Micro Focus Visual COBOL for Visual Studio 2017

自習書

はじめに

Micro Focus Visual COBOL for Visual Studio 2017 は、Microsoft の最新 Windows 開発環境である Visual Studio 2017 の強力な統合開発環境(IDE)上で COBOL アプリケーションプログラム開発を可能とする COBOL 開発環境製品です。 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 プログラミング経験のない C#などの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も最適な製品です。

本書は、Micro Focus Visual COBOL for Visual Studio 2017 を学ぶための自習書です。 本書の読者は、プログラミングの基礎知識をもち、かつ Windows の基本操作を理解しているものとします。 なお、本書に沿って製品を実際に操作しながら学習するためには、以下の製品が必要です。

Micro Focus Visual COBOL 3.0J for Visual Studio 2017

また、本書に掲載している画面イメージは Windows 10 Pro 64 bit 版でキャプチャしています。他の Windows OS では多少異なる場合がありますが、ご了承ください。

Visual COBOL は Microsoft が提供する Visual Studio のバージョン固有の機能に関連するものを除いて各 Visual Studio 版で共通機能を提供しています。そのため、本書で紹介する内容は Visual Studio 2012 版、Visual Studio 2013 版、Visual Studio 2015 版のいずれでも同様にお試しいただくことができます。

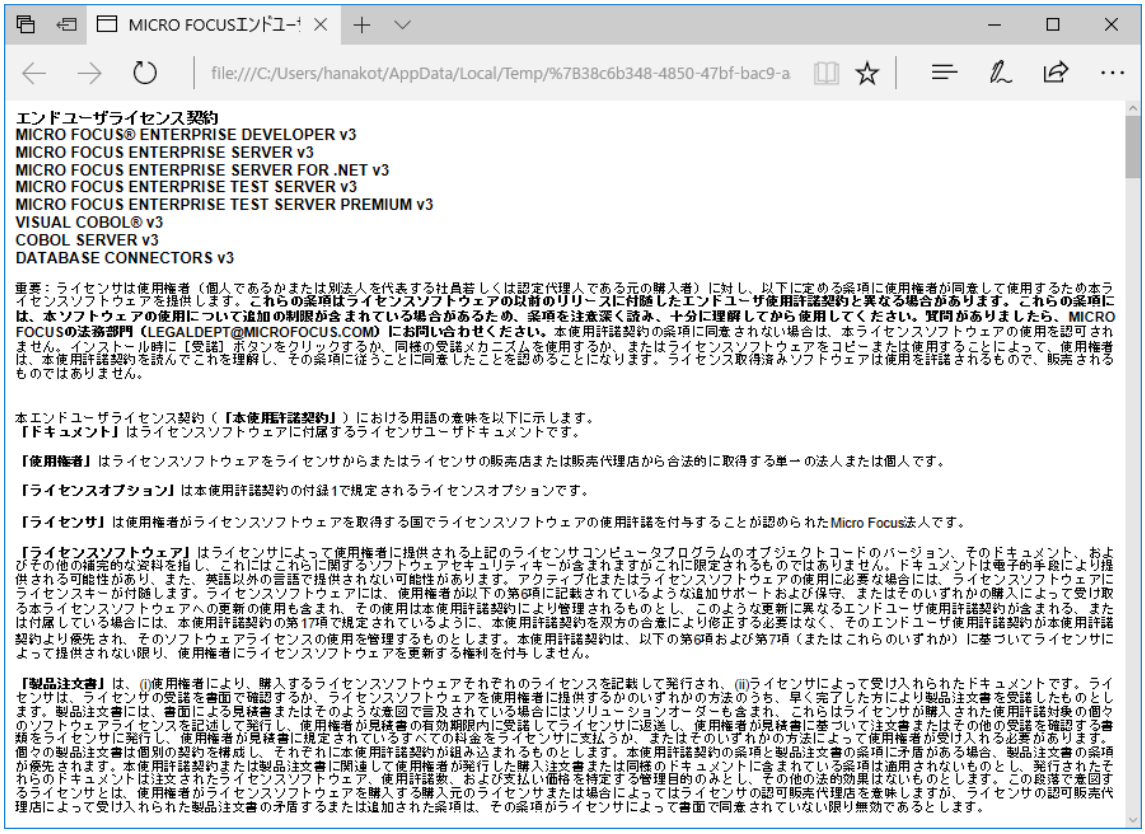
第1章　　自習環境の準備

Micro Focus Visual COBOL for Visual Studio 2017 は、COBOL プログラミングの IDE として Microsoft Visual Studio 2017 の IDE を利用します。 自習環境用に、Microsoft Visual Studio 2017 (Professional / Enterprise / Community Edition のいずれか)をセットアップ済みの PC を準備してください。

