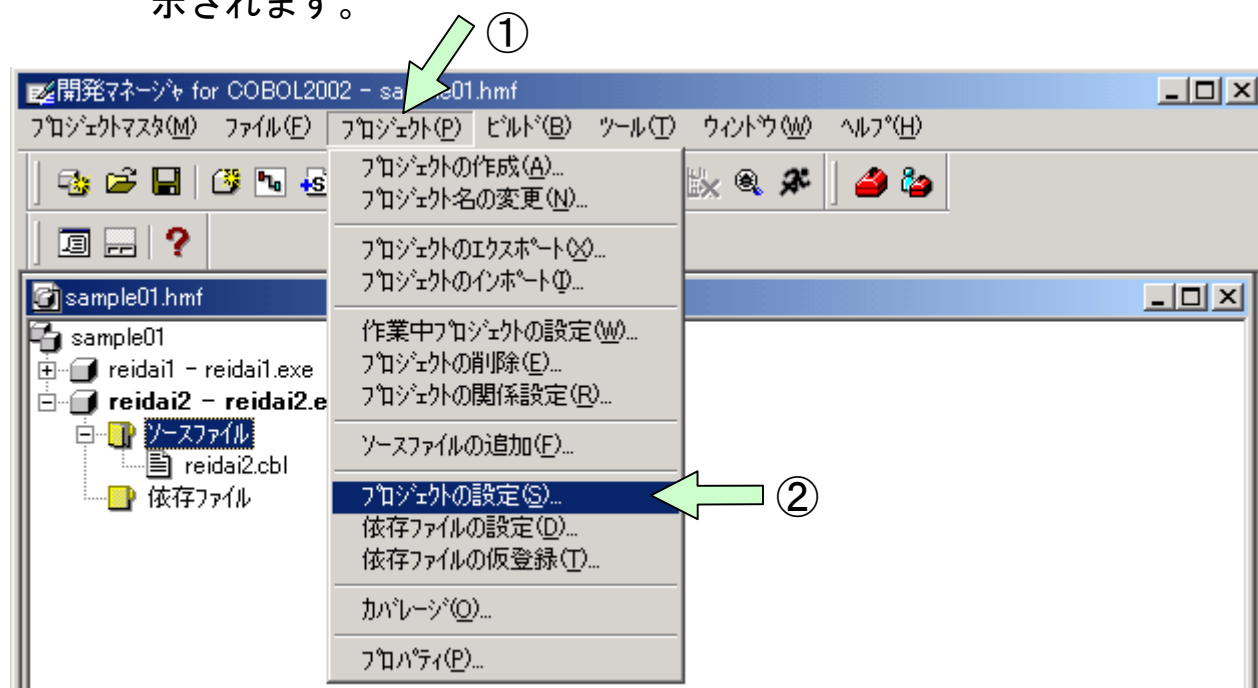
000100 01 YYMMDD.
000200 02 年 PIC 9(2).
000300 02 月 PIC 9(2).
000400 02 日 PIC 9(2).

```

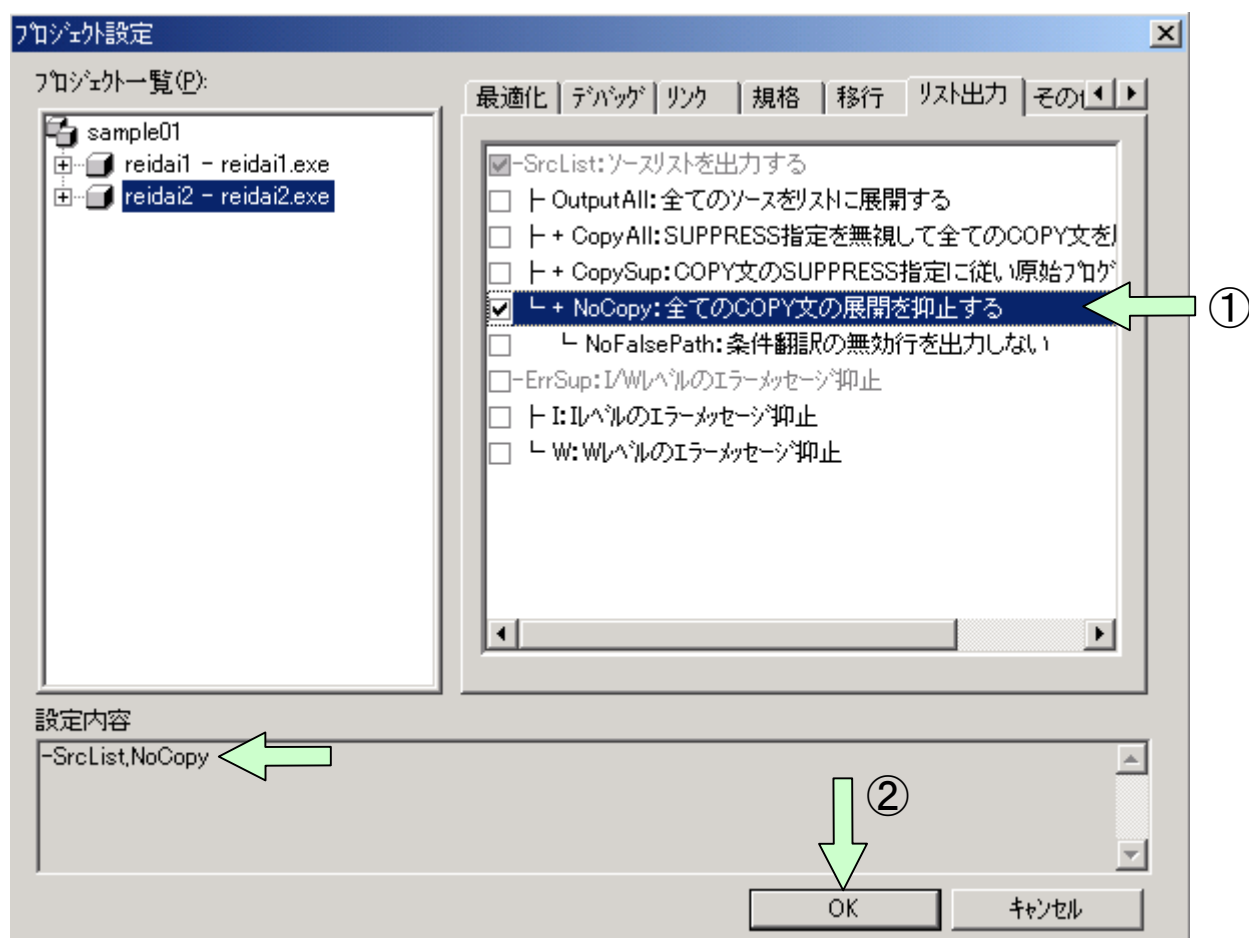
行: 1      カラム: 8      挿入



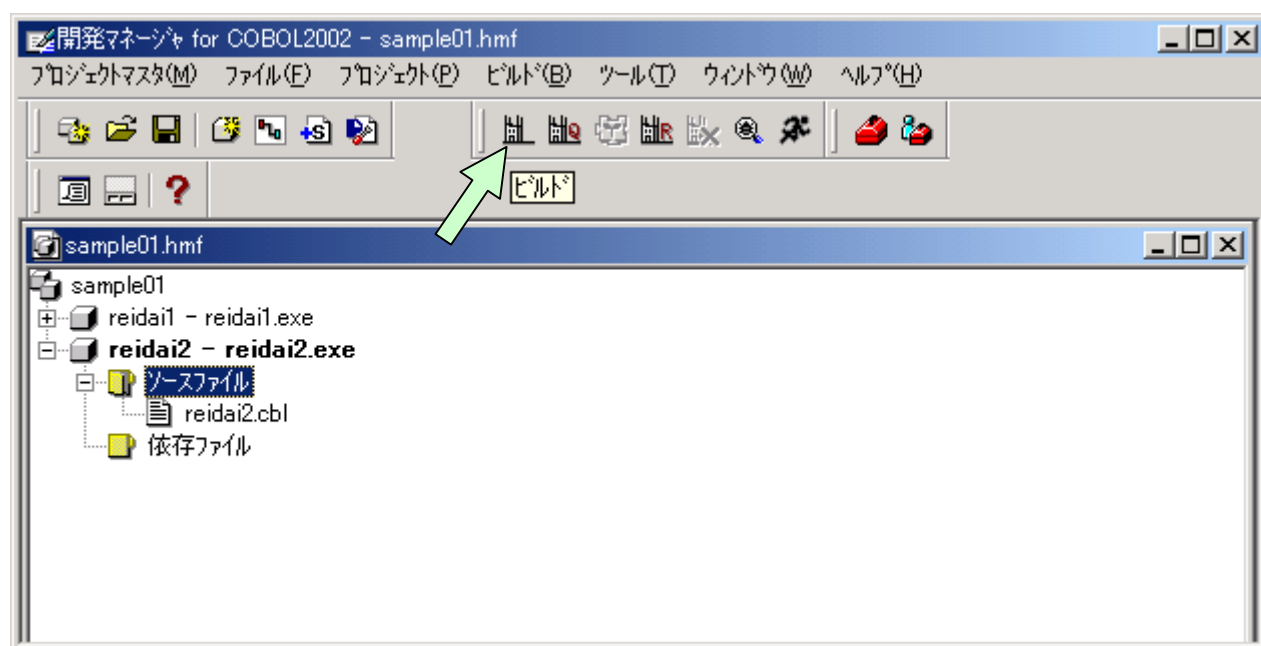
[手順 1] COBOL2002開発マネージャのメニューバーの「プロジェクト(P)」、「プロジェクトの設定(S)」の順にクリックします。すると「プロジェクト設定」画面が表示されます。



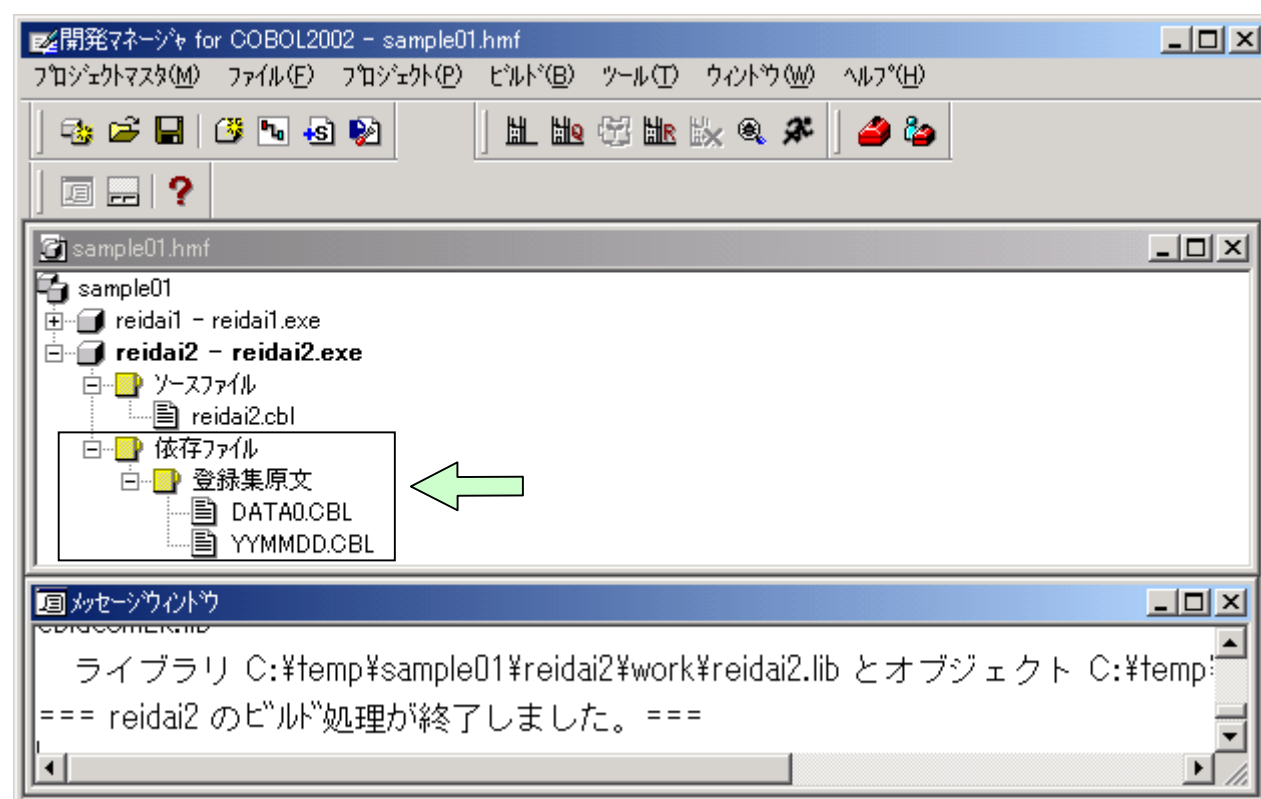
[手順 2] 「プロジェクト設定」画面の「リスト出力」タブの中から「-SrcList, NoCopy」オプションをクリックし、「OK」ボタンをクリックします（口内にレ印が付き、「設定内容」の欄に表示されます）。



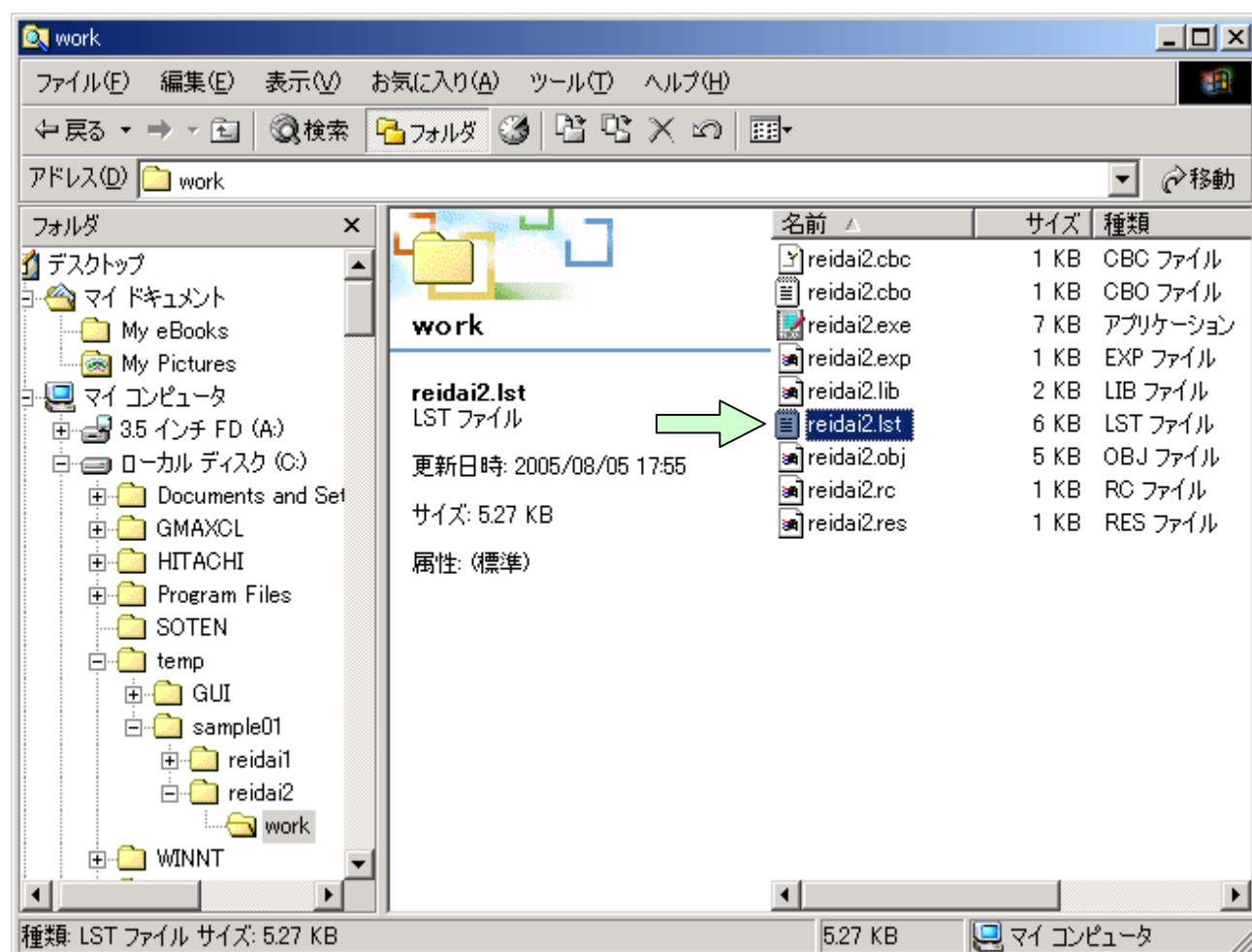
〔手順3〕 開発マネージャの画面に戻ったら、「ビルド」を行います。



〔手順4〕 ビルドが終了すると「依存ファイル」が表示されます。＋ボタンを押して登録されている登録集原文を見てください。依存関係が自動的に確立されていることがわかります。また、workフォルダの下には、拡張子が.lstのファイルが生成されていますので開いて見てください。



- [手順5] エクスプローラによりworkフォルダの下に実行可能ファイル等と共にコンパイルリストファイル「reidai2.lst」が生成されます。  
コンパイルリストファイルは、COBOLエディタ、メモ帳等で開いて見ることができます。



コンパイルリストを以下に示します(メモ帳で開いています)。  
「-SrcList, NoCopy」 オプションなので、COPY文の内容は展開されません。

