
Micro Focus Visual COBOL 5.0 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

Visual Studio 2019

1. 目的

Micro Focus Visual COBOL for Visual Studio 2019 は、マイクロソフト社の最新開発環境である Visual Studio 2019 の IDE（統合開発環境）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 プログラマが既存の COBOL 資産を Windows 環境で活用するだけでなく、COBOL 言語の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 Visual Studio 2019 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 5.0J for Visual Studio 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

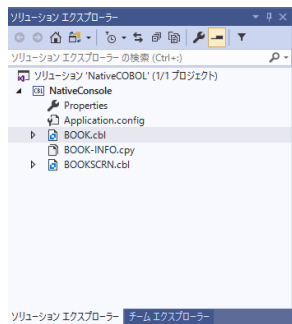
下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダに解凍しておいてください。

[サンプルプログラムのダウンロード](#)

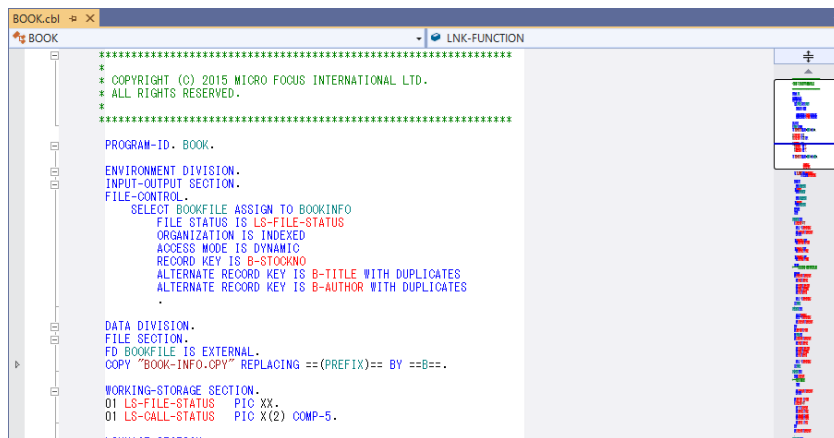
内容

1. 目的
2. 前提
3. チュートリアル
 - 3.1. Visual Studio 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

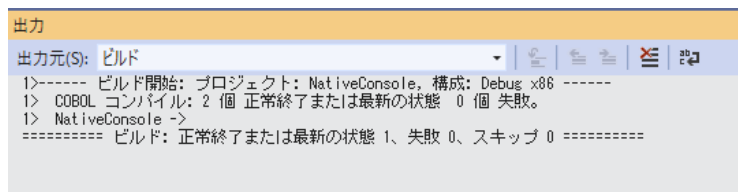
Visual Studio 2019 のソリューションとプロジェクトには、アプリケーションの作成に必要な参照、データ接続、フォルダ、およびファイルを表す項目が含まれています。ソリューションには複数のプロジェクトを含めることができ、プロジェクトには、通常、複数の項目が含まれます。ソリューションエクスプローラーには、ソリューション、それらのプロジェクト、そのプロジェクト内の項目が表示されます。ソリューションエクスプローラーを使用すると、編集するファイルを開く、プロジェクトに新規ファイルを追加する、ソリューション、プロジェクト、および項目のプロパティを表示するなどの操作を実行できます。



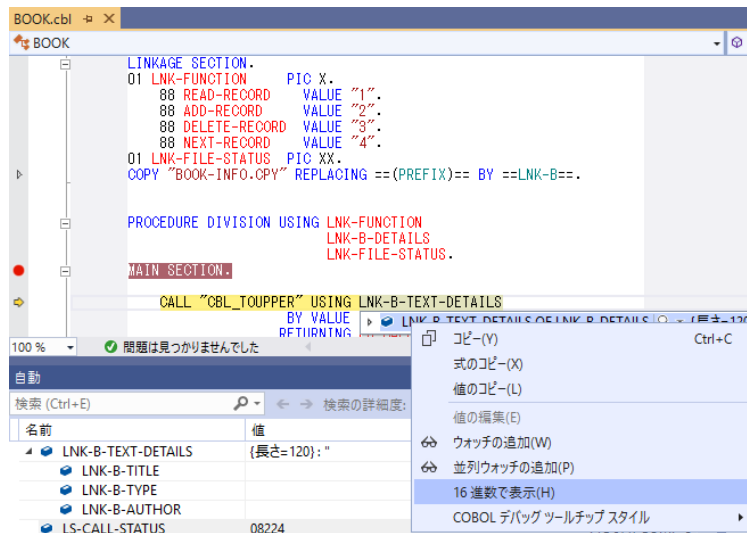
Visual Studio 2019 のソースコードエディターには、COBOL 予約語とデータ名や手続き名などの利用者語を色分け表示や、COBOL スニペットなど COBOL 言語固有の機能拡張が含まれます。ソースコードを入力するとバックグラウンドチェックを実行して、赤の波線でエラー箇所を強調表示します。そのエラー箇所にマウスポインタを移動すればエラー内容の確認、定義への移動、他の参照検索などの操作が可能です。



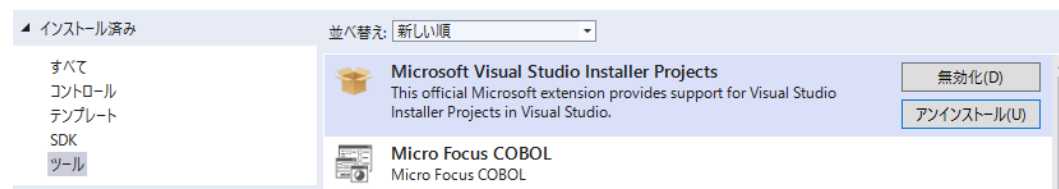
Visual Studio 2019 のビルド構成では、プラットフォームの選択、プロジェクトまたはソリューションのビルド方法を定義します。プロジェクトタイプごとに、デバッグとリリースのデフォルト構成があり、独自の構成を作成することも可能です。コンソールウィンドウにはビルド時のメッセージやアプリケーションのコンソール出力等が表示されます。問題ウィンドウには、不正な構文、キーワードのスペルミス、型の不一致などのコンパイルエラーが表示されます。



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、デバッガーを使用します。Visual Studio 2019 のデバッガーは、コードのステップ実行、様々な条件を設定したブレイクポイントで実行、変数ウィンドウやウォッチ式などのツール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。



デバッグが完了したアプリケーションは、Windows インストーラーを使用するか、ファイルを手動でコピーして、本番環境に配置します。Visual Studio 2019 では、Visual Studio の Marketplace より「Microsoft Visual Studio Installer Projects」をダウンロードし、インストールすることで下図のような各種インストーラー作成用のプロジェクトをご利用できます。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 Studio 2019 を起動します。

Windows のスタートメニューから Visual Studio 2019 をクリックします。Microsoft Visual Studio 2019 が起動されるので「新しいプロジェクトの作成」を選択します。

- 2) 使用するテンプレートを選択します。

フィルター画面が表示されるので、言語に “COBOL”、プラットフォームに “Windows”、プロジェクト タイプに “ネイティブ” を選択し、一覧から「コンソールアプリケーション」を選んで、[次へ(N)] ボタンをクリックします。



- 3) 以下の入力を行い、[作成(C)] ボタンをクリックします。

プロジェクト名： ConsoleHello

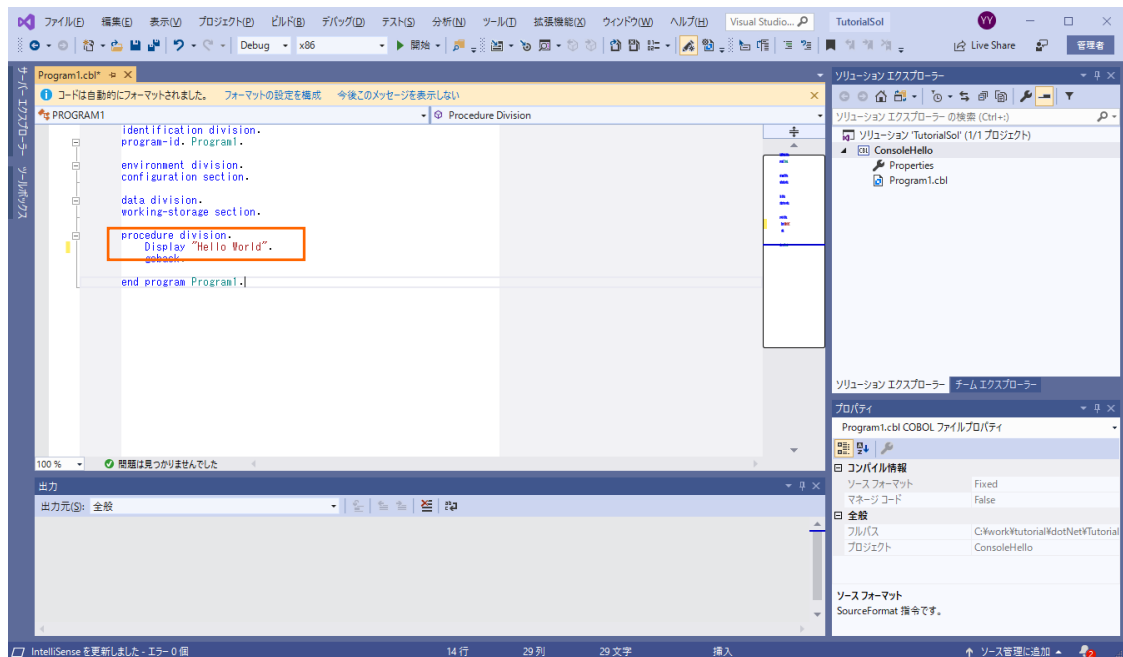
ソリューション名： TutorialSol



- 4) Visual Studio のコードエディターで COBOL のソースコードを編集します。

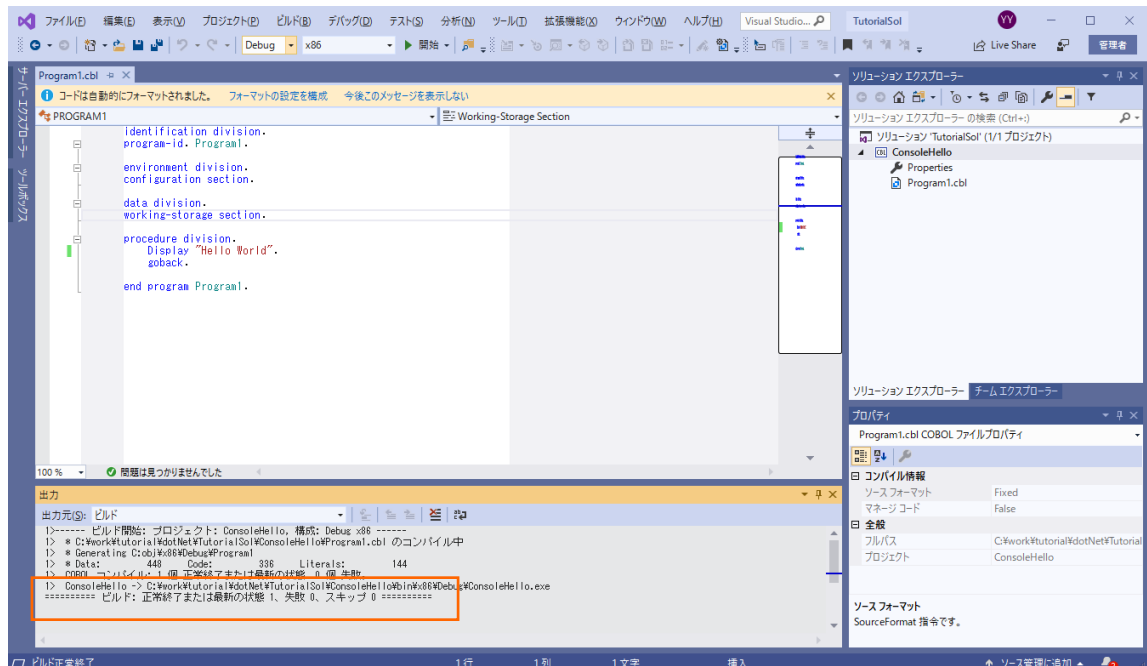
ConsoleHello プロジェクトの作成が成功すると、COBOL 専用のコードエディターが起動します。エディター画面には、コンソールアプリケーションのひな形が表示されています。COBOL ソースは、見出し部 (identification division)、環境部 (environment division)、データ部 (data division)、手続き部 (procedure division) で構成されますが、今回は「Hello World」を表示して終了するプログラムなので、手続き部に DISPLAY 文を書き加えるだけです。なお、COBOL 正書法ではエディター画面左右にあるグレー部分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。