1　ダウンロードした **vcvs2017_30.exe** をダブルクリックします。

2　表示されるセットアップ画面で **エンドユーザ使用許諾契約書** をクリックします。

3　使用許諾契約書の内容を確認します。

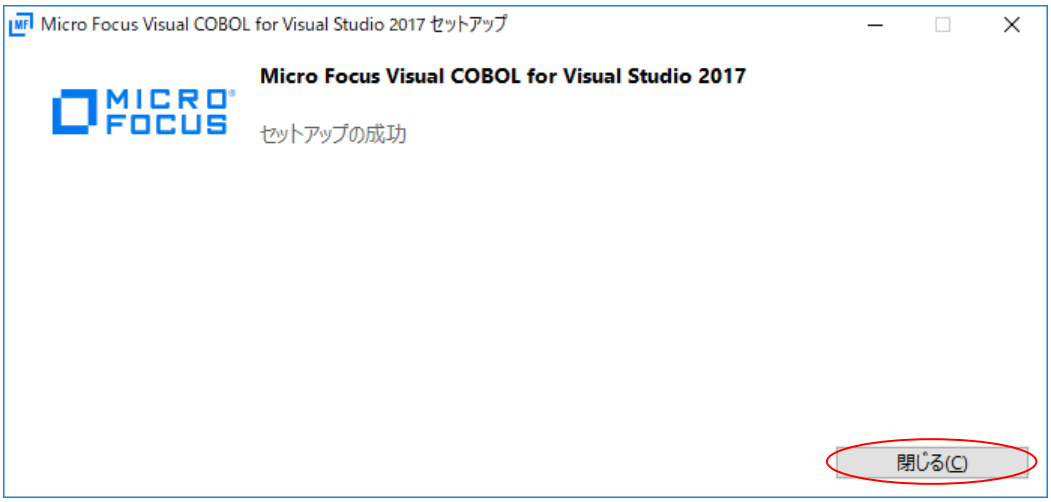


4　インストールを開始します。

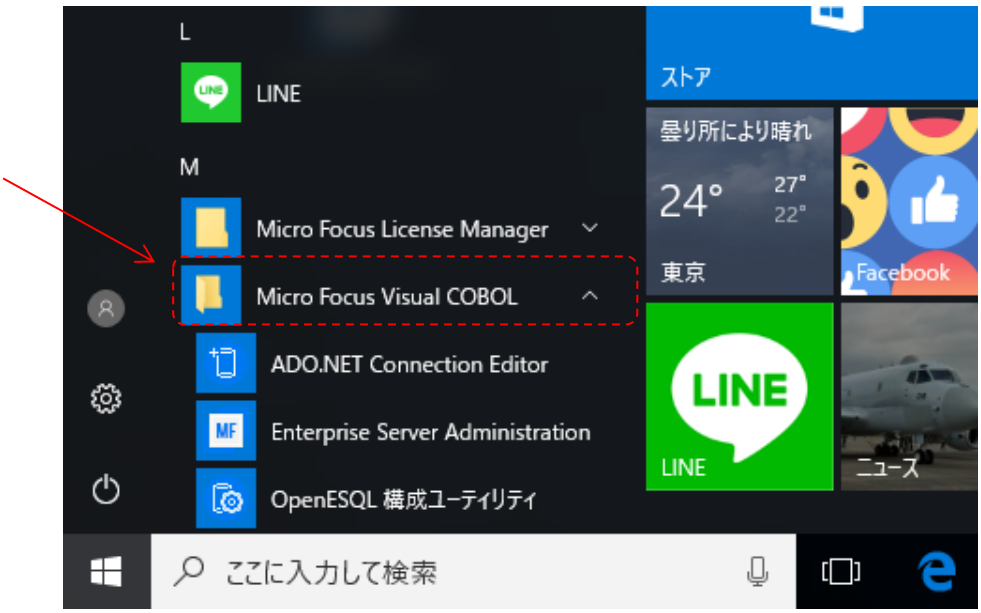
問題がなければ、**同意します(A)** にチェックを入れ［**インストール(I)**］ボタンを押下してインストールを開始します。

5　セットアップを終了します。

[閉じる(C)] ボタンを押下します。



以上で、自習環境の準備は終了しました。 Windows のスタートメニューに Visual COBOL が登録されていることを確認してください。

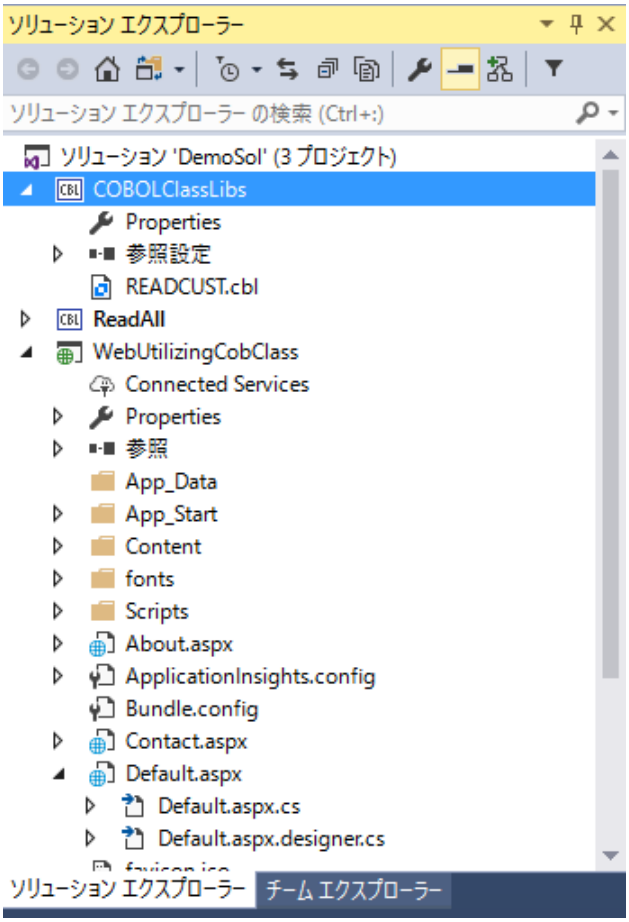


第2章　Visual Studio 2017 IDE に慣れよう

Microsoft Visual Studio 2017 の IDE を初めて利用する COBOL プログラマのために、概要を簡単に説明します。 既に Microsoft Visual Studio 2017 を習熟済みの方は、本章を読み飛ばしてください。

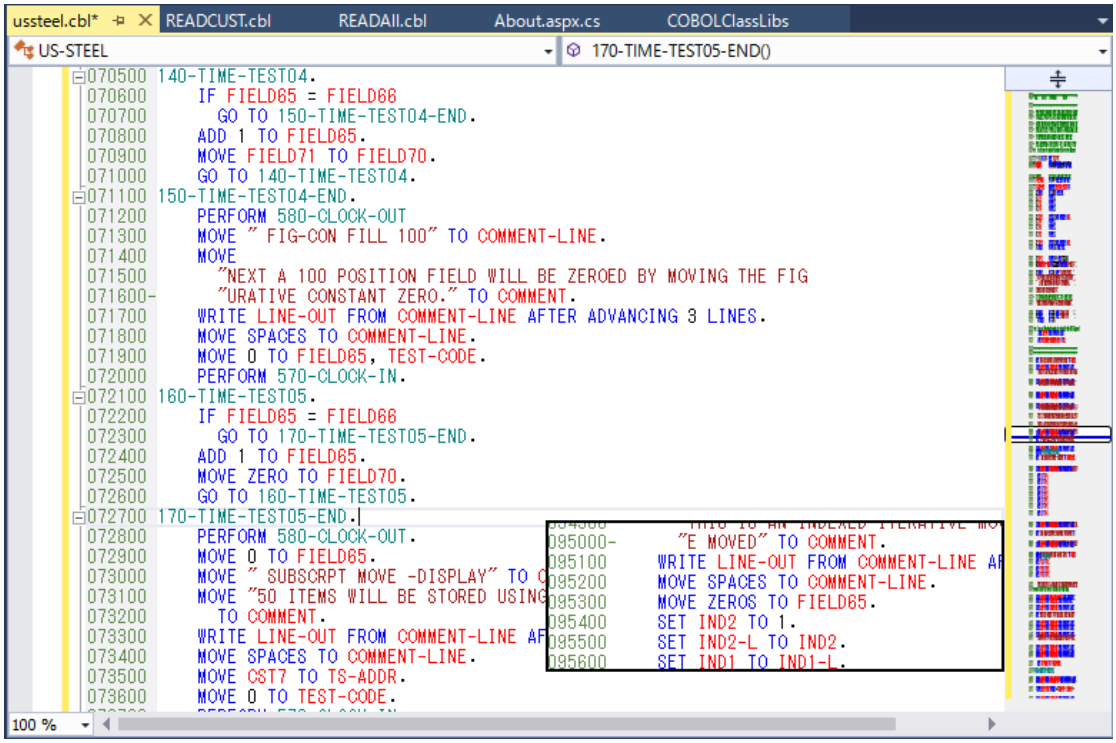
Microsoft Visual Studio 2017 の IDE は、メニューバー、ツールバー、左、下または右にドッキングまたは自動的に非表示になる各種ツールウィンドウ、エディター領域など、複数の要素で構成されます。 IDE 内の要素の配置は、適用した設定とその後に加えたカスタマイズ内容によって異なります。

Visual Studio 2017 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、フォルダー、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。

