
Micro Focus Visual COBOL 6.0 チュートリアル

COBOL 開発：ステップバイステップチュートリアル

Eclipse

1. 目的

Micro Focus Visual COBOL for Eclipse は、高機能なオープンソースの IDE（統合開発環境）として広く普及する Eclipse 上で COBOL のアプリケーション開発を行うための製品です。COBOL プログラムが既存の COBOL 資産を Windows、UNIX/Linux といったオープン環境で活用するだけでなく、COBOL プログラミング経験のないプログラマが初めて COBOL アプリケーション開発を行う場合に最適な製品です。

このドキュメントは、Micro Focus Visual COBOL for Eclipse を学ぶためのステップバイステップのチュートリアルです。

2. 前提

- 本チュートリアルで使用したマシン OS : Windows 10
- Visual COBOL 6.0J for Eclipse がすでにインストールされていること
インストール手順は以下の FAQ サイトを参照してください。
- https://support.amc.rocketsoftware.co.jp/SupportInf/amc_faqpublic.aspx?VC01002
- プログラミングの基礎知識を有していること
- Windows の基本操作を理解していること

下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダに解凍しておいてください。

[サンプルプログラムのダウンロード](#)

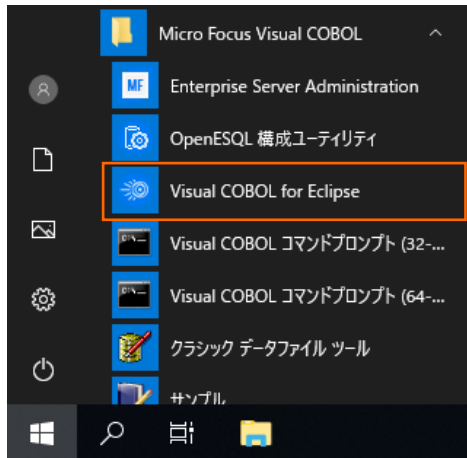
内容

1. 目的
2. 前提
3. チュートリアル
 - 3.1. Eclipse の IDE に慣れてみよう
 - 3.2. はじめての Visual COBOL プロジェクト
 - 3.3. ファイルの入出力

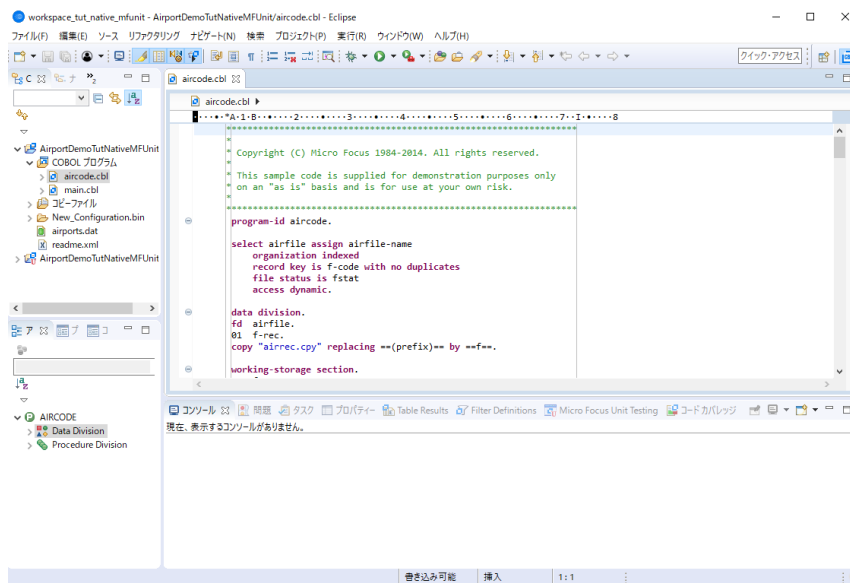
3. チュートリアル

3.1. Eclipse の IDE に慣れてみよう

- 1) スタートメニューより、Eclipse を起動します。

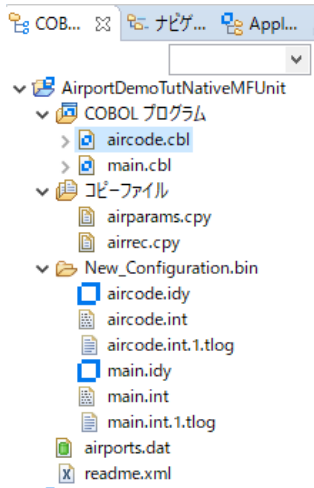


- 2) Eclipse を起動して任意の COBOL プロジェクトを作成すると以下のような画面が表示されます。

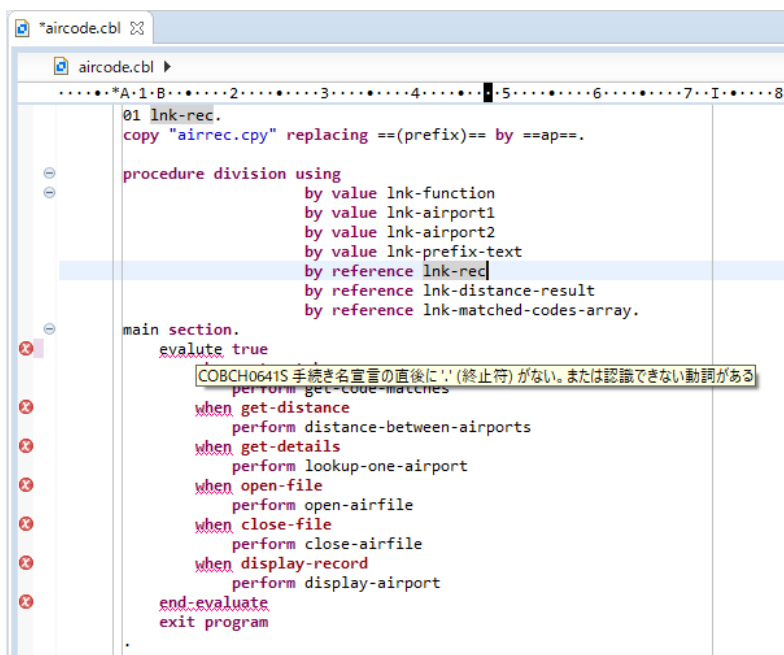


Eclipse の IDE は、ビュー、エディター、ツールバー、メニューバー、ステータスバー、パースペクティブから構成されるワークベンチというウィンドウ内で作業します。パースペクティブは、行いたい作業によって切り替えて利用するワークベンチのレイアウト（表示するビュー、メニュー、ツールバーの種類や場所）のことを指します。パースペクティブを切り替えることによって目的の作業に応じた構成要素を揃えることができます。各要素の配置はカスタマイズ可能です。

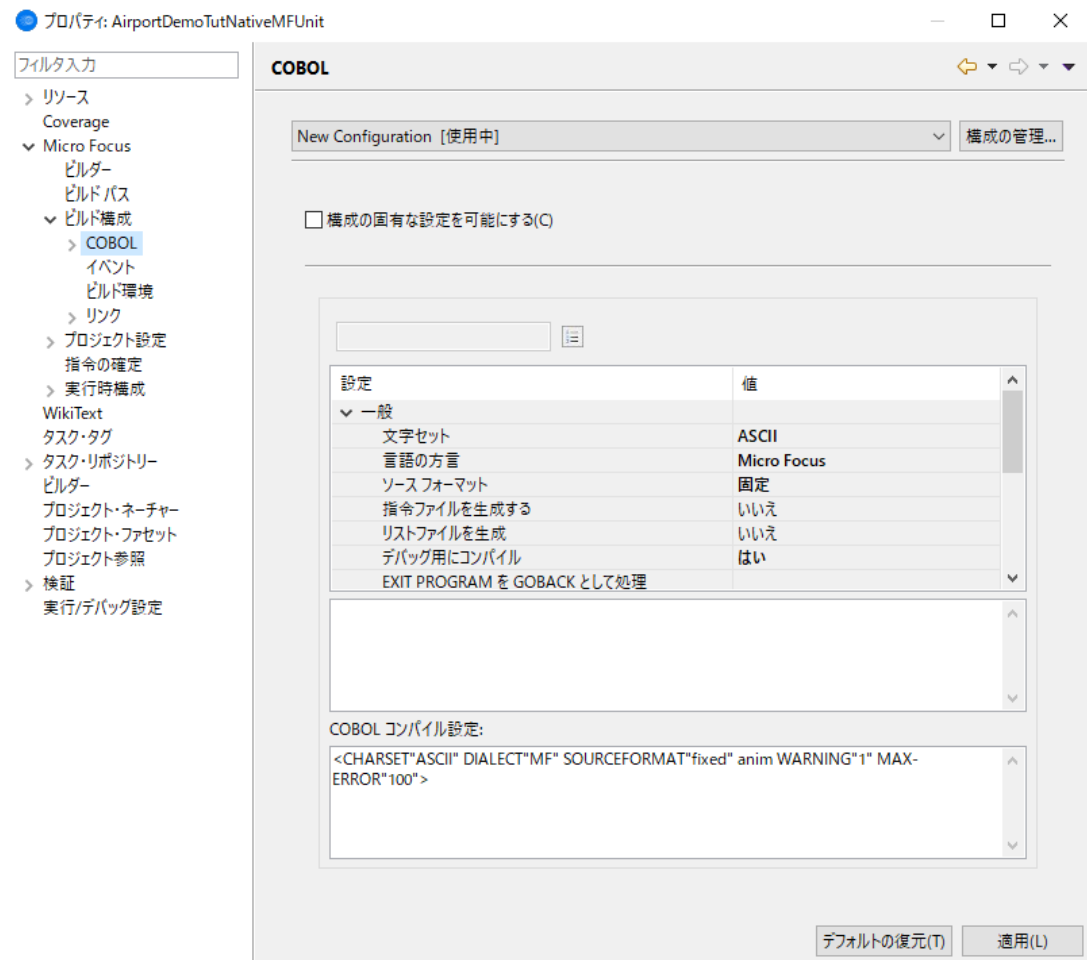
Eclipse のワークスペースとプロジェクトには、アプリケーションの作成に必要なビルドパス、データ接続、フォルダ、およびファイルを表す項目等が含まれています。ワークスペースには複数のプロジェクトを含めることができ、プロジェクトには、通常、複数の項目が含まれます。COBOL エクスプローラーには、ワークスペースに紐づけられたプロジェクト、それらのプロジェクト内の項目が階層状に表示されます。COBOL エクスプローラー上で目的の要素を検索し、編集するファイルを開く、プロジェクトに新規ファイルを追加する、プロジェクトおよび項目のプロパティを表示するなどの操作を実行できます。パースペクティブによっては異なる名称のビューが同等の用途のために用意されています。例えば Java パースペクティブであればパッケージエクスプローラーという Java 開発時に必要な要素をフィルター表示するビューが紐づけられています。



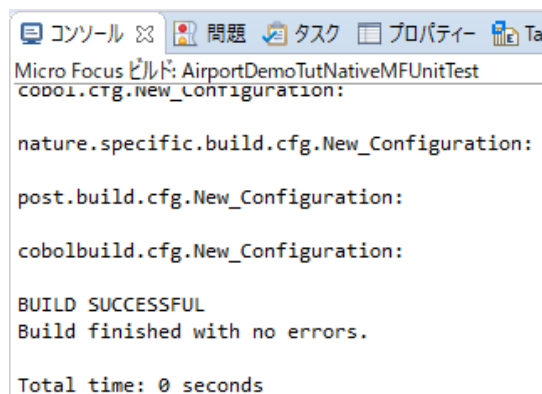
COBOL パースペクティブに紐づくエディターには、COBOL 予約語とデータ名や手続き名などの利用者語を色分け表示や、COBOL スニペットなど COBOL 言語固有の機能拡張が含まれます。ソースコードを入力するとバックグラウンドチェックを実行して、赤の波線でエラー箇所を強調表示します。そのエラー箇所にマウスポインタを移動すればエラー内容の確認、定義への移動、他の参照検索などの操作が可能です。



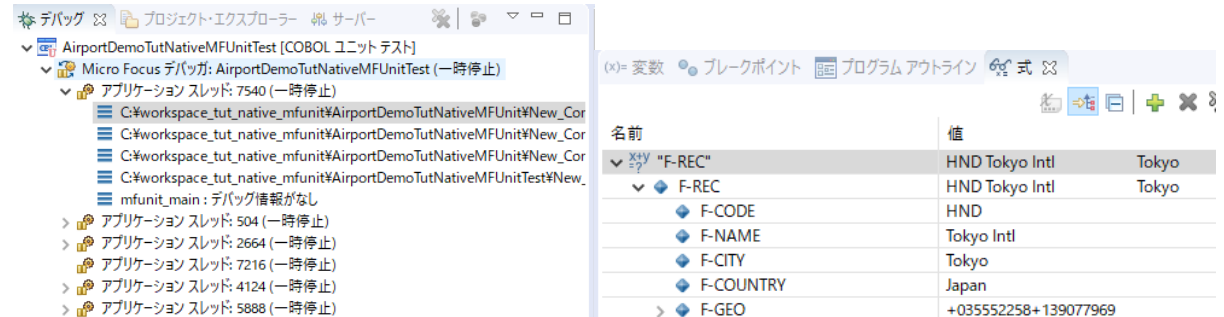
プロジェクト名を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[プロパティ(R)] を選択します。その後、表示される画面にて、[Micro Focus] > [ビルド構成] > [COBOL] と遷移する画面では COBOL アプリケーションのビルド方法等を構成します。



コンソールビューにはビルド時のメッセージやアプリケーションのコンソール出力等が表示されます。問題ビューには、不正な構文、キーワードのスペルミス、型の不一致などのコンパイルエラーが表示されます。



ビルドしたアプリケーションは、実行時の論理エラーやセマンティックエラーなどの問題を検出して修正するために、デバッグ機能を利用します。Eclipse のデバッガーは、コードのステップ実行や、様々な条件を設定したブレークポイントで実行を中断させ、変数ビューを使用してローカル変数やその他の関連データを調べることができます。



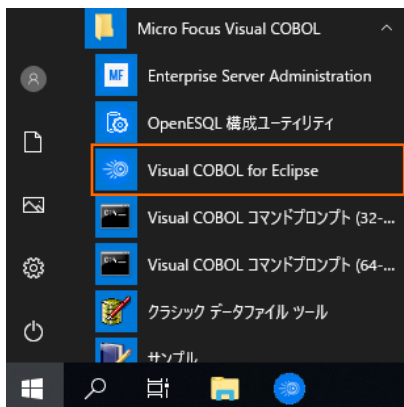
名前	値
▼ "F-REC"	HND Tokyo Intl Tokyo
▼ F-REC	HND Tokyo Intl Tokyo
F-CODE	HND
F-NAME	Tokyo Intl
F-CITY	Tokyo
F-COUNTRY	Japan
F-GEO	+035552258+139077969

デバッグが完了したアプリケーションは、アプリケーションサーバ等との連携機能を利用して自動配備するか、ファイルを手動でコピーして、本番環境に配置します。なお、本番環境には COBOL Server が事前にインストールされている必要があります。

3.2. はじめての Visual COBOL プロジェクト

- 1) Visual COBOL for Eclipse を起動します。

Windows のスタートメニューから [Micro Focus Visual COBOL] > [Visual COBOL for Eclipse] を選択します。



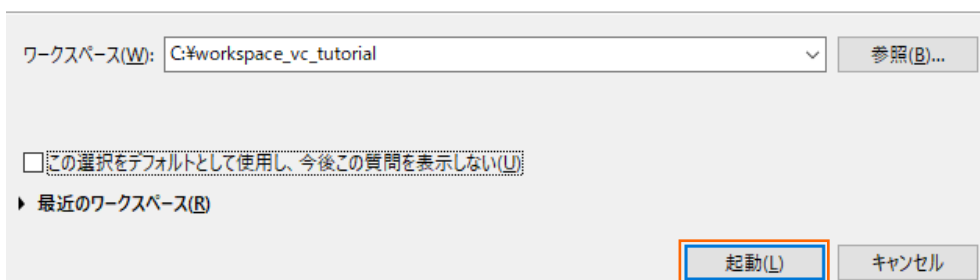
ワークスペースを保存する任意のフォルダを指定します。[参照(B)] ボタンをクリックしエクスプローラー経由で選択することも可能です。ワークスペースを選択後、[起動(L)] ボタンをクリックします。

Eclipse ランチャー

×

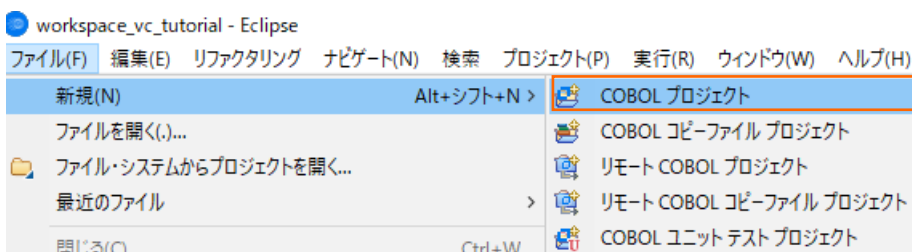
ディレクトリーをワークスペースとして選択

Eclipse は、ワークスペースディレクトリを使用して、環境設定と開発成果物を保存します。



- 2) COBOL プロジェクトの作成

メニューより、[ファイル(F)] > [新規(N)] > [COBOL プロジェクト] を選択します。



プロジェクト名に “ConsoleHello” を入力し、プロジェクトテンプレートを稼働環境に合わせて選択した上、[終了(F)] ボタンをクリックします。


COBOL プロジェクトの新規作成
— □ ×

COBOL プロジェクト


ワークスペースまたは外部の場所にCOBOL プロジェクトを作成します。

プロジェクト名(P):