```

reidai2.lst - メモ帳
ファイル(F) 編集(E) 書式(O) ヘルプ(H)

*****
* コード              意  味
*   * : 更新
*   # : INITIALIZE又はCORRESPONDINGで更新される下位項目
*   A : ALTERで参照
*   D : データ部又は環境部で参照
*   E : PERFORMの出口
*   G : GO TOで参照
*   P : PERFORMで参照
*   Q : IF/EVALUATE/PERFORM...UNTIL/SEARCH...WHEN/探索条件で参照
*   S : 添字で参照
* なし : その他
*****

A 000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    reidai2.
000300*
000400 ENVIRONMENT  DIVISION.
000500*
000600 DATA        DIVISION.
000700 WORKING-STORAGE SECTION.
000800 COPY DATA0.
000900 COPY YYMMDD  SUPPRESS.
001000*
001100 PROCEDURE  DIVISION.
001200*
001300 Mein-Sec SECTION.
001400     PERFORM  初期処理.
001500     PERFORM  比較処理.
001600     PERFORM  出力処理.
001700     STOP   RUN.
  
```

COPY文の内容は展開されません。

「-SrcList, CopySup」オプションを指定してコンパイルすると、コンパイルリストは次のようになります。

```

A 000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.    reidai2.
000300*
000400 ENVIRONMENT  DIVISION.
000500*
000600 DATA    DIVISION.
000700 WORKING-STORAGE SECTION.
000800 COPY DATA0.
801  C1 000100 01 DATA0.
802  C1 000200 02 DATA1 PIC X(10) VALUE ALL '*'.
803  C1 000300 02 DATA2 PIC X(20) VALUE SPACE.
804  C1 000400 02 DATA3 PIC X(10) VALUE ALL '*'.
000900 COPY YYMMDD SUPPRESS.
001000*
001100 PROCEDURE  DIVISION.
001200*
001300 Mein-Sec SECTION.
001400     PERFORM  初期処理.
001500     PERFORM  比較処理.
001600     PERFORM  出力処理.
001700     STOP  RUN.
001800*
001900 初期処理  SECTION.
002000     ACCEPT YYMMDD FROM DATE.
002100*
002200 比較処理  SECTION.
002300     IF 月 = 9
002400     THEN
1 002500         MOVE 'September!!' TO DATA2
002600     ELSE

```

「-SrcList, CopyAll」オプションを指定してコンパイルすると、コンパイルリストは次のようになります。