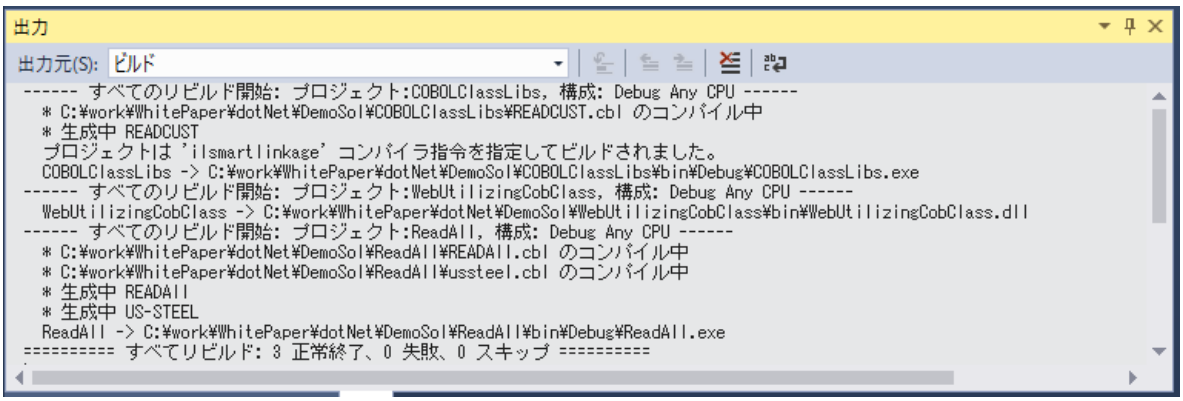


Visual Studio 2017 のソースコードエディターには、COBOL 予約語とデータ名や手続

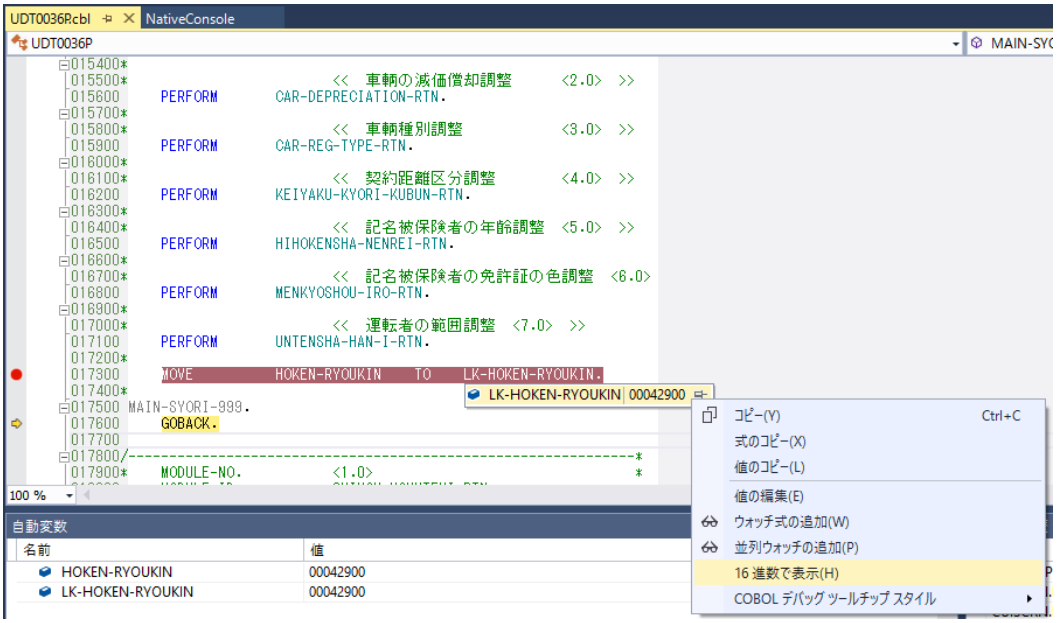
き名などの利用者語を色分け表示したり、COBOL スニペットなど COBOL 言語固有の機能拡張が含まれます。ソースコードを入力するとバックグラウンドチェックを実行して、赤の波線でエラー箇所を強調表示します。 そのエラー箇所にマウスポインタを移動すればエラー内容を確認したり、定義への移動、他の参照検索などの操作が可能です。



Visual Studio 2017 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。

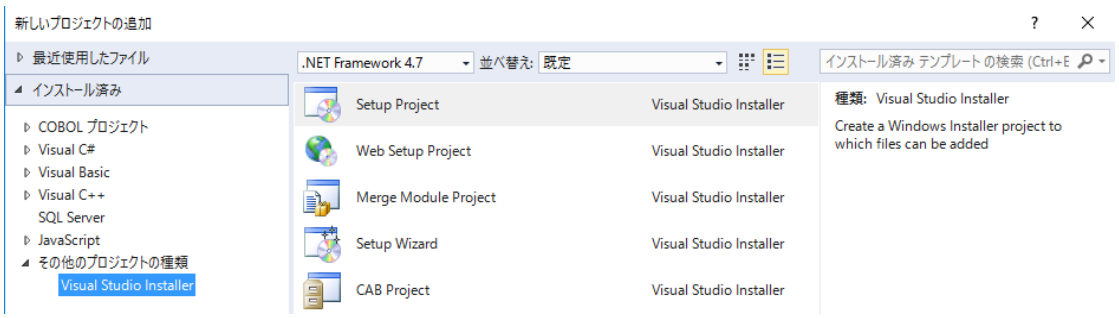


ビルドしたアプリケーションは、実行時の論理エラーやセマンティックエラーなどの問題を検出して修正するために、デバッガーを使用します。 Visual Studio 2017 のデバッガーは、コードをステップ実行したり様々な条件を設定したブレークポイントで実行を中断させ、変数ウィンドウやウォッチ式などのツールを使用してローカル変数やその他の関連データを調べることができます。



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 2017 では、Visual Studio の Marketplace より「Microsoft Visual Studio 2017 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 installer 作成用のプロジェクトをご利用できます。



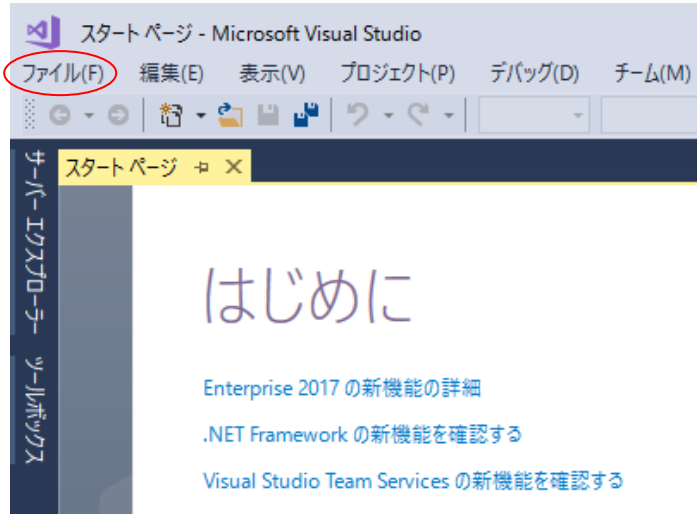
なお、本番環境には COBOL Server が事前にインストールされている必要があります。

第3章　はじめての Visual COBOL

それでは、Windows のコマンドプロンプト画面に「Hello World」を表示する COBOL アプリケーションを Visual COBOL for Visual Studio 2017 で作成します。