プロジェクトテンプレートを選択

 Micro Focus テンプレート [32 ビット]
 Micro Focus テンプレート [64 ビット]

[テンプレートの設定を構成...](#)

☐ テンプレートの参照

場所: 参照...
 ファイルシステムを選択: default ▼

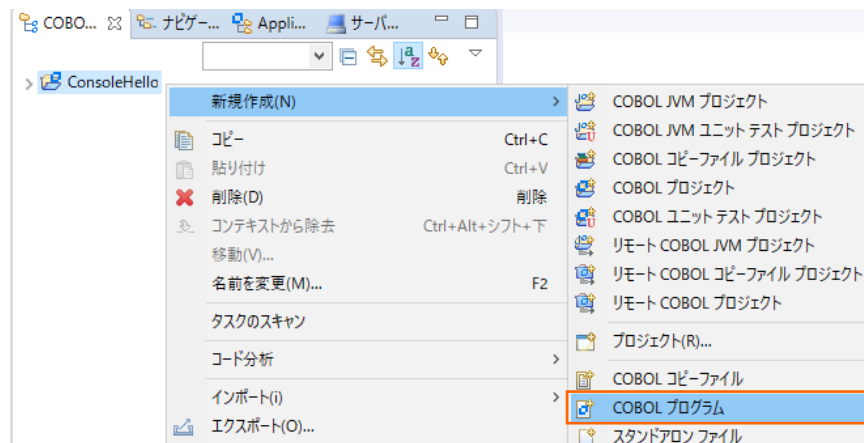
☒ デフォルト・ロケーションの使用(D)

ロケーション(L): 参照(R)...
 ファイル・システムを選択(Y): デフォルト ▼

?
終了(F)
キャンセル

3) COBOL プログラムを追加します。

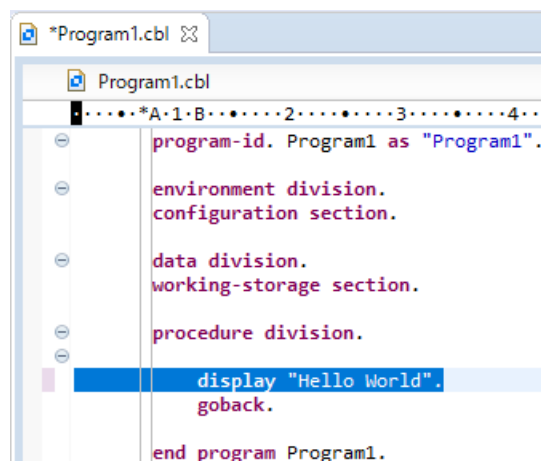
COBOL エクスプローラービューにて ConsoleHello プロジェクトを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[新規作成(N)] > [COBOL プログラム] を選択します。



4) エディターで COBOL ソースコードを入力します。

COBOL プロジェクトにて COBOL プログラムを新規に作成するとプログラムを構成する見出し部 (identification division)、環境部 (environment division)、データ部 (data division)、手続き部 (procedure division) や program-id 段落が埋め込まれたかたちで生成されます。今回は “Hello World” を表示して終了するプログラムなので、手続き部 3 行目の goback 文の手前に「display "Hello World".」を追加します。

なお、COBOL 正書法ではエディタービュー左右にある線で区切られた領域を特別な領域として利用するので、通常のソースコードはこれを避けて入力します。



5) COBOL アプリケーションをビルドします。

終止符 (ピリオド) を含めてスペルミスがなければ、エディタービュー上の Program1.cbl をアクティブにした状態で、メニューより、[ファイル(F)] > [保管(S)] または Ctrl + S キーで保存します。これにより自動的にコンパイラが起動されビルド処理が始まります。コンソールビューに正常にビルドできた旨のメッセージが出力されていることを確認します。

```

コンソール 問題 タスク プロパティ
Micro Focus ビルド: ConsoleHello

nature.specific.build.cfg.New_Configuration:

post.build.cfg.New_Configuration:

cobolbuild.cfg.New_Configuration:

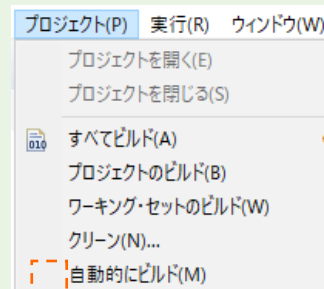
BUILD SUCCESSFUL
Build finished with no errors.

Total time: 0 seconds
-----ビルド完了-----

```

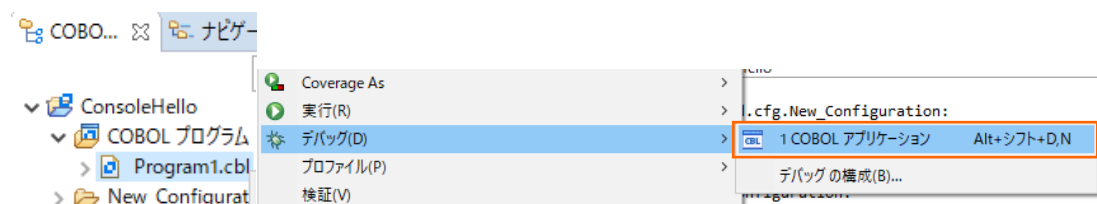
補足)

Visual COBOL for Eclipse では、上記のように保存したタイミングでビルドが自動実行されるようになっています。自動実行を抑止する場合は、メニューより、[プロジェクト(P)] > [自動的にビルド(M)] を選択し、未チェック状態にします。

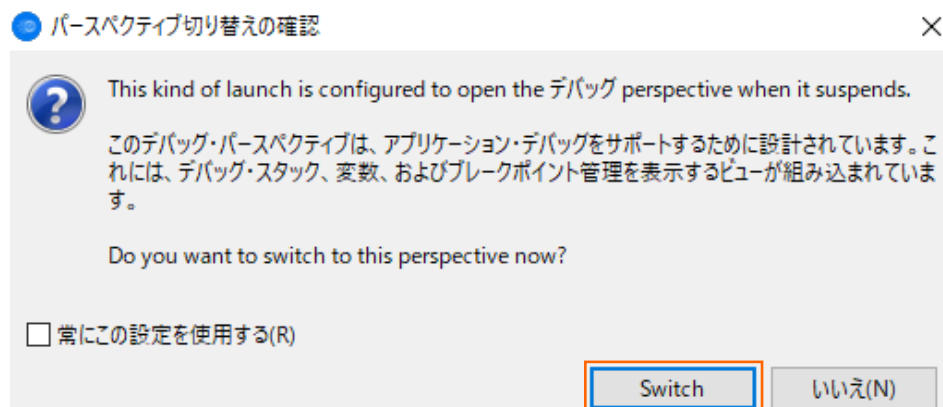


6) COBOL アプリケーションをデバッグ実行します。

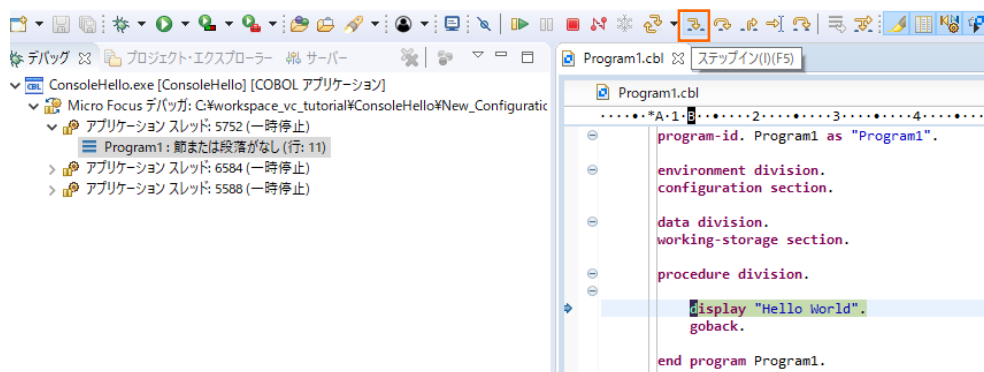
COBOL エクスプローラー上の Program1.cbl を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[デバッグ(D)] > [COBOL アプリケーション] を選択します。



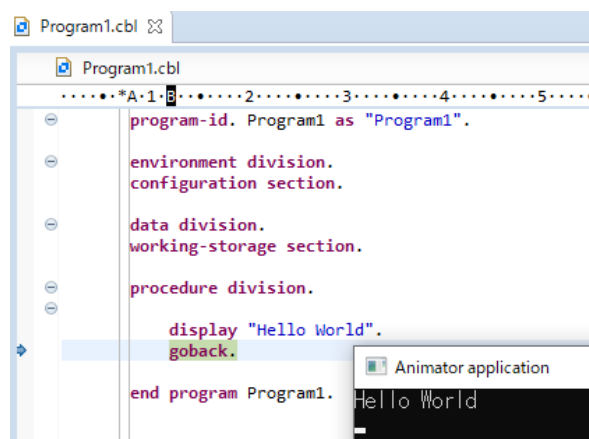
パースペクティブ切り替えの確認ダイアログが表示された場合、[Switch] ボタンをクリックします。なお、「常にこの設定を使用する」にチェックをすることで、今後、この問合せを抑止できます。



デバッグパースペクティブに切り替わりましたら、DISPLAY 文を実行する手前でステップが一時停止しています。[ステップイン] アイコンを一回クリックし DISPLAY 文を実行します。



Animator application 画面に「Hello World」が表示されたことを確認できましたら、[ステップイン] アイコンを再度クリックしデバッグを終了します。

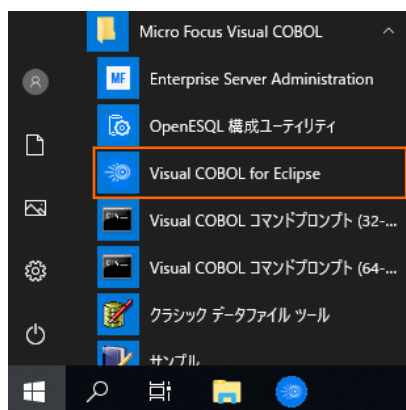


3.3. ファイルの入出力

エクセルやメモ帳で作成した CSV ファイルを読み込んで、固定長順編成ファイルを作成する COBOL アプリケーションを Visual COBOL for Eclipse で作成します。

- 1) Visual COBOL for Eclipse を終了した場合は起動します。

Windows のスタートメニューから [Micro Focus Visual COBOL] > [Visual COBOL for Eclipse] を選択します。



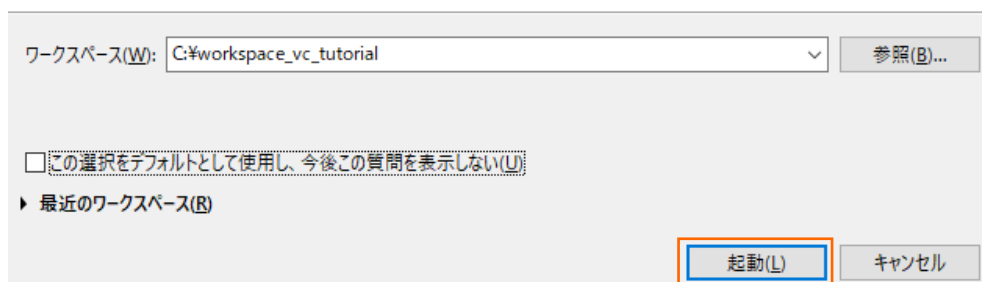
ワークスペースは 3.2 で使用したものでも、新規のものでも構いません。（本チュートリアルでは、同一のワークスペースを使用します。）ワークスペースを選択後、[起動(L)] ボタンをクリックします。

Eclipse ランチャー

×

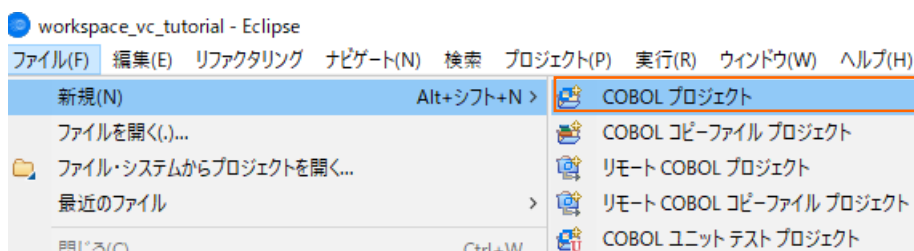
ディレクトリーをワークスペースとして選択

Eclipse は、ワークスペースディレクトリを使用して、環境設定と開発成果物を保存します。



- 2) COBOL プロジェクトを作成します。

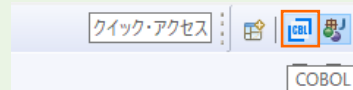
メニューより、[ファイル(F)] > [新規(N)] > [COBOL プロジェクト] を選択します。



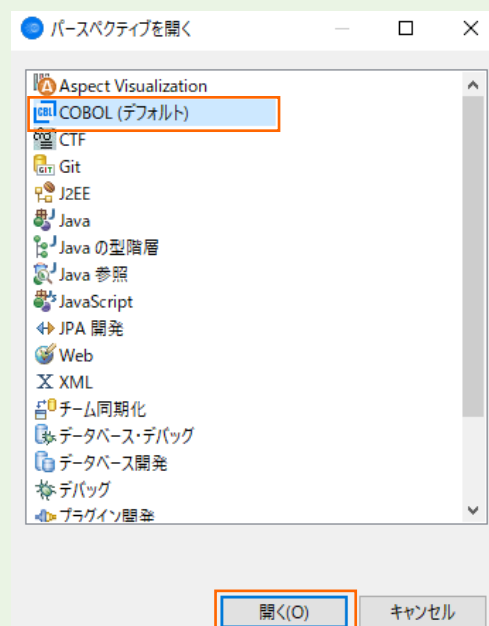
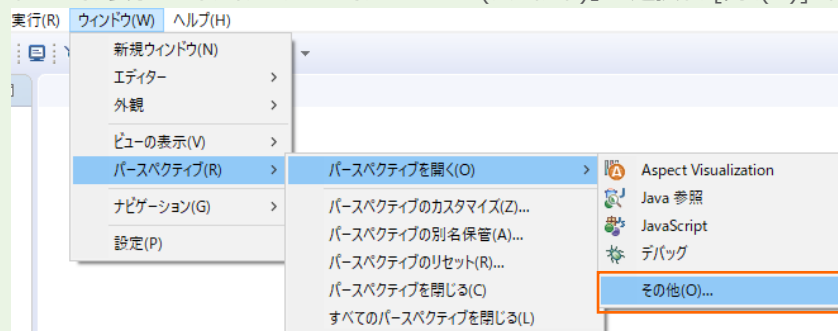
補足)

新規メニューに表示される項目は、現在、アクティブのパーズペクティブにより変更されます。例えば、COBOL パーズペクティブがアクティブとなっている場合、上記 COBOL プロジェクトは表示されますが、Java パーズペクティブがアクティブの場合、異なる内容が表示されます。この場合は、Eclipse 画面の右上よりパーズペクティブを切り替えます。

以下は、Java パーズペクティブがアクティブな状態です。COBOL パーズペクティブアイコンをクリックすることで、パーズペクティブの切り替えできます。



もしくは、メニューより、[ウィンドウ(W)] > [パーズペクティブ(R)] > [パーズペクティブを開く(O)] > [その他(O)] を選択した上で、表示されるダイアログ上にて、「COBOL (デフォルト)」を選択し、[開く(O)] ボタンをクリックします。



プロジェクト名に “LoadCSVFile” を入力し、プロジェクトテンプレートを稼働環境に合わせて選択した上、[終了(F)] ボタンをクリックします。


COBOL プロジェクトの新規作成
—
□
×

COBOL プロジェクト

ワークスペースまたは外部の場所にCOBOL プロジェクトを作成します。

プロジェクト名(P):

プロジェクトテンプレートを選択

 Micro Focus テンプレート [32 ビット]
  Micro Focus テンプレート [64 ビット]

[テンプレートの設定を構成...](#)

☐ テンプレートの参照

場所: 参照...