```

A 000100 IDENTIFICATION DIVISION.
  000200 PROGRAM-ID.    reidai2.
  000300*
  000400 ENVIRONMENT  DIVISION.
  000500*
  000600 DATA    DIVISION.
  000700 WORKING-STORAGE SECTION.
  000800 COPY  DATA0.
801  C1  000100 01  DATA0.
802  C1  000200 02  DATA1 PIC X(10) VALUE ALL '*'.
803  C1  000300 02  DATA2 PIC X(20) VALUE SPACE.
804  C1  000400 02  DATA3 PIC X(10) VALUE ALL '*'.
  000900 COPY  YMMDD  SUPPRESS.
901  C1  000100 01  YMMDD.
902  C1  000200 02  年  PIC  9(2).
903  C1  000300 02  月  PIC  9(2).
904  C1  000400 02  日  PIC  9(2).
  001000*
  001100 PROCEDURE  DIVISION.
  001200*
  001300 Mein-Sec SECTION.
  001400     PERFORM  初期処理.
  001500     PERFORM  比較処理.
  001600     PERFORM  出力処理.
  001700     STOP   RUN.
  001800*
  001900 初期処理  SECTION.
  002000     ACCEPT YMMDD FROM DATE.
  002100*
  002200 比較処理  SECTION.
  
```

SUPPRESS指定に関係なく、全てのCOPY文が展開されます。

## (g) オンラインマニュアルの使用方法

- ・ オンラインマニュアルにはCOBOL2002の全機能が掲載されています。
- ・ キーワード検索等により、探したい項目を瞬時に検索することができます。

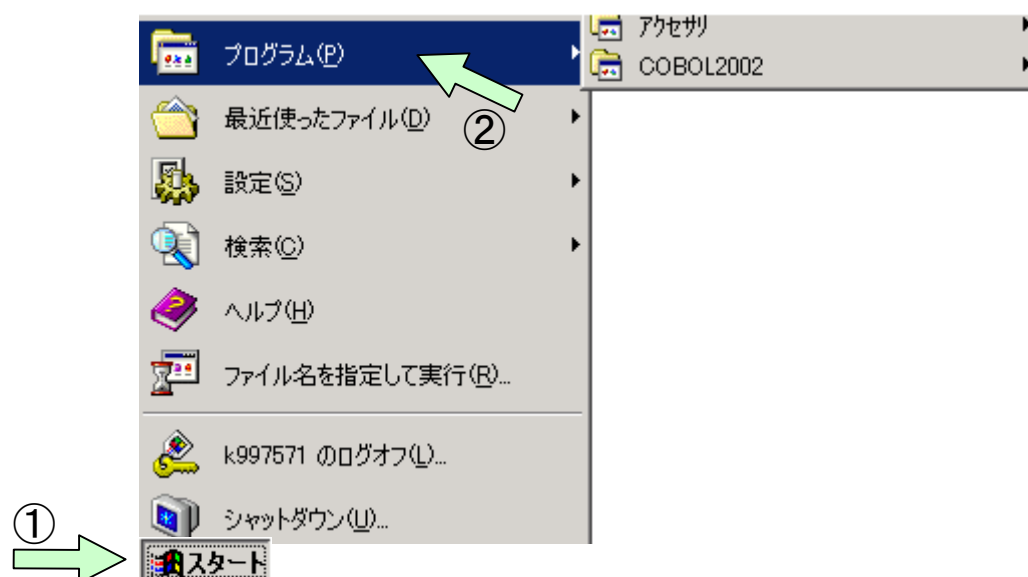
### 〈オンラインマニュアルの構成〉

オンラインマニュアルは、次のマニュアルから構成されています。

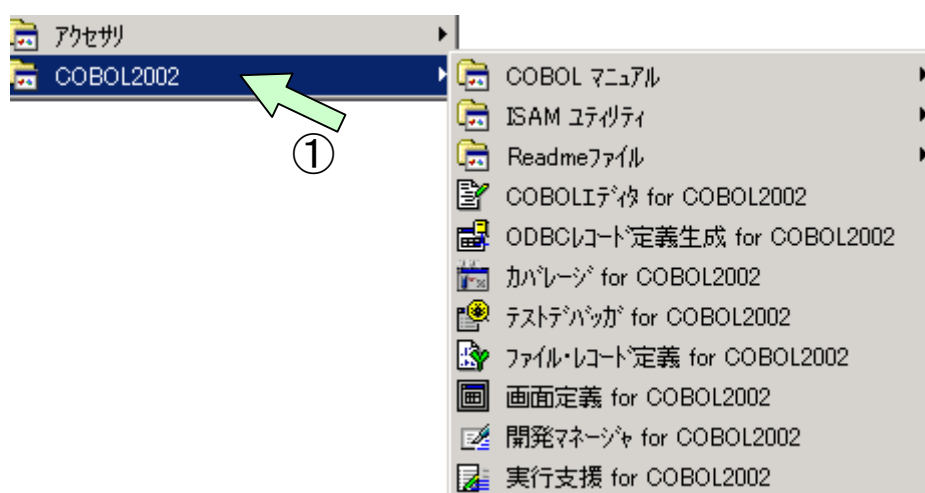
- ・ 「総合目次」
- ・ 「COBOL2002言語 標準仕様編」
- ・ 「COBOL2002 言語 拡張仕様編」
- ・ 「COBOL2002操作ガイド」
- ・ 「COBOL2002 ユーザーズガイド」
- ・ 「COBOL2002 メッセージ」
- ・ 「COBOL2002操作入門」
- ・ 「索引順編成ファイル管理ISAM」

# オンラインマニュアルの参照方法

[手順 1] スタートボタンを押し、「プログラム(P)」の所にマウスポインタを移動します。すると起動できるプログラムの一覧が出てきます。



[手順 2] プログラムの一覧の中から「COBOL2002」の所にマウスポインタを移動します。するとCOBOL2002の中の使用できるツール一覧が表示されます。

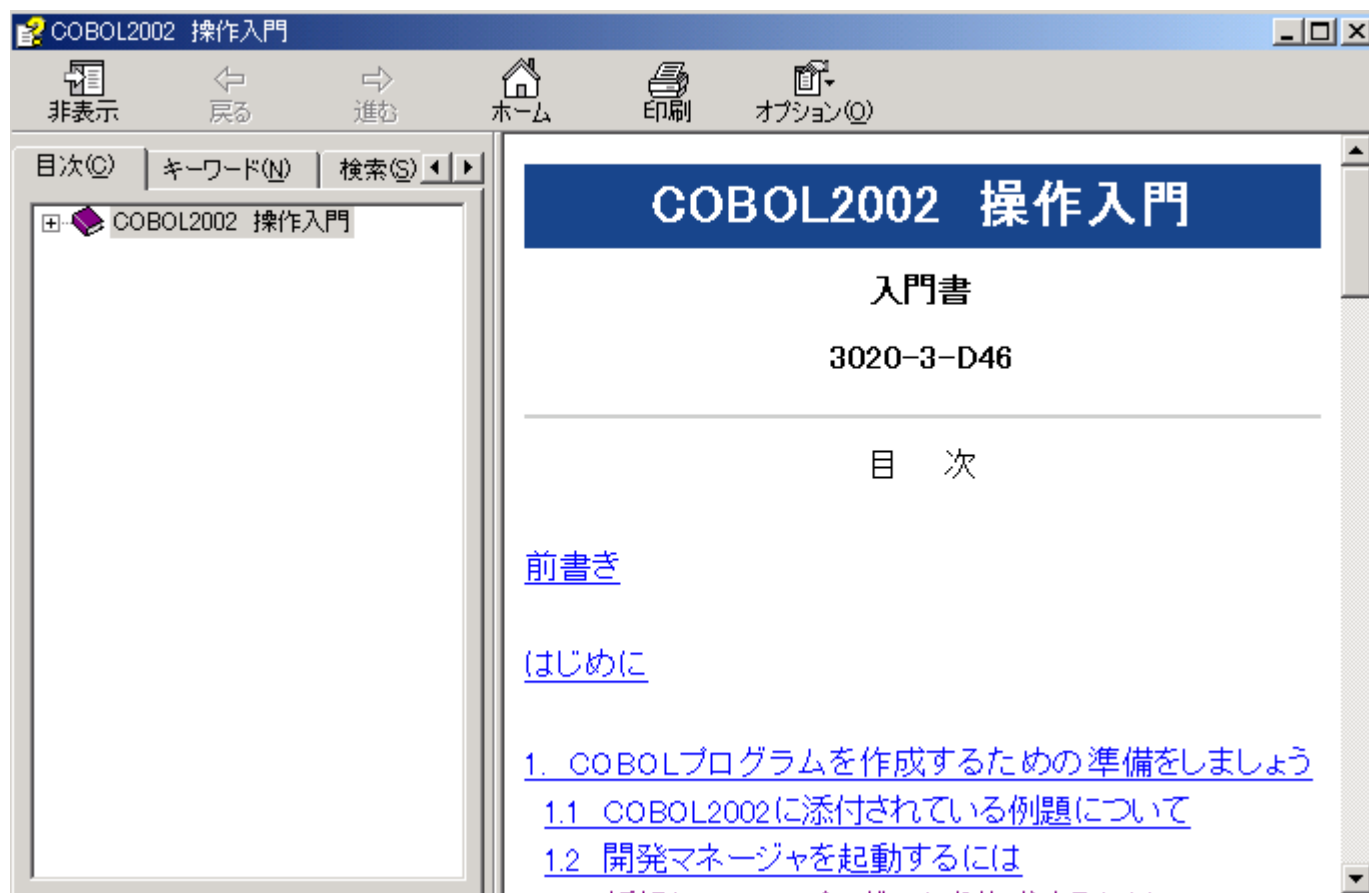




[手順3] COBOL2002の中の使用できるツールの中から、「COBOLマニュアル」の所にマウスポインタを移動すると、COBOLマニュアルの一覧が表示されます。ここで参照したいマニュアルを選択します。

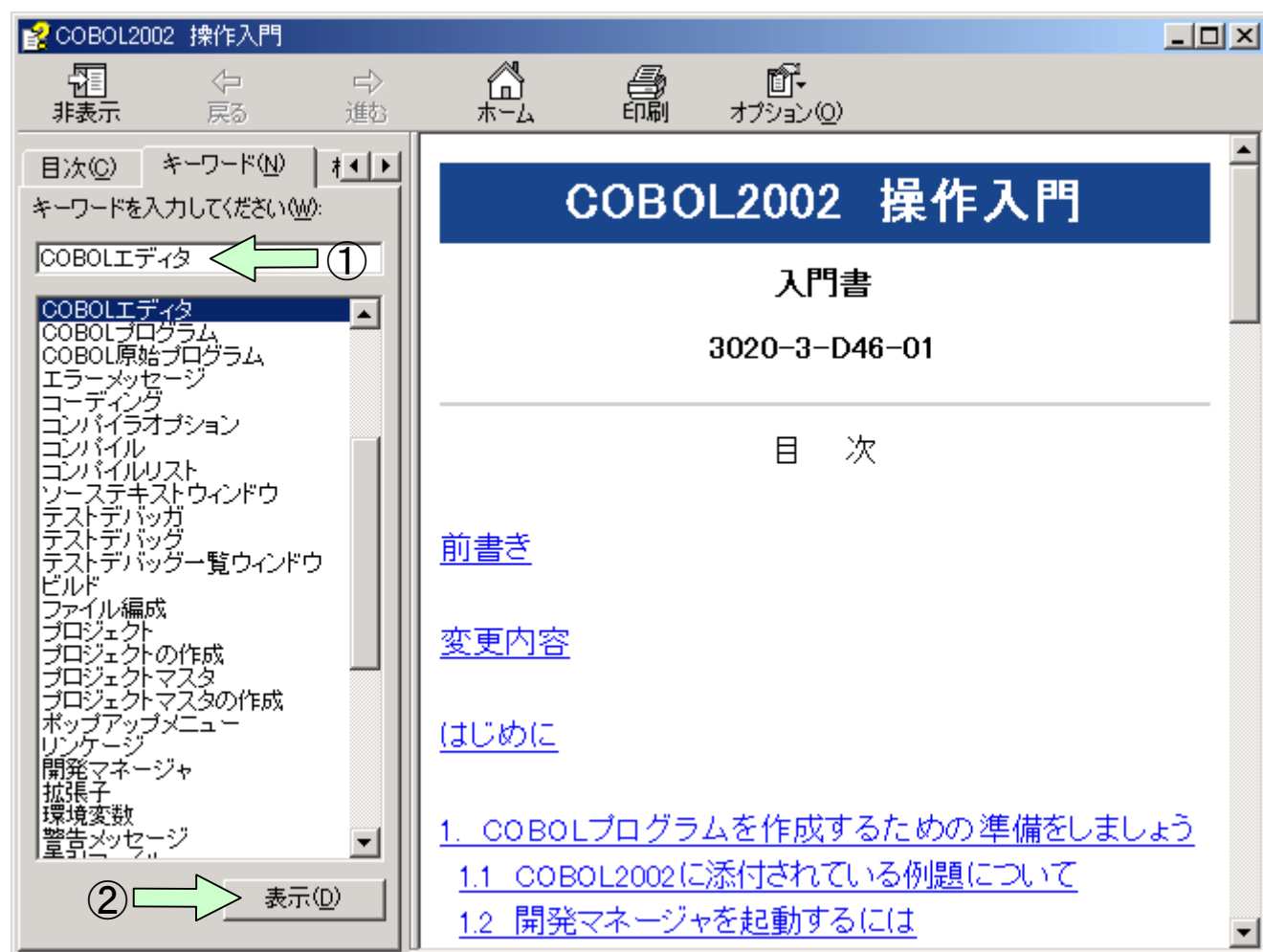


下記のようなCOBOL2002操作入門マニュアル画面が表示されます。「キーワード (N)」、「検索 (S)」機能により探したい項目、文字列を素早く検索することができます。次ページ以降は検索手順について説明します。

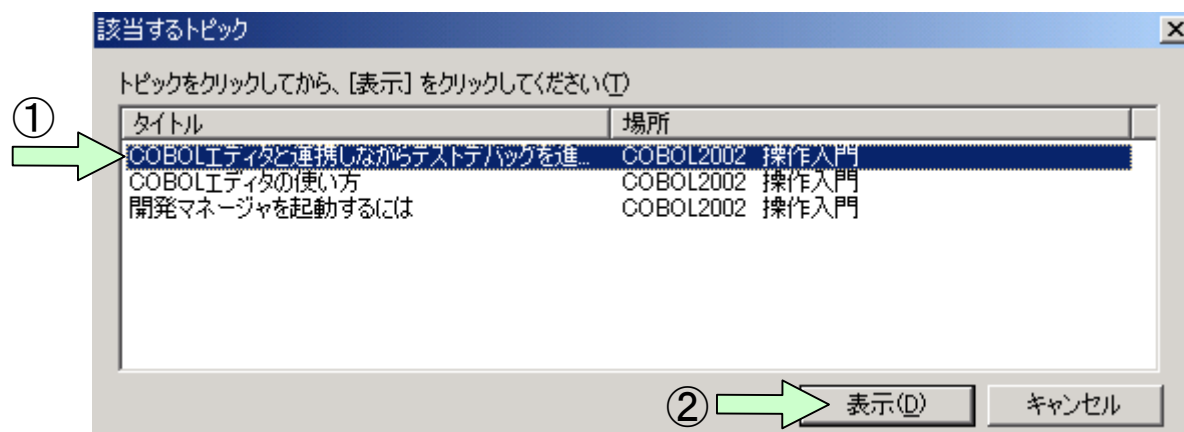


## (a) キーワードによる検索方法

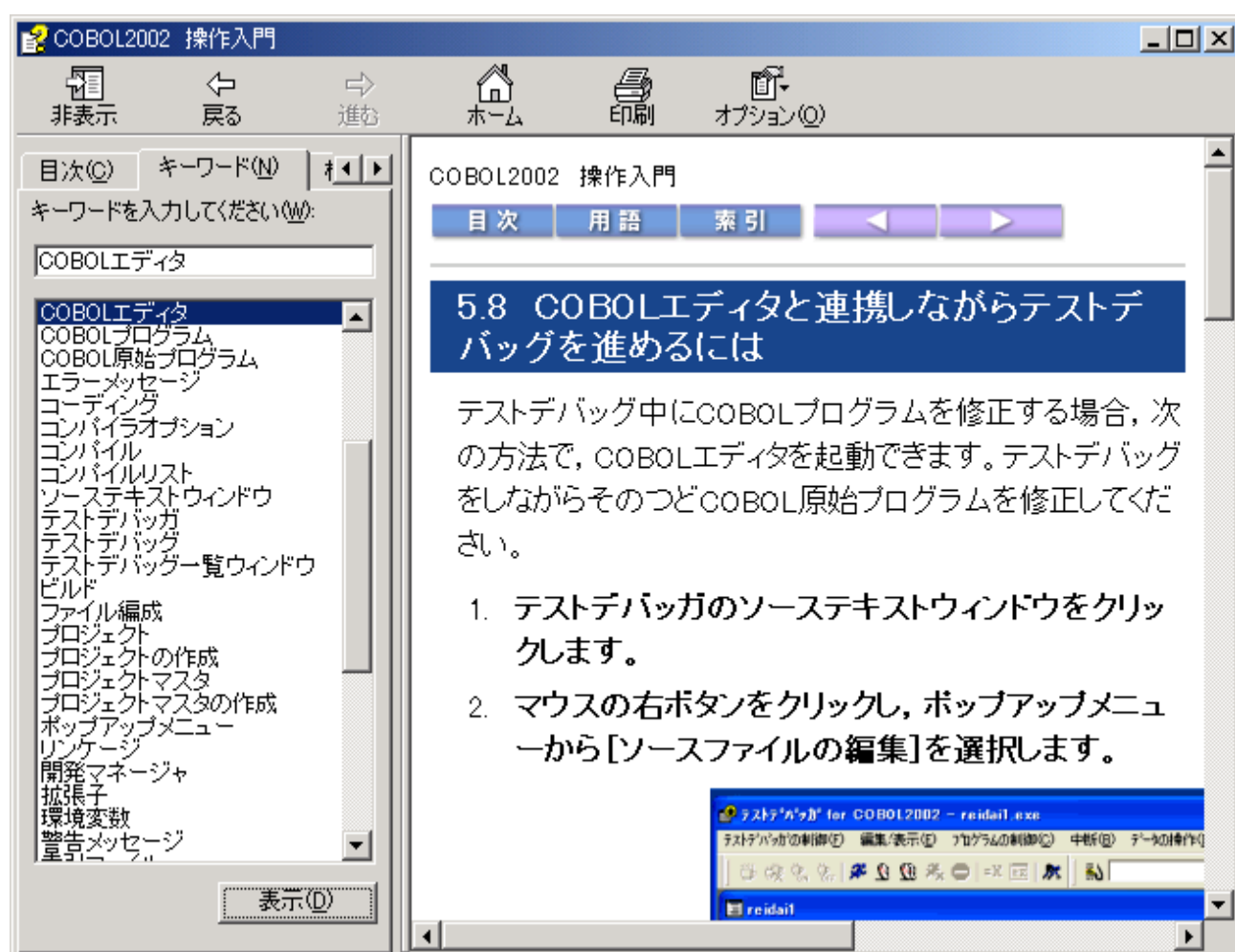
[手順 1] 例えば「COBOLエディタ」に関する項目を探したい場合は、キーワード入力画面で「COBOLエディタ」というキーワードを入力するか、選択画面から該当するキーワードを選択し、「表示」ボタンをクリックします。



[手順 2] すると、トピック画面が表示されますので、該当するトピックをクリックしてから、「表示」をクリックします。



該当する箇所が表示されます。



## (b) 検索語句入力による文字列検索方法

[手順 1] 例えば「COBOL2002」という文字列を探したい場合は、検索項目入力画面で検索文字列「COBOL2002」を入力し、「検索開始」ボタンを押してください。

COBOL2002 操作入門

非表示 戻る 進む ホーム 印刷 オプション(O)

キーワード(N) 検索(S) お気に入りに追加

探したい語句を入力してください(W):

COBOL2002

検索開始(L) 表示(D)

トピックの選択(T): 検索結果: 0

タイトル	場所	ランク
------	----	-----

☐ 以前の結果から検索(U)  
☒ 類似する文字に合致(M)  
☐ タイトルのみ検索(R)

# COBOL2002 操作入門

## 入門書

3020-3-D46

### 目次

[前書き](#)

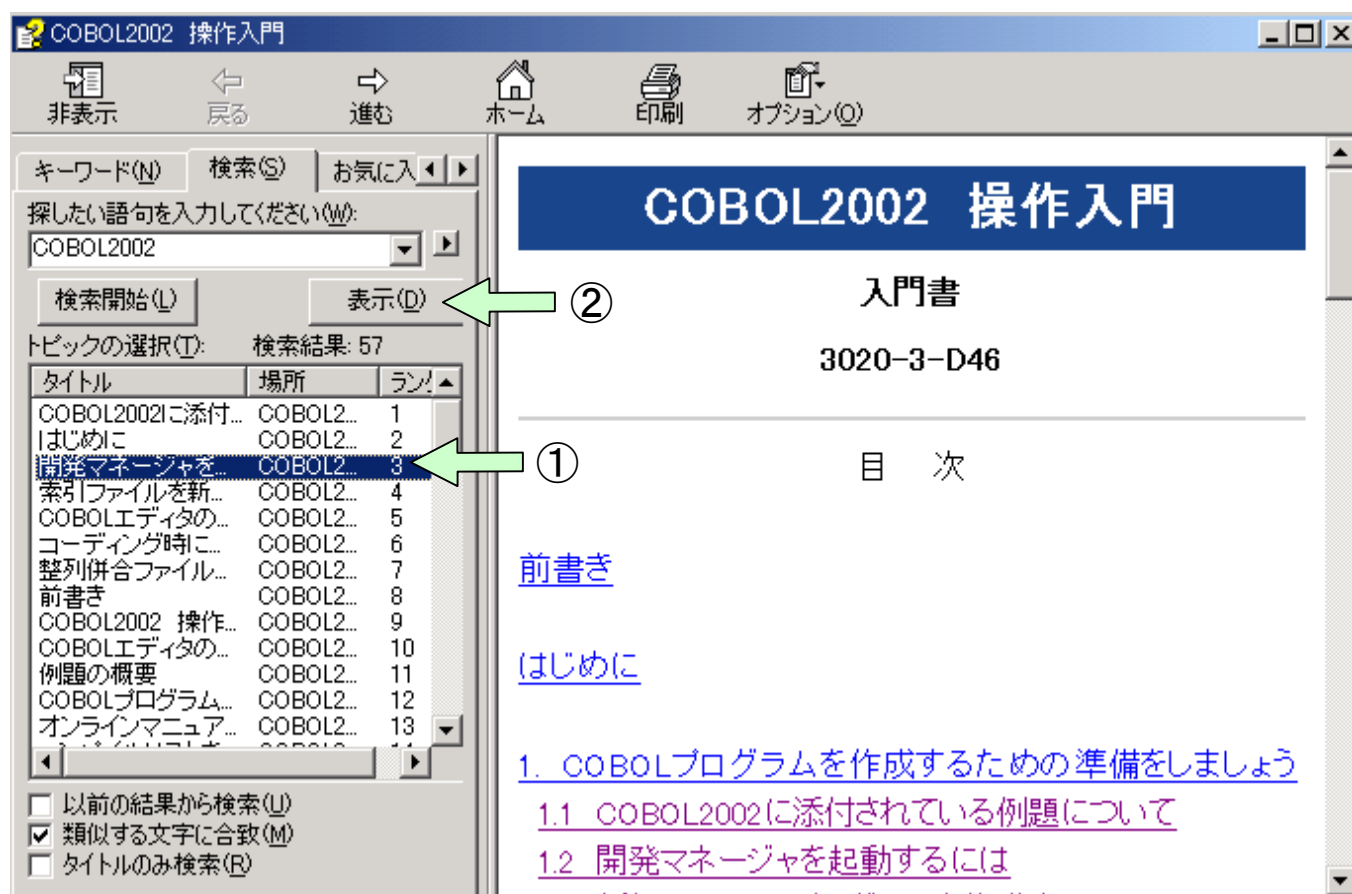
[はじめに](#)

1. COBOLプログラムを作成するための準備をしましょう

1.1 COBOL2002に添付されている例題について

1.2 開発マネージャを起動するには

[手順2] すると、該当するトピック画面が表示されますので、トピックをクリックしてから、「表示」をクリックしてください。



以下のように指定した語句「COBOL2002」がリバーズ表示されます。「キーワード (N)」と「検索 (S)」の違いを次に示します。

- ・ 項目の内容を検索したい場合は「キーワード (N)」を使用します。
- ・ 項目の文字列を検索したい場合は「検索 (S)」を使用します。

COBOL2002 操作入門

目次 用語 索引

## 1.2 開発マネージャを起動するには

COBOLプログラムを効率良く開発するために、**COBOL2002**では開発マネージャを用意しています。

COBOL原始プログラムを編集するためのCOBOLエディタの起動や、COBOLプログラムのコンパイルなど、すべて、この開発マネージャから実行できます。

このマニュアルでは、Windows XPのデフォルトでの表示の場合を例にして説明します。

1. タスクバーの[スタート]ボタンをクリックします。

## **(h) 登録集原文の指定方法**

# 1. 登録集原文の指定方法

複数のCOBOLプログラムに共通の記述を別のソースファイルとしておき、COPY文で取り込むことができます。登録集原文の概念図を下欄に示します。このようにすると、コーディング量を削減でき、記述ミスもなくなるため、開発効率を高めることができます。ここでは、第1章で実習した「reidai1.cbl」というソースファイルの「DATA0」と「YYMMDD」を登録集原文としたものとして説明します。このソースファイルを「reidai2.cbl」とします。

登録集原文はソースファイルとは別のフォルダにまとめて格納するのが一般的です。コンパイルする際には、「CBLLIB」というコンパイル時環境変数で登録集原文が格納されているフォルダを指定します。

<reidai2.cbl>

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    reidai2.
:
DATA DIVISION.
WORKING-STORAGE SECTION.
*
COPY DATA0.
*
COPY YYMMDD.
*
PROCEDURE DIVISION.
:
```