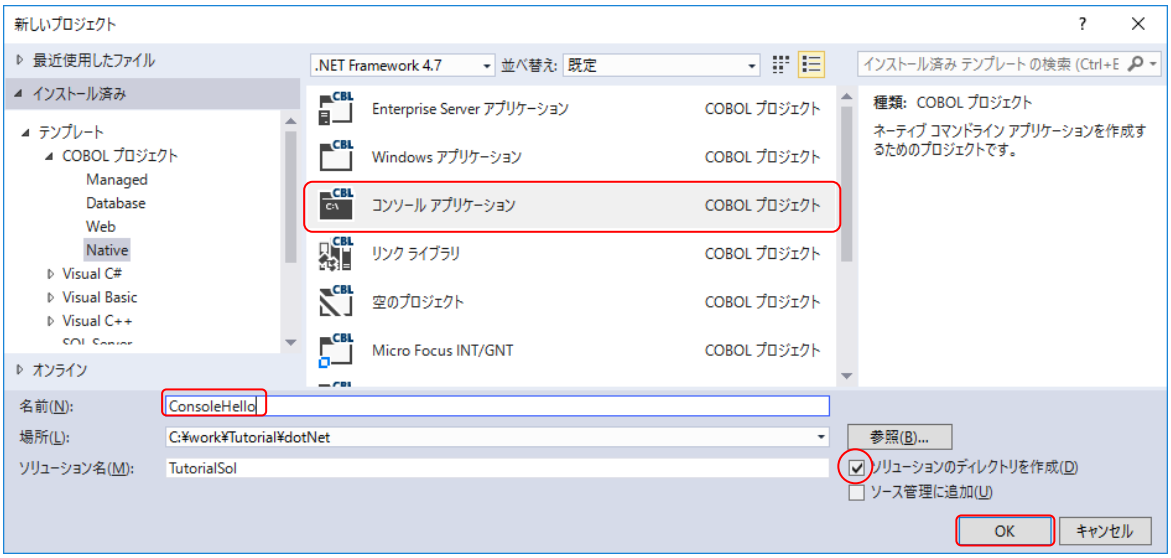
1　Visual COBOL for Visual Studio 2017 を起動します。

Windows のスタートメニューから、**Visual COBOL for Visual Studio 2017** をクリックします。 Microsoft Visual Studio 2017 のスタートページが表示されたら、**ファイル(F)**メニューから**新規作成(N)**、**プロジェクト(P)** を選択します。



2　使用するテンプレートを選択します。

インストールされたテンプレートの一覧から **COBOL プロジェクト**、**Native**、**コンソールアプリケーション**を選択します。 **ソリューションのディレクトリを作成(D)** がチェックされていることを確認し、名前(N)に **ConsoleHello** と入力し、[**ＯＫ**] ボタンを押下します。

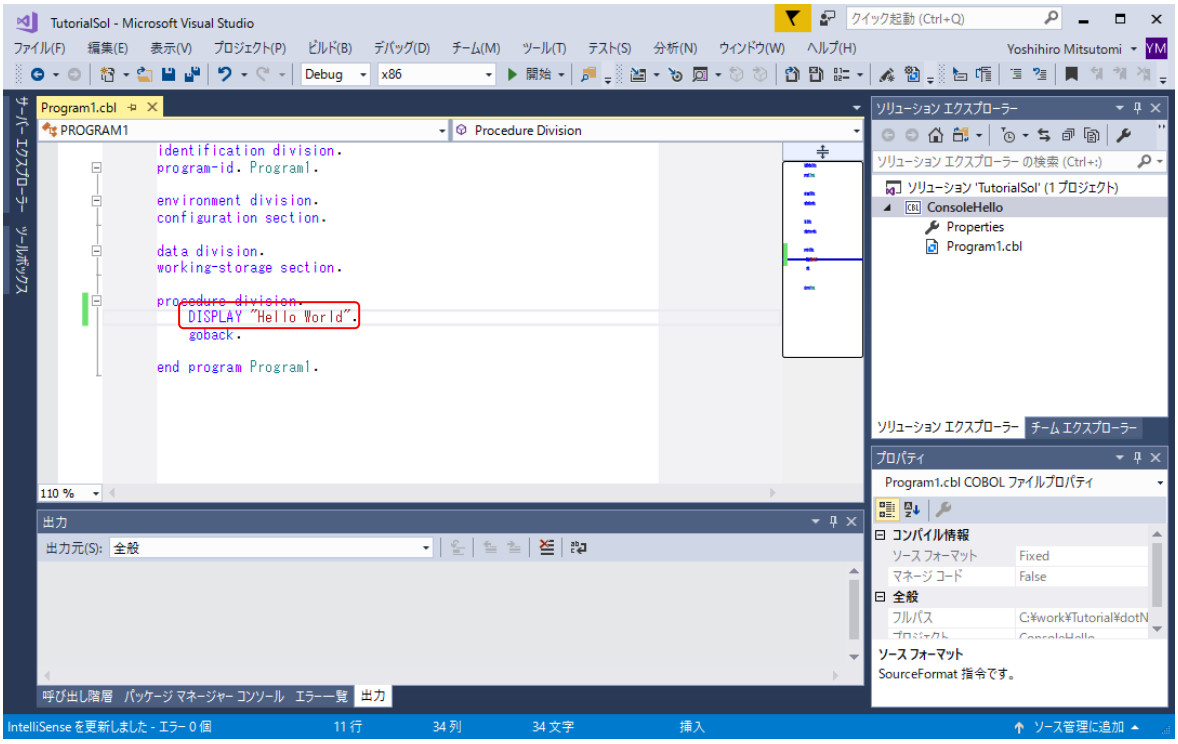


3　コードエディターで COBOL ソースコードを入力します。

プロジェクト「ConsoleHello」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。

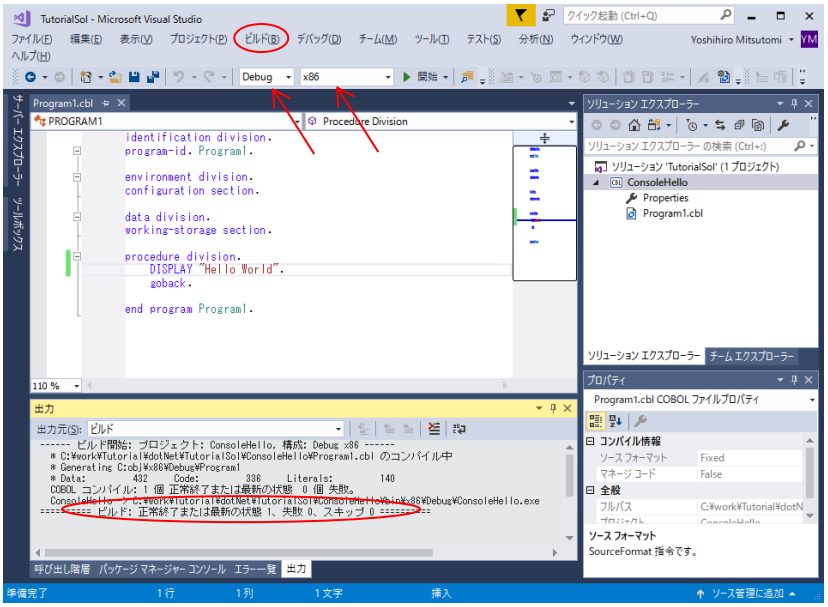
なお、COBOL 正書法ではエディター画面左右にあるグレー部分を特別な領域として利用するので、通常のソースコードはこれを避けて入力します。

```
DISPLAY "Hello World".
```