ファイルシステムを選択: default ▾

☒ デフォルト・ロケーションの使用(D)

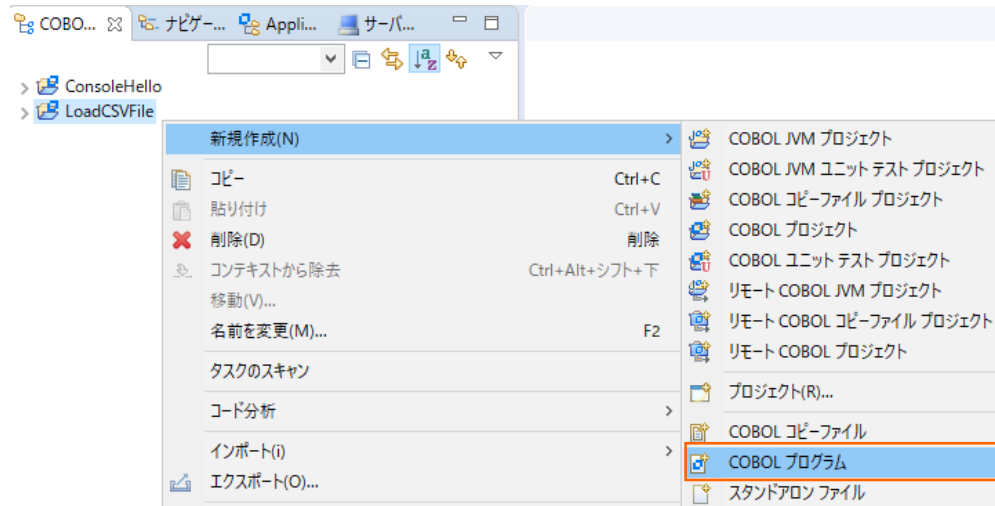
ロケーション(L): 参照(R)...

ファイル・システムを選択(Y): デフォルト ▾

?
終了(E)
キャンセル

3) COBOL プログラムを追加します。

COBOL エクスプローラービューにて LoadCSVFile プロジェクトを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[新規作成(N)] > [COBOL プログラム] を選択します。

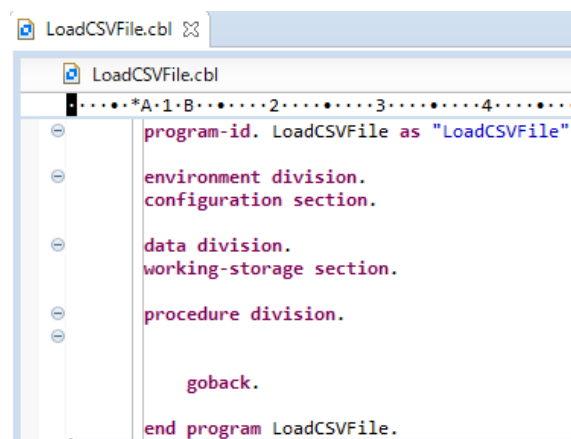


プログラム名に “LoadCSVFile.cbl” を入力し、[終了] ボタンをクリックします。



4) コードエディターで COBOL ソースコードを入力します。

COBOL ソースファイルを新規に作成した直後の段階ではコンソールアプリケーションのひな形が埋め込まれています。ここでは、環境部 (environment division)、データ部 (data division)、手続き部 (procedure division) を書き換えます。



```

program-id. LoadCSVFile as "LoadCSVFile".

environment division.
configuration section.

data division.
working-storage section.

procedure division.

    goback.

end program LoadCSVFile.

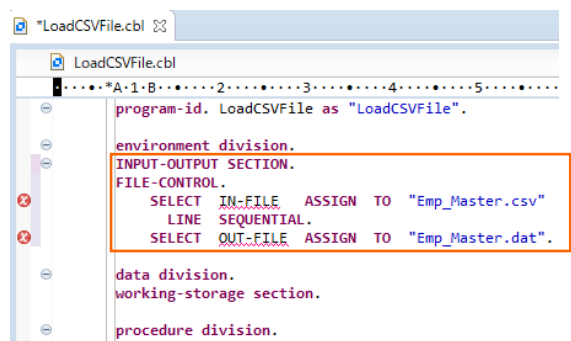
```

まず、環境部の構成節 (configuration section) を削除し、以下の入出力節 (input-output section) を追加します。また、データ部のファイル定義が未入力なので IN-FILE と OUT-FILE がエラーとなりますが、ここでは無視して構いません。

```

INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT IN-FILE  ASSIGN TO "Emp_Master.csv"
           LINE SEQUENTIAL.
    SELECT OUT-FILE ASSIGN TO "Emp_Master.dat".

```



```

program-id. LoadCSVFile as "LoadCSVFile".

environment division.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT IN-FILE  ASSIGN TO "Emp_Master.csv"
           LINE SEQUENTIAL.
    SELECT OUT-FILE ASSIGN TO "Emp_Master.dat".

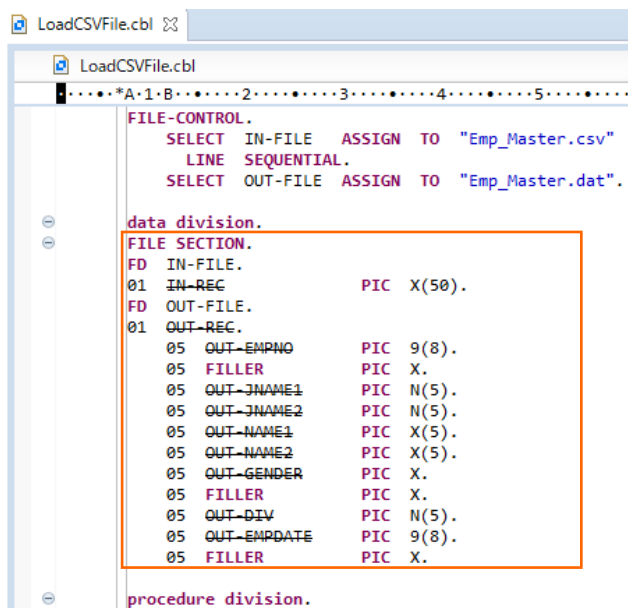
data division.
working-storage section.

procedure division.

```

次に、データ部の作業場所節（working-storage section）を削除し、以下のファイル節（FILE SECTION）を追加します。なお、データ部のファイル定義を入力したので、環境部のエラーは無くなります。

```
FILE SECTION.
FD IN-FILE.
01 IN-REC          PIC X(50).
FD OUT-FILE.
01 OUT-REC.
    05 OUT-EMPNO    PIC 9(8).
    05 FILLER       PIC X.
    05 OUT-JNAME1   PIC N(5).
    05 OUT-JNAME2   PIC N(5).
    05 OUT-NAME1    PIC X(5).
    05 OUT-NAME2    PIC X(5).
    05 OUT-GENDER   PIC X.
    05 FILLER       PIC X.
    05 OUT-DIV      PIC N(5).
    05 OUT-EMPDATE  PIC 9(8).
    05 FILLER       PIC X.
```



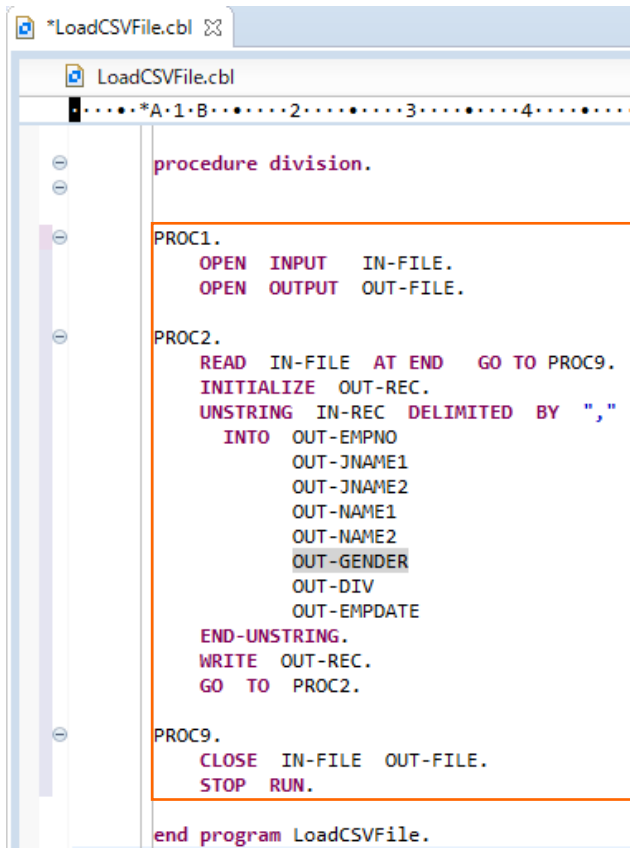
```
LoadCSVFile.cbl
...*A.1.B...2...3...4...5...
FILE-CONTROL.
  SELECT IN-FILE  ASSIGN TO "Emp_Master.csv"
  LINE SEQUENTIAL.
  SELECT OUT-FILE ASSIGN TO "Emp_Master.dat".

data division.
FILE SECTION.
FD IN-FILE.
01 IN-REC          PIC X(50).
FD OUT-FILE.
01 OUT-REC.
    05 OUT-EMPNO    PIC 9(8).
    05 FILLER       PIC X.
    05 OUT-JNAME1   PIC N(5).
    05 OUT-JNAME2   PIC N(5).
    05 OUT-NAME1    PIC X(5).
    05 OUT-NAME2    PIC X(5).
    05 OUT-GENDER   PIC X.
    05 FILLER       PIC X.
    05 OUT-DIV      PIC N(5).
    05 OUT-EMPDATE  PIC 9(8).
    05 FILLER       PIC X.

procedure division.
```

最後に、手続き部の goback 文を削除し、以下の手続き文を追加します。

```
PROC1.  
    OPEN  INPUT  IN-FILE.  
    OPEN  OUTPUT OUT-FILE.  
  
PROC2.  
    READ  IN-FILE AT END  GO TO PROC9.  
    INITIALIZE OUT-REC.  
    UNSTRING IN-REC DELIMITED BY ","  
        INTO  OUT-EMPNO  
            OUT-JNAME1  
            OUT-JNAME2  
            OUT-NAME1  
            OUT-NAME2  
            OUT-GENDER  
            OUT-DIV  
            OUT-EMPDATE  
    END-UNSTRING.  
    WRITE OUT-REC.  
    GO TO PROC2.  
  
PROC9.  
    CLOSE IN-FILE OUT-FILE.  
    STOP  RUN.
```



```
*LoadCSVFile.cbl

LoadCSVFile.cbl
.....*A.1.B.....2.....3.....4.....

procedure division.

PROC1.
  OPEN INPUT IN-FILE.
  OPEN OUTPUT OUT-FILE.

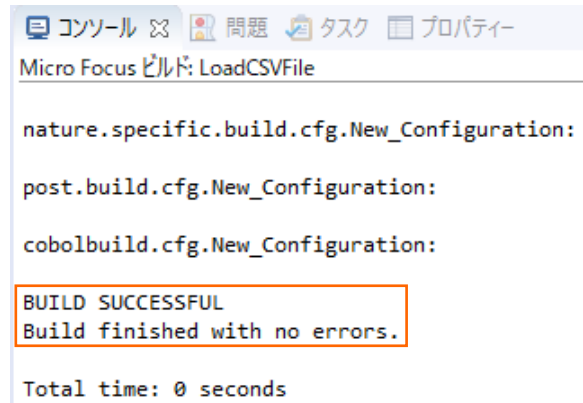
PROC2.
  READ IN-FILE AT END GO TO PROC9.
  INITIALIZE OUT-REC.
  UNSTRING IN-REC DELIMITED BY ","
    INTO OUT-EMPNO
        OUT-JNAME1
        OUT-JNAME2
        OUT-NAME1
        OUT-NAME2
        OUT-GENDER
        OUT-DIV
        OUT-EMPDATE
  END-UNSTRING.
  WRITE OUT-REC.
  GO TO PROC2.

PROC9.
  CLOSE IN-FILE OUT-FILE.
  STOP RUN.

end program LoadCSVFile.
```

5) COBOL アプリケーションをビルドします。

終止符 (ピリオド) を含めてエラーがなければ、LoadCSVFile.cbl への変更を、メニューより、[ファイル(F)] > [保管(S)]
または Ctrl + S キーで保存します。自動的にビルドが実行され、コンソールビューに正常にビルドできた旨のメッセージが出力されていることを確認します。



```

Micro Focus ビルド: LoadCSVFile

nature.specific.build.cfg.New_Configuration:

post.build.cfg.New_Configuration:

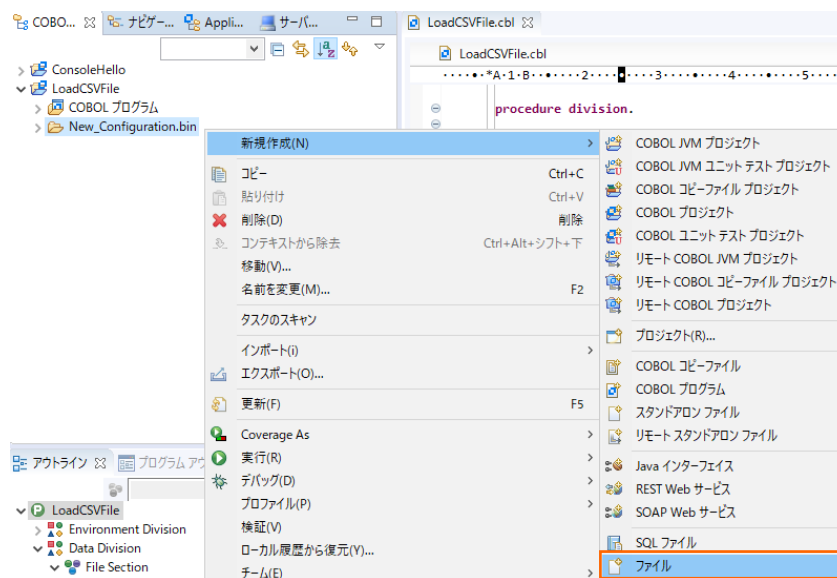
cobolbuild.cfg.New_Configuration:

BUILD SUCCESSFUL
Build finished with no errors.

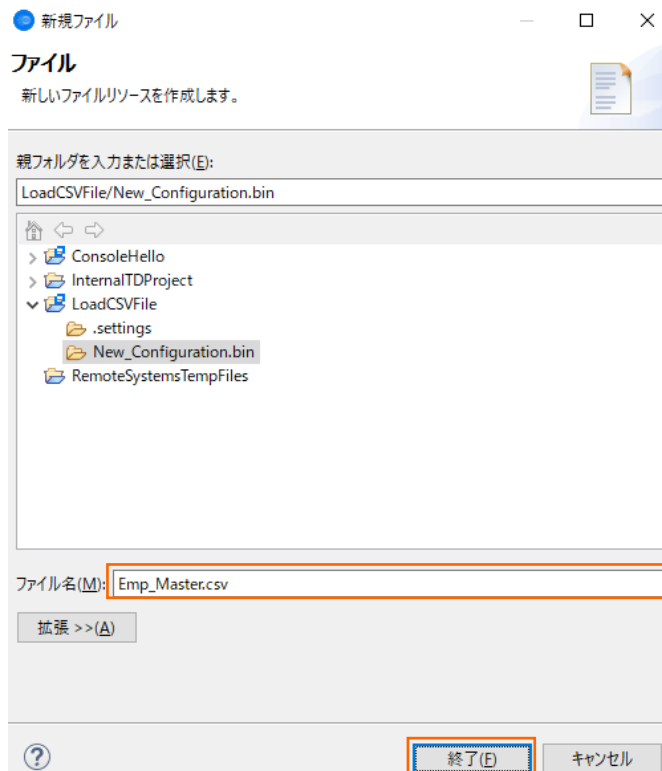
Total time: 0 seconds
  
```

6) CSV ファイルを作成します。

LoadCSVFile プロジェクト配下の New_Configuration.bin フォルダを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[新規作成(N)] > [ファイル] を選択します。



ファイル名に “Emp_Master.csv” を入力し、[終了(F)] ボタンをクリックします。



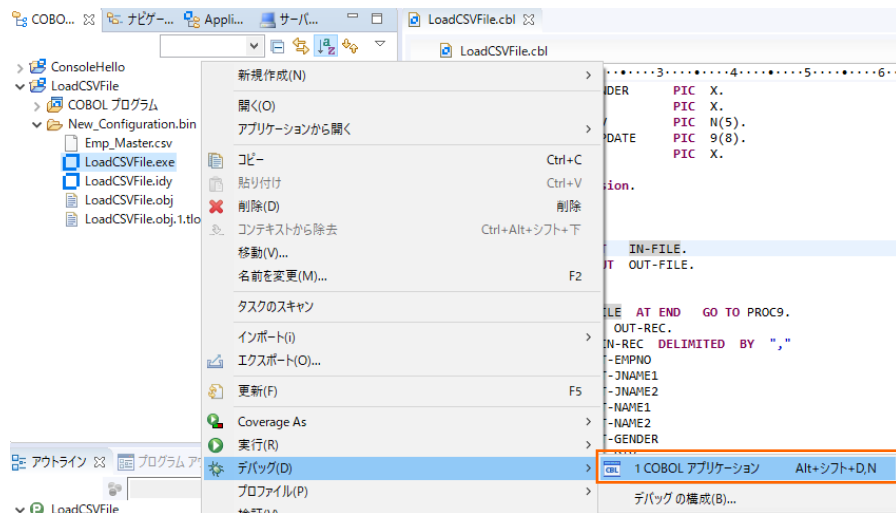
「この種類のファイル(.csv)を開くには、どのアプリをしますか？」の問いがポップアップされた場合はメモ帳など、お使いのエディターを選択します。