今回は、procedure division の下に「Display "Hello World".」とタイプします。



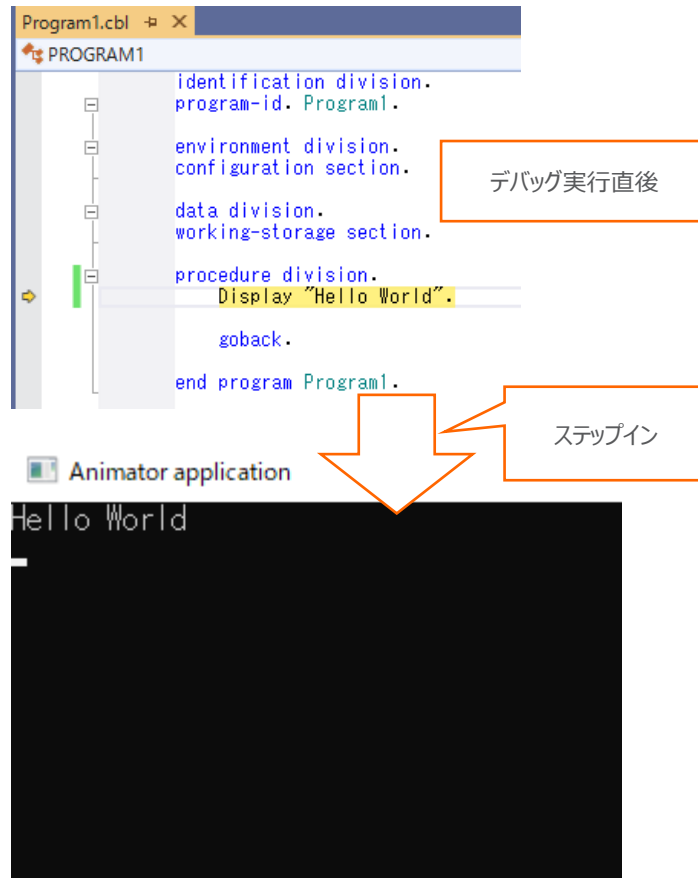
5) COBOL アプリケーションをビルドします。

終止符（ピリオド）を含めてスペルミスがなければ、ソリューション構成が Debug、ソリューションプラットフォームが x86 であることを確認して、メニューより、[ビルド(B)] > [ソリューションのビルド(B)] を選択します。出力ウィンドウにビルド結果が表示されるので、すべてのビルドが正常終了したことを確認します。



6) COBOL アプリケーションをデバッグ実行します。

メニューより、[デバッグ(D)] > [ステップイン(I)] を選択すると、コマンドプロンプト画面が開き、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 `display` 文を実行する前の状態で停止します。今回は調べるローカル変数がないので、そのまま [ステップイン(I)] を選択し、ステップ実行を進めます。

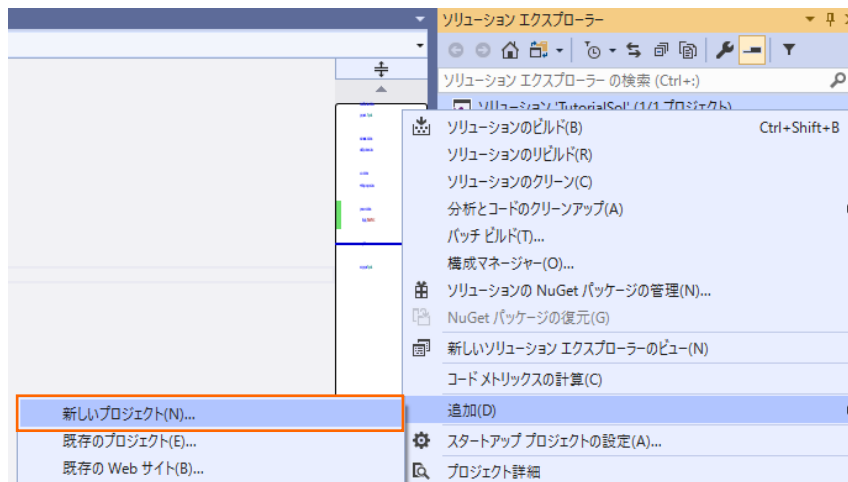


コマンドプロンプト画面に “Hello World” が表示されたことを確認して、デバッグを終了します。

次に、ウィンドウ画面のボタンを押して “Hello World” を表示する COBOL アプリケーションを作成します。

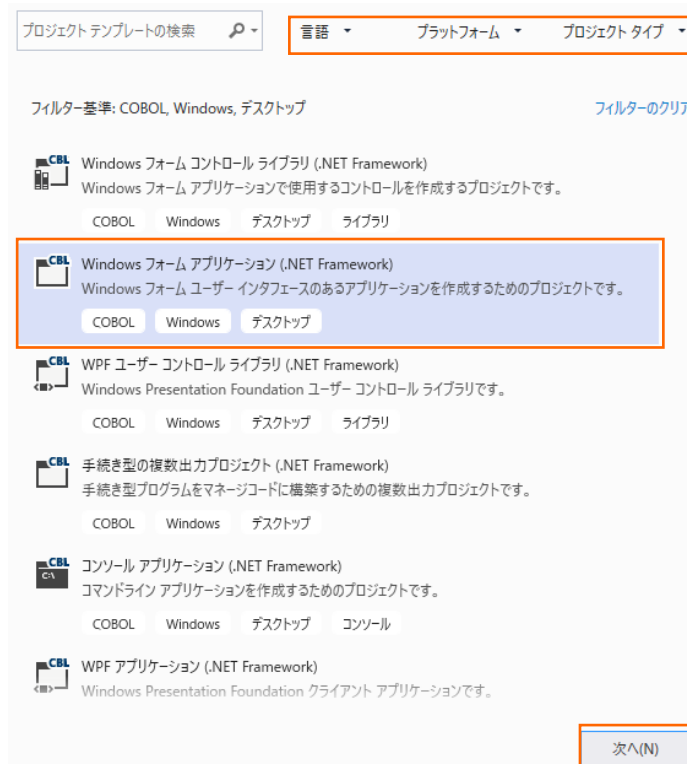
7) 作成したソリューションへプロジェクトを追加します。

これまでに作成したソリューション中のソリューションエクスプローラーにて、ソリューションを選択した状態で、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、[追加(D)] > [新しいプロジェクト(N)...] を選択します。



8) 使用するテンプレートを選択します。

フィルター画面が表示されるので、言語に “COBOL”、プラットフォームに “Windows”、プロジェクト タイプに “デスクトップ” を選択し、一覧から「Windows フォームアプリケーション(.NET Framework)」を選んで、[次へ(N)] ボタンをクリックします。



- 9) プロジェクト名に “WinHello” を入力し、[作成(C)] ボタンをクリックします。

新しいプロジェクトを構成します

Windows フォーム アプリケーション (.NET Framework) COBOL Windows デスクトップ

プロジェクト名
WinHello

場所
C:\Users\%tarot%\source\repos\TutorialSol

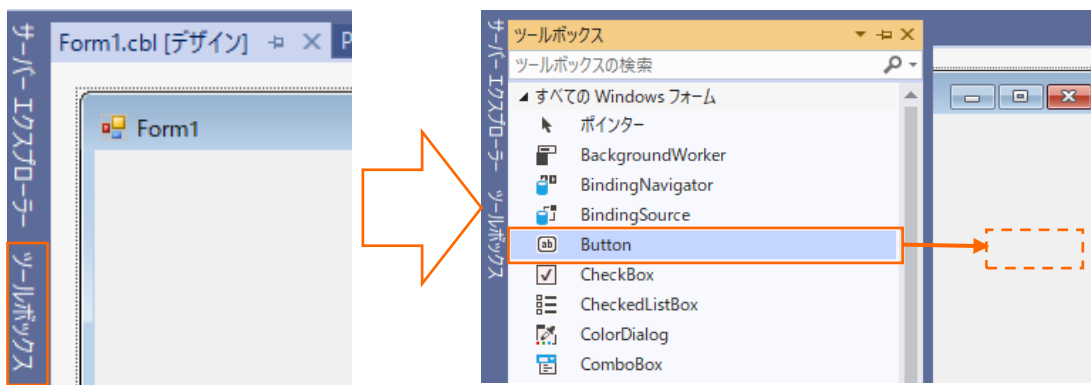
フレームワーク
.NET Framework 4.7.2

戻る(B) 作成(C)

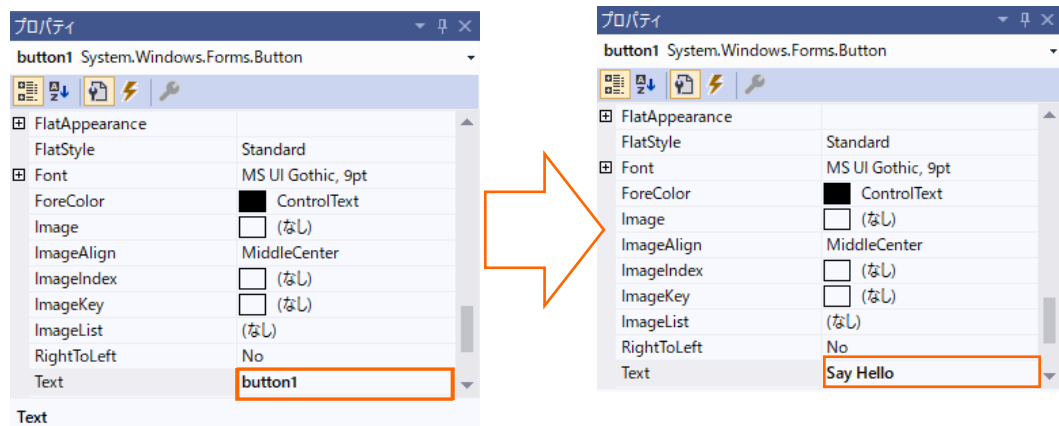
- 10) フォームデザイナーでウィンドウを作成します。

WinHello プロジェクトの作成が成功すると、フォームデザイナーが起動します。

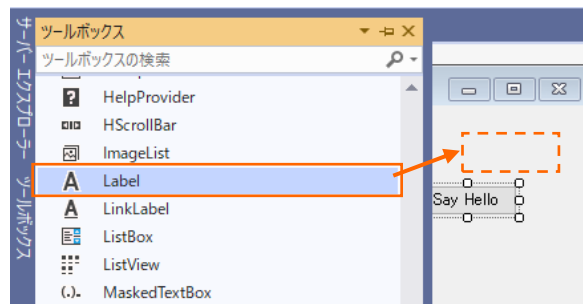
デザイナー画面に Form1 ウィンドウが表示されるので、画面左に表示される ツールボックス を選択して展開します。表示されたツールボックス中のすべての Windows フォームを展開します。続いて、Button コントロールを選択し、Form1 ウィンドウ上にドラッグ&ドロップします。



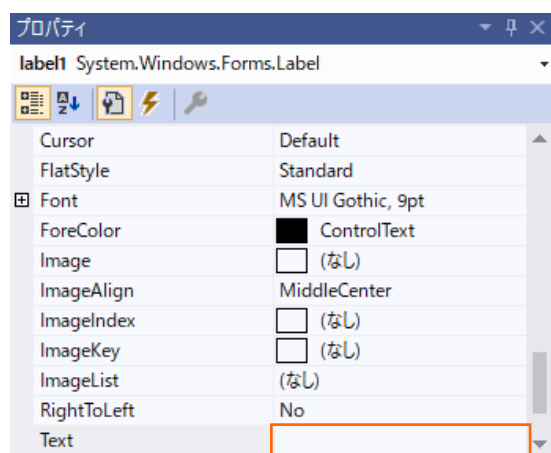
Form1 ウィンドウ上にボタンが表示されると、プロパティが Button1 ボタンに切り替わります。プロパティを下方方向にスクロールして「Text」を選択し、値を“Say Hello”に変更します。



ツールボックスをスクロールして Label コントロールを選択し、Form1 ウィンドウ上にドラッグ&ドロップします。



プロパティをスクロールして「Text」を選択し、テキストの値を削除します。

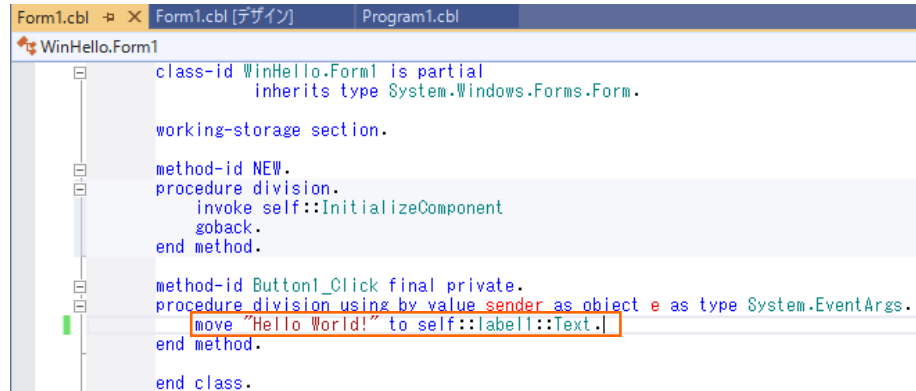


11) コードエディターで COBOL ソースコードを入力します。

デザイナー画面上の [Say Hello] ボタンをダブルクリックすると、COBOL 専用のコードエディターが起動します。

エディター画面には、Windows フォームアプリケーションのひな形が表示されます。ここでは [Say Hello] ボタンをクリックした時の処理を記述するので、Button1_Click メソッドの手続き部に以下の move 文を追加します。

move "Hello World!" to self::label1::Text.



```

class-id WinHello.Form1 is partial
    inherits type System.Windows.Forms.Form.

working-storage section.

method-id NEW.
    procedure division.
        invoke self::InitializeComponent
        goback.
    end method.

method-id Button1_Click final private.
    procedure division using by value sender as object e as type System.EventArgs.
        move "Hello World!" to self::label1::Text.
    end method.

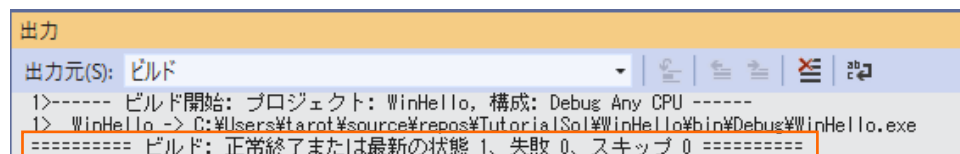
end class.
    
```