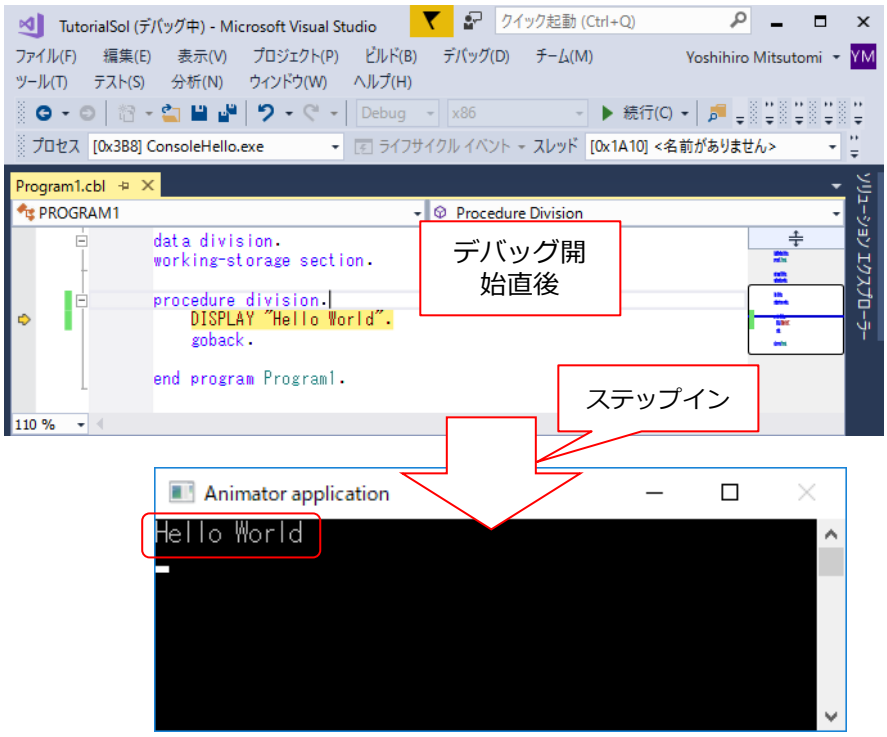
4　COBOL アプリケーションをビルドします。

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、**ビルド(B)** メニューから **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。



5　COBOL アプリケーションをデバッグ実行します。

デバッグ(D)メニューから **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。



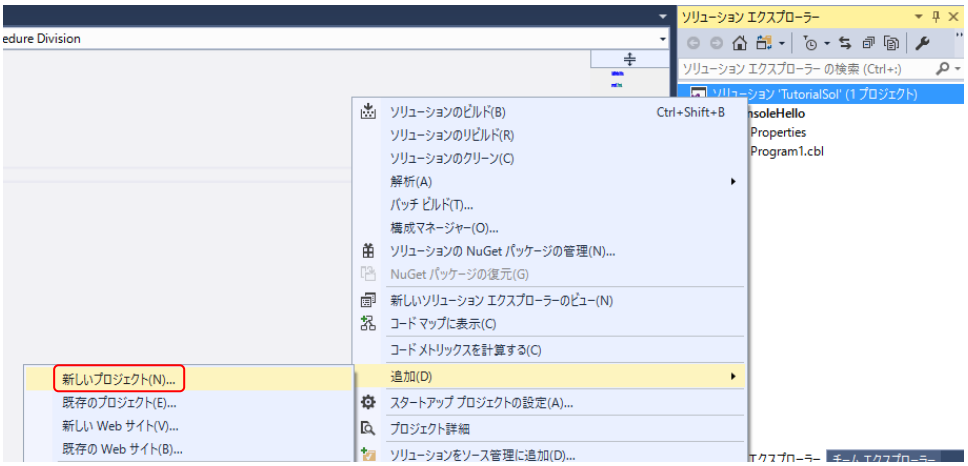
コマンドプロンプト画面に「Hello World」が表示されたことを確認して、デバッグを終了します。

第4章　Visual COBOL の画面操作

続いて、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 Visual COBOL for Visual Studio 2017 で作成します。

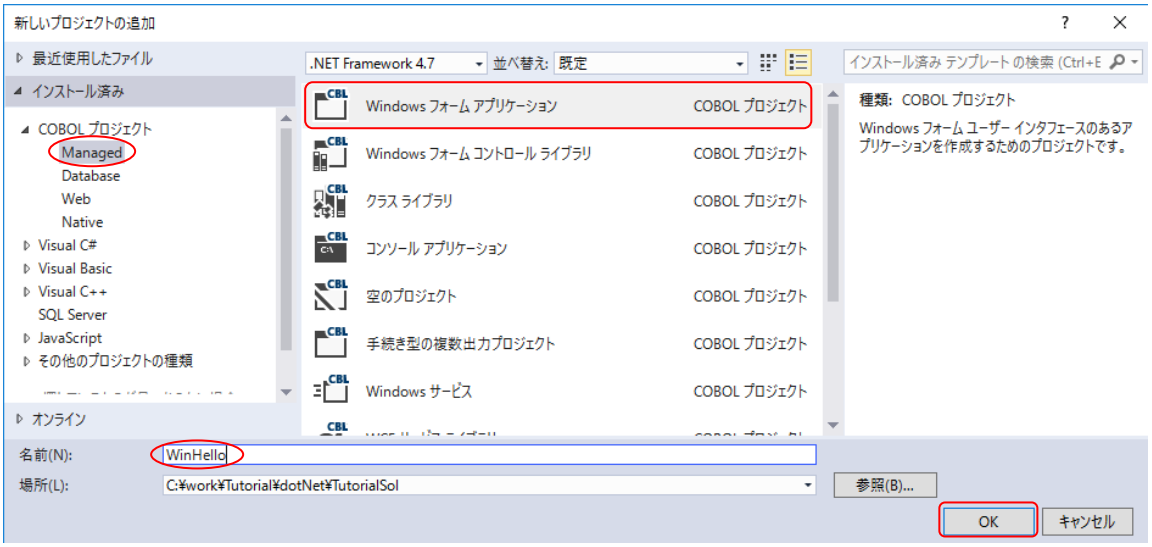
1　作成したソリューションへプロジェクトを追加します。

第３章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、ソリューションを右クリックし、**追加(D)** > **新しいプロジェクト(N)...** へとナビゲートします。