エディター上で、以下のデータを入力し、保存した上で、エディターを閉じます。

```
11111113,佐藤,隆,サウ,タシ,M,営業部,19980401,0
22222226,鈴木,尚之,スヅキ,ナオキ,M,技術部,19981015,0
33333339,田中,直美,タナカ,ナミ,F,総務部,19990401,0
44444442,山田,洋一,ヤマダ,ヨウイチ,M,営業部,20000701,0
55555555,伊藤,弘子,イトウ,ヒロコ,F,技術部,20010401,0
66666668,木村,貴弘,キムラ,タカヒロ,M,営業部,20021220,0
77777771,中村,慎司,ナカムラ,シンジ,M,技術部,20030401,0
88888884,橋本,悦子,ハシモト,エツコ,F,総務部,20040805,0
99999997,三井,薫,ミツイ,カオル,F,営業部,20050401,0
```

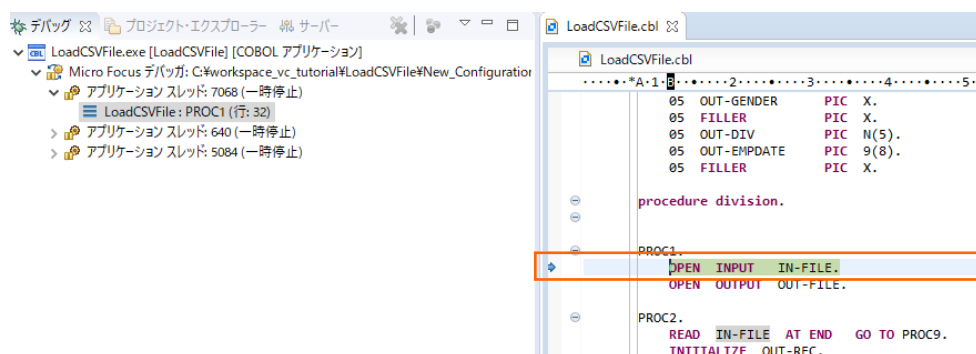
7) COBOL アプリケーションをデバッグ実行します。

LoadCSVFile プロジェクトの New_Configuration.bin フォルダ配下にある LoadCSVFile.exe を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[デバッグ(D)] > [COBOL アプリケーション] を選択します。

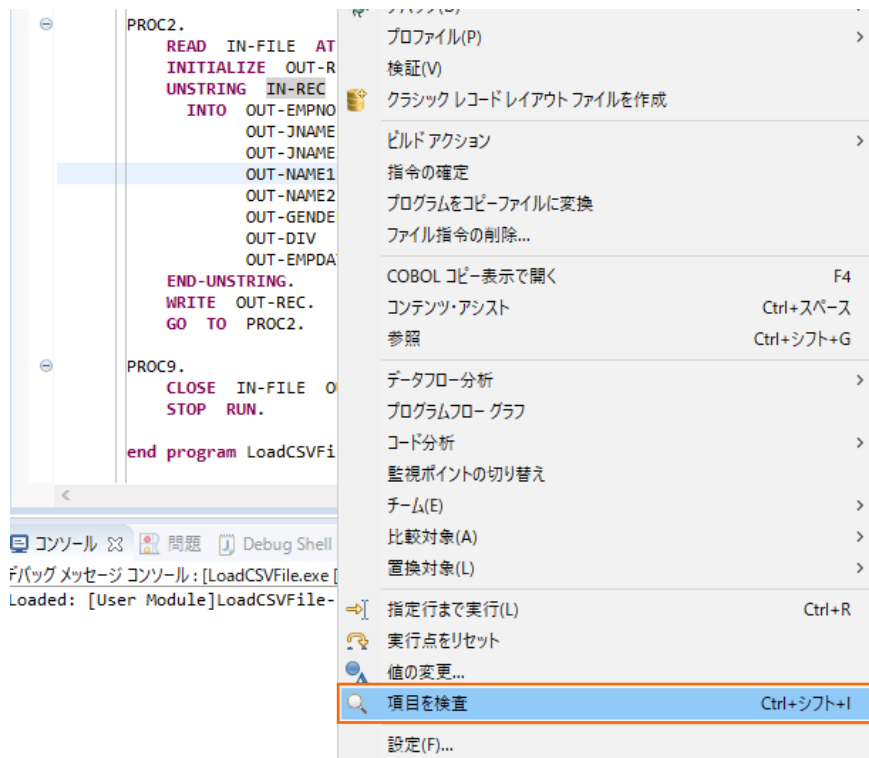


3.1 の 6) 同様、パースペクティブ切り替えの確認ダイアログが表示された場合、[Switch] ボタンをクリックします。

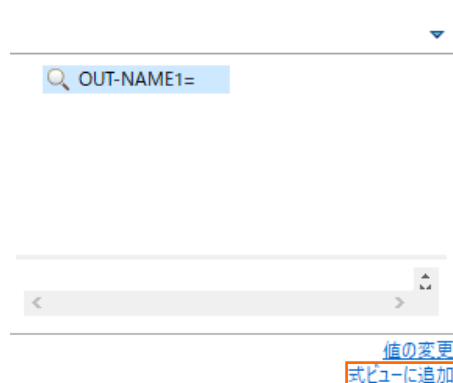
デバッガーは手続き部の最初の COBOL 文である open 文の処理前に一時停止している状態となっています。



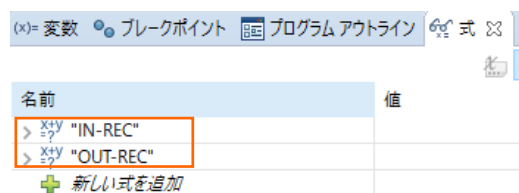
入力ファイルから読み込んだレコードの内容を確認するため、IN-REC に格納される値の変遷を追います。UNSTRING 文の IN-REC を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[項目を検査] を選択します。



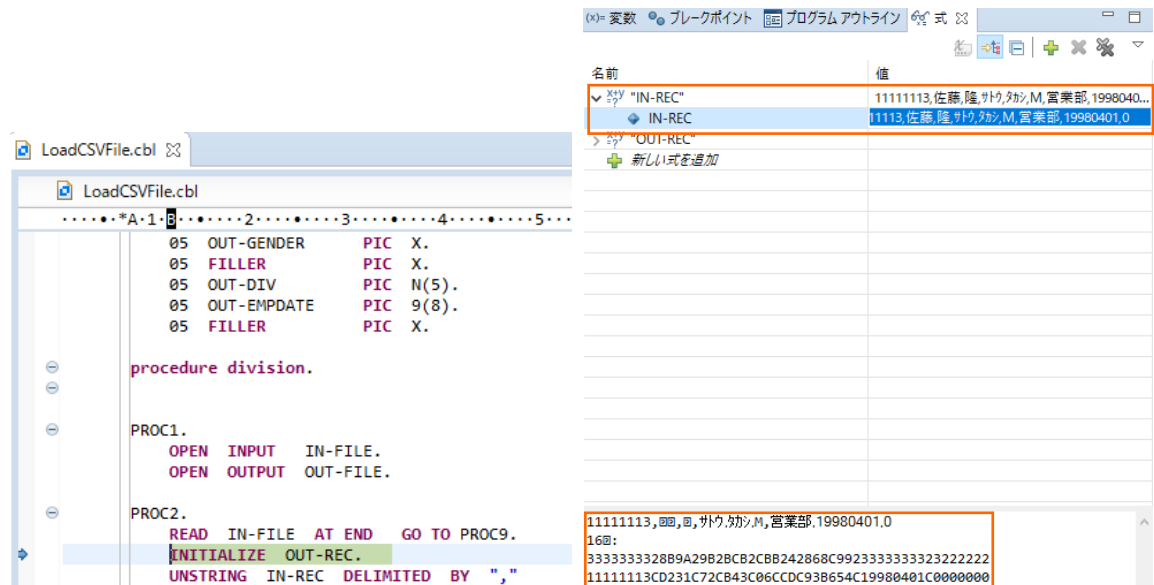
[式ビューに追加] をクリックします。



出力ファイルに書き出すレコードの内容もトレースするため、INITIALIZE 文の OUT-REC を同様の手順で式ビューに追加します。



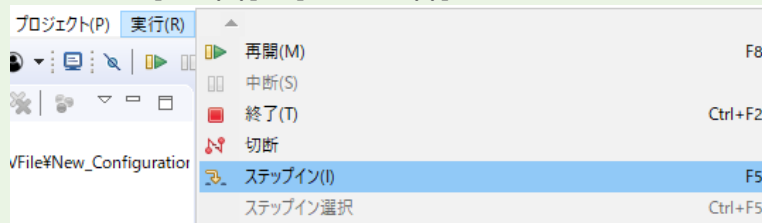
F5 キー（ステップイン）を 3 回押すと、デバッガーは READ 文実行後、処理を中断します。式ビューの IN-REC の値には CSV ファイルから読み込んだ最初のレコードが表示されます。



The screenshot shows the Eclipse IDE with a COBOL program named 'LoadCSVFile.cbl' open. The code is in the left pane, and the debugger's 'Variables' view is on the right. The 'Variables' view shows a table with two columns: '名前' (Name) and '値' (Value). The 'IN-REC' variable is highlighted, showing its value as '11111113,佐藤,隆,サトウ,タカシ,M,営業部,19980401,0'. Below the table, the 'OUT-REC' variable is also visible, showing its value as '11111113,00,0,サトウ,タカシ,M,営業部,19980401,0'. The code in the left pane shows a 'READ IN-FILE AT END GO TO PROC9.' statement, which is the line where the debugger has stopped.

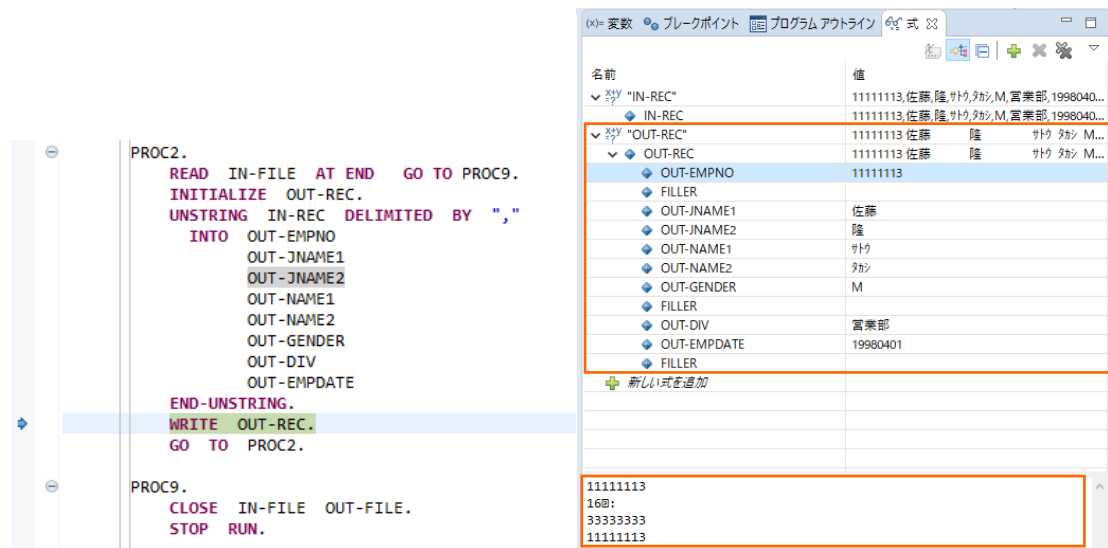
補足)

ステップインには、F5 キーがホットキーとして割り当てられているため、F5 キーを押すことでステップイン指示が可能です。メニューからは、[実行(R)] > [ステップイン(I)] を選択します。



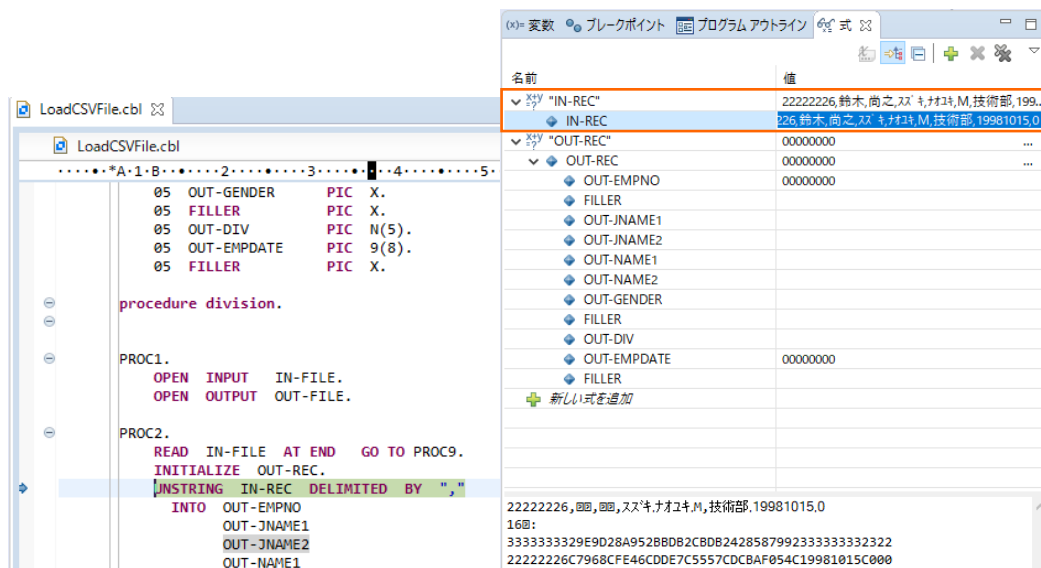
The screenshot shows the Eclipse IDE's 'Run' menu. The menu is open, and the 'Step In' option is highlighted. The menu items are: '再開(M)' (F8), '中断(S)', '終了(T)' (Ctrl+F2), '切断', 'ステップイン(I)' (F5), and 'ステップイン選択' (Ctrl+F5). The 'Step In' option is the one that triggers the F5 key action.

さらに F5 キーを 2 回押すと、デバッガーは UNSTRING 文を実行後、処理を中断します。式ビューの OUT-REC の値には出力ファイルへ書き出す最初のレコードが表示されます。



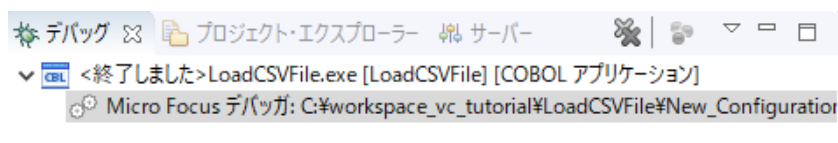
The screenshot shows the COBOL source code for PROC2 and PROC9. PROC2 contains a READ statement, an INITIALIZE statement for OUT-REC, and an UNSTRING statement. The Data Explorer on the right shows the values for the OUT-REC structure, including OUT-EMPNO, OUT-JNAME1, OUT-JNAME2, OUT-NAME1, OUT-NAME2, OUT-GENDER, OUT-DIV, and OUT-EMPDATE.

さらに F5 キーを 4 回押すと、デバッガーは INITIALIZE 文を実行後、処理を中断します。ウォッチ式の IN-REC の値には CSV ファイルから読み込んだ 2 番目のレコードが表示され、OUT-REC の値は INITIALIZE 文で初期化されています。



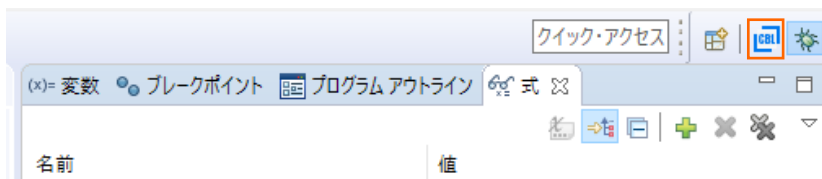
The screenshot shows the Watch window with the IN-REC variable selected, displaying its value as a CSV record. The Data Explorer on the right shows the values for the OUT-REC structure, which has been initialized to zero.

F8(再開)キーを打鍵するか CSV ファイルからすべてのレコードを読み込むまで F5 キーを押すと、デバッガーは終了します。

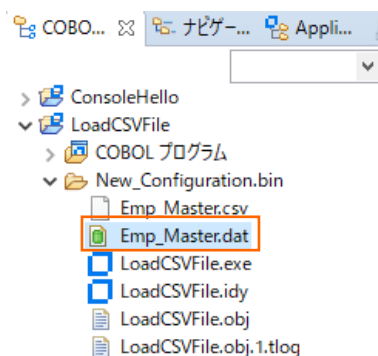


The screenshot shows the Debug console with the message "<終了しました>LoadCSVFile.exe [LoadCSVFile] [COBOL アプリケーション]".