12) COBOL アプリケーションをビルドします。

メニューより、[ビルド(B)] > [WinHello のビルド(U)] を選択します。

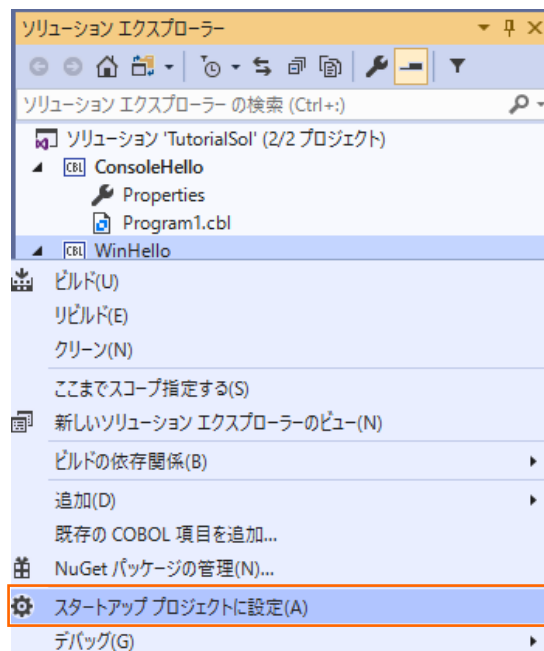


出力ウィンドウにビルド結果が表示されますので、ビルドが正常終了したことを確認します。

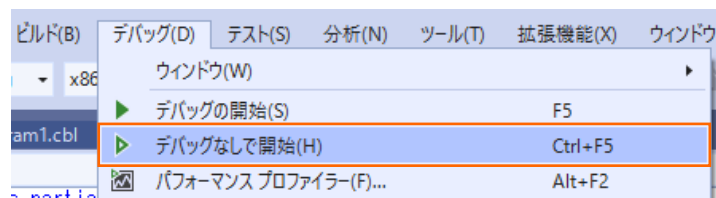


13) COBOL アプリケーションを実行します。

ソリューションエクスプローラーにて WinHello 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、[スタートアッププロジェクトに設定(A)] を選択します。



メニューより、[デバッグ(D)] > [デバッグなしで開始(H)] を選択すると、Form1 ウィンドウが開きます。



Form1 ウィンドウの [Say Hello] ボタンをクリックして “Hello World!” の表示を確認します。



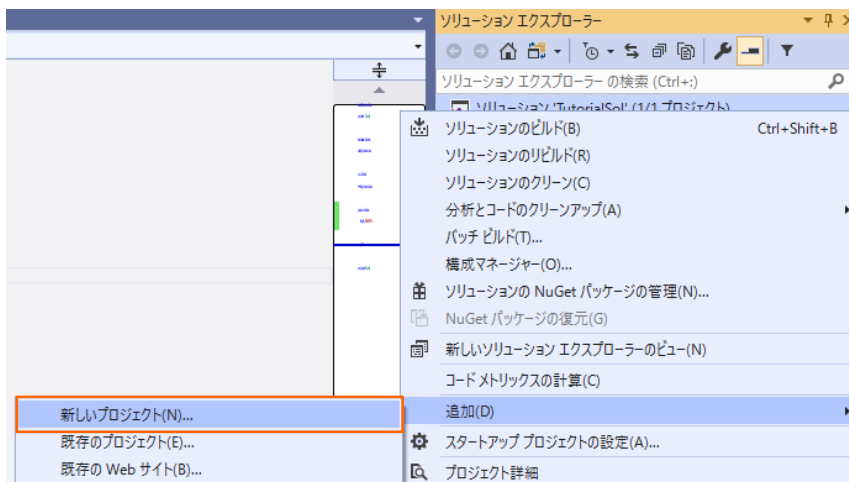
アプリケーションの終了は、[X] ボタンをクリックします。

3.3. ファイルの入出力

続いて、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作成します。

- 1) 作成したソリューションへプロジェクトを追加します。

3.2 で作成したソリューション中のソリューションエクスプローラーにて、ソリューションを選択した状態で、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、[追加(D)] > [新しいプロジェクト(N)...] を選択します。



- 2) 使用するテンプレートを選択します。

フィルター画面が表示されるので、言語に “COBOL”、プラットフォームに “Windows”、プロジェクト タイプに “ネイティブ” を選択し、一覧から「コンソールアプリケーション」を選んで、[次へ(N)] ボタンをクリックします。



プロジェクト名に “LoadCSVFile” を入力し、[作成(C)] ボタンをクリックします。

新しいプロジェクトを構成します

コンソール アプリケーション COBOL Windows ネーティブ コンソール

プロジェクト名

LoadCSVFile

場所

C:\Users*tarot*source\repos\TutorialSol

戻る(B)

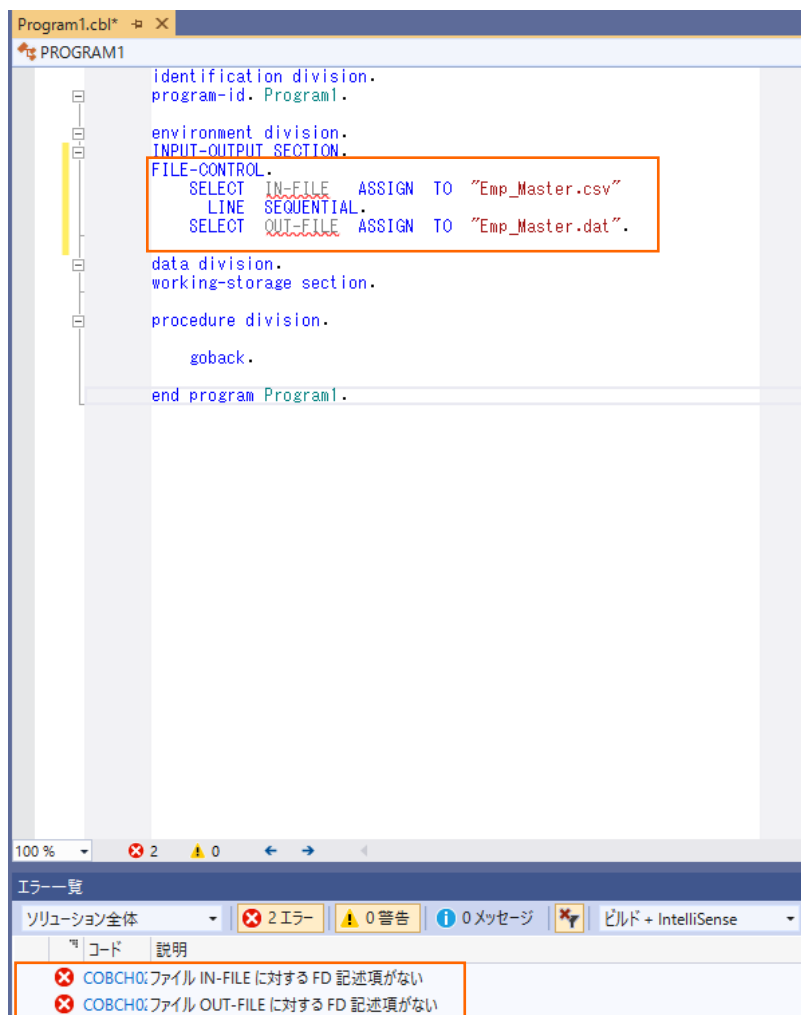
作成(C)

3) コードエディターで COBOL ソースコードを入力します。

LoadCSVFile プロジェクトの作成が成功すると、COBOL 専用のコードエディターが起動します。エディター画面にコンソールアプリケーションのひな形が表示されるので、環境部 (environment division)、データ部 (data division)、手続き部 (procedure division) を書き換えます。

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".
```

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.
  
```

```

Program1.cbl  X
PROGRAM1
identification division.
program-id. Program1.

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.
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.

    goback.

end program Program1.

```

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.

```

```

Program1.cbl*  X
PROGRAM1
05 OUT-DIV PIC N(5).
05 OUT-EMPDATE PIC 9(8).
05 FILLER PIC X.

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 Program1.
  
```

4) COBOL アプリケーションをビルドします。

メニューより、[ビルド(B)] > [LoadCSVFile のビルド(U)] を選択します。



出力ウィンドウにビルド結果が表示されますので、ビルドが正常終了したことを確認します。

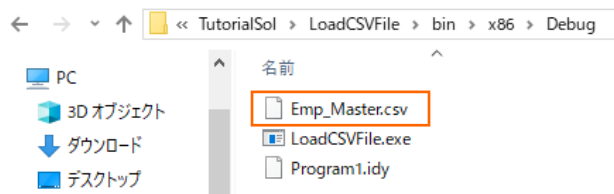
```

出力
出力元(S): ビルド
1>----- ビルド開始: プロジェクト: LoadCSVFile, 構成: Debug x86 -----
1> * C:\Users\*source\repos*TutorialSol\LoadCSVFile\Program1.cbl のコンパイル中
1> * Generating C:\obj\x86\Debug\Program1
1> * Data: 1584 Code: 1832 Literals: 664
1> COBOL コンパイル: 1 個 正常終了または最新の状態 0 個 失敗。
1> LoadCSVFile -> C:\Users\*source\repos*TutorialSol\LoadCSVFile\bin\x86\Debug\LoadCSVFile.exe
===== ビルド: 正常終了または最新の状態 1、失敗 0、スキップ 0 =====
  
```

5) CSV ファイルを作成します。

デバッグフォルダ(<3.2 節の 3)「場所」で指定したフォルダ>¥TutorialSol¥LoadCSVFile¥bin¥x86¥debug) にメモ帳などを利用して以下の Emp_Master.csv ファイルを作成します。

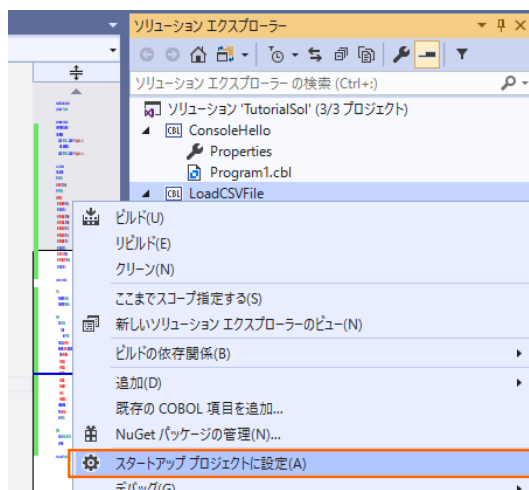
```
11111113,佐藤,隆,サウ,タカシ,M,営業部,19980401,0
22222226,鈴木,尚之,スズキ,ナオキ,M,技術部,19981015,0
33333339,田中,直美,タナカ,ナミ,F,総務部,19990401,0
44444442,山田,洋一,ヤマダ,ヨウイチ,M,営業部,20000701,0
55555555,伊藤,弘子,イトウ,ヒロコ,F,技術部,20010401,0
66666668,木村,貴弘,キムラ,タカヒロ,M,営業部,20021220,0
77777771,中村,慎司,ナカムラ,シンジ,M,技術部,20030401,0
88888884,橋本,悦子,ハシモト,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
```



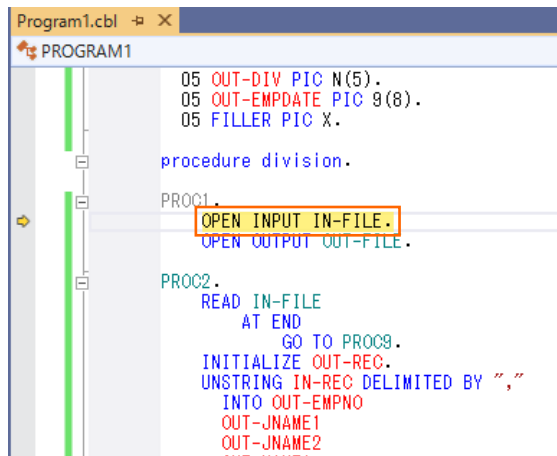
6) COBOL アプリケーションをデバッグ実行します。

ソリューションエクスプローラーにて 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、

[スタートアッププロジェクトに設定(A)] を選択します。



続いて、メニューより、[デバッグ(D)] > [ステップイン(I)] を選択して、デバッガーによる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 open 文で実行を中断します。



```

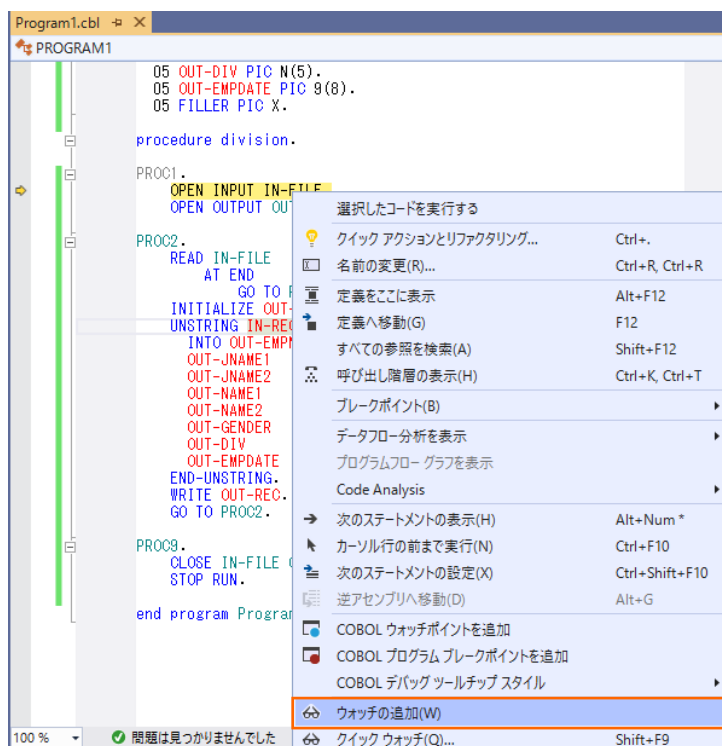
Program1.cbl
PROGRAM1
05 OUT-DIV PIC N(5).
05 OUT-EMPDATE PIC 9(8).
05 FILLER PIC X.