2　使用するテンプレートを選択します。

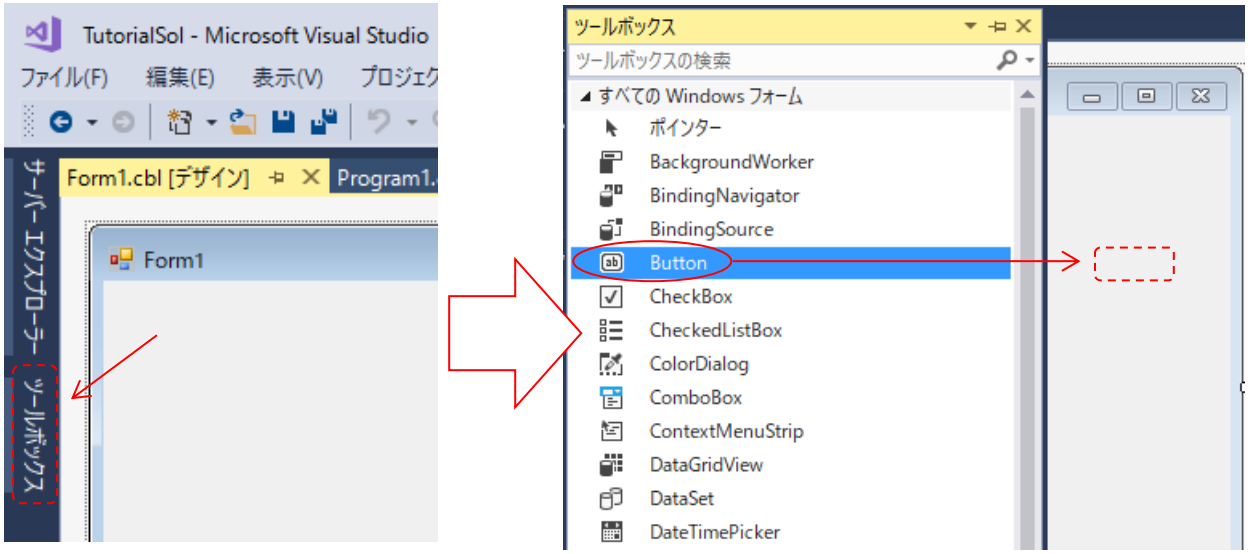
インストールされたテンプレートの一覧から **COBOL プロジェクト**、**Managed**、**Windows フォームアプリケーション**を選択します。名前(N)に **WinHello** と入力し、[**OK**] ボタンを押下します。



3　フォームデザイナーでウィンドウを作成します。

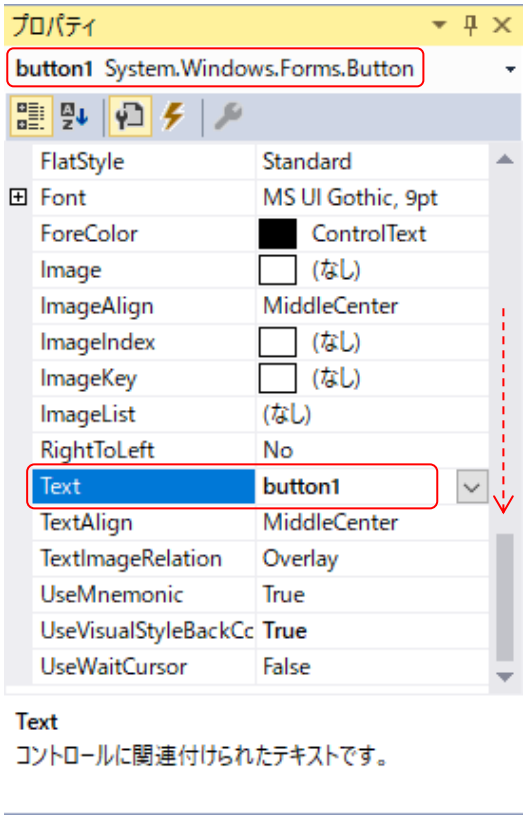
プロジェクト「WinHello」の作成が成功すると、フォームデザイナーが起動します。

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。

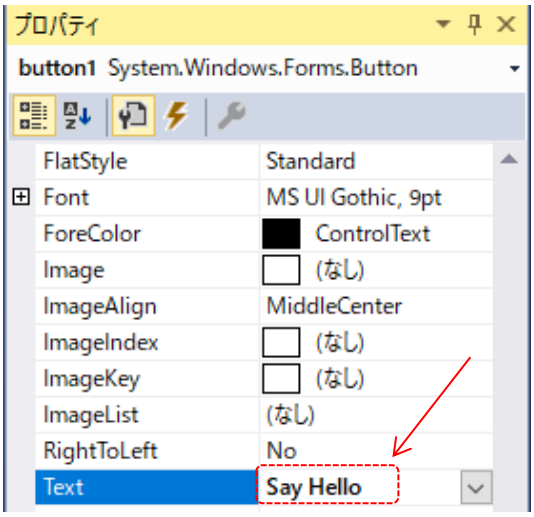


Form1 ウィンドウ上にボタンが表示されると、プロパティが **Button1** ボタンに切り替わります。

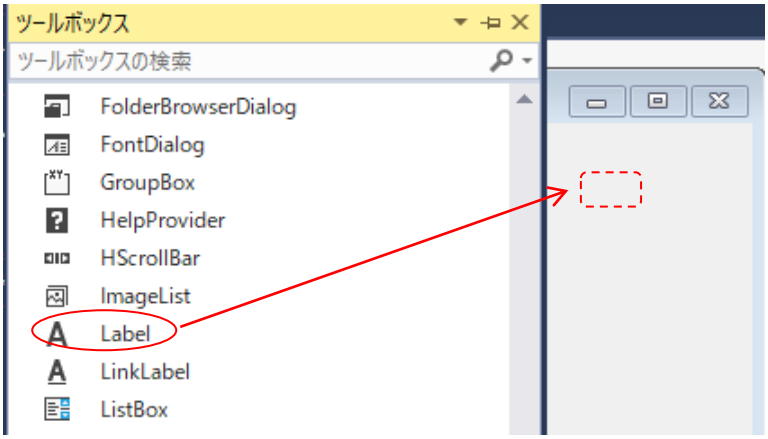
プロパティを下方向にスクロールして「表示」セクションの **Text** を選択します。



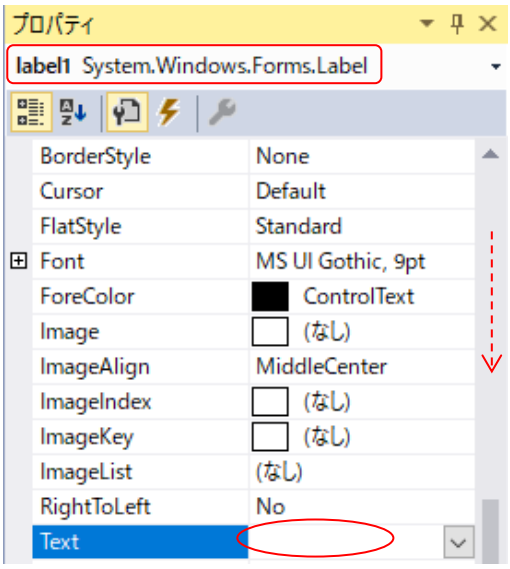
テキストの値を「Button1」から「Say Hello」に変更します。



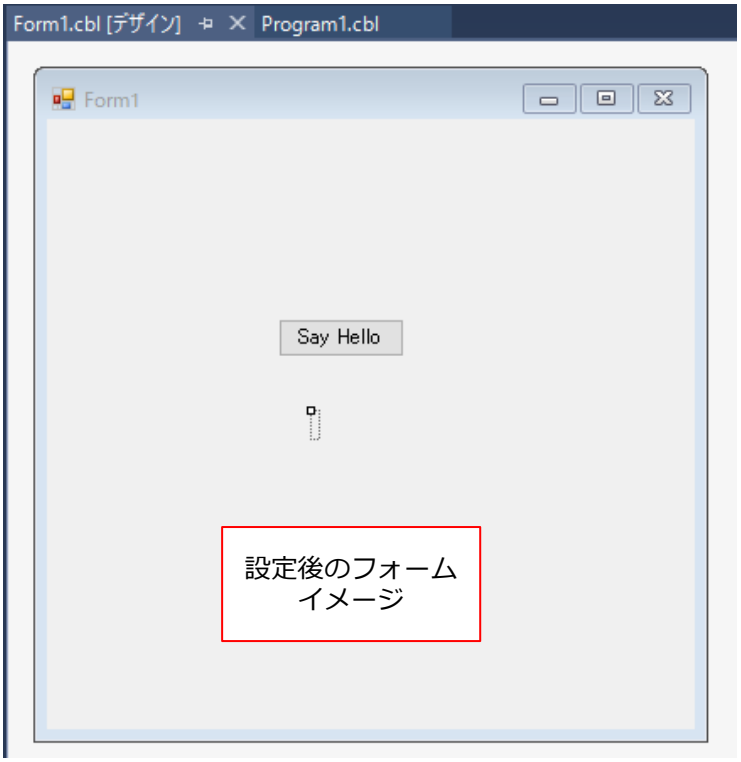
ツールボックスをスクロールして **Label** コントロールを選択し、**Form1** ウィンドウ上にドラッグ&ドロップします。