登録集原文

<DATA0.cbl>

```
01 DATA0.
02 DATA1 PIC X(10) VALUE ALL '*'.
02 DATA2 PIC X(20) VALUE SPACE.
02 DATA3 PIC X(10) VALUE ALL '*'.
```

<YYMMDD.cbl>

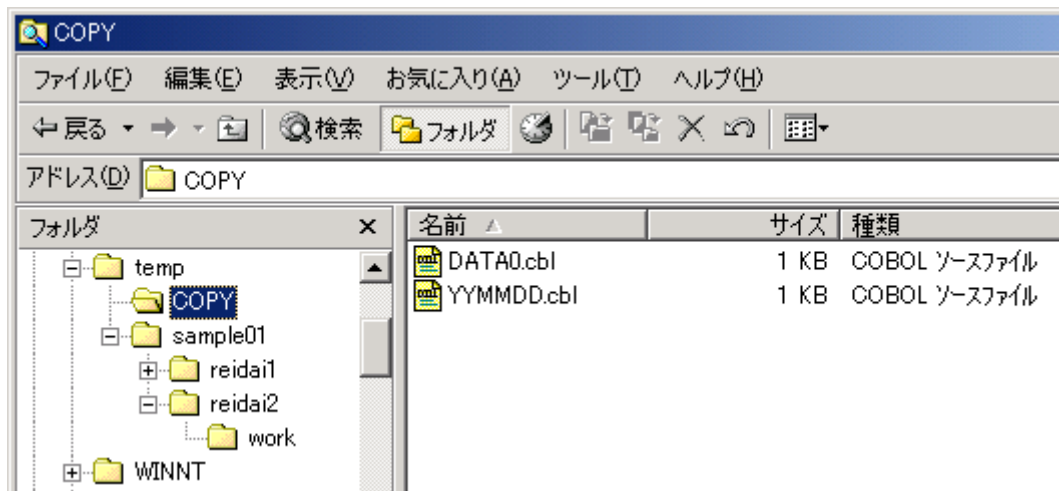
```
01 YYMMDD.
02 年 PIC 9(2).
02 月 PIC 9(2).
02 日 PIC 9(2).
```

:



フォルダの構成の例を次に示します。

ここでは、「COPY」というフォルダに登録集原文を格納しています。



[手順 1] reidai2というプロジェクトをこれまでと同様の手順で作成します。  
 ソースファイル「reidai2. cbl」を次に示します。  
 「reidai2. cbl」は、「reidai1. cbl」の「DATA0」と「YYMMDD」をCOPY文に変更したものです。

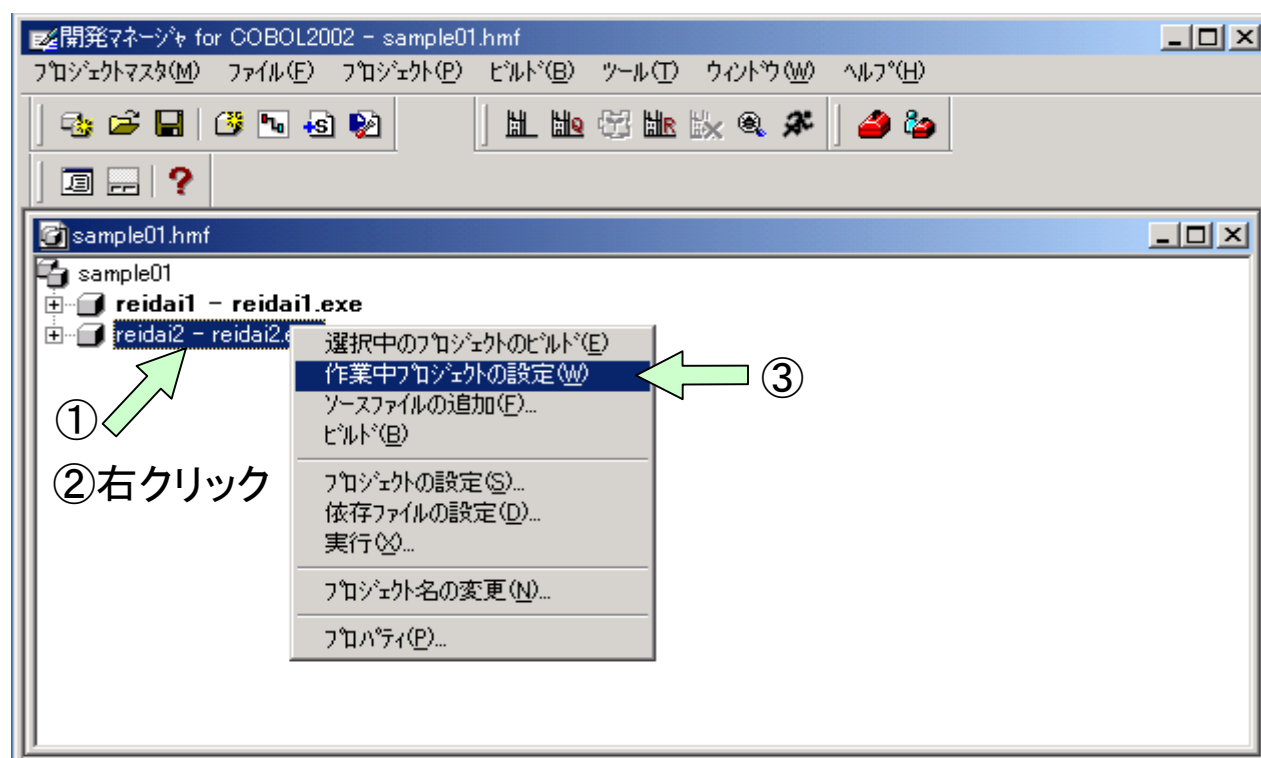
```

000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.      reidai2.
000300*
000400 ENVIRONMENT  DIVISION.
000500*
000600 DATA    DIVISION.
000700 WORKING-STORAGE  SECTION.
000800*
000900 COPY  DATA0.
001000*
001100 COPY  YYMMDD.
001200*
001300 PROCEDURE  DIVISION.

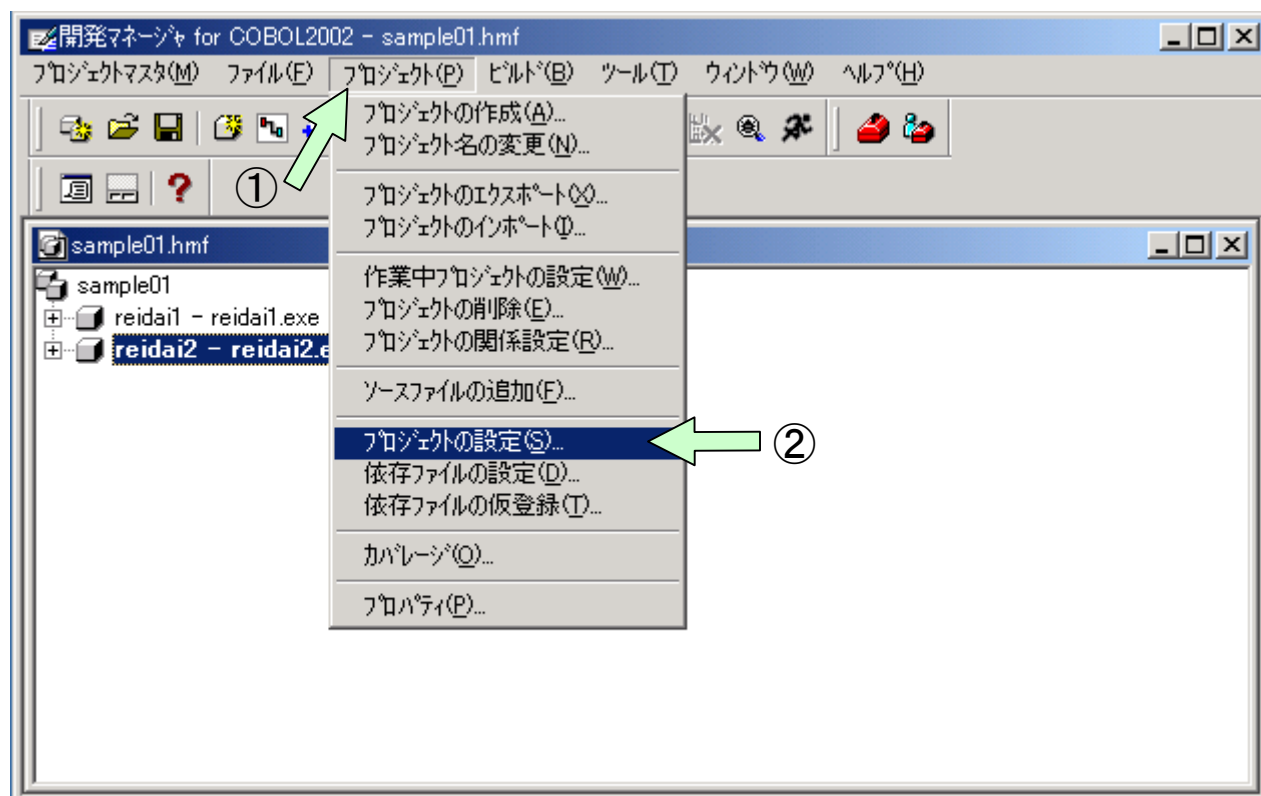
```