ワークスペース右上にて COBOL アイコン を選択し、COBOL パースペクティブに戻ります。

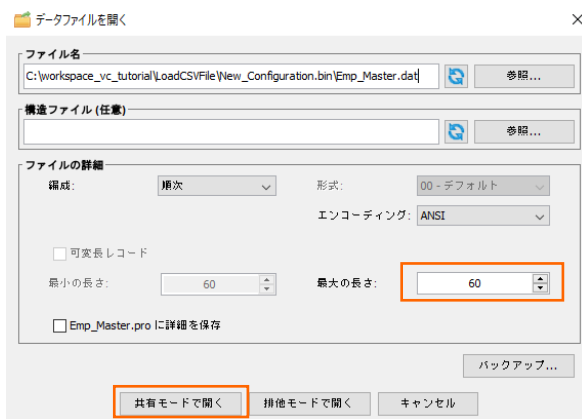
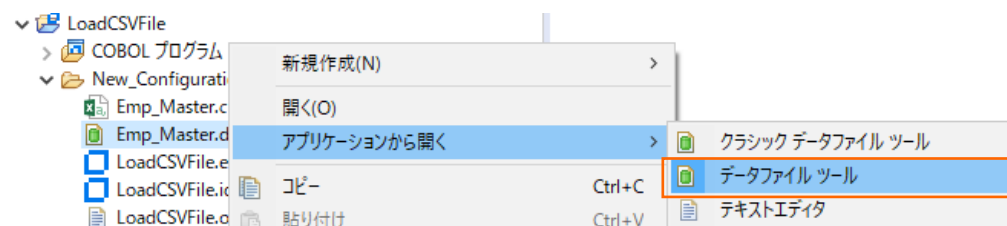


COBOL エクスプローラービュー上で LoadCSVFile プロジェクトの New_Configuration.bin フォルダを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、「更新(F)」を選択します。



Emp_Master.dat が作成されていることを確認します。

Emp_Master.csv を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[アプリケーションから開く] > [データファイルツール] を選択します。Visual COBOL に付属する データファイルエディタが起動します。「最大の長さ」欄には “60” を指定して、[共有モードで開く] ボタンをクリックします。



社員 9 名分のデータが正常に固定長順編成ファイルに書き込まれていることを確認します。

Micro Focus データファイル ツール - Emp_Master.dat

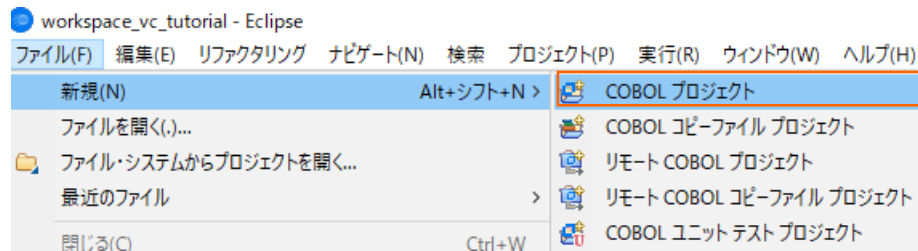
ファイル 編集 表示 構造 検索 ツール ヘルプ

ファイル		レコード		ナビゲーション	
Emp_Master.dat					
レコード 1 - 9					
11111113	佐藤 隆	サトウ タカシ	M	営業部	19980401
22222226	鈴木 尚之	スズキ ナオキ	M	技術部	19981015
33333339	田中 直美	タナカ ナミ	F	総務部	19990401
44444442	山田 洋一	ヤマダ ヨウイチ	M	営業部	20000701
55555555	伊藤 弘子	イトウ ヒロコ	F	技術部	20010401
66666668	木村 貴弘	キムラ タカヒロ	M	営業部	20021220
77777771	中村 慎司	ナカムラ シンジ	M	技術部	20030401
88888884	橋本 悦子	ハシモト エツコ	F	総務部	20040805
99999997	三井 薫	ミツイ カオル	F	営業部	20050401

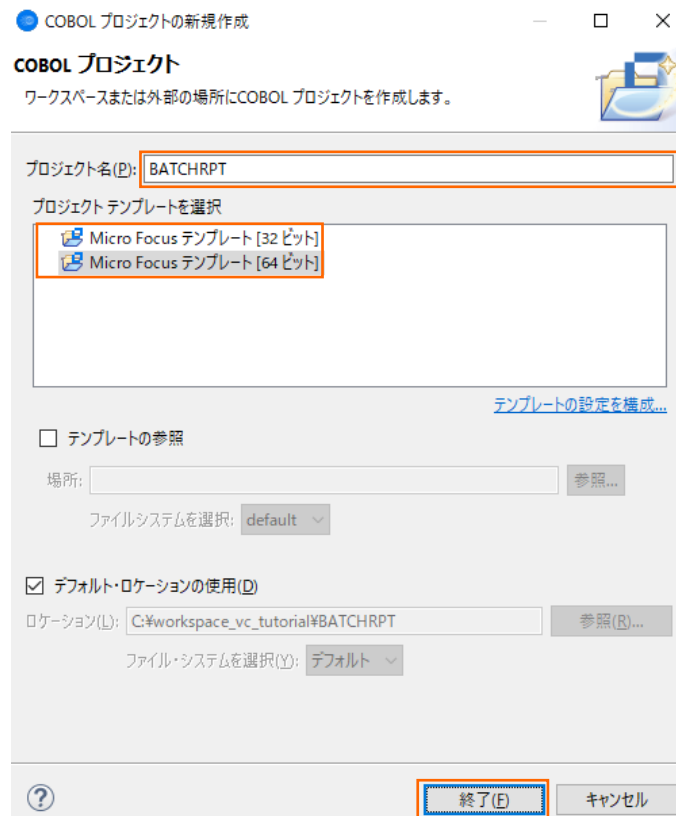
続いて、この固定長順編成ファイルを読み込んでレポートファイルを作成するバッチアプリケーションを作成します。

8) COBOL プロジェクトを作成します。

メニューより、[ファイル(F)] > [新規(N)] > [COBOL プロジェクト] を選択します。

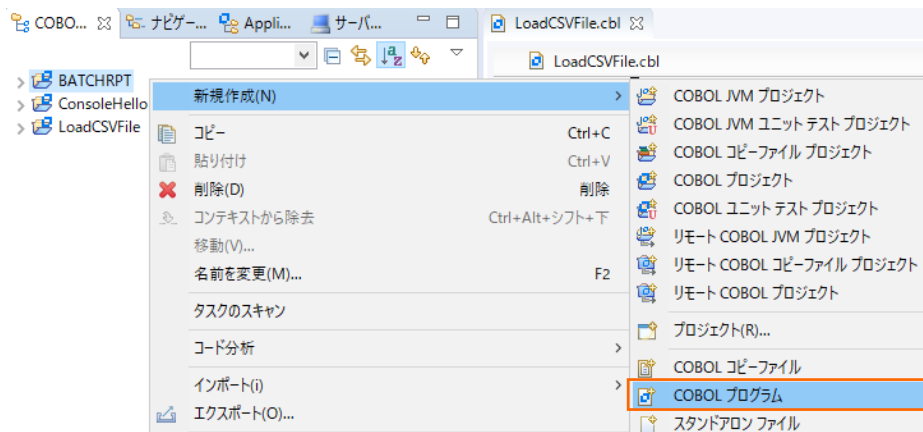


プロジェクト名に “BATCHRPT” を入力し、プロジェクトテンプレートを稼働環境に合わせて選択した上、[終了(F)] ボタンをクリックします。



9) COBOL プログラムを追加します。

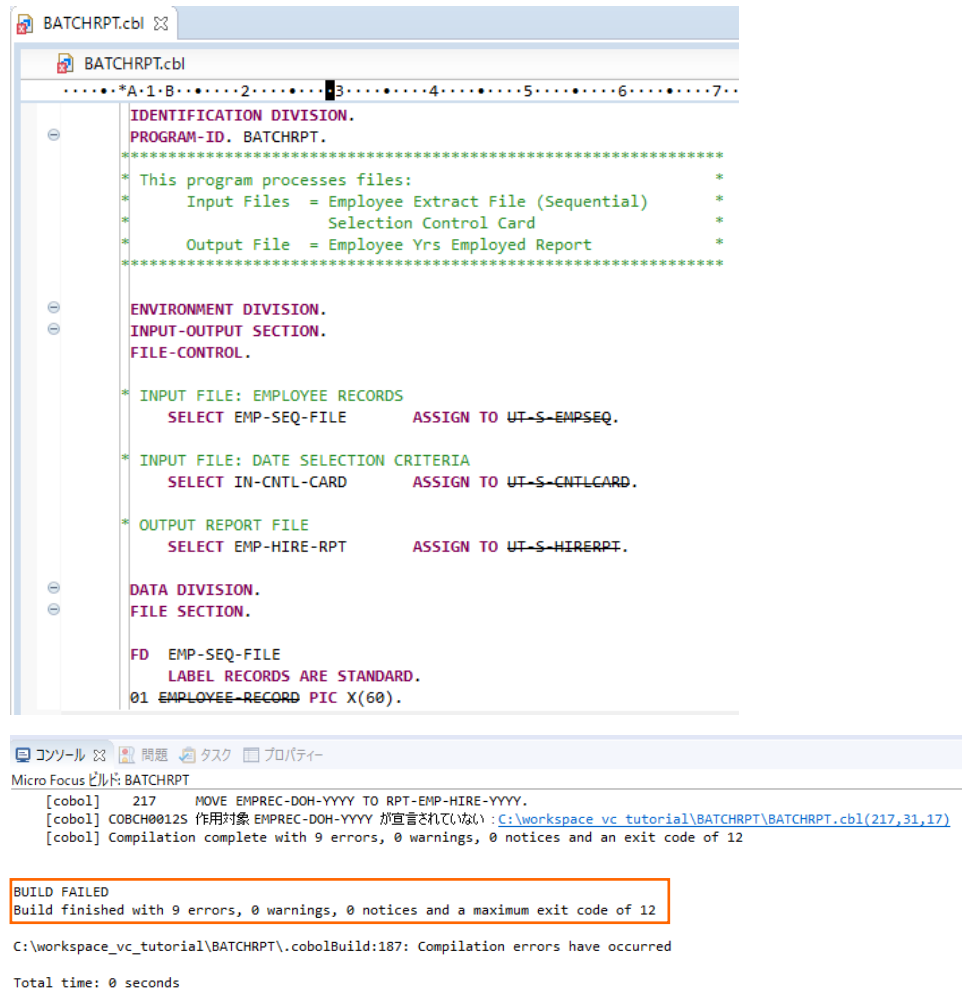
COBOL エクスプローラービューにて BATCHRPT プロジェクトを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[新規作成(N)] > [COBOL プログラム] を選択します。



新規ファイル名に “BATCHRPT.cbl” を入力し、[終了(F)] ボタンをクリックします。



BATCHRPT.cbl の中身を、サンプルプログラム BATCHRPT.cbl の内容で上書き保存します。



The screenshot shows the Eclipse IDE with the `BATCHRPT.cbl` file open. The code is a COBOL program for processing employee records. The console output shows the compilation process and a failure due to missing copybooks.

```

IDENTIFICATION DIVISION.
PROGRAM-ID. BATCHRPT.
*****
* This program processes files:
*   Input Files  = Employee Extract File (Sequential)
*               = Selection Control Card
*   Output File  = Employee Yrs Employed Report
*****

ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.

* INPUT FILE: EMPLOYEE RECORDS
  SELECT EMP-SEQ-FILE      ASSIGN TO UT-S-EMPSEQ.

* INPUT FILE: DATE SELECTION CRITERIA
  SELECT IN-CNTL-CARD      ASSIGN TO UT-S-CNTLCARD.

* OUTPUT REPORT FILE
  SELECT EMP-HIRE-RPT      ASSIGN TO UT-S-HIRERPT.

DATA DIVISION.
FILE SECTION.

FD  EMP-SEQ-FILE
   LABEL RECORDS ARE STANDARD.
01  EMPLOYEE-RECORD PIC X(60).
  
```

Console Output:

```

[console] 217 MOVE EMPREC-DOH-YYYY TO RPT-EMP-HIRE-YYYY.
[console] COBCH00125 作用対象 EMPREC-DOH-YYYY が宣言されていない : C:\workspace_vc_tutorial\BATCHRPT\BATCHRPT.cbl(217,31,17)
[console] Compilation complete with 9 errors, 0 warnings, 0 notices and an exit code of 12

BUILD FAILED
Build finished with 9 errors, 0 warnings, 0 notices and a maximum exit code of 12

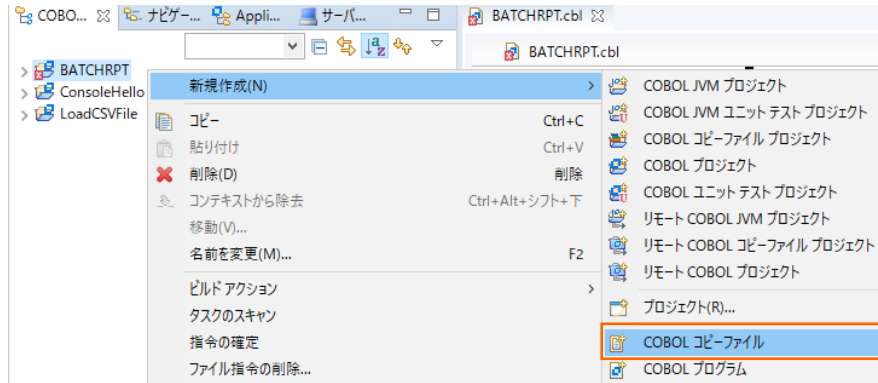
C:\workspace_vc_tutorial\BATCHRPT\cobolBuild:187: Compilation errors have occurred

Total time: 0 seconds
  
```

自動ビルドが実行された結果、エラーが報告されますが、これは、参照しているコピーブックが存在しないためです。
次の手順にて、コピーブックを作成するため、ここでは無視して構いません。

10) コピーブックを作成します。

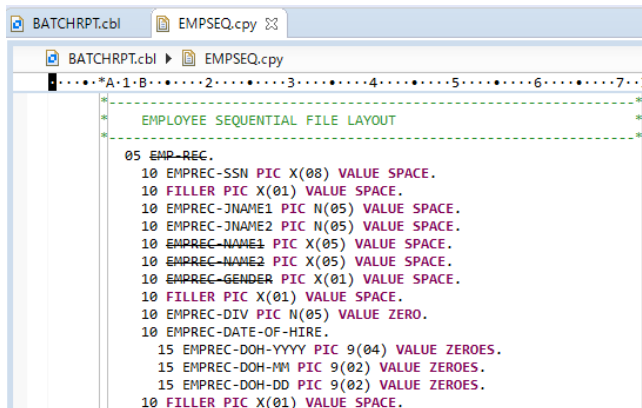
COBOL エクスプローラービューにて BATCHRPT プロジェクトを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[新規作成(N)] > [COBOL コピーファイル] を選択します。



新規ファイル名に “EMPSEQ.cpy” を入力し、[終了(F)] ボタンをクリックします。



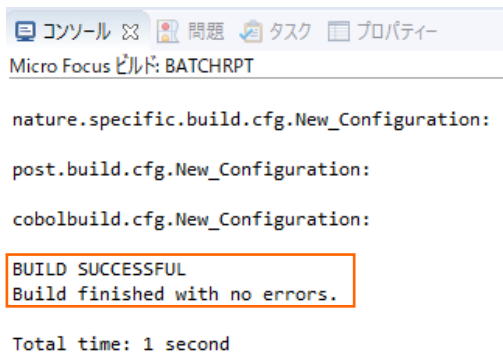
作成された EMPSEQ.cpy を、サンプルプログラム EMPSEQ.cpy の内容で上書き保存します



```

EMPLOYEE SEQUENTIAL FILE LAYOUT
05 EMP-REC.
  10 EMPREC-SSN PIC X(08) VALUE SPACE.
  10 FILLER PIC X(01) VALUE SPACE.
  10 EMPREC-JNAME1 PIC N(05) VALUE SPACE.
  10 EMPREC-JNAME2 PIC N(05) VALUE SPACE.
  10 EMPREC-NAME1 PIC X(05) VALUE SPACE.
  10 EMPREC-NAME2 PIC X(05) VALUE SPACE.
  10 EMPREC-GENDER PIC X(01) VALUE SPACE.
  10 FILLER PIC X(01) VALUE SPACE.
  10 EMPREC-DIV PIC N(05) VALUE ZERO.
  10 EMPREC-DATE-OF-HIRE.
    15 EMPREC-DOH-YYYY PIC 9(04) VALUE ZEROES.
    15 EMPREC-DOH-MM PIC 9(02) VALUE ZEROES.
    15 EMPREC-DOH-DD PIC 9(02) VALUE ZEROES.
  10 FILLER PIC X(01) VALUE SPACE.
  