プロパティをスクロールして「表示」セクションの **Text** を選択し、テキストの値を削除します。



以上でウィンドウ画面の作成は終了です。

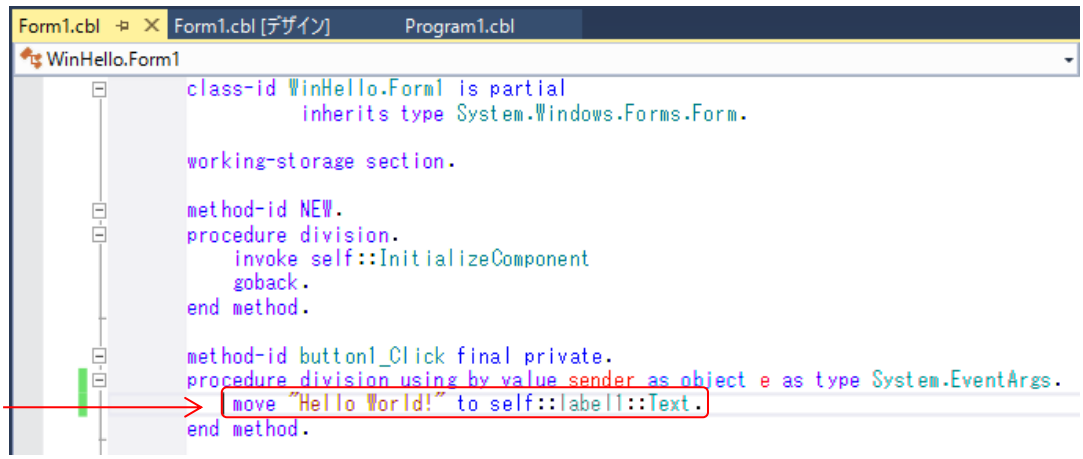


4　コードエディターで COBOL ソースコードを入力します。

次に、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。

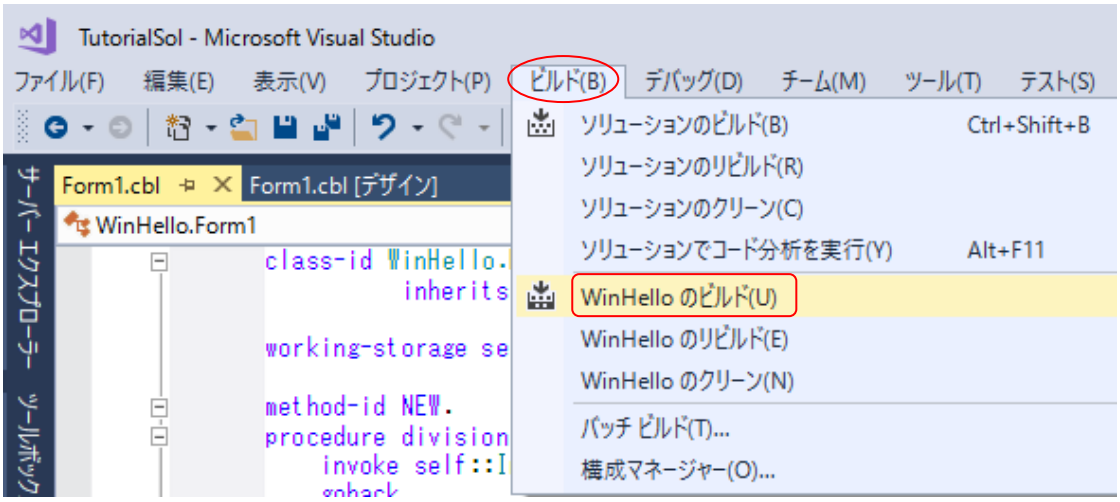
エディター画面には、Windows フォームアプリケーションのひな形が表示されます。 ここでは **Say Hello** ボタンをクリックした時の処理を記述するので、**button1_Click** メソッドの手続き部に以下の move 文を追加します。