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
  
```

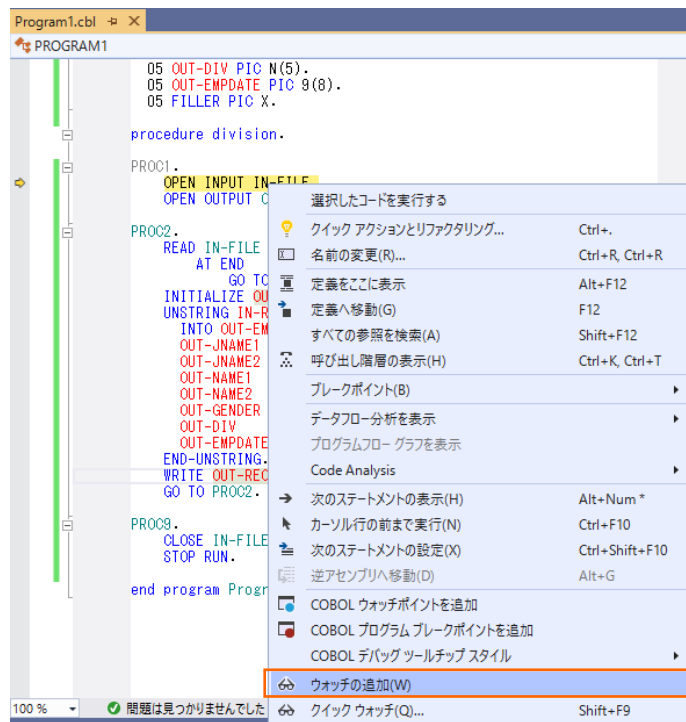
補足)

ステップインには、F11 キーがホットキーとして割り当てられているため、F11 キーを押すことでステップイン指示が可能です。後述する作業では F11 キーを使用してステップイン指示を実施します。

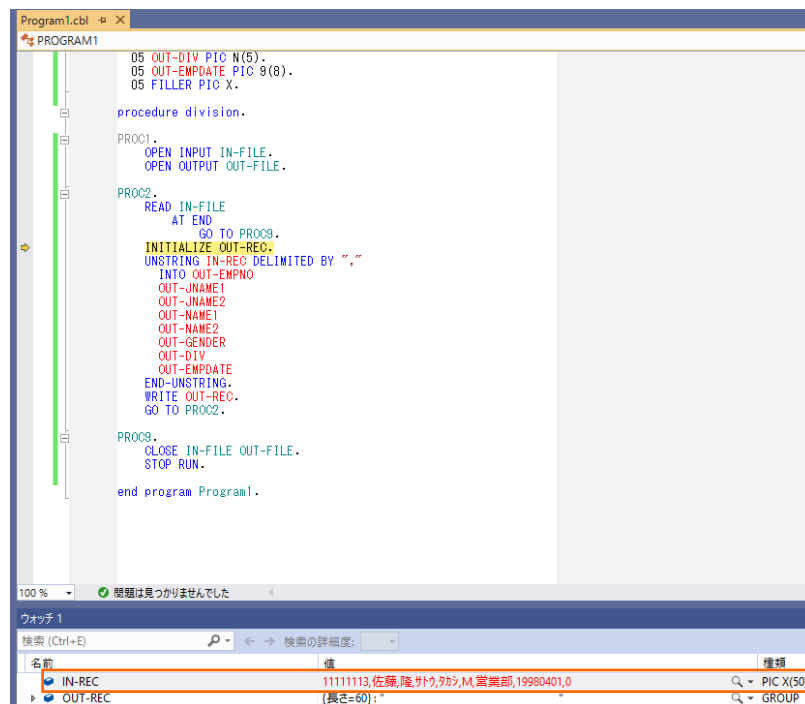
入力ファイルから読み込んだレコードの内容を確認するため、unstring 文の in-rec 上でマウスの右クリックにてコンテキストメニューを表示し、[ウォッチの追加(W)] を選択します。



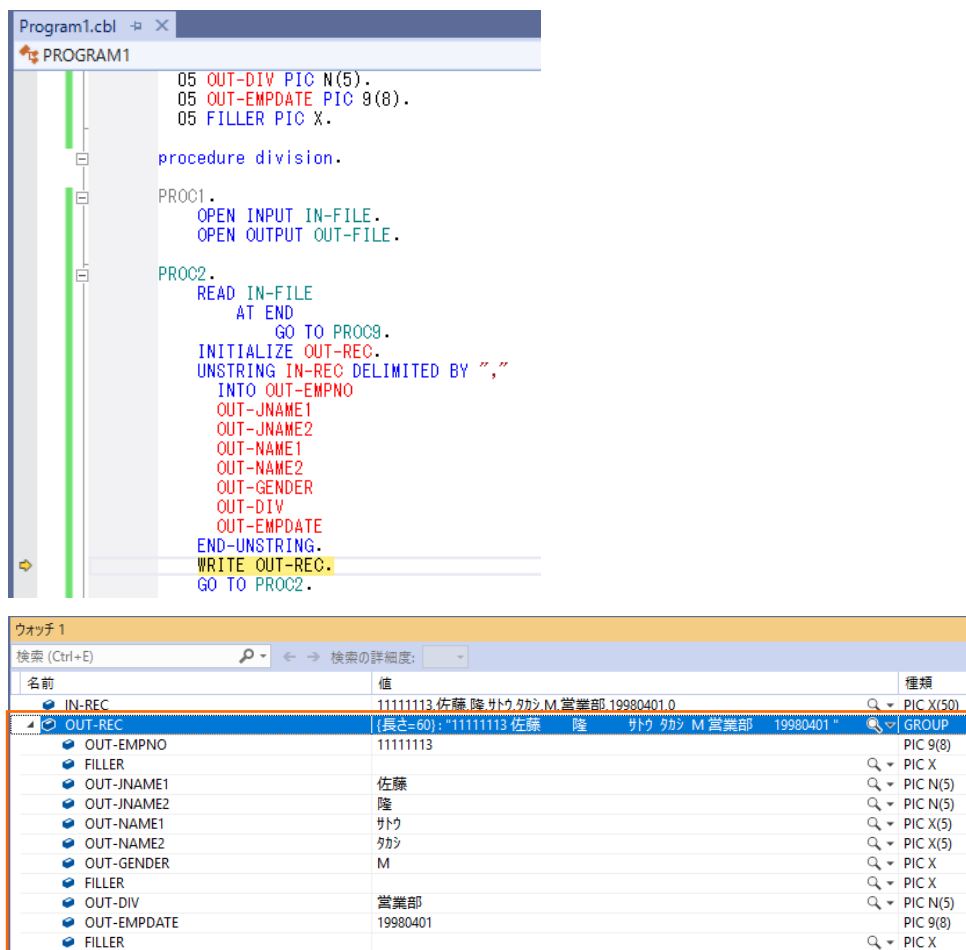
同様に出力ファイルに書き出すレコードの内容を確認するため、initialize 文の out-rec 上でもマウスの右クリックにてコンテキストメニューを表示し、[ウォッチの追加(W)] を選択します。



F11 キーを 3 回押すと、デバッガーは read 文実行後、処理を中断します。ウォッチ式の 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。



さらに F11 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。ウォッチ式の 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 Visual Studio 2019 interface with a COBOL program and a Watch window.

Program1.cbl

```

PROGRAM1
05 OUT-DIV PIC N(5).
05 OUT-EMPDATE PIC 9(8).
05 FILLER PIC X.

procedure division.

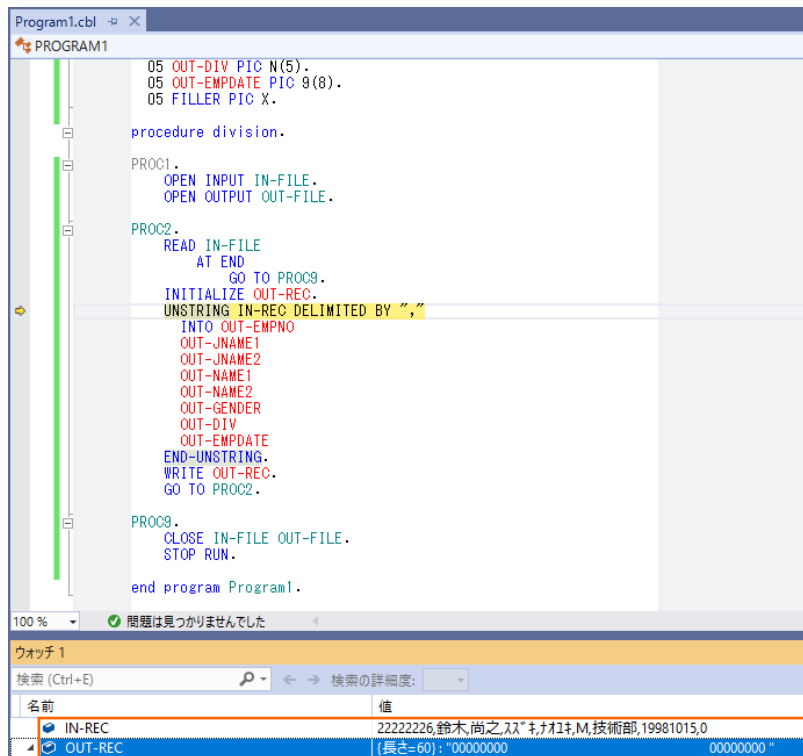
PROC1.
  OPEN INPUT IN-FILE.
  OPEN OUTPUT OUT-FILE.

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.
  
```

ウォッチ 1

名前	値	種類
IN-REC	11111113 佐藤 隆 サトウ タカシ M 営業部 19980401.0	PIC X(50)
OUT-REC	{長さ=60}: "11111113 佐藤 隆 サトウ タカシ M 営業部 19980401 "	GROUP
OUT-EMPNO	11111113	PIC 9(8)
FILLER		PIC X
OUT-JNAME1	佐藤	PIC N(5)
OUT-JNAME2	隆	PIC N(5)
OUT-NAME1	サトウ	PIC X(5)
OUT-NAME2	タカシ	PIC X(5)
OUT-GENDER	M	PIC X
FILLER		PIC X
OUT-DIV	営業部	PIC N(5)
OUT-EMPDATE	19980401	PIC 9(8)
FILLER		PIC X

さらに F11 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。



```

PROGRAM1
05 OUT-DIV PIC N(5).
05 OUT-EMPDATE PIC 9(8).
05 FILLER PIC X.

procedure division.

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

PROC2.
  READ IN-FILE
  AT END
    GO TO PROC3.
  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.

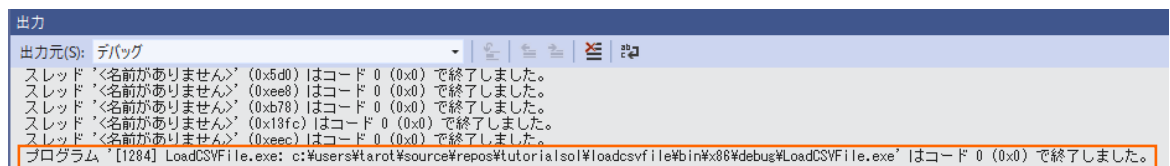
PROC3.
  CLOSE IN-FILE OUT-FILE.
  STOP RUN.

end program Program1.
  
```

ウォッチ 1

名前	値
IN-REC	22222226,鈴木,尚之,ス,キ,オ,1,1,M,技術部,19981015,0
OUT-REC	{長さ=60}: "00000000" 00000000 "

メニューより、[デバッグ(D)] > [続行(C)] を選択するか、CSV ファイルからすべてのレコードを読み込むまで F11 キーを押すと、デバッガーは終了します。



出力

出力元(S): デバッグ