```

自動的にビルドが実行され、正常にビルドが行われたことを確認します。



```

コンソール 問題 タスク プロパティ
Micro Focus ビルド: BATCHRPT

nature.specific.build.cfg.New_Configuration:

post.build.cfg.New_Configuration:

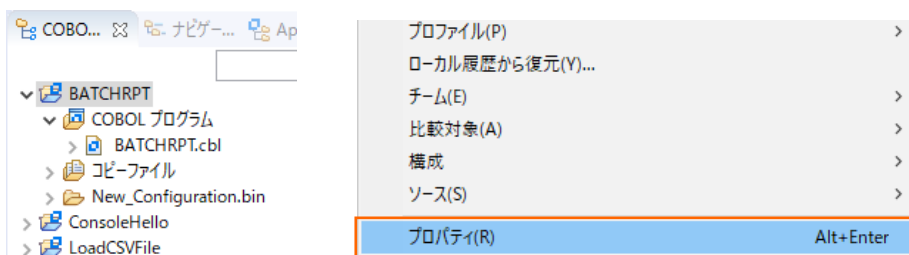
cobolbuild.cfg.New_Configuration:

BUILD SUCCESSFUL
Build finished with no errors.

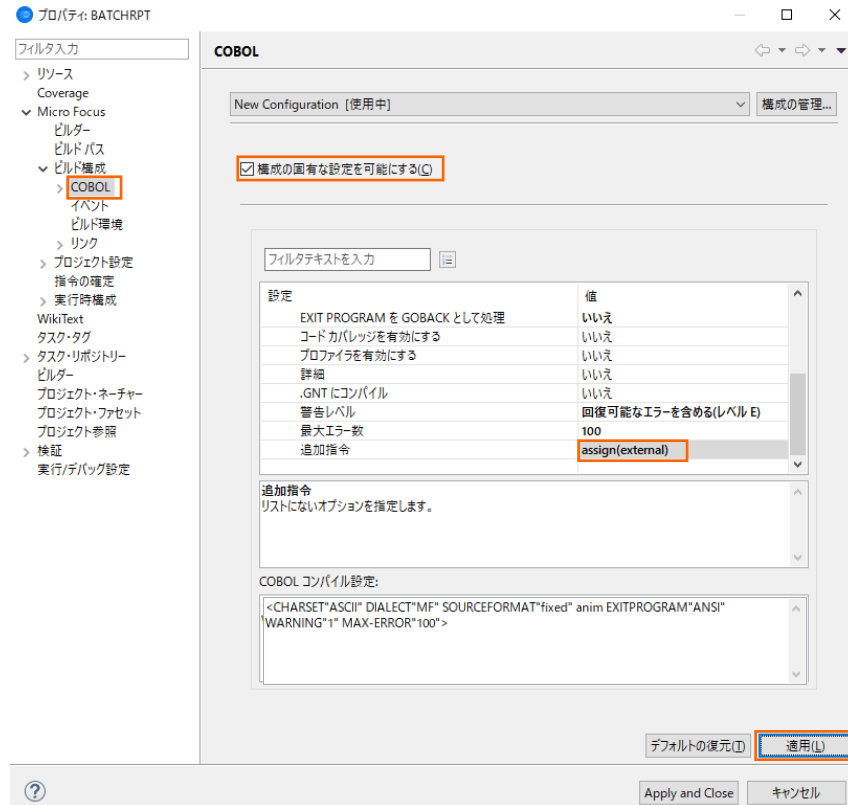
Total time: 1 second
  
```

11) COBOL コンパイル指令を追加します。

COBOL エクスプローラービューにて BATCHRPT プロジェクトを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[プロパティ(R)] を選択します



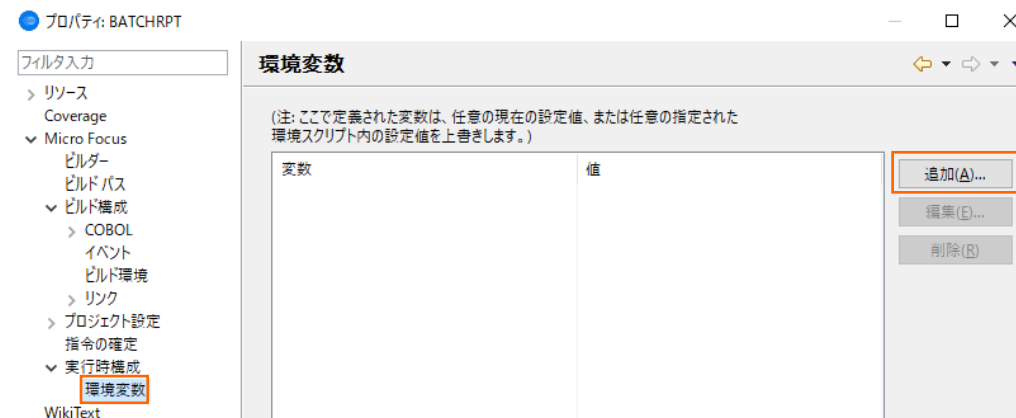
左側のツリーより、[Micro Focus] > [ビルド構成] > [COBOL] を選択します。続いて、画面右側より、「構成の固有な設定を可能にする」にチェックを行い、画面中央の「設定」欄を下にスクロールした上で、「追加指令」に “assign(external)” を入力します。その後、[適用(L)] ボタンをクリックします。



12) 環境変数を構成します。

前手順で開かれているプロパティ画面をクローズした場合、前手順に従い、プロパティ画面を開いてください。

左側のツリーより、[Micro Focus] > [実行時構成] > [環境変数] を選択し、[追加(A)] ボタンをクリックします。



以下の入力を行い、[OK] ボタンをクリックします。

変数： dd_EMPSEQ

値： Emp_Master.dat

● 変数を追加

環境変数を追加または変更します

変数: dd_EMPSEQ

値: Emp_Master.dat

?

OK キャンセル

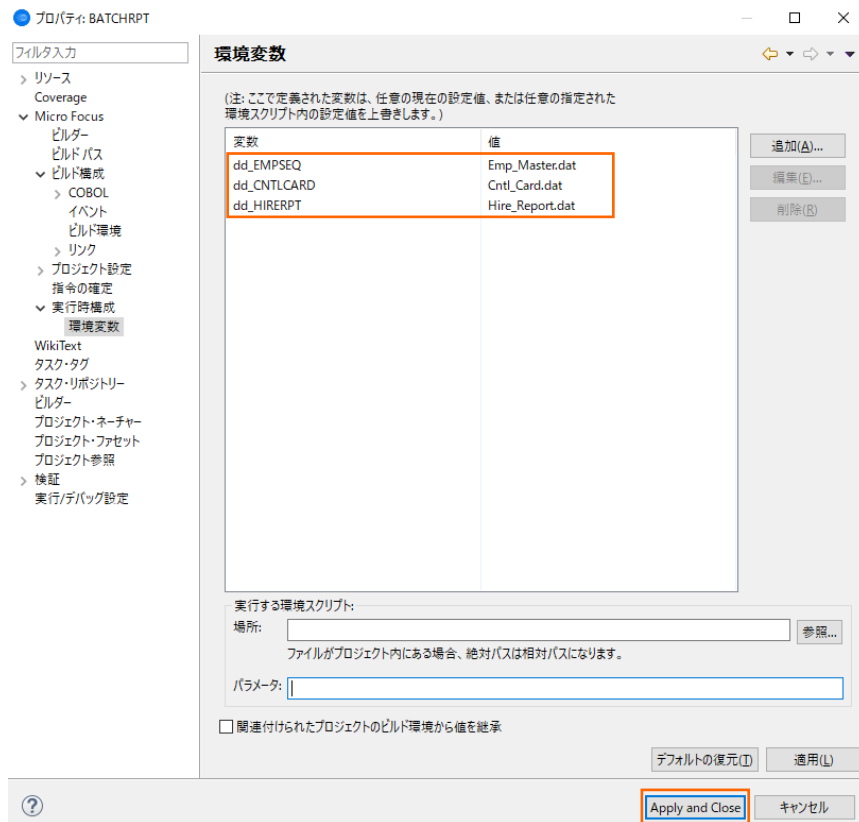
観光変数の一覧に dd_EMPSEQ の情報が登録されます。

環境変数	
(注: ここで定義された変数は、任意の現在の設定値、または任意の指定された環境スクリプト内の設定値を上書きします。)	
変数	値
dd_EMPSEQ	Emp_Master.dat

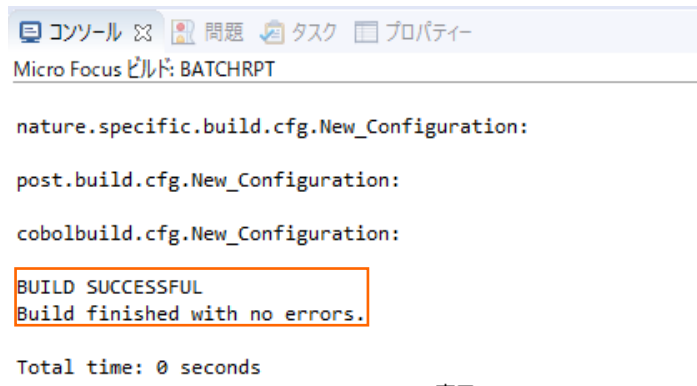
同じ手順で、以下の環境変数の情報を追加してください。

変数	値
dd_CNTLCARD	Cntl_Card.dat
dd_HIRERPT	Hire_Report.dat

下図のように環境変数の情報が表示されていることを確認して、[Apply and Close] ボタンをクリックします。

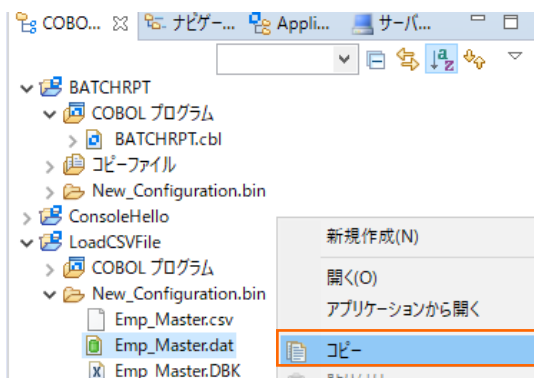


ダイアログがクローズすると、自動的にビルドが実行され、正常に終了することを確認します。

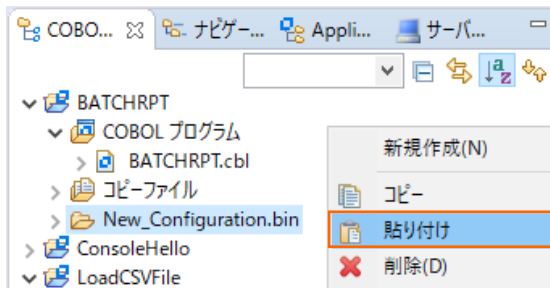


13) 入力ファイルをコピーします。

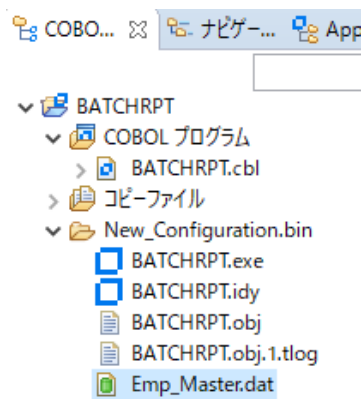
7) にて作成したファイルを入力ファイルとして使用します。COBOL エクスプローラービューにて LoadCSVFile プロジェクトの New_Configuration.bin フォルダ配下にある Emp_Master.dat を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[コピー] を選択します。



COBOL エクスプローラービューにて BATCHRPT プロジェクト配下の New_Configuration.bin フォルダを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[貼り付け] を選択します。

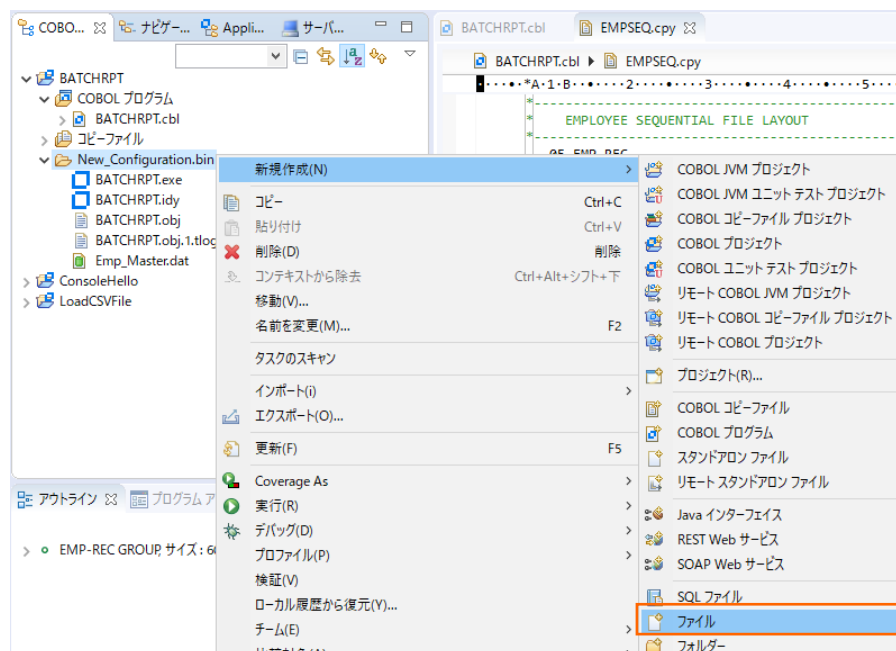


Emp_Master.dat がコピーされます。

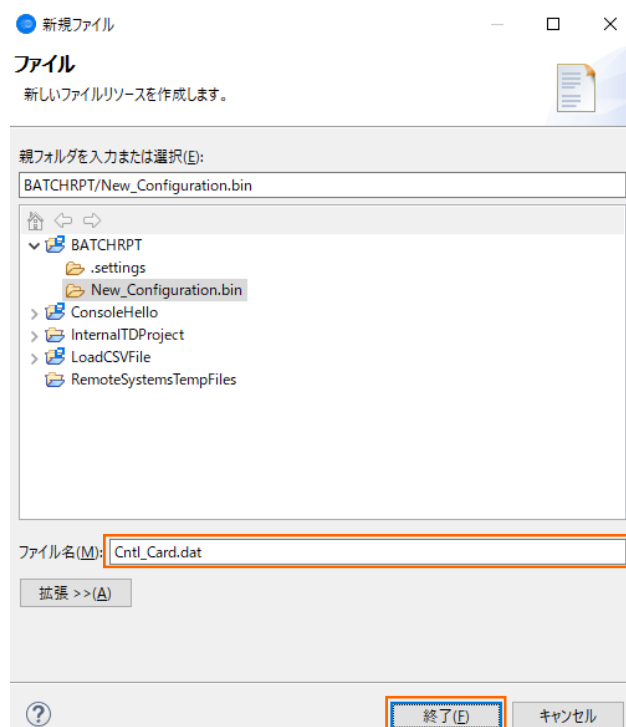


14) 制御ファイルを作成します。

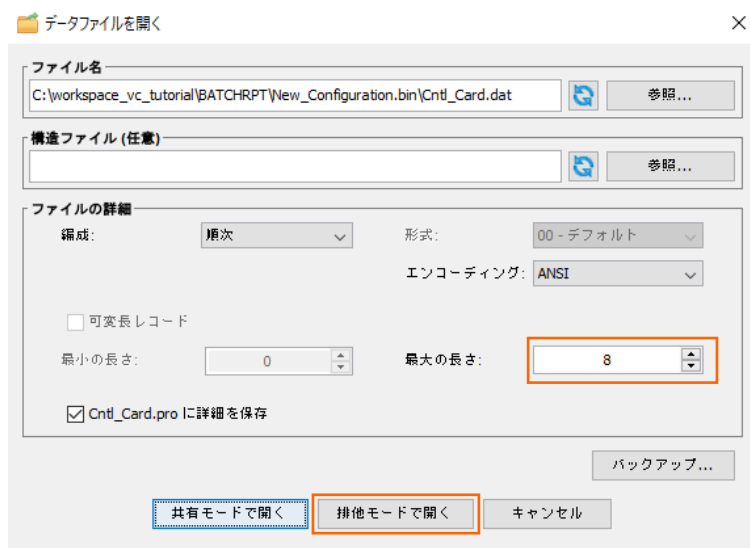
COBOL エクスプローラービューにて BATCHRPT プロジェクト配下の New_Configuration.bin フォルダを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[新規作成(N)] > [ファイル] を選択します。



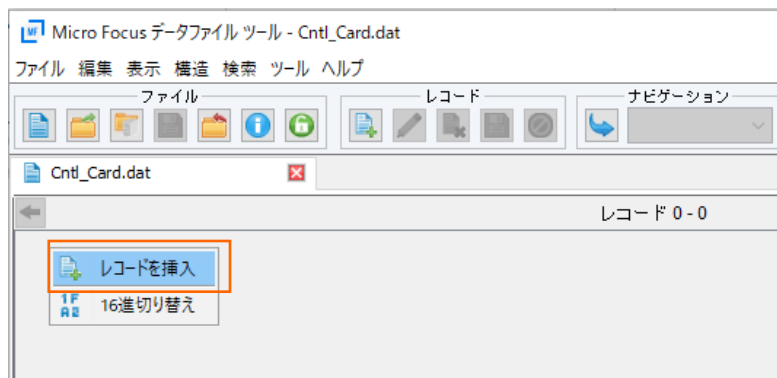
ファイル名に "Cntl_Card.dat" を指定し [終了(F)] ボタンをクリックします。