以降は「reidai1. cbl」と同じです。

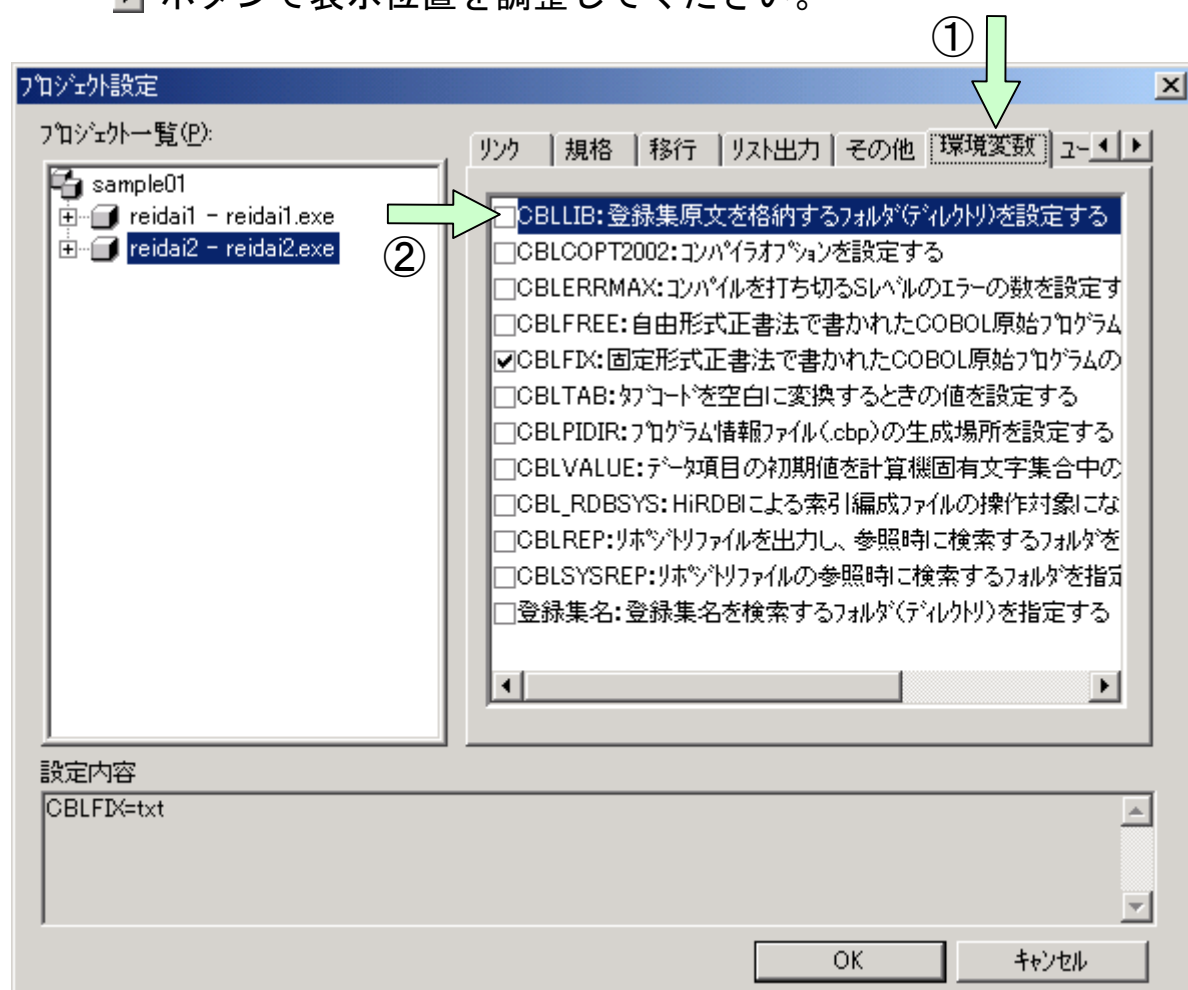
[手順2] プロジェクト「reidai2」を作成したら、作業中のプロジェクトを「reidai2」に設定します。「reidai2」をポイントし右クリックし、プルダウンメニューの中から「作業中プロジェクトの設定(W)」を選択します。



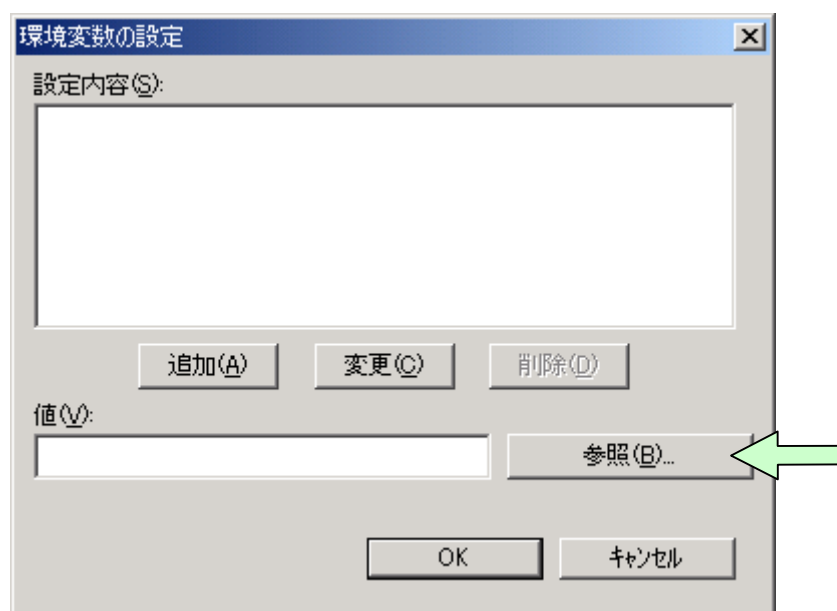
[手順3] 次に環境変数「CBLLIB」で登録集原文のフォルダを指定します。開発マネージャのメニューバーの「プロジェクト(P)」をクリックし、プルダウンメニューの「プロジェクトの設定(S)」を選択します。



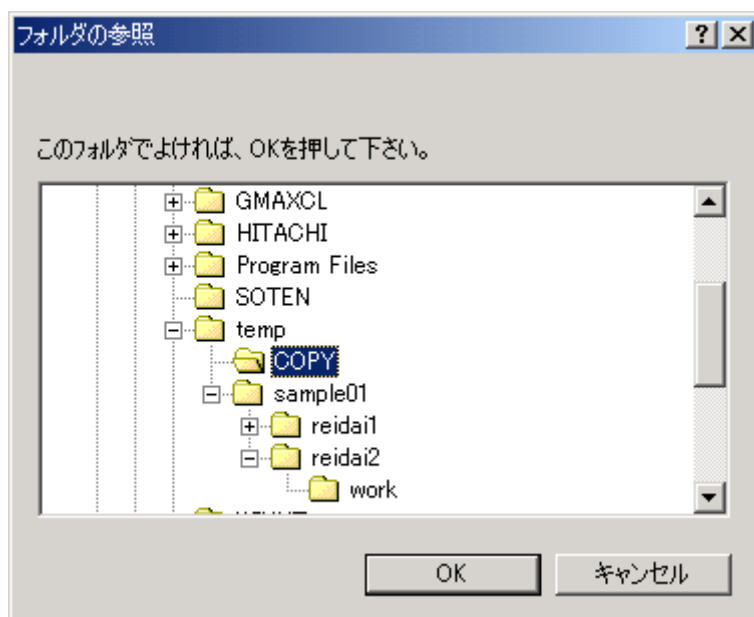
- [手順4] 「プロジェクトの設定」画面が表示されるので、「環境変数」タブをクリックし、「CBLLIB」をチェックします。「環境変数」タブが見えないときは、 ボタンで表示位置を調整してください。