```

スレッド '<名前がありません>' (0x5d0) はコード 0 (0x0) で終了しました。
スレッド '<名前がありません>' (0xee8) はコード 0 (0x0) で終了しました。
スレッド '<名前がありません>' (0xb78) はコード 0 (0x0) で終了しました。
スレッド '<名前がありません>' (0x18fc) はコード 0 (0x0) で終了しました。
スレッド '<名前がありません>' (0xeec) はコード 0 (0x0) で終了しました。
プログラム '[1284] LoadCSVFile.exe: c:\Users\tarot\source\repos\tutorial\src\loadcsvfile\bin\x86\debug\LoadCSVFile.exe' はコード 0 (0x0) で終了しました。
  
```

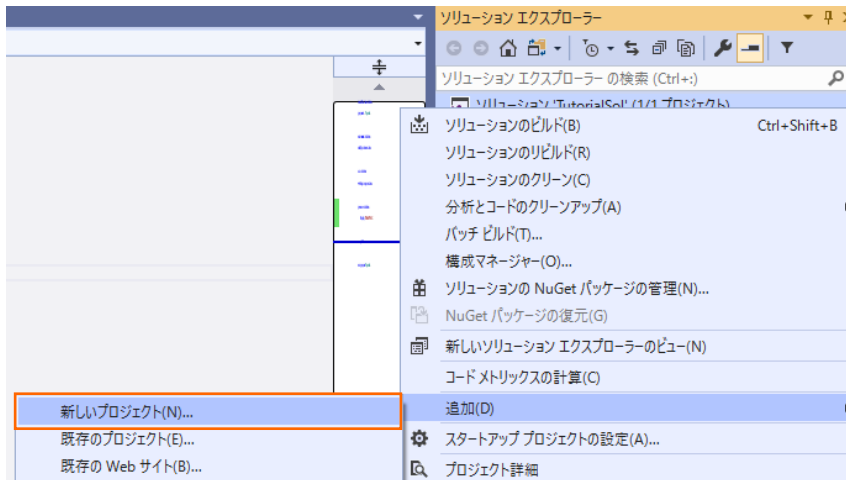
デバッグフォルダ(<3.2 節の 3)「場所」で指定したフォルダ> ¥TutorialSol¥LoadCVSFile¥bin¥x86¥debug)に Emp_Master.dat ファイルが作成されます。テキストエディタなどでファイルを開き、社員 9 名分のデータが表示されることを確認します。エディター上で 60 桁で折り返し設定を行うと、以下のように表示されます。

Emp_Master.dat														
	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	11111113	佐藤												19980401
	22222226	鈴木												19981015
	33333339	田中												19990401
	44444442	山田												20000701
	55555555	伊藤												20010401
	66666668	木村												20021220
	77777771	中村												20030401
	88888884	橋本												20040805
	99999997	三井												20050401

さきほど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。

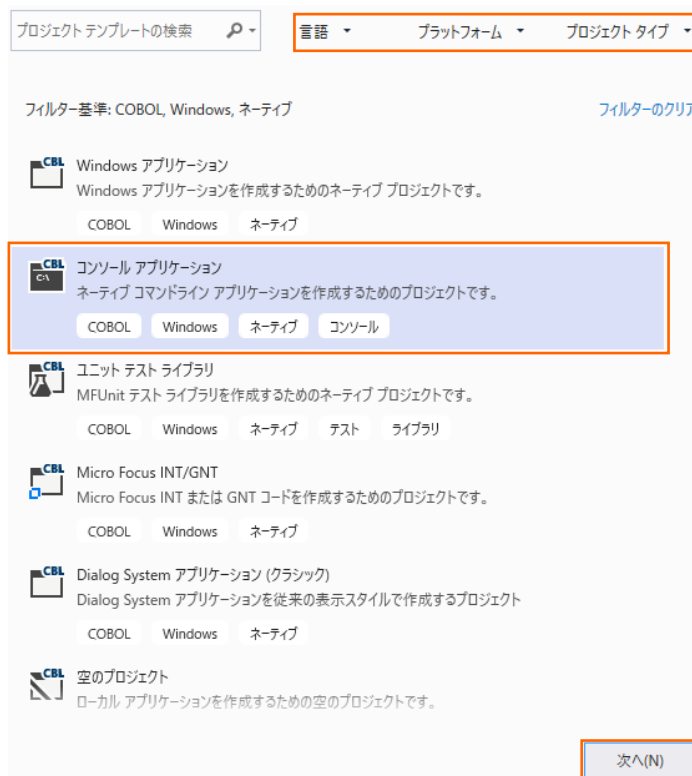
7) ソリューションに新規プロジェクトを追加します。

3.2 で作成したソリューション中のソリューションエクスプローラーにて、TutorialSol ソリューションを選択した状態で、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、[追加(D)] > [新しいプロジェクト(N)...] を選択します。



8) 使用するテンプレートを選択します。

フィルター画面が表示されるので、言語に “COBOL”、プラットフォームに “Windows”、プロジェクト タイプに “ネイティブ” を選択し、一覧から「コンソールアプリケーション」を選んで、[次へ(N)] ボタンをクリックします。



- 9) プロジェクト名に “BATCHRPT” を入力し、[作成(C)] ボタンをクリックします。

新しいプロジェクトを構成します

コンソール アプリケーション COBOL Windows ネーティブ コンソール

プロジェクト名

BATCHRPT

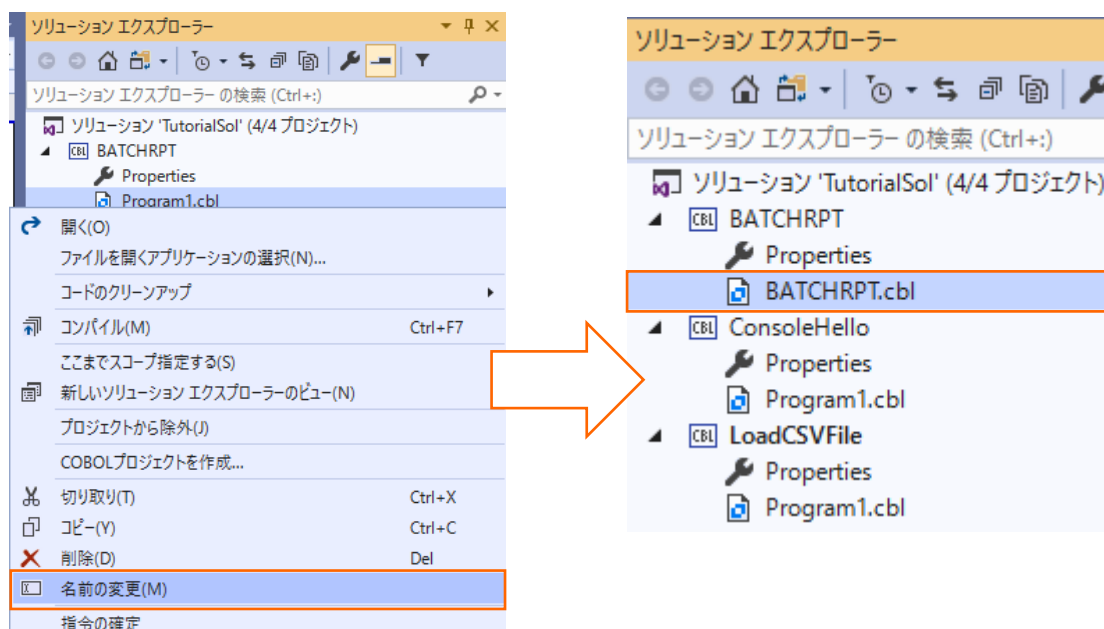
場所

C:\Users\Tarot\source\repos\TutorialSol

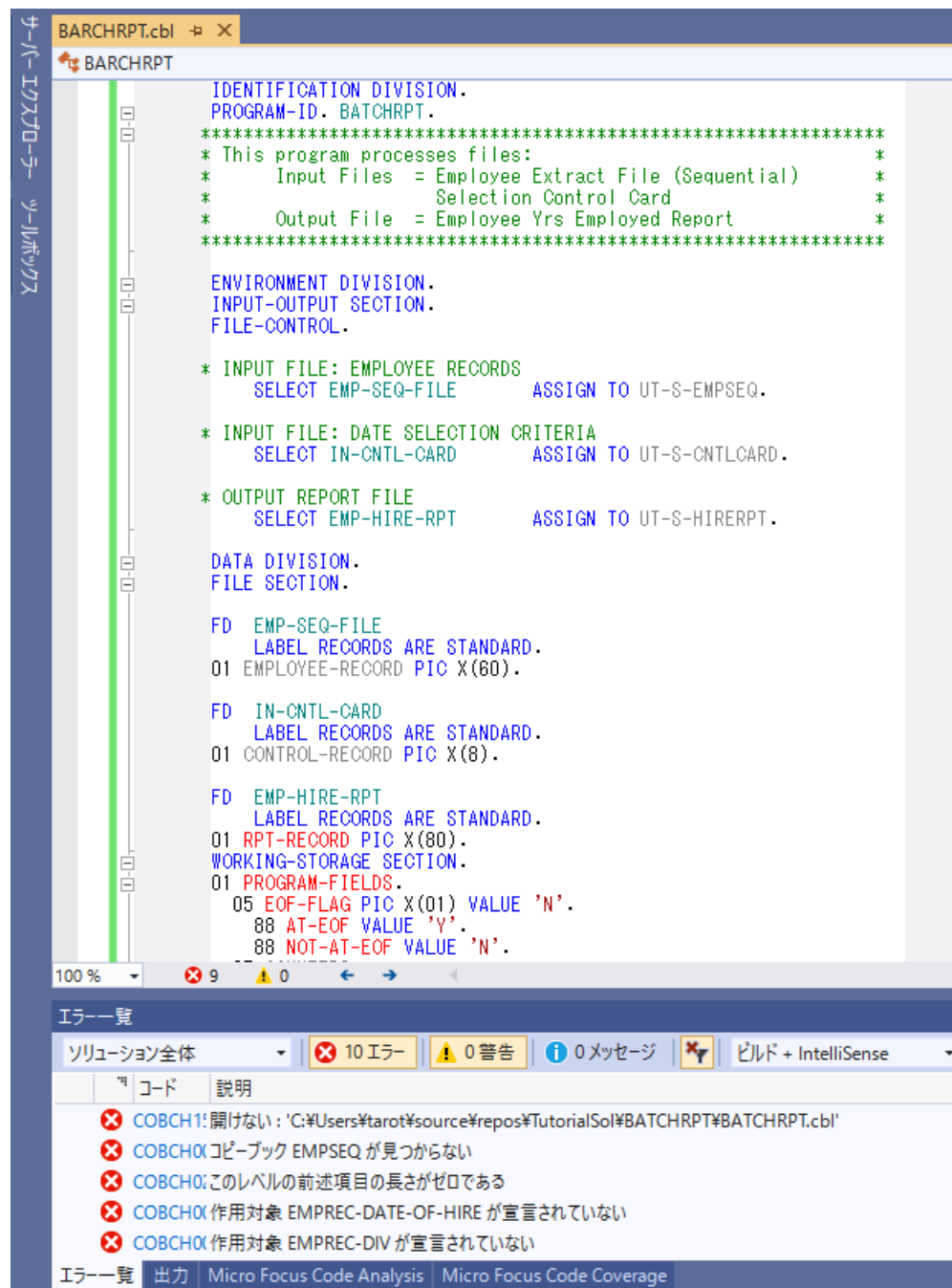
戻る(B) 作成(C)

- 10) コードエディターで COBOL ソースコードを入力します。

BATCHRPT プロジェクトの作成が成功すると、COBOL 専用のコードエディターが起動します。エディター画面にコンソールアプリケーションのひな形が表示されるので、ソリューションエクスプローラーで BATCHRPT プロジェクト配下のソースプログラム「Program1.cbl」を選択した上で、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、[名前の変更(M)] を選択します。その後、プログラム名を “BATCHRPT.cbl” に書き換えます。



サンプルプログラム BARCHRPT.cbl の内容でコードを上書きします。



The screenshot displays the Visual Studio 2019 interface with the COBOL source code for `BARCHRPT.cbl` open in the editor. The code is structured into several divisions and sections:

```

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(80).

FD IN-CNTL-CARD
  LABEL RECORDS ARE STANDARD.
01 CONTROL-RECORD PIC X(8).