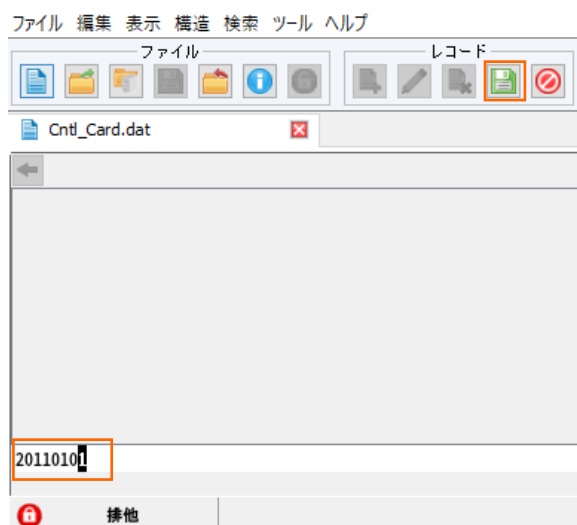
データファイルツールが起動します。「最大の長さ」に 8 を入力して、[排他モードで開く] ボタンをクリックします。



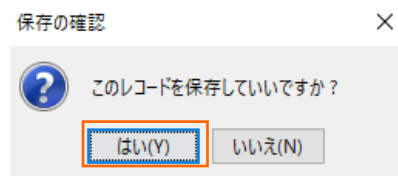
ツール上にて、マウスの右クリックにてコンテキストメニューを表示し、[レコードを挿入] を選択します。



20110101 を入力し、保存します。



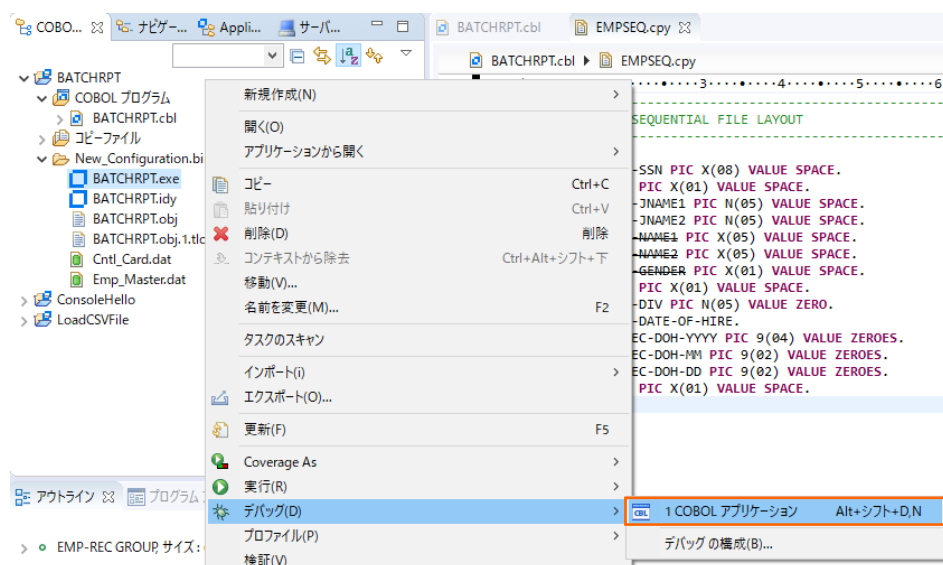
保存アイコンをクリックした後、確認ダイアログが表示されますので、[はい(Y)] ボタンをクリックします。



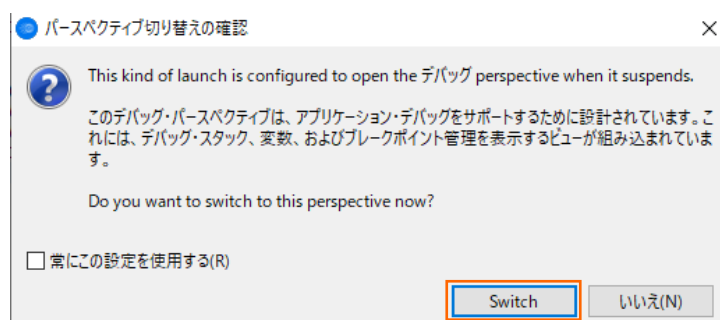
データファイルツールを終了します。

15) COBOL アプリケーションをデバッグ実行します。

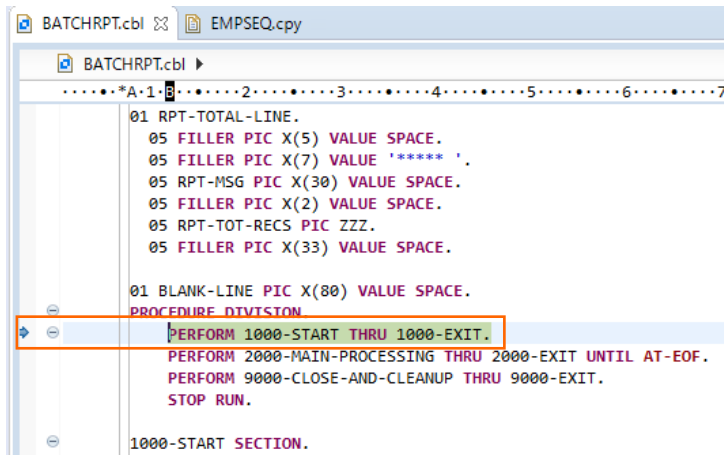
BATCHRPT プロジェクトの New_Configuration.bin フォルダ配下にある BATCHRPT.exe を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[デバッグ(D)] > [COBOL アプリケーション] を選択します。



パースペクティブ切り替えの確認メッセージには [Switch] ボタンをクリックします。



デバッガーがステップ実行を開始します。デバッガーは手続き部の最初の COBOL 文である PERFORM 文の処理前に一時停止している状態となっています。



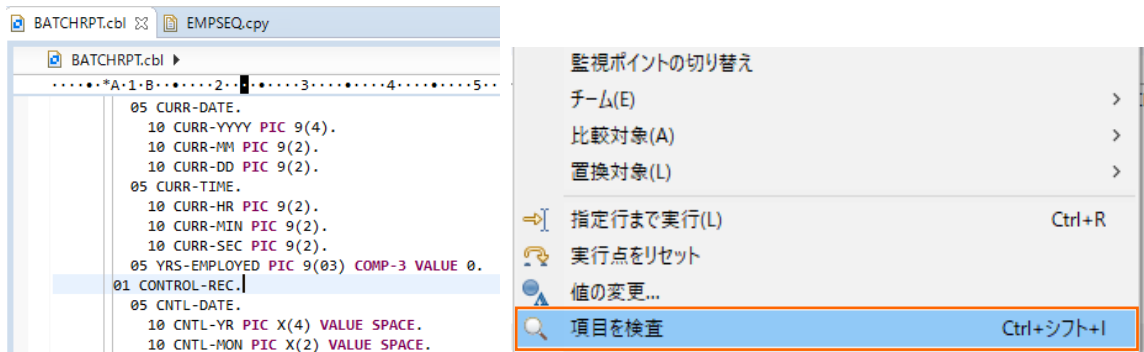
```

01 RPT-TOTAL-LINE.
05 FILLER PIC X(5) VALUE SPACE.
05 FILLER PIC X(7) VALUE '*****'.
05 RPT-MSG PIC X(30) VALUE SPACE.
05 FILLER PIC X(2) VALUE SPACE.
05 RPT-TOT-RECS PIC ZZZ.
05 FILLER PIC X(33) VALUE SPACE.

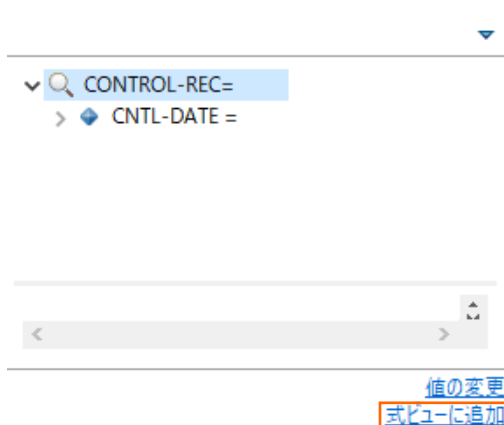
01 BLANK-LINE PIC X(80) VALUE SPACE.
PROCEDURE DIVISION.
PERFORM 1000-START THRU 1000-EXIT.
PERFORM 2000-MAIN-PROCESSING THRU 2000-EXIT UNTIL AT-EOF.
PERFORM 9000-CLOSE-AND-CLEANUP THRU 9000-EXIT.
STOP RUN.

1000-START SECTION.
  
```

制御ファイルから読み込んだレコードの内容を確認するため、CONTROL-REC に格納される値の変遷を追います。データ部の CONTROL-REC を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[項目を検査] を選択します。



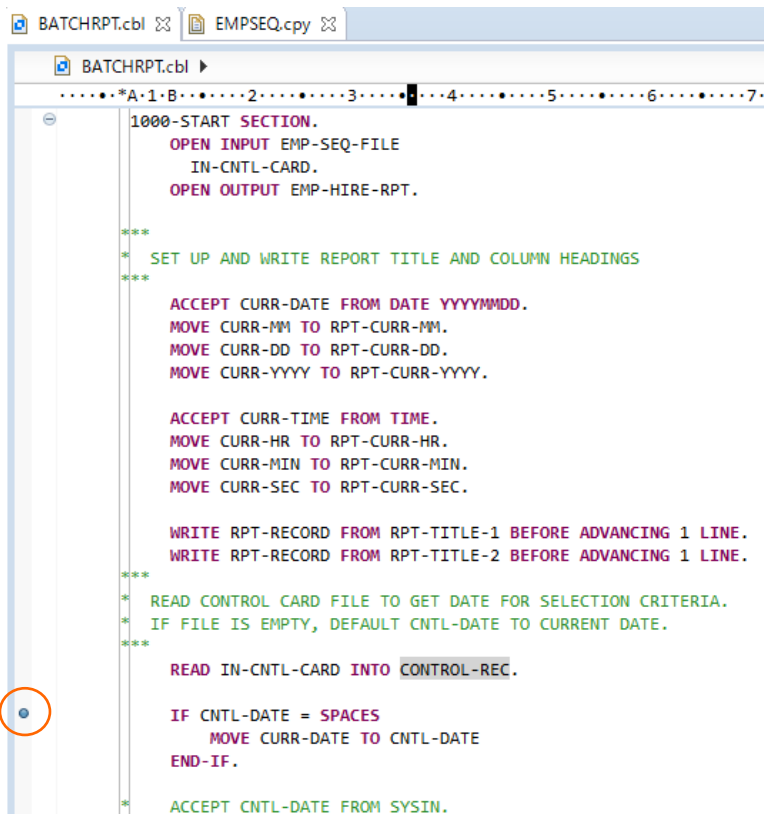
[式ビューに追加] をクリックします。



入力ファイルから読み込んだレコードの内容を確認するため、データ部の EMP-RECORD-IO-AREA を同じ要領で式ビューに追加します。(以前の手順で追加した IN-REC 及び OUT-REC は式ビューから削除しても構いません。)

(x)= 変数 ● ブレークポイント プログラム アウトライン 式	
名前	値
> X+y "CONTROL-REC"	
> X+y "EMP-RECORD-IO-AREA"	0 0 0 ...
+ 新しい式を追加	

手続き部 1000-START 節の READ 文に続く IF 文でエディタービューの左端をダブルクリックし、ブレークポイントを設定します。



```

BATCHRPT.cbl EMPSEQ.cpy
BATCHRPT.cbl ▶
.....*A.1.B.....2.....3.....4.....5.....6.....7.....
1000-START SECTION.
  OPEN INPUT EMP-SEQ-FILE
  IN-CNTL-CARD.
  OPEN OUTPUT EMP-HIRE-RPT.

***
* SET UP AND WRITE REPORT TITLE AND COLUMN HEADINGS
***

  ACCEPT CURR-DATE FROM DATE YYYYMMDD.
  MOVE CURR-MM TO RPT-CURR-MM.
  MOVE CURR-DD TO RPT-CURR-DD.
  MOVE CURR-YYYY TO RPT-CURR-YYYY.

  ACCEPT CURR-TIME FROM TIME.
  MOVE CURR-HR TO RPT-CURR-HR.
  MOVE CURR-MIN TO RPT-CURR-MIN.
  MOVE CURR-SEC TO RPT-CURR-SEC.

  WRITE RPT-RECORD FROM RPT-TITLE-1 BEFORE ADVANCING 1 LINE.
  WRITE RPT-RECORD FROM RPT-TITLE-2 BEFORE ADVANCING 1 LINE.

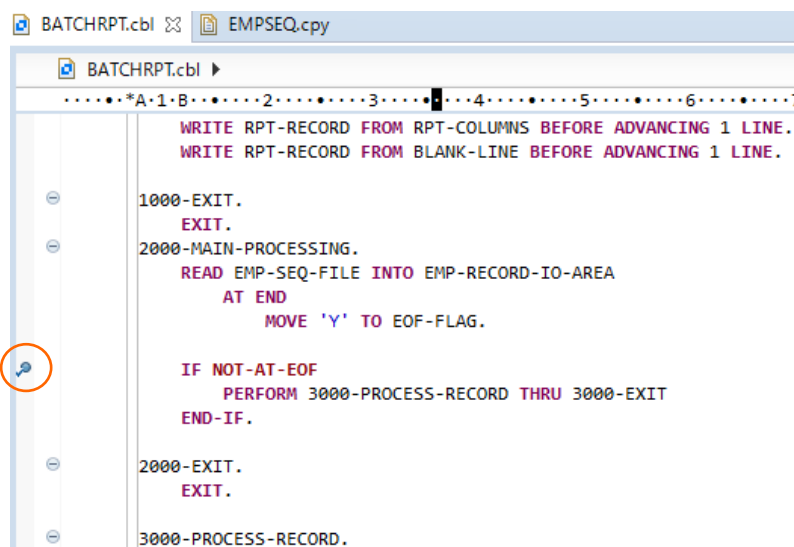
***
* READ CONTROL CARD FILE TO GET DATE FOR SELECTION CRITERIA.
* IF FILE IS EMPTY, DEFAULT CNTL-DATE TO CURRENT DATE.
***

  READ IN-CNTL-CARD INTO CONTROL-REC.

  IF CNTL-DATE = SPACES
    MOVE CURR-DATE TO CNTL-DATE
  END-IF.

* ACCEPT CNTL-DATE FROM SYSIN.
  
```

同様に手続き部 2000-MAIN-PROCESSING 段落の READ 文に続く IF 文に、ブレークポイントを設定します。



```

.....*A-1-B.....2.....3.....4.....5.....6.....
WRITE RPT-RECORD FROM RPT-COLUMNS BEFORE ADVANCING 1 LINE.
WRITE RPT-RECORD FROM BLANK-LINE BEFORE ADVANCING 1 LINE.

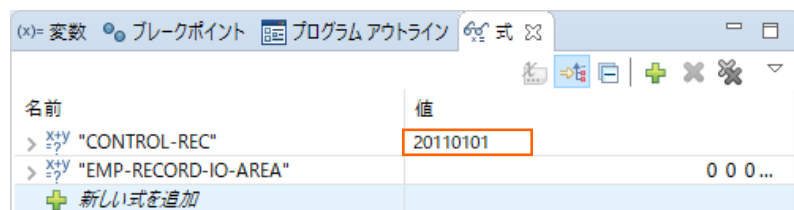
1000-EXIT.
EXIT.
2000-MAIN-PROCESSING.
  READ EMP-SEQ-FILE INTO EMP-RECORD-IO-AREA
  AT END
    MOVE 'Y' TO EOF-FLAG.

  IF NOT-AT-EOF
    PERFORM 3000-PROCESS-RECORD THRU 3000-EXIT
  END-IF.

2000-EXIT.
EXIT.

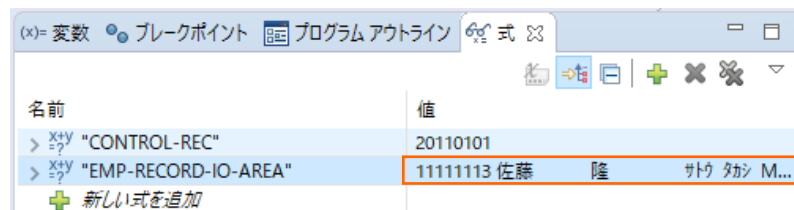
3000-PROCESS-RECORD.
  