- [手順5] 「環境変数の設定」画面が表示されるので、「参照(B)」ボタンをクリックします。

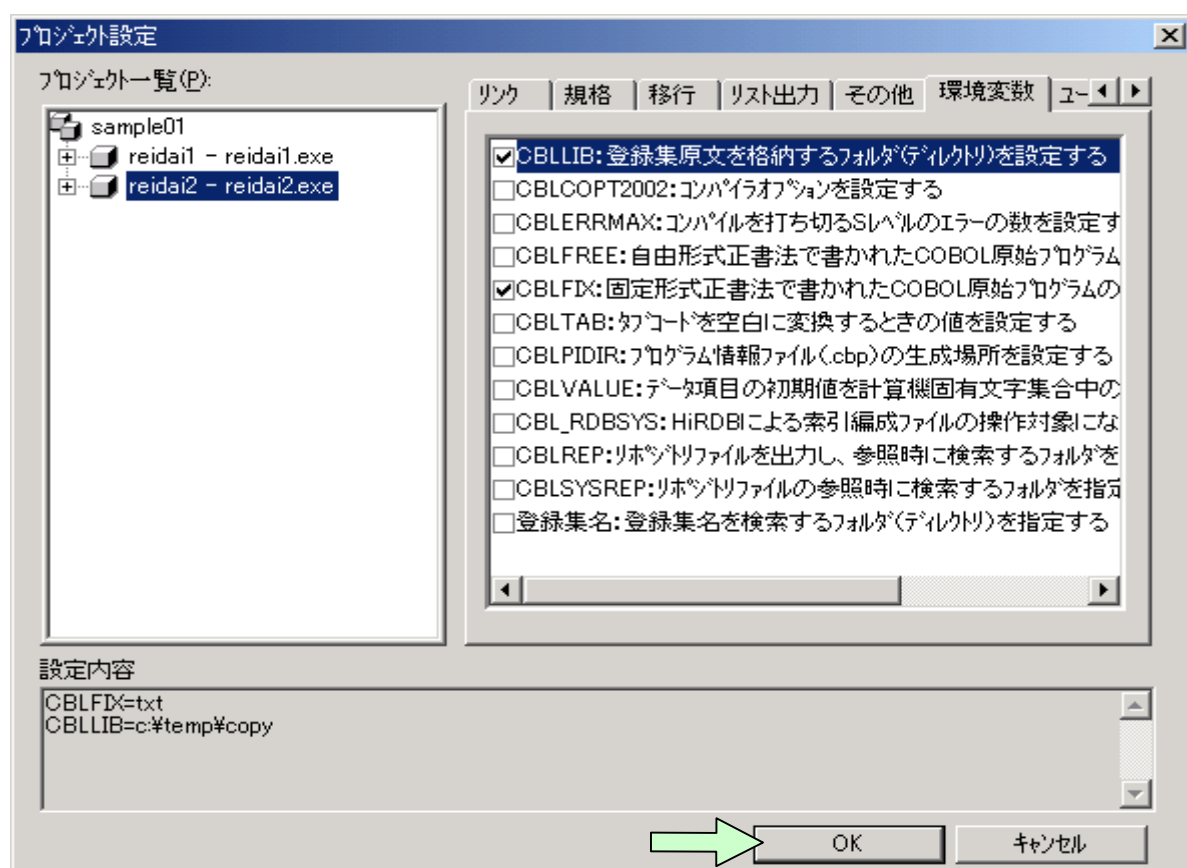


[手順6] 「フォルダの参照」画面が表示されるので、「COPY」フォルダを指定して「OK」ボタンをクリックします。

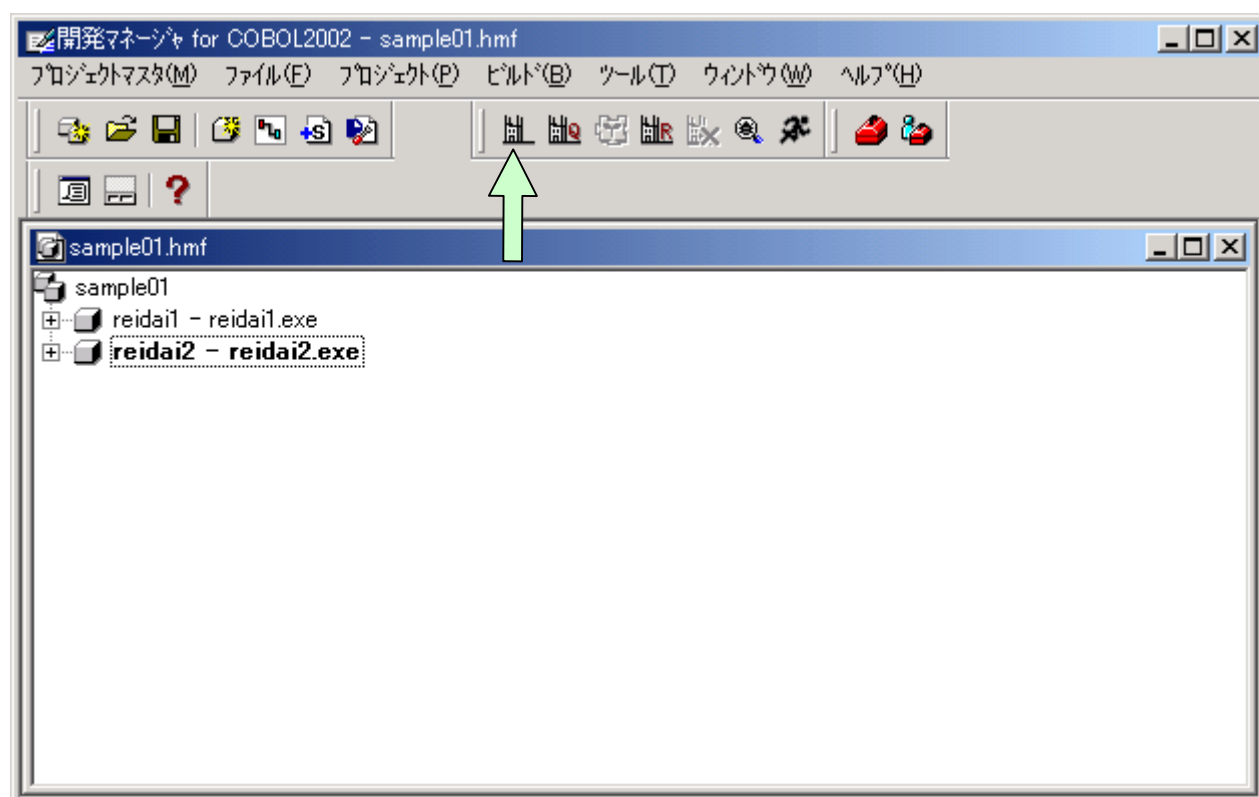


「OK」ボタンをクリックします。

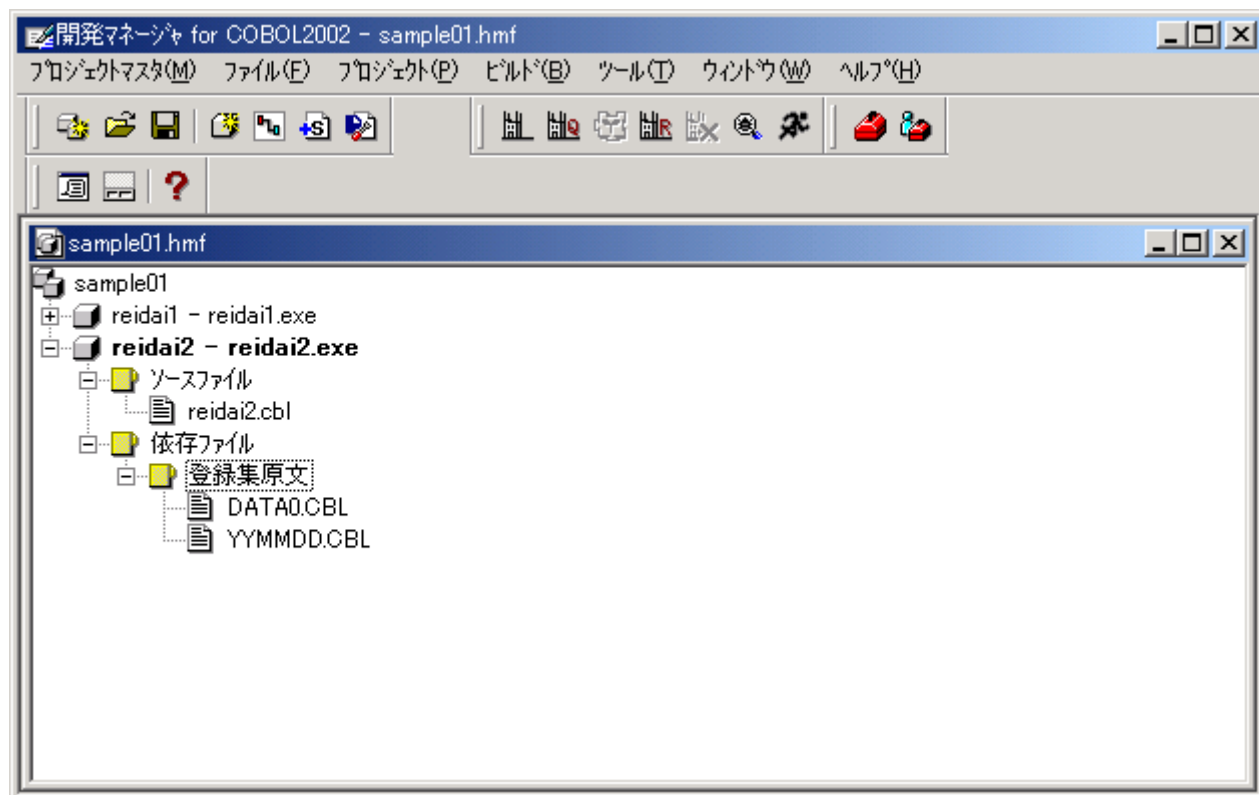
[手順7] 「プロジェクトの設定」画面に戻るので、「OK」ボタンをクリックします。



[手順8] ビルドします。



[手順9] プロジェクト「reidai2」を全て表示すると、依存ファイルとして登録集原文「DATA0.cbl」と「YYMMDD.cbl」が登録されているのがわかります。ビルドすると、開発マネージャがCOPY文を認識して自動的に依存関係を構築するのです。



## (i)サブプログラムの追加方法

1つのメインプログラムと複数のサブプログラムから構成される場合のプロジェクトの設定方法

# 1. プロジェクトの設定方法

ここでは、一つのメインプログラムと複数のサブプログラムから構成される場合のプロジェクトの設定方法について説明します。

(プロジェクト「reidai3」を例題として説明します。)

第1章で実習した「reidai1」のプログラムの「初期処理」、「比較処理」、「出力処理」を各サブプログラムとして、メインプログラムから呼ばれるようにした例で設定方法を示します。

各サブプログラムの名称は次のとおりです。

## ■メインプログラム : reidai3. cbl

```

:
PROCEDURE DIVISION.
Mein-Sec SECTION.
    CALL 'REIDAI3A' USING YYMMDD.
    CALL 'REIDAI3B' USING DATA0 YYMMDD.
    CALL 'REIDAI3C' USING DATA0.
    STOP RUN.

```

## ■サブプログラム「初期処理」 : reidai3a. cbl

```

IDENTIFICATION DIVISION.
PROGRAM-ID. reidai3a.
ENVIRONMENT DIVISION.
DATA DIVISION.
LINKAGE SECTION.
01 YYMMDD.
    02 年 PIC 9(2).
    02 月 PIC 9(2).
    02 日 PIC 9(2).
PROCEDURE DIVISION USING YYMMDD.
初期処理 SECTION.
    ACCEPT YYMMDD FROM DATE.
    EXIT PROGRAM.

```

## ■サブプログラム「比較処理」 : reidai3b. cbl

```

IDENTIFICATION DIVISION.
PROGRAM-ID. reidai3b.
ENVIRONMENT DIVISION.
DATA DIVISION.
LINKAGE SECTION.
01 DATA0.
    02 DATA1 PIC X(10).
    02 DATA2 PIC X(20).
    02 DATA3 PIC X(10).
01 YYMMDD.
    02 年 PIC 9(2).
    02 月 PIC 9(2).
    02 日 PIC 9(2).
PROCEDURE DIVISION USING DATA0 YYMMDD.
比較処理 SECTION.
    IF 月 = 9
    THEN
        MOVE 'September!!' TO DATA2
    ELSE
        MOVE 'Not September!!' TO DATA2
    END-IF.
    EXIT PROGRAM.

```

## ■サブプログラム「出力処理」 : reidai3c. cbl

```

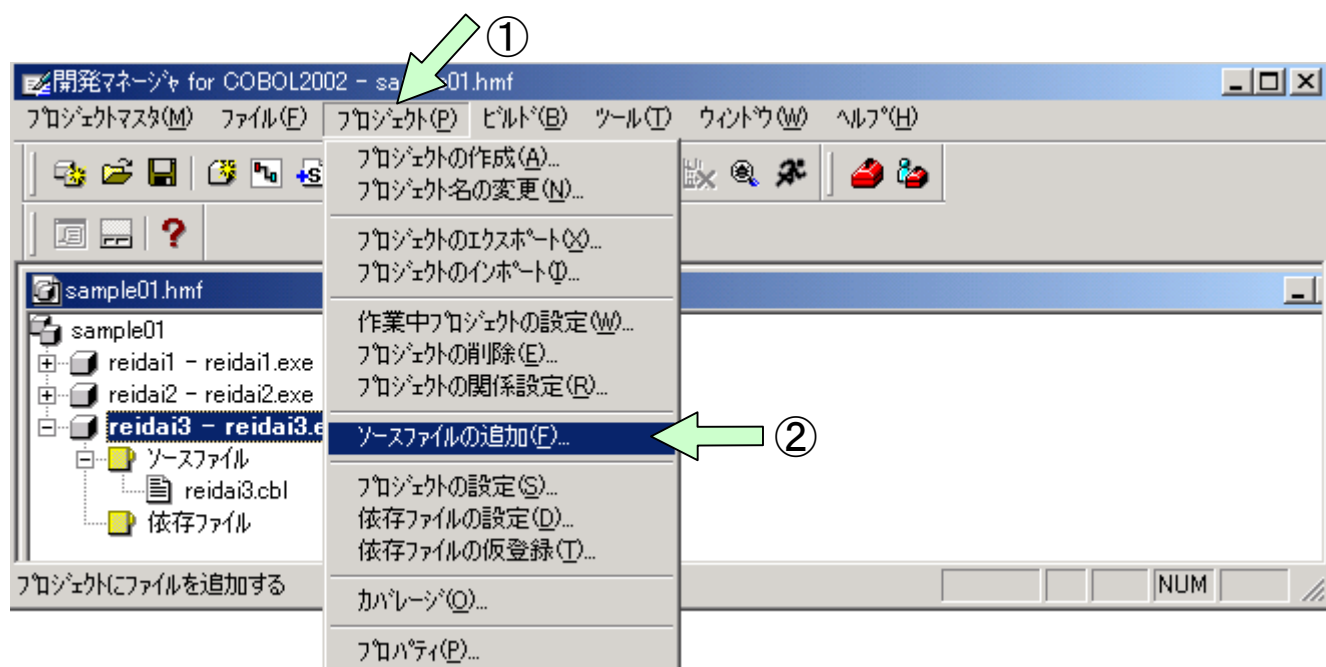
IDENTIFICATION DIVISION.
PROGRAM-ID. reidai3a.
ENVIRONMENT DIVISION.
DATA DIVISION.
LINKAGE SECTION.
01 DATA0.
    02 DATA1 PIC X(10).
    02 DATA2 PIC X(20).
    02 DATA3 PIC X(10).
PROCEDURE DIVISION USING DATA0.
出力処理 SECTION.
    DISPLAY DATA0.
    EXIT PROGRAM.