FD EMP-HIRE-RPT
  LABEL RECORDS ARE STANDARD.
01 RPT-RECORD PIC X(80).
WORKING-STORAGE SECTION.
01 PROGRAM-FIELDS.
   05 EOF-FLAG PIC X(01) VALUE 'N'.
   88 AT-EOF VALUE 'Y'.
   88 NOT-AT-EOF VALUE 'N'.
  
```

Below the code editor, the **エラー一覧** (Error List) window is visible, showing 10 errors. The errors are related to missing copybooks and undeclared variables:

- COBCH1! 開けない: 'C:\Users\tarot\source\repos\TutorialSol\BATCHRPT\BATCHRPT.cbl'
- COBCH0X コピーブック EMPSEQ が見つからない
- COBCH0X このレベルの前述項目の長さがゼロである
- COBCH0X 作用対象 EMPREC-DATE-OF-HIRE が宣言されていない
- COBCH0X 作用対象 EMPREC-DIV が宣言されていない

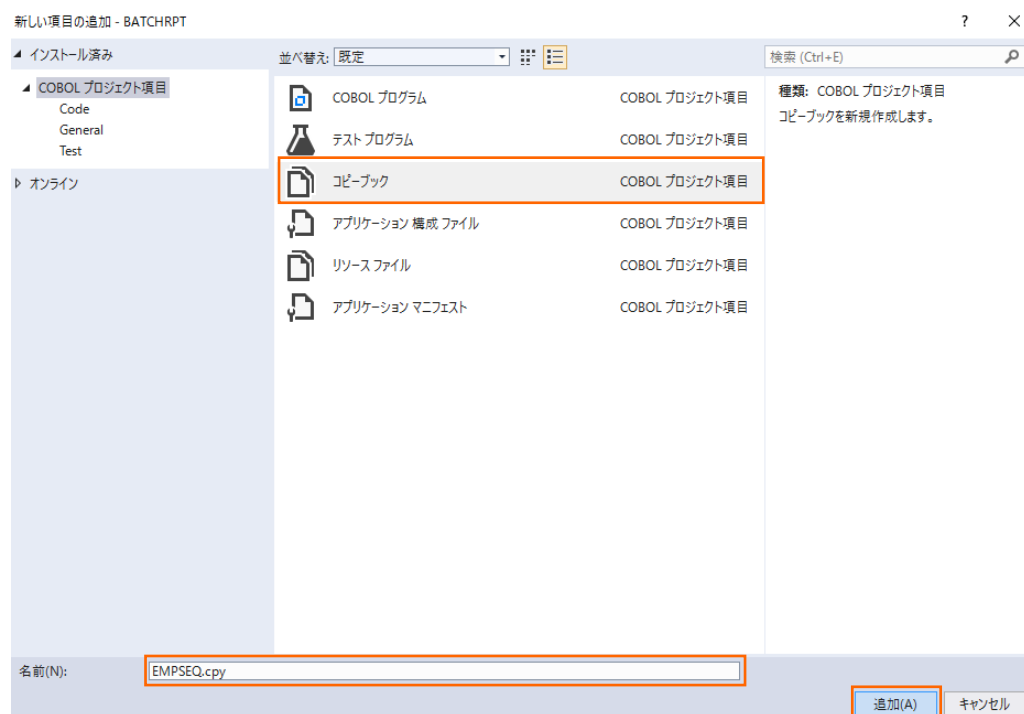
参照しているコピーブックが存在しないため、エラーが報告されますが、ここでは無視して構いません。

続いて、参照されるコピーブックを作成します。

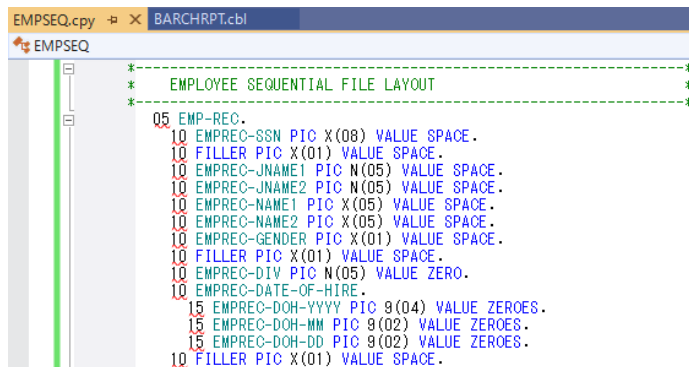
TutorialSol ソリューション配下の BARCHRPT プロジェクト名を選択した上で、マウスの右クリックにてコンテキストメニューを表示し、[追加(D)] > [新しい項目(W)] を選択します。



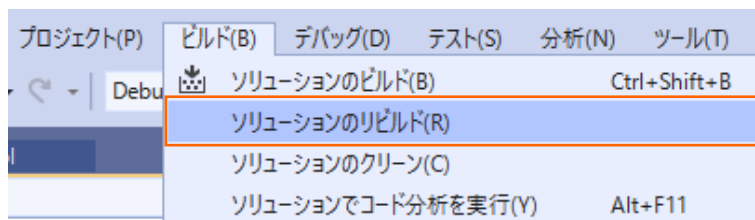
「コピーブック」を選択し、名前に “EMPSEQ.cpy” を入力し、[追加(A)] ボタンをクリックします。



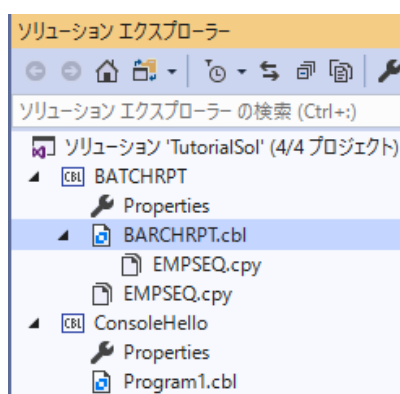
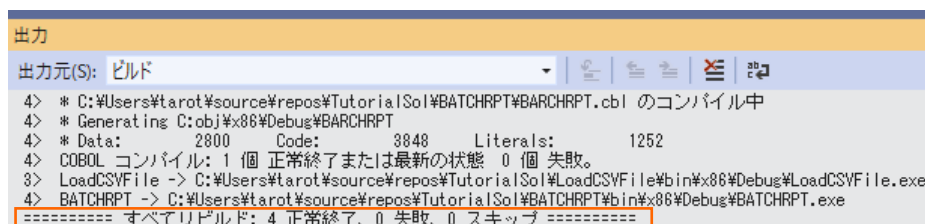
作成された EMPSEQ.cpy の中身を、サンプルプログラム EMPSEQ.cpy で上書きします。



メニューより、[ビルド(B)] > [ソリューションのリビルド(R)] を選択します。



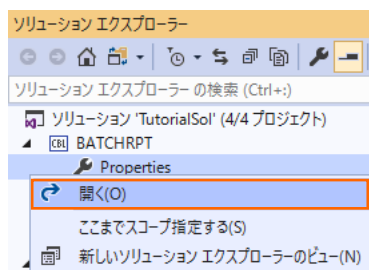
さきほどまでのエラーが全て解消されていることを確認します。



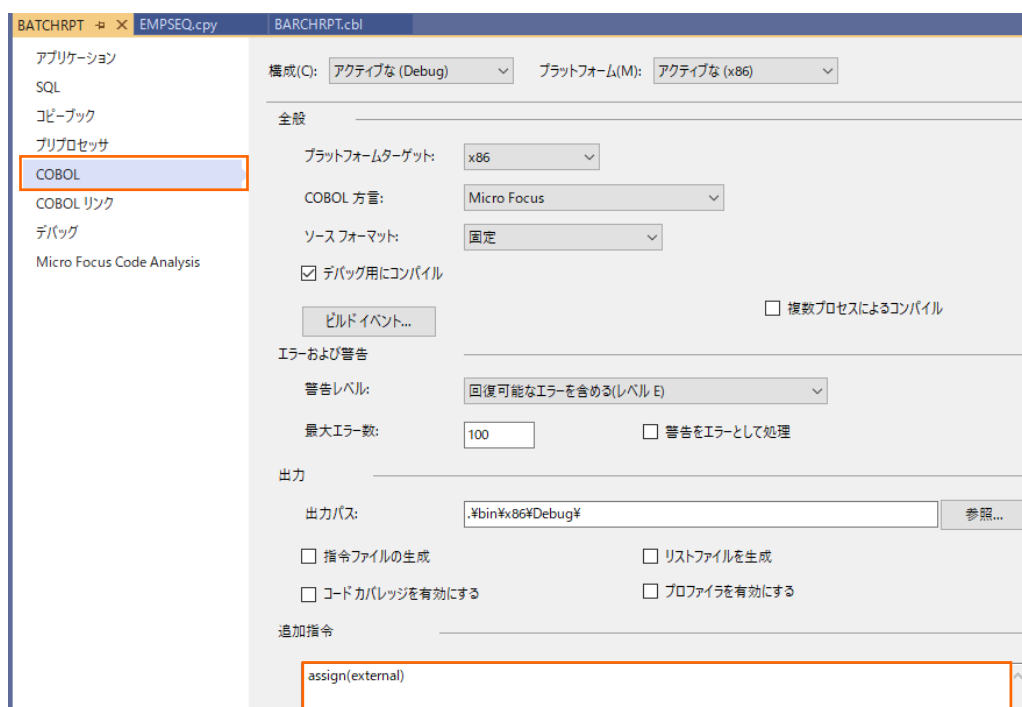
補足)
ソリューションエクスプローラーで、プログラムが参照しているコピーブックファイルを上記のようにツリー上で表示できます。ただし、この表示は Visual Studio の再起動を行い、ソリューションの再読み込みが必要です。

11) COBOL コンパイル指令を追加します。

ファイル名の割り当てを EXTERNAL (外部割り当て) に変更するため、ソリューションエクスプローラーにて BATCHPR1 プロジェクト配下の Properties を選択し、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、[開く(O)] を選択します。



COBOL タブを選択し 追加指令に “assign(external)” を入力し、プロパティファイルを保存します。



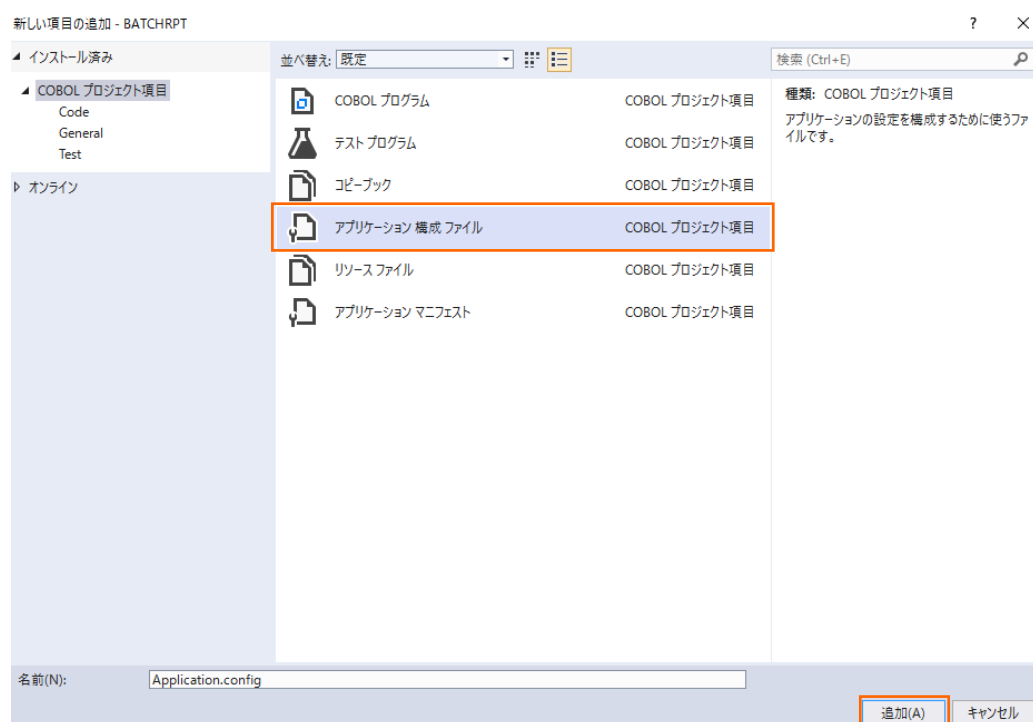
12) アプリケーション構成ファイルを作成します。

TutorialSol ソリューション配下の BARCHRPT プロジェクト名を選択し、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、[追加(D)] > [新しい項目(W)] を選択します。



13) 使用するテンプレートを選択します。

「アプリケーション構成ファイル」を選択し、[追加(A)] をクリックします。

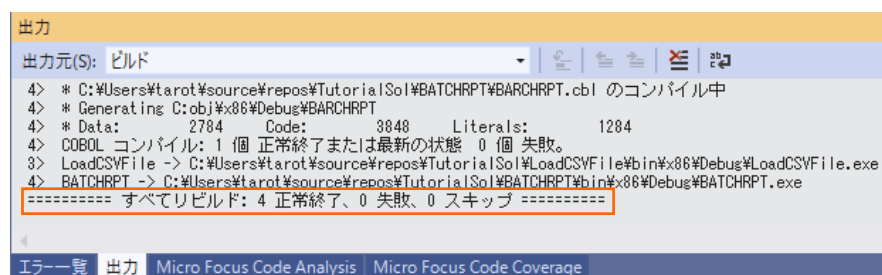


14) COBOL アプリケーションをビルドします。

メニューより、[ビルド(B)] > [ソリューションのリビルド(R)] を選択します。

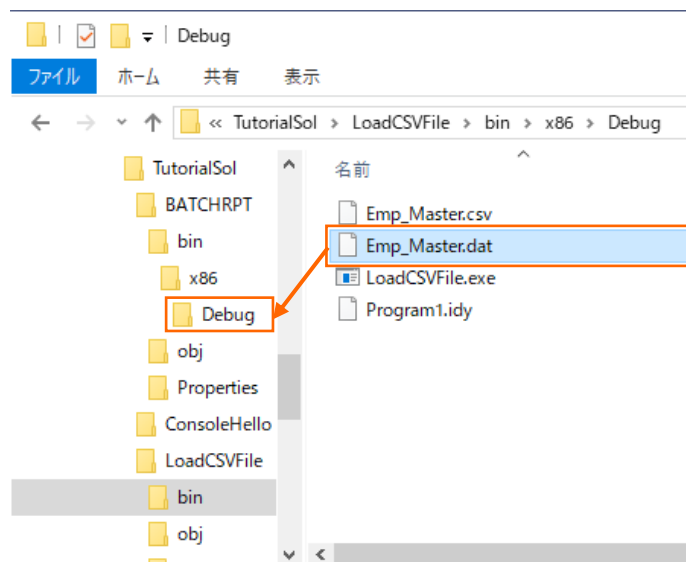


出力ウィンドウにビルド結果が表示されるので、すべてのビルドが正常終了したことを確認します。



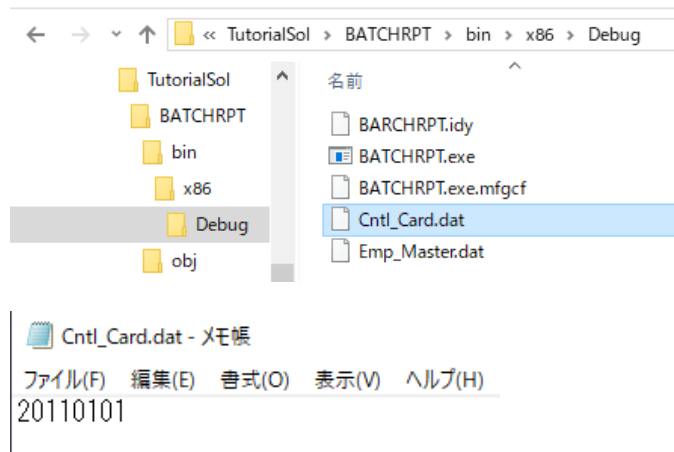
15) 入力ファイルをコピーします。

手順 6) の最後で作成された Emp_Master.dat ファイルをデバッグフォルダ (<3.2 節の 3) 「場所」で指定したフォルダ > ¥TutorialSol¥BATCHRPT¥bin¥x86¥debug) にコピーします。



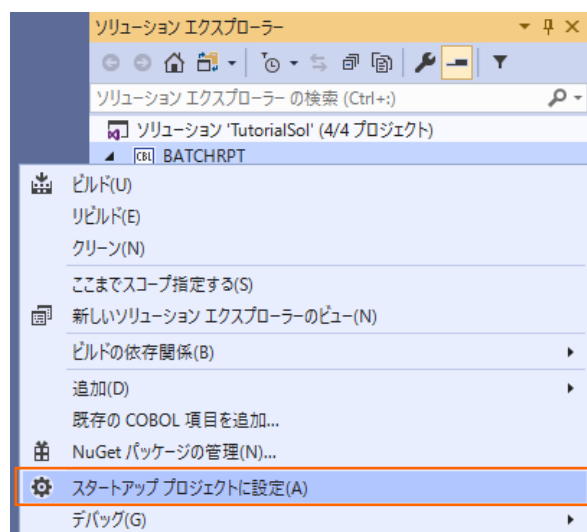
16) 制御ファイルを作成します。

以下の内容が記載された Cntl_Card.dat ファイルをデバッグフォルダ（＜3.2 節の 3）「場所」で指定したフォルダ＜br/>¥TutorialSol¥BATCHRPT¥bin¥x86¥debug）に作成します。

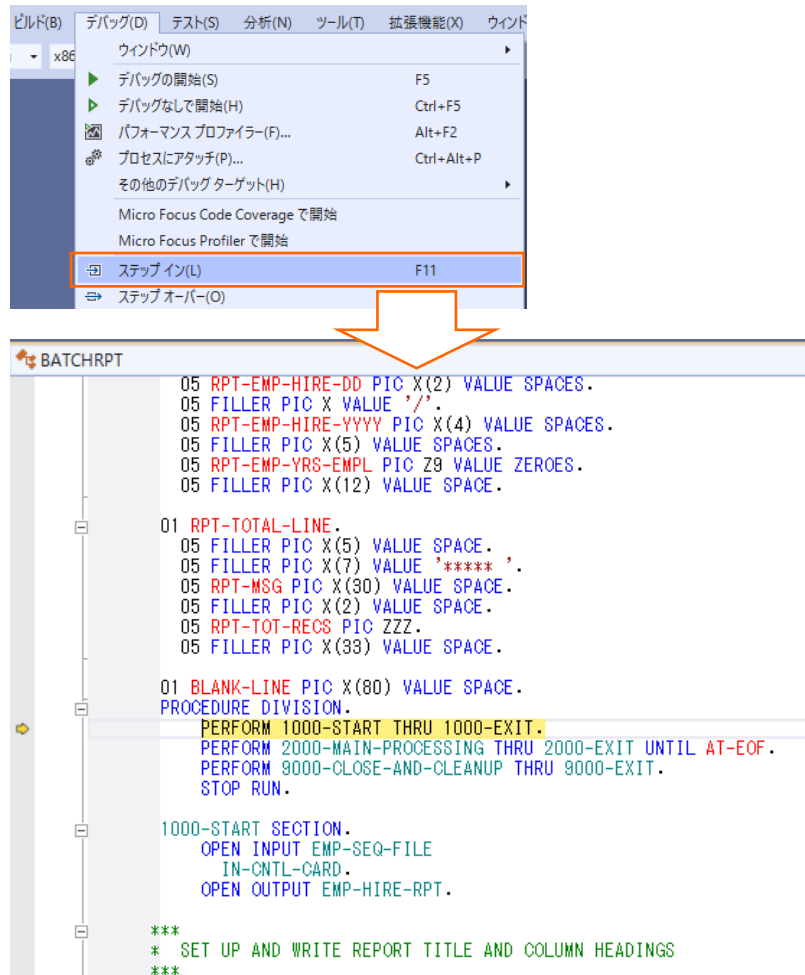


17) COBOL アプリケーションをデバッグ実行します。

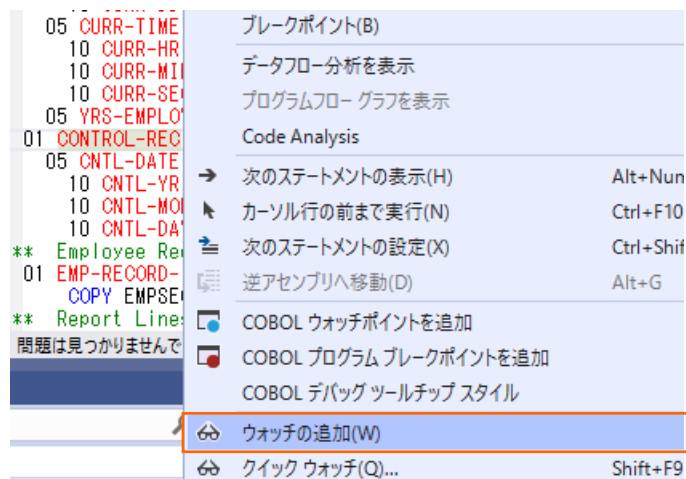
ソリューションエクスプローラーにて BATCHRPT プロジェクトを選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[スタートアッププロジェクトに設定(A)] を選択します。



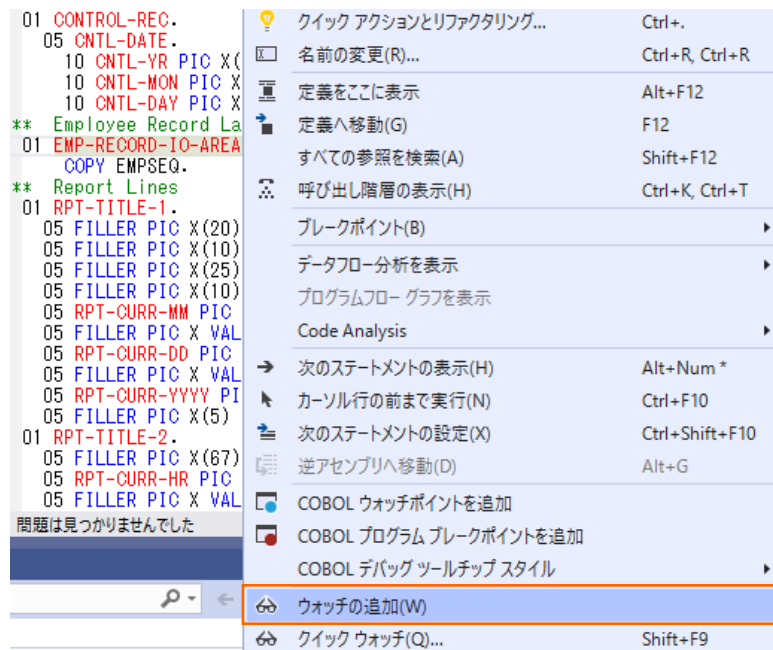
メニューより、[デバッグ(D)] > [ステップイン(I)] を選択するか F11 キーを押すと、コマンドプロンプト画面が開き、デバッガーがステップ実行を開始します。デバッガーは手続き部の最初の COBOL 文である PERFORM 文を実行する手前で処理を中断します。



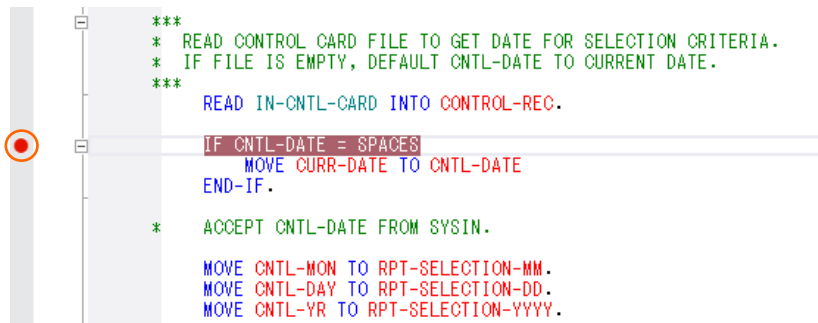
制御ファイルから読み込んだレコードの内容を確認するため、データ部の CONTROL-REC 上でマウスの右クリックにてコンテキストメニューを表示し、[ウォッチの追加(W)] を選択します。



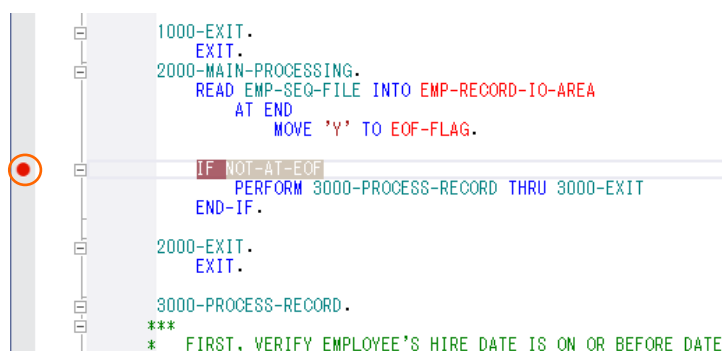
同様に入力ファイルから読み込んだレコードの内容を確認するため、データ部の EMP-RECORD-IO-AREA 上でマウスの右クリックにてコンテキストメニューを表示し、[ウォッチの追加(W)] を選択します。



手続き部 1000-START 節の READ 文に続く IF 文でエディター画面の左端をクリックし、ブレークポイントを設定します。

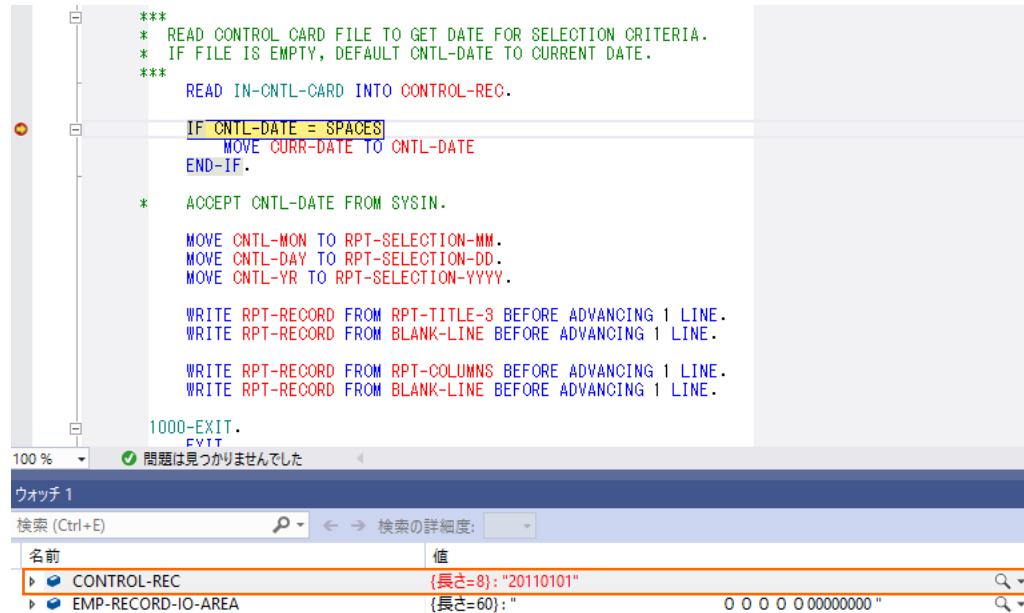


同様に手続き部 2000-MAIN-PROCESSING 段落の READ 文に続く IF 文でエディター画面の左端をクリックし、ブレークポイントを設定します。



メニューより、[デバッグ(D)] > [続行(C)] を選択するか F5 キーを押すと、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。

ウォッチ式の 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。