```

メニューより、[実行(R)] > [再開(M)] を選択するか、ホットキーである F8 キーを押すことで、デバッガーは最初のブレークポイントで実行を中断します。式ビュー中の CONTROL-REC の値に制御ファイルから読み込んだレコードが表示されます。



名前	値
> X+Y "CONTROL-REC"	20110101
> X+Y "EMP-RECORD-IO-AREA"	0 0 0 ...
+ 新しい式を追加	

再度、F8 キーを押すと、デバッガーは 2 番目のブレークポイントで実行を中断します。式ビューの EMP-RECORD-IO-AREA の値に入力ファイルから読み込んだ 1 番目のレコードが表示されます。



名前	値
> X+Y "CONTROL-REC"	20110101
> X+Y "EMP-RECORD-IO-AREA"	11111113 佐藤 隆 サウ タシ M...
+ 新しい式を追加	

再度 F8 キーを押すと、デバッガーは同一のブレークポイントで実行を中断します。ウォッチ式の EMP-RECORD-IO-AREA の値には、入力ファイルから読み込んだ 2 番目のレコードが表示されます

名前				値
> X+y	"CONTROL-REC"			20110101
✓ X+y	"EMP-RECORD-IO-AREA"			22222226 鈴木 尚之 スキナオキ ...
✓	EMP-RECORD-IO-AREA			22222226 鈴木 尚之 スキナオキ ...
✓	EMP-REC			22222226 鈴木 尚之 スキナオキ ...
✓	EMPREC-SSN			22222226
✓	FILLER			
✓	EMPREC-JNAME1			鈴木
✓	EMPREC-JNAME2			尚之
✓	EMPREC-NAME1			スキ
✓	EMPREC-NAME2			ウ2
✓	EMPREC-GENDER			M
✓	FILLER			
✓	EMPREC-DIV			技術部
> ✓	EMPREC-DATE-OF-HIRE			19981015
✓	FILLER			

+ 新しい式を追加

ブレイクポイントは、現在設定中の行の左端をダブルクリックすることで解除できます。さきほど設定した2つのブレイクポイントを解除してください。

BATCHRPT.cbl EMPSEQ.cpy

BATCHRPT.cbl ▶

```

1.....*A.1.B.....2.....3.....4.....5.....6.....7.
READ CONTROL-CARD FILE TO GET DATE FOR SELECTION CRITERIA.
IF FILE IS EMPTY, DEFAULT CNTL-DATE TO CURRENT DATE.
***
READ IN-CNTL-CARD INTO CONTROL-REC.

IF CNTL-DATE = SPACES
    MOVE CURR-DATE TO CNTL-DATE
END-IF.

*
ACCEPT CNTL-DATE FROM SYSIN.

MOVE CNTL-MON TO RPT-SELECTION-MM.
MOVE CNTL-DAY TO RPT-SELECTION-DD.
MOVE CNTL-YR TO RPT-SELECTION-YYYY.

WRITE RPT-RECORD FROM RPT-TITLE-3 BEFORE ADVANCING 1 LINE.
WRITE RPT-RECORD FROM BLANK-LINE BEFORE ADVANCING 1 LINE.

WRITE RPT-RECORD FROM RPT-COLUMNS BEFORE ADVANCING 1 LINE.
WRITE RPT-RECORD FROM BLANK-LINE BEFORE ADVANCING 1 LINE.

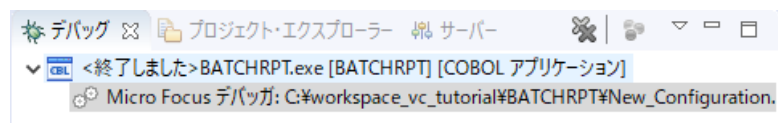
1000-EXIT.
EXIT.
2000-MAIN-PROCESSING.
    READ EMP-SEQ-FILE INTO EMP-RECORD-IO-AREA
    AT END
        MOVE 'Y' TO EOF-FLAG.

    IF NOT-AT-EOF
        PERFORM 3000-PROCESS-RECORD THRU 3000-EXIT
    END-IF.

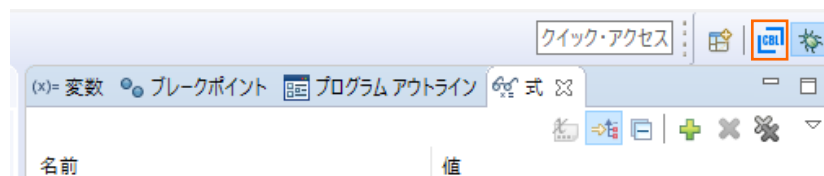
2000-EXIT.

```

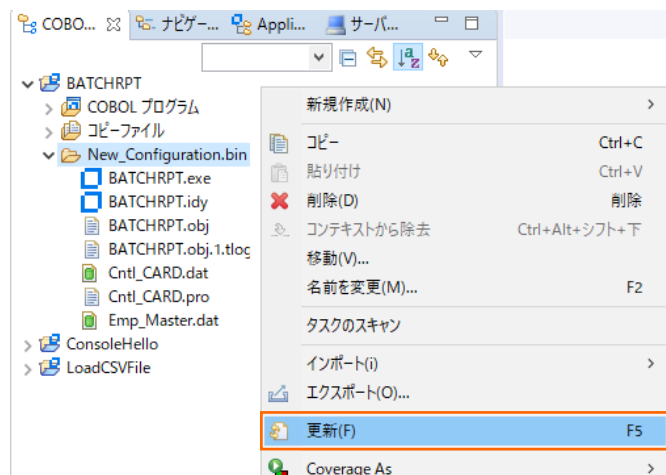
メニューより、[実行(R)] > [再開(M)] を選択するか F8 キーを押してプログラムを実行します。今は、ブレークポイントが設定されていないため、そのままプログラムは正常終了します。



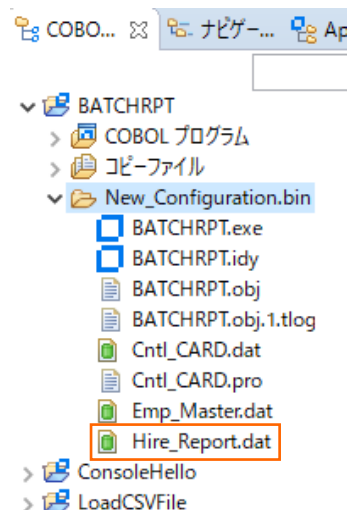
ワークスペース右上にて COBOL アイコン を選択し、COBOL パースペクティブに戻ります。



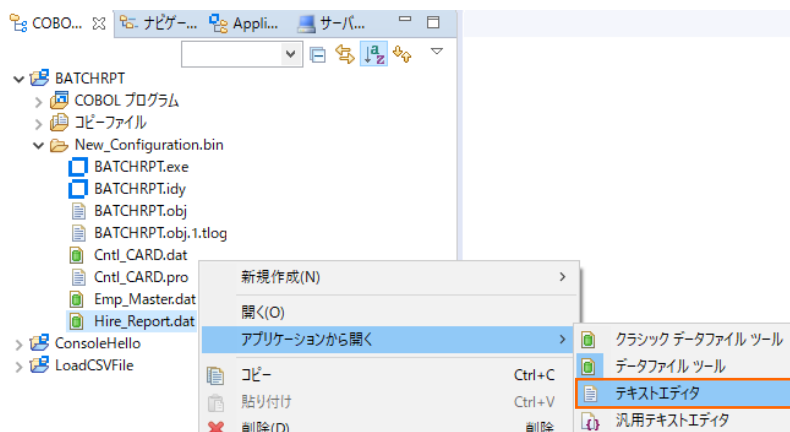
COBOL エクスプローラービューにて BATCHRPT プロジェクト配下の New_Configuration.bin フォルダを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[更新(F)] を選択します。



Hire_Report.dat が作成されていることを確認します。



Hire_Report.dat を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[アプリケーションから開く] > [テキストエディタ] を選択します。



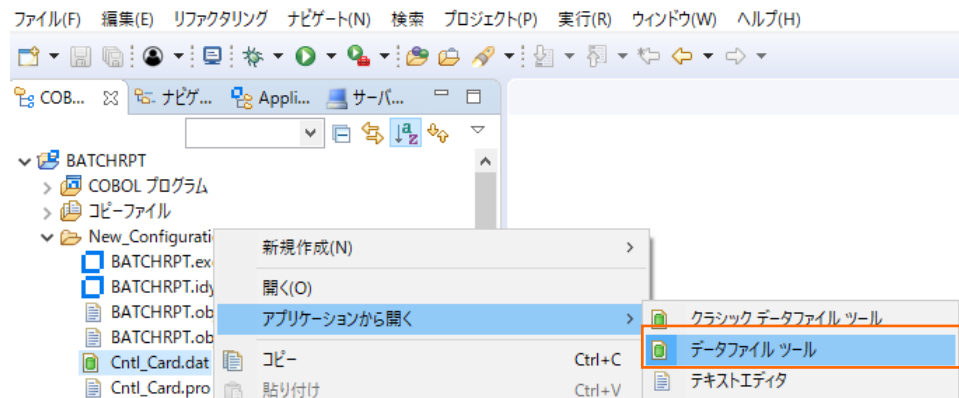
社員 9 名分のデータが表示されることを確認します。



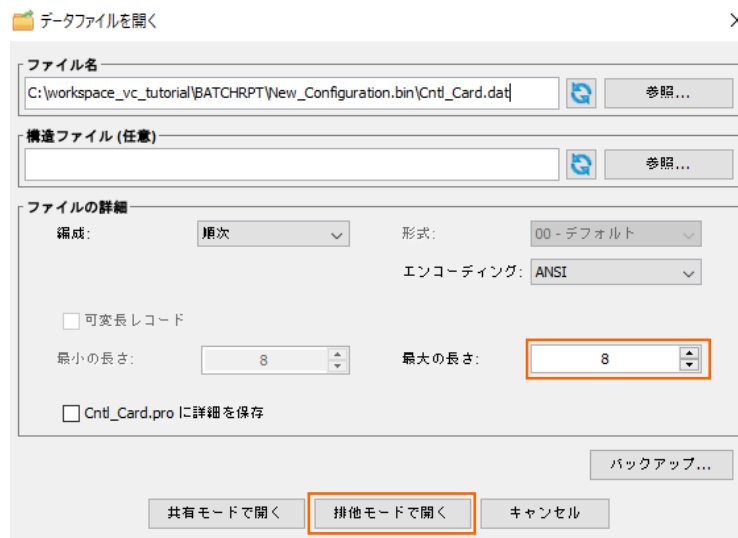
補足)

Eclipse におけるデフォルトのテキストエディタフォントがプロポーショナルになっている場合は多少見た目が異なる可能性があります。この場合、テキストエディタ上で右クリックから [設定(F)] を選択し [一般] > [外観] > [色とフォント] で表示されるページにてフォントを変更できます。

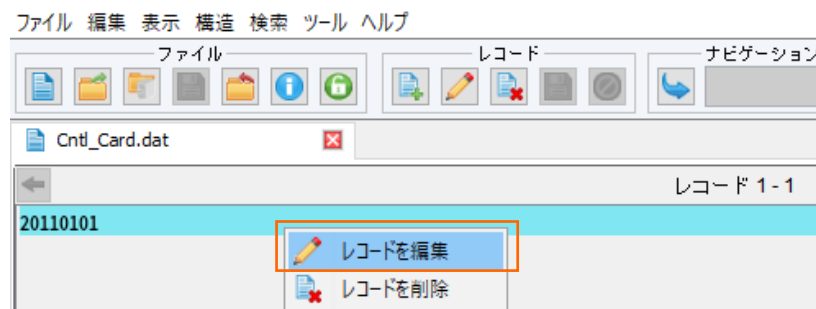
続いて、制御ファイルの値を変更してアプリケーションを実行します。COBOL エクスプローラー上で Cntl_Card.dat ファイルを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[アプリケーションから開く] > [データファイル ツール] を選択します。



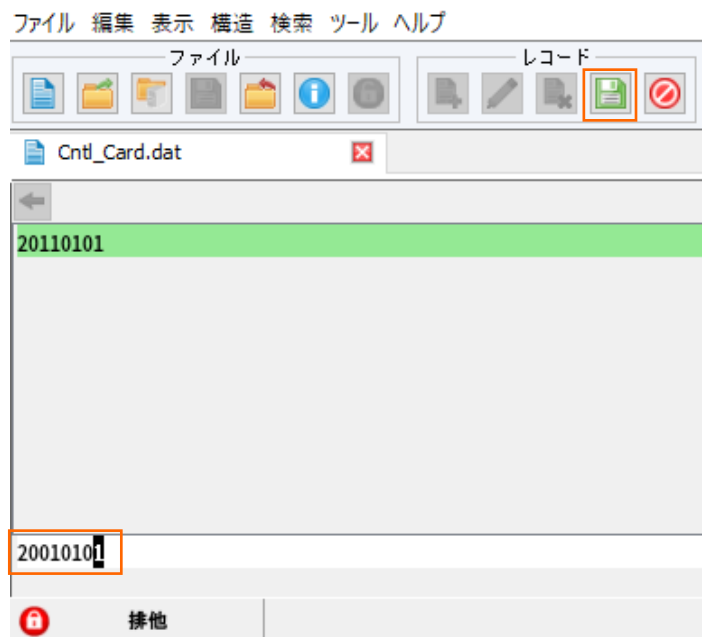
「最大の長さ」に 8 が入力されていることを確認して、[排他モードで開く] ボタンをクリックします。



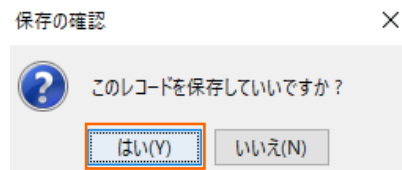
20110101 のレコードを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[レコードを編集] を選択します。



ツール上で、“20110101” の値を “20010101” に更新したうえで、保存アイコンをクリックします。

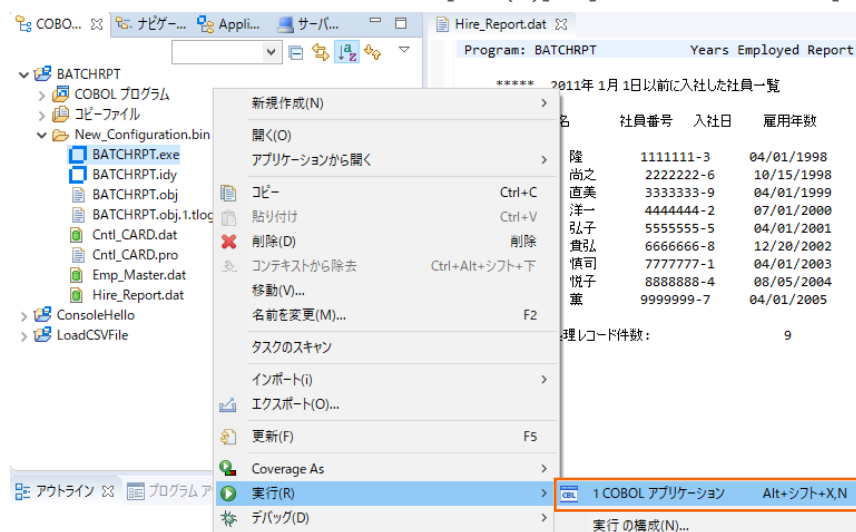


さきほど同様、確認ダイアログが表示されますので、[はい(Y)] ボタンをクリックします。



保存後、ツールを終了してください。

BATCHRPT プロジェクトの New_Configuration.bin フォルダ配下にある BATCHRPT.exe を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[実行(R)] > [COBOL アプリケーション] を選択します。

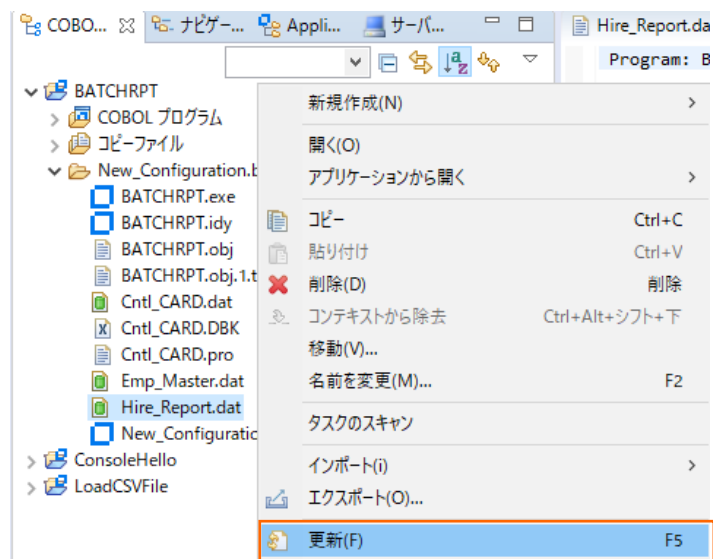


以下のコンソール画面がポップアップされたら、メッセージに従い、任意のキーを押し、アプリケーションを終了します。

```

C:\Windows\SYSTEM32\cmd.exe
*** REPORT CREATED SUCCESSFULLY ***
*** VIEW: HIRERPT.DAT ***
続行するには何かキーを押してください . . .
  
```

Hire_Report.dat を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[更新(F)] を選択します。



2000 年 1 月 1 日以前に入社した社員 4 名分のデータだけが表示されることを確認します。

Program: BATCHRPT					Years Employed Report	07/10/2019
						10:05:18
*****					2001年 1月 1日以前に入社した社員一覧	
部署名	社員名	社員番号	入社日	雇用年数		
営業部	佐藤 隆	1111111-3	04/01/1998	21		
技術部	鈴木 尚之	2222222-6	10/15/1998	21		
総務部	田中 直美	3333333-9	04/01/1999	20		
営業部	山田 洋一	4444444-2	07/01/2000	19		
*****					処理レコード件数:	4

免責事項

ここで紹介したソースコードは、機能説明のためのサンプルであり、製品の一部ではございません。ソースコードが実際に動作するか、御社業務に適合するかなどに関しまして、一切の保証はございません。ソースコード、説明、その他すべてについて、無謬性は保障されません。

ここで紹介するソースコードの一部、もしくは全部について、弊社に断りなく、御社の内部に組み込み、そのままご利用頂いても構いません。

本ソースコードの一部もしくは全部を二次的著作物に対して引用する場合、著作権法の精神に基づき、適切な扱いを行ってください。