```

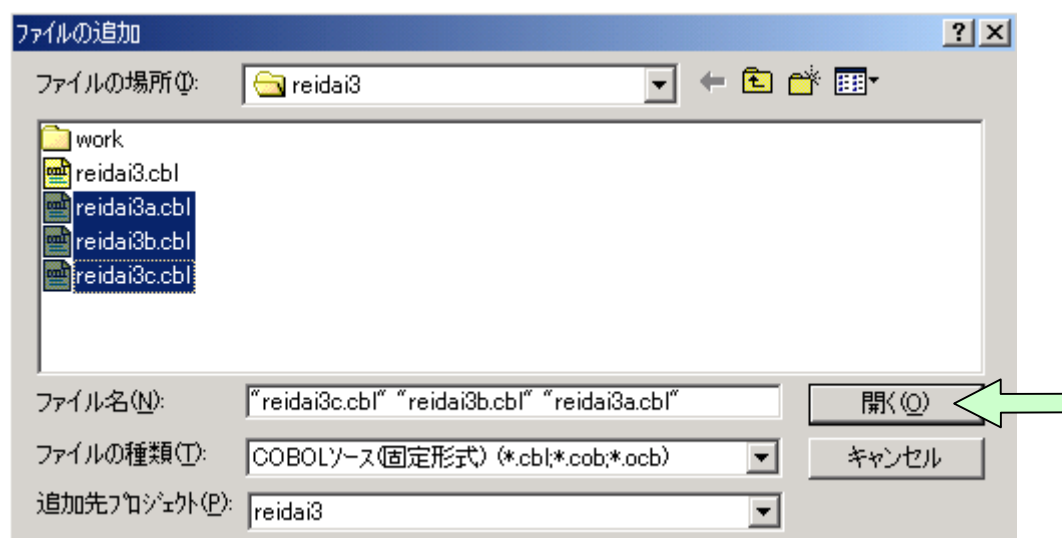
メインプログラムのプロジェクト「reidai3」は、これまで示した手順で作成し、各サブプログラムはあらかじめ「reidai3. cbl」と同じフォルダに格納してあるものとして手順を示します。



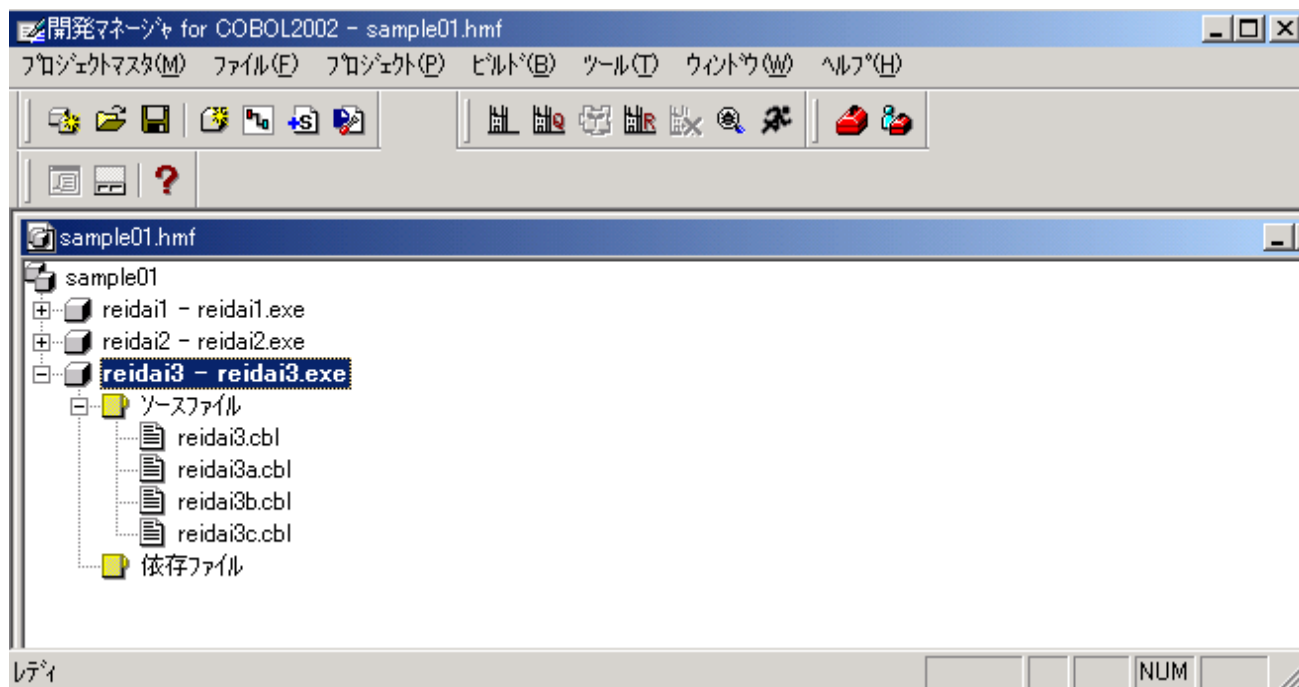
[手順 1] 開発マネージャのメニューバーの「プロジェクト(P)」をクリックし、プルダウンメニューの中の「ソースファイルの追加(F)」をクリックします。  
すると、ファイルの追加画面が表示されます。



[手順 2] ここで追加するファイルを選択し、開くボタンを押下します。  
複数のサブプログラムを追加できます。



[手順3] reidai3のプロジェクトに3つのサブプログラムが追加されました。  
 この後、ビルドを行います。ビルドすると4つのプログラムが結合されて一つの実行可能ファイル(.exe)になります。エラーがなければ続いて実行してください。



#### [注意事項]

- ①メインプログラムの終了は「STOP RUN」文、サブプログラムの終了は「EXIT PROGRAM」文を使用します。
- ②メイン側のプログラムからサブプログラムを呼び出す「CALL」文では、サブプログラム名をアポストロフィで囲んで指定します。

[例] CALL 'REIDAI3A' USING DATA0 YYMMDD.

プログラム名は基本的に大文字を使用します。これは、サブプログラムの「PROGRAM-ID」段落で定義したプログラム名が大文字と解釈されるからです。

[例] PROGRAM-ID. reidai3a. <-- プログラム名は「REIDAI3A」と解釈

なお、「PROGRAM-ID」段落のプログラム名をアポストロフィで囲んで指定すると、指定したとおりに解釈されます。

[例] PROGRAM-ID. 'reidai3a'. <-- プログラム名は「reidai3a」と解釈

- ③サブプログラムで受け取るデータは「LINKAGE SECTION」として定義します。「LINKAGE SECTION」では、「VALUE」句は使用できません。

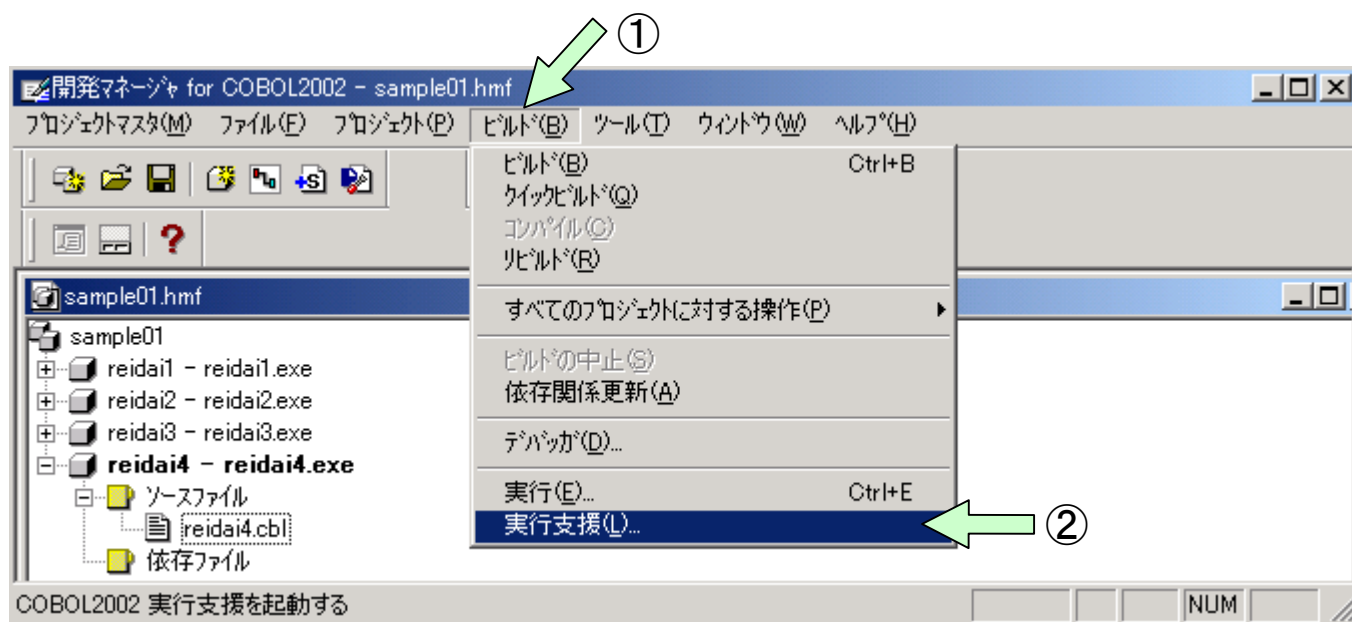
## (j)索引ファイルを新規作成 する方法

索引ファイルを新規に作成する方法について説明します。

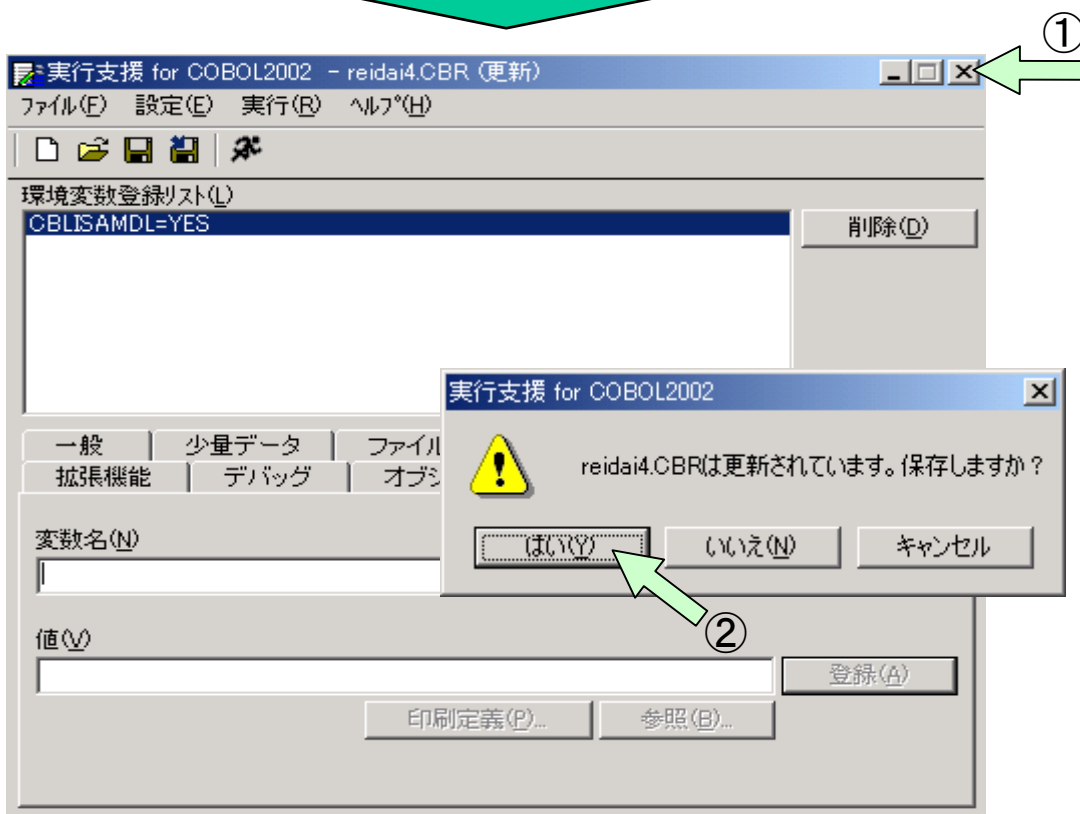
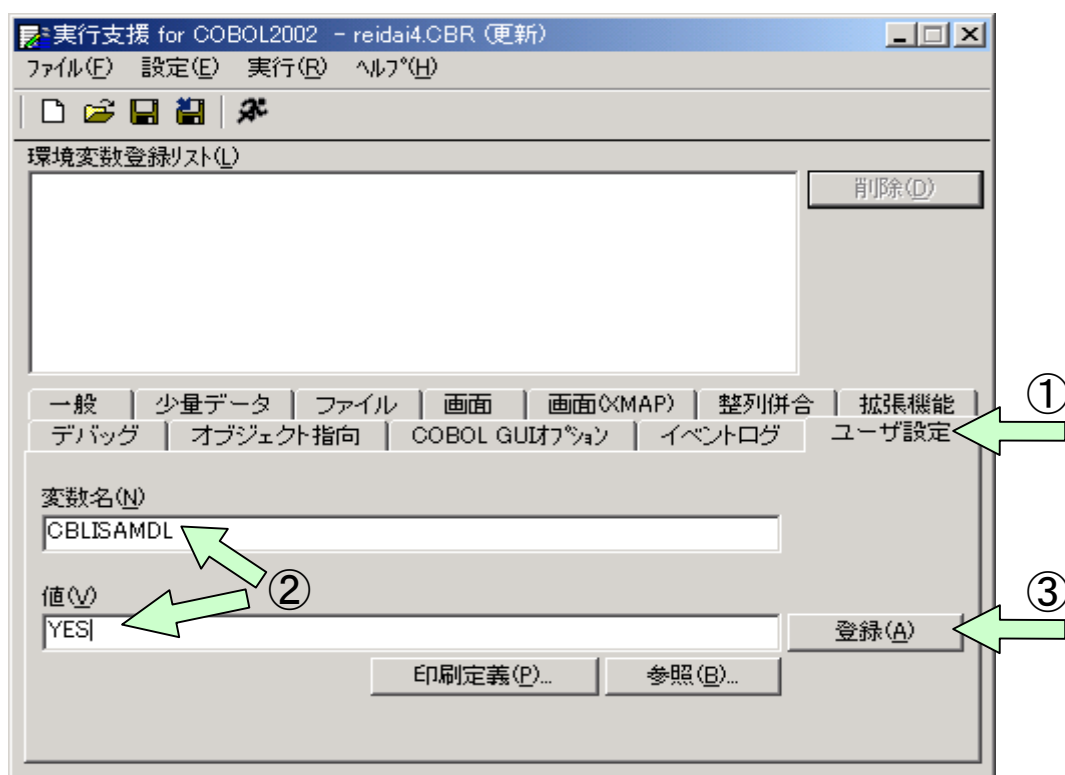
「OPEN OUTPUT」で開いてWRITE文でレコードを書き出すと、通常は新規作成になりますが、索引ファイルの場合は追加モードでレコードが書き出されます。すなわち、一度索引ファイルを作成した後、同じキーで索引ファイルを作成し直そうとしても、「重複キー」エラーになってしまいます。