```

***
* 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.

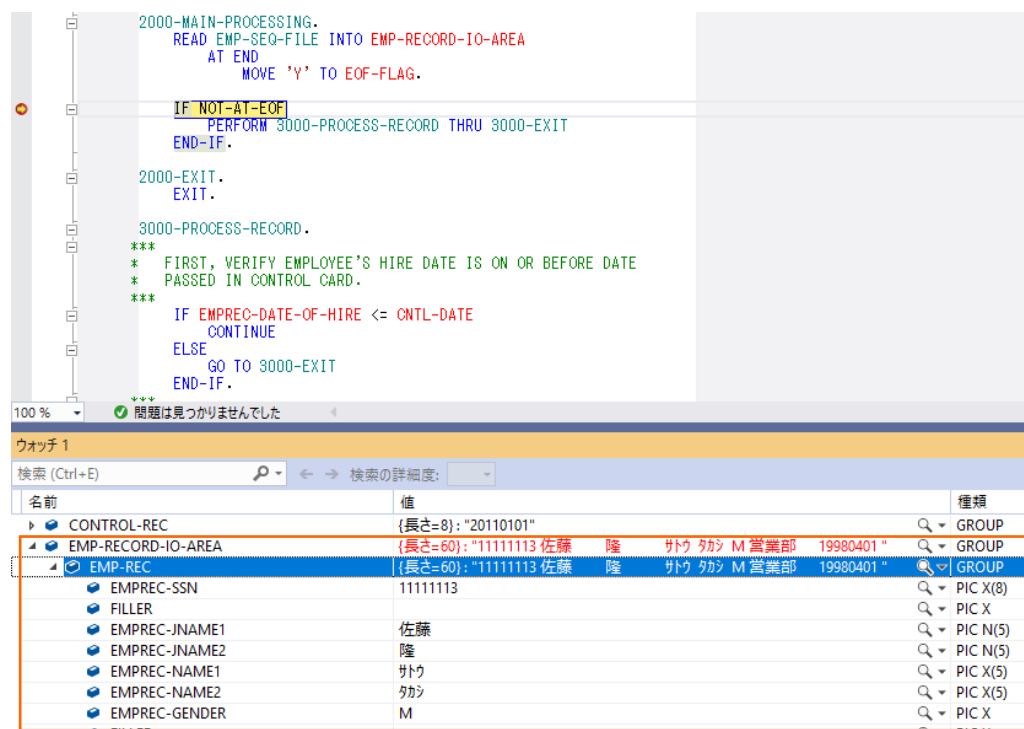
WRITE RPT-RECORD FROM RPT-COLUMNS BEFORE ADVANCING 1 LINE.
WRITE RPT-RECORD FROM BLANK-LINE BEFORE ADVANCING 1 LINE.

1000-EXIT.
EXIT.
  
```

名前	値
CONTROL-REC	{長さ=8}: "20110101"
EMP-RECORD-IO-AREA	{長さ=60}: " 0 0 0 0 00000000 "

再度、メニューより、[デバッグ(D)] > [続行(C)] を選択するか F5 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。

ウォッチしている 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。



```

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

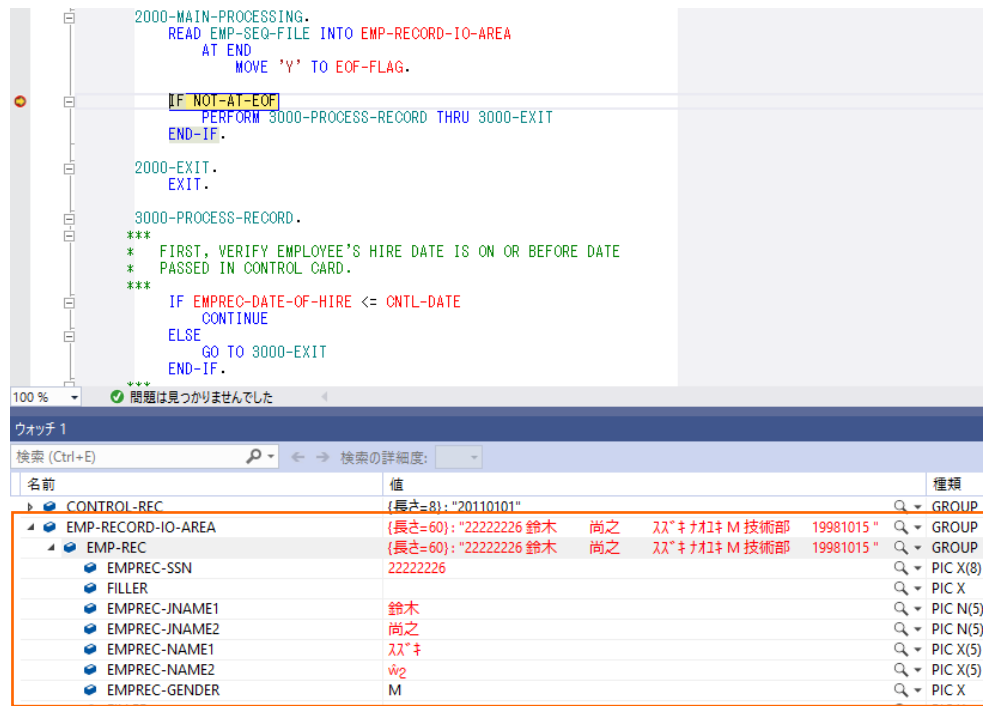
  IF NOT-AT-EOF
    PERFORM 3000-PROCESS-RECORD THRU 3000-EXIT
  END-IF.

2000-EXIT.
EXIT.

3000-PROCESS-RECORD.
***
* FIRST, VERIFY EMPLOYEE'S HIRE DATE IS ON OR BEFORE DATE
* PASSED IN CONTROL CARD.
***
IF EMPREC-DATE-OF-HIRE <= CNTL-DATE
  CONTINUE
ELSE
  GO TO 3000-EXIT
END-IF.
  
```

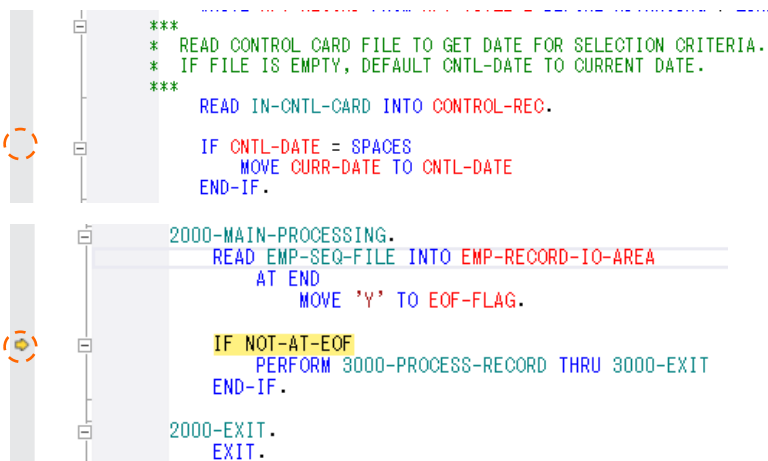
名前	値	種類
CONTROL-REC	{長さ=8}: "20110101"	GROUP
EMP-RECORD-IO-AREA	{長さ=60}: "11111113 佐藤 隆 サトウ タカシ M 営業部 19980401 "	GROUP
EMP-REC	{長さ=60}: "11111113 佐藤 隆 サトウ タカシ M 営業部 19980401 "	GROUP
EMPREC-SSN	11111113	PIC X(8)
FILLER		PIC X
EMPREC-JNAME1	佐藤	PIC N(5)
EMPREC-JNAME2	隆	PIC N(5)
EMPREC-NAME1	サトウ	PIC X(5)
EMPREC-NAME2	タカシ	PIC X(5)
EMPREC-GENDER	M	PIC X

再度、メニューより、[デバッグ(D)] > [続行(C)] を選択するか F5 キーを押すと、同じブレークポイント位置で停止しますが、2 番目のデータが表示されます。



名前	値	種類
CONTROL-REC	{長さ=8}: "20110101"	GROUP
EMP-RECORD-IO-AREA	{長さ=60}: "22222226 鈴木 尚之 ス*ナオキ M 技術部 19981015 "	GROUP
EMP-REC	{長さ=60}: "22222226 鈴木 尚之 ス*ナオキ M 技術部 19981015 "	GROUP
EMPREC-SSN	22222226	PIC X(8)
FILLER		PIC X
EMPREC-JNAME1	鈴木	PIC N(5)
EMPREC-JNAME2	尚之	PIC N(5)
EMPREC-NAME1	ス*ナ	PIC X(5)
EMPREC-NAME2	オキ	PIC X(5)
EMPREC-GENDER	M	PIC X

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



```

***
* 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.

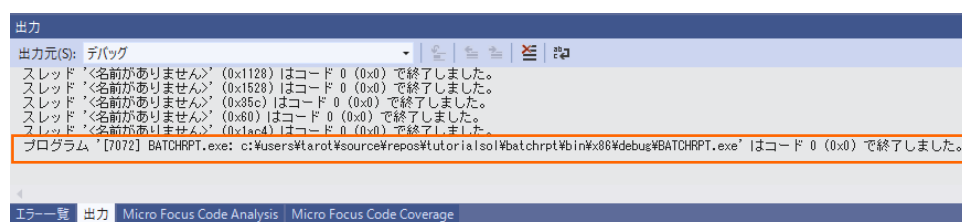
IF CNTL-DATE = SPACES
  MOVE CURR-DATE TO CNTL-DATE
END-IF.

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.
  
```

メニューより、[デバッグ(D)] > [続行(C)] を選択するか F5 キーを押して、プログラムを終了します。



出力

出力元(S): デバッグ

スレッド '<名前がありません>' (0x1128) はコード 0 (0x0) で終了しました。

スレッド '<名前がありません>' (0x1528) はコード 0 (0x0) で終了しました。

スレッド '<名前がありません>' (0x35c) はコード 0 (0x0) で終了しました。

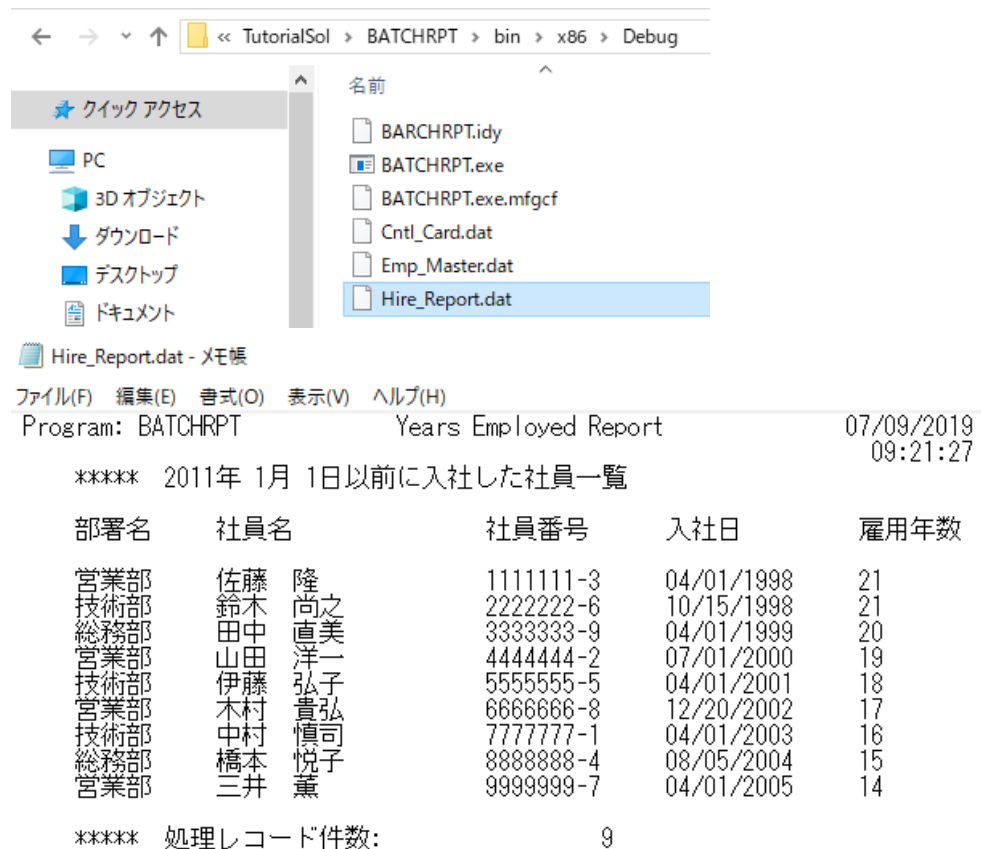
スレッド '<名前がありません>' (0x80) はコード 0 (0x0) で終了しました。

スレッド '<名前がありません>' (0x1ac4) はコード 0 (0x0) で終了しました。

プログラム '[7072] BATCHRPT.exe: c:\users\larot\source\repos\tutorial\batchrpt\bin\x86\debug\BATCHRPT.exe' はコード 0 (0x0) で終了しました。

エラー一覧 出力 Micro Focus Code Analysis Micro Focus Code Coverage

デバッグフォルダ（＜3.2 節の 3）「場所」で指定したフォルダ＞¥TutorialSol¥BATCHRPT¥bin¥x86¥debug）に Hire_Report.dat ファイルが作成されるので、メモ帳などのエディターでファイルを開き、社員 9 名分のデータが表示されることを確認します。



名前

- BARCHRPT.idy
- BATCHRPT.exe
- BARCHRPT.exe.mfgrcf
- Cntl_Card.dat
- Emp_Master.dat
- Hire_Report.dat

Hire_Report.dat - メモ帳

ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

Program: BATCHRPT Years Employed Report 07/09/2019 09:21:27

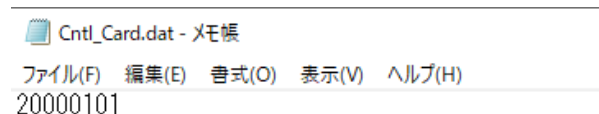
***** 2011年 1月 1日以前に入社した社員一覧

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
技術部	伊藤 弘子	5555555-5	04/01/2001	18
営業部	木村 貴弘	6666666-8	12/20/2002	17
技術部	中村 慎司	7777777-1	04/01/2003	16
総務部	橋本 悦子	8888888-4	08/05/2004	15
営業部	三井 薫	9999999-7	04/01/2005	14

***** 処理レコード件数: 9

続いて、基準日により結果が変更されることを確認します。

デバッグフォルダ（＜3.2 節の 3）「場所」で指定したフォルダ＞¥TutorialSol¥BATCHRPT¥bin¥x86¥debug）に Cntl_Card.dat ファイルの中身を 20000101 に更新します。

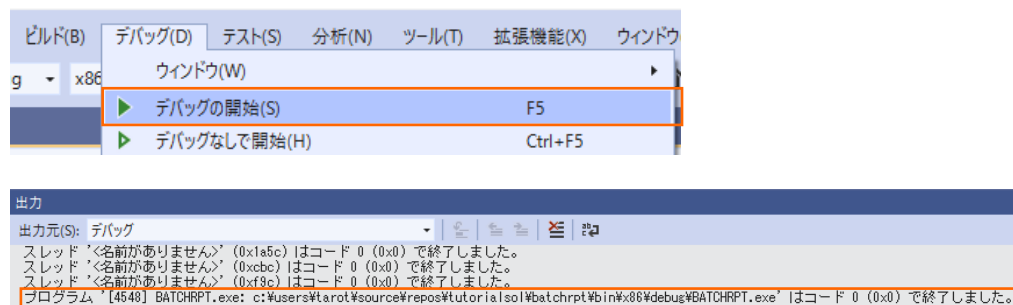


Cntl_Card.dat - メモ帳

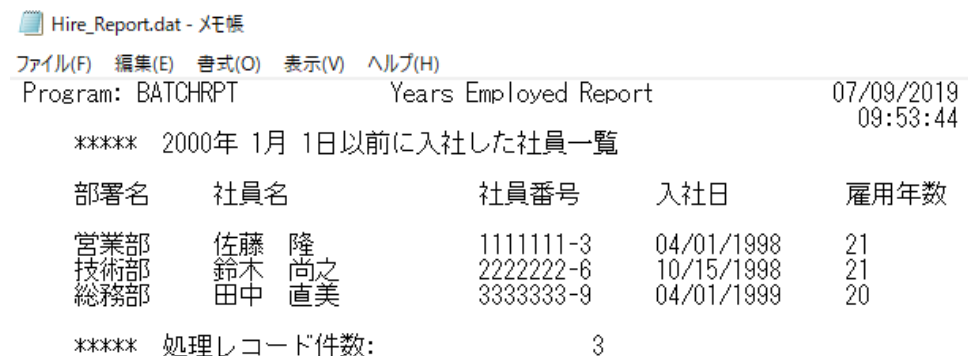
ファイル(F) 編集(E) 書式(O) 表示(V) ヘルプ(H)

20000101

メニューより、[デバッグ(D)] > [デバッグの開始(S)] を選択するか F5 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。



デバッグフォルダ (<3.2 節の 3) 「場所」で指定したフォルダ > ¥TutorialSol¥BATCHRPT¥bin¥x86¥debug) に Hire_Report.dat ファイルには、2000 年 1 月 1 日以前に入社した社員 3 名分のデータだけが表示されることを確認します。



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。