索引ファイルの新規作成をするときは、実行時環境変数「CBLISAMDL=YES」を指定します。これにより、既存の索引ファイルが削除され、新規作成することが出来ます。

〔手順 1〕 開発マネージャのメニューバーの「ビルド (B)」をクリックし、プルダウンメニューの中の「実行支援 (L)」をクリックします。すると、実行支援画面が表示されます。



- [手順 2] 実行時環境変数は、「ユーザ設定」タブで設定します。  
 変数名に「CBLISAMD L」、値に「YES」と入力して「登録」ボタンを押すと、  
 「環境変数登録リスト(L)」に登録されます。閉じる(×)ボタンを押すと  
 「保存しますか?」と聞いてきますので、必ず「はい」をクリックして  
 ください。



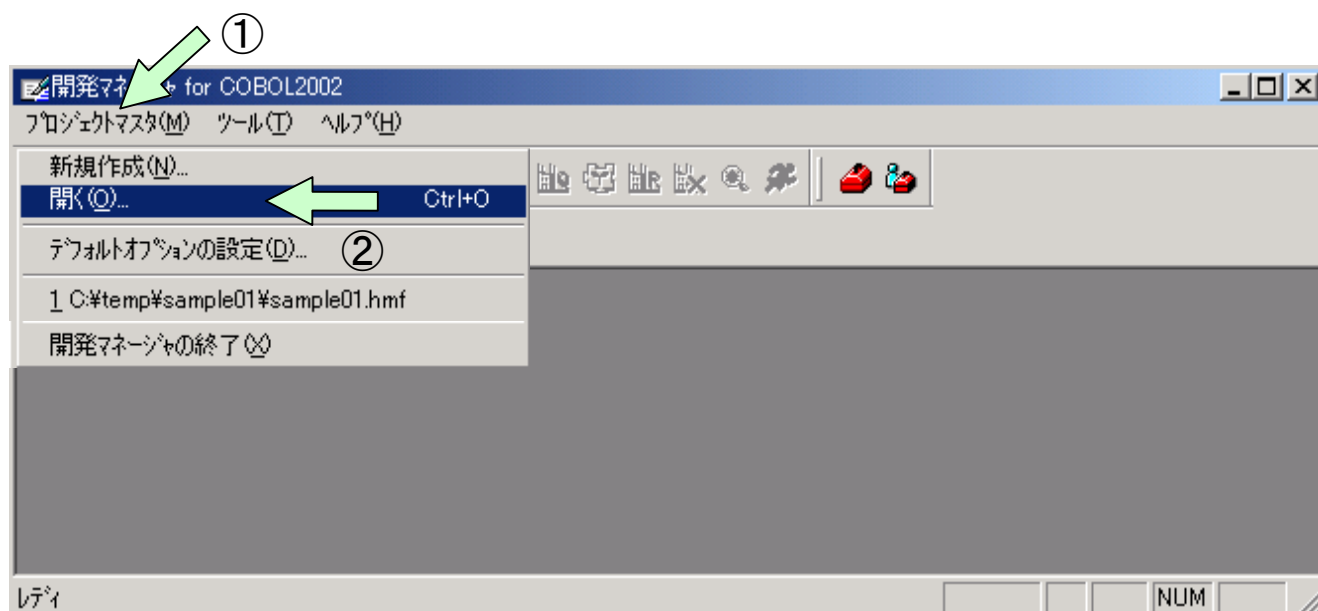
## (k) 既存のプロジェクトマスタ ファイルの開き方

ー既に作成済みのプロジェクトマスタファイルを開くにはー

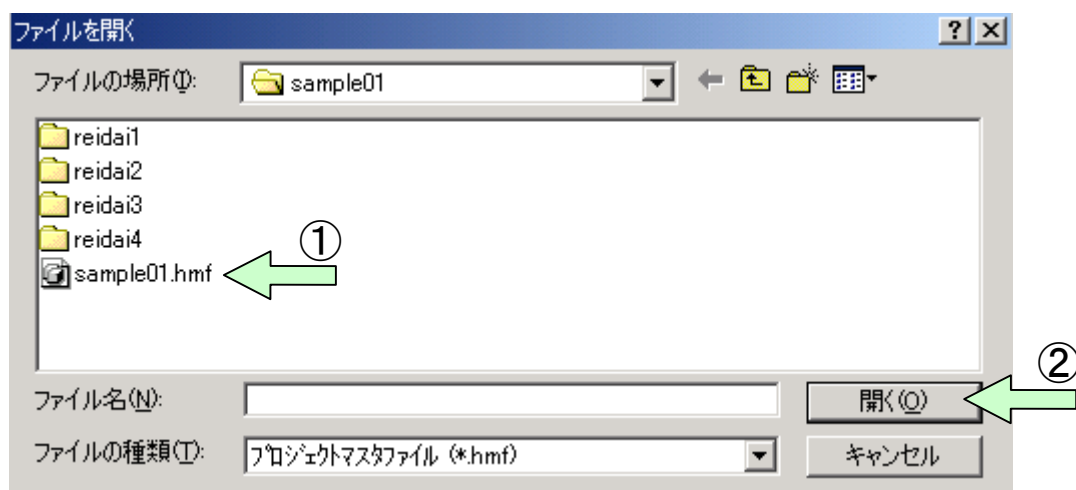
# 1. 既存のプロジェクトマスタファイルの開き方

ここでは、既に作成済みのプロジェクトマスタファイルの開き方について説明します。

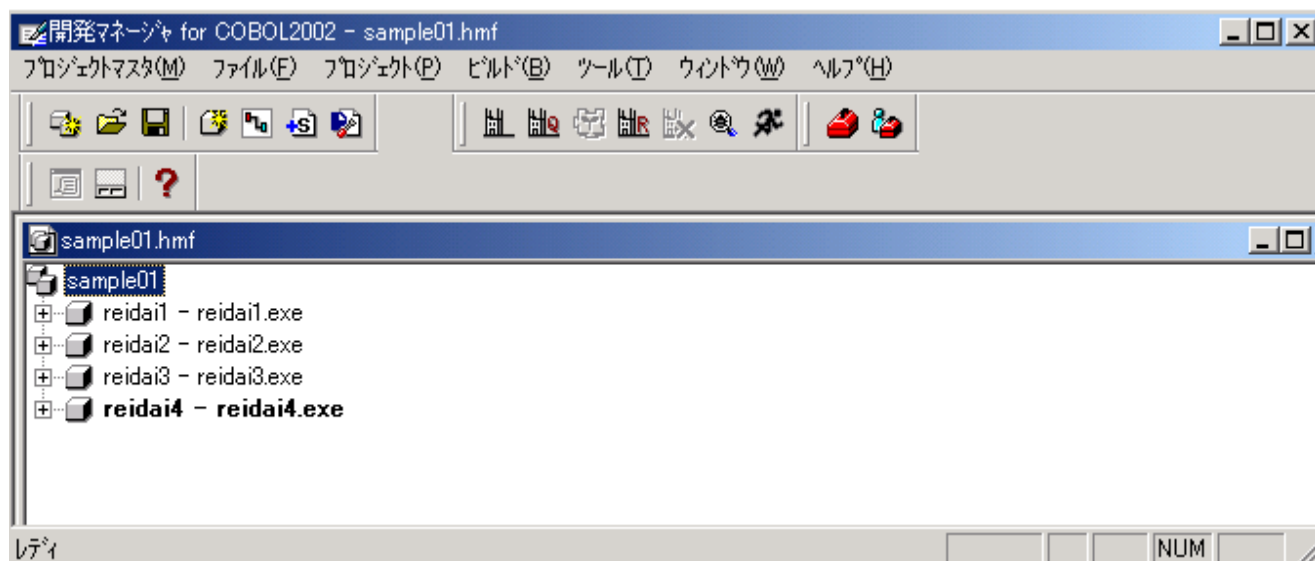
[手順 1] 開発マネージャのメニューバーの「プロジェクトマスタ(M)」、「開く(O)」をクリックします。すると「ファイルを開く」画面が表示されます。



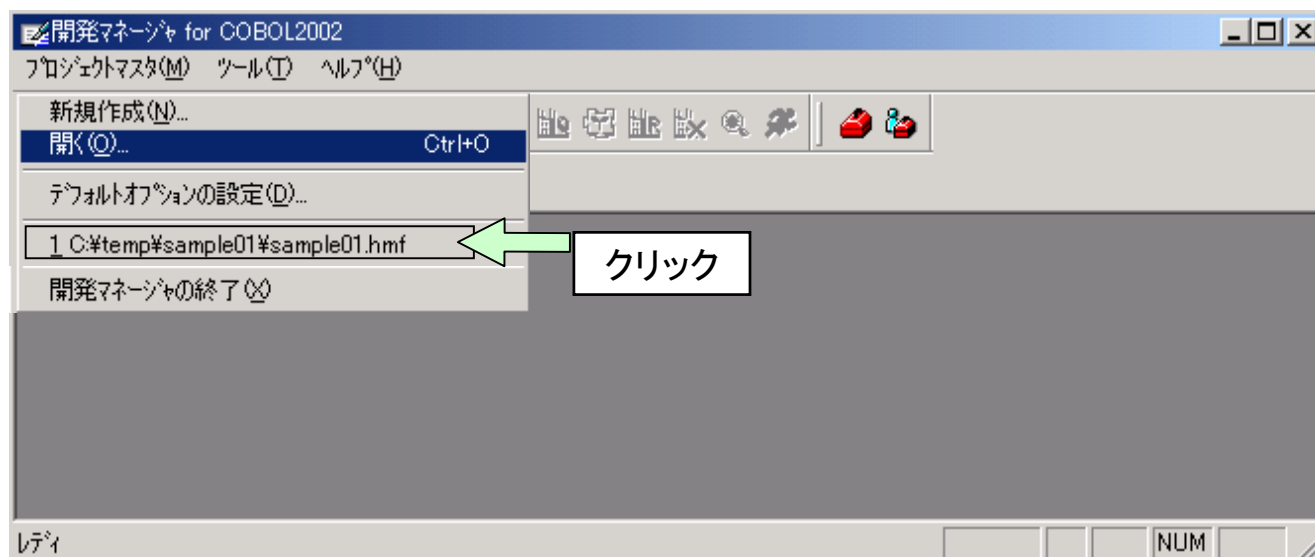
[手順 2] ここで開きたいプロジェクトマスタファイルをクリックし、「開く(O)」ボタンを押します。すると選択したプロジェクトマスタファイルが開きます。下記の例では「SAMPLE01」というプロジェクトマスタファイルを開きます。



[手順3] プロジェクトマスタファイル「sample01」が開けられました。



[補足] 手順1で、「プロジェクトマスタ(M)」をクリックしたとき、最近使用したプロジェクトマスタファイルが表示されます。開きたいファイルが表示されているときは、そのファイルをクリックしてください。  
あるいは、エクスプローラで直接プロジェクトマスタファイル(. hmf)をダブルクリックして開くこともできます。



#### 《他社所有名称に対する表示》

- ・ Btrieveは、米国Pervasive, Inc. の商品名称です。
- ・ ODBCは、米国Microsoft Corp. が提唱するデータベースアクセス機構です。
- ・ OLEは、米国Microsoft Corp. が開発したソフトウェア名称です。OLEは、Object Linking and Embeddingの略です。
- ・ Windowsは、米国およびその他の国における米国Microsoft Corp. の登録商標です。
- ・ その他記載の会社名・製品名は、それぞれの会社の商標もしくは登録商標です。