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

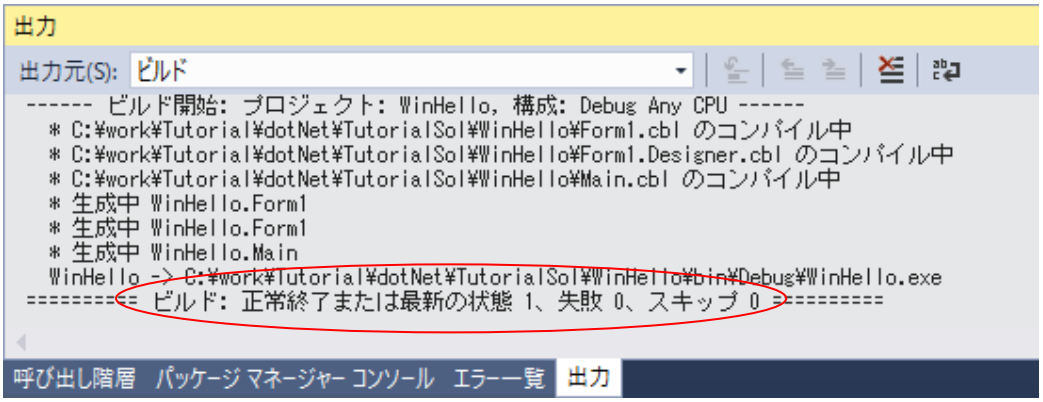


5　COBOL アプリケーションをビルドします。

スペルミスがなければ、**ビルド(B)**メニューから **WinHello のビルド(U)** を選択します。

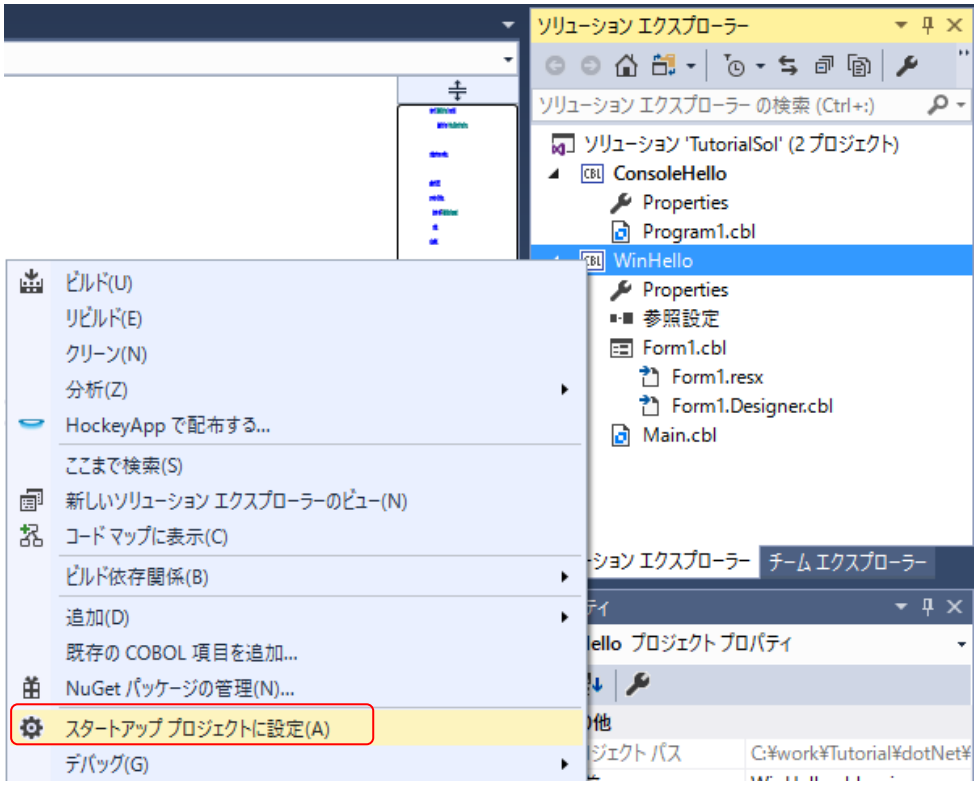


出力ウィンドウにビルド結果が表示されますので、ビルドが正常終了したことを確認します。

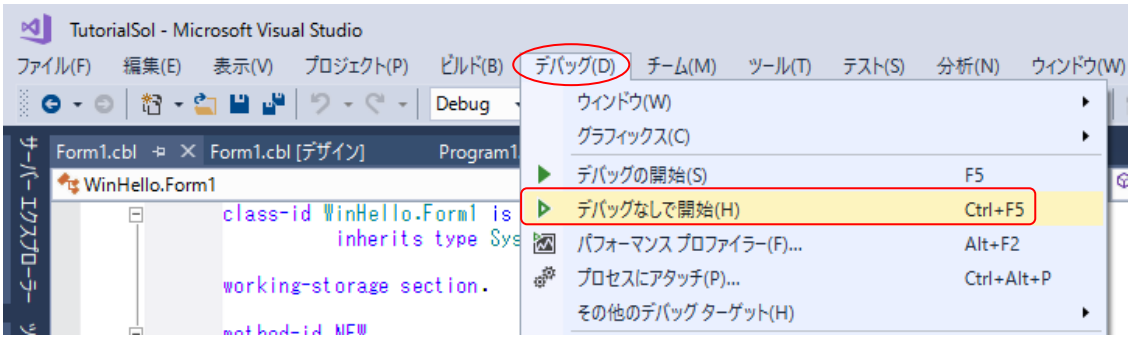


6　COBOL アプリケーションを実行します。

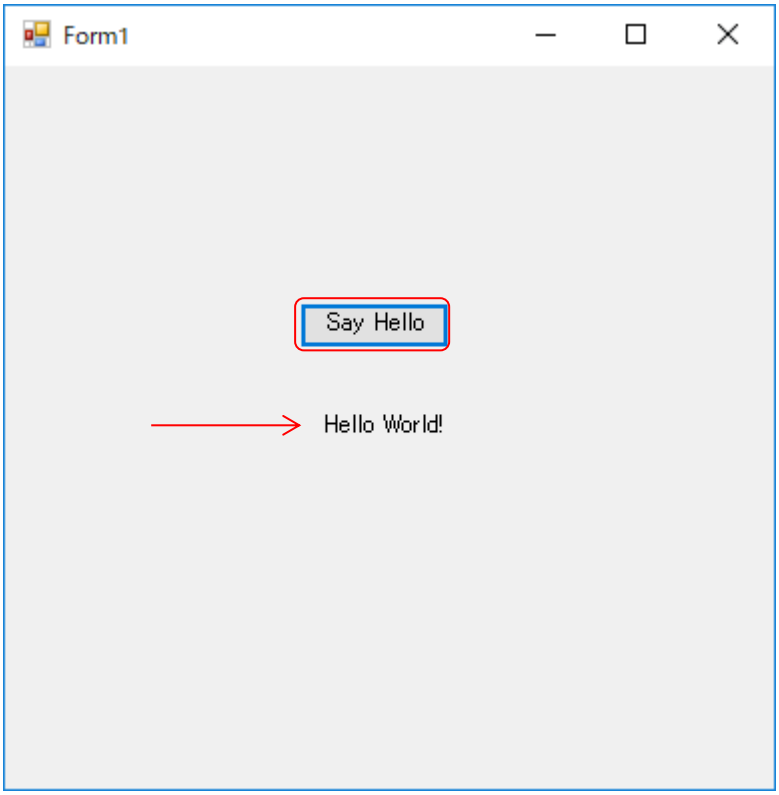
ソリューションエクスプローラーにて **WinHello** プロジェクトを右クリックし、**スタートアップに設定(A)** を選択します。



デバッグ(D)メニューから **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。



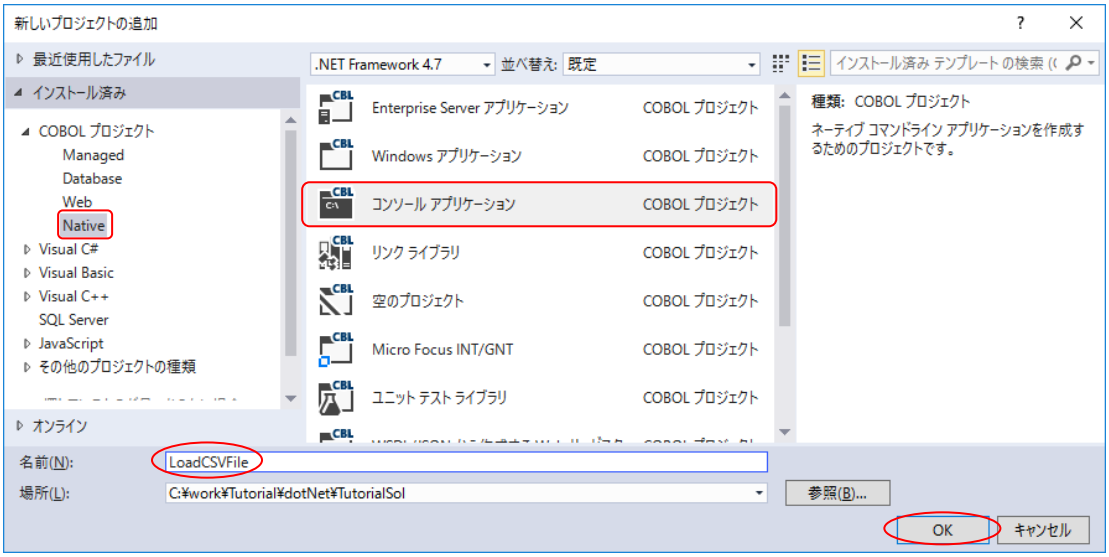
[×] アイコンをクリックして、アプリケーションを終了させます。

第5章　Visual COBOL のファイル入出力

次に、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 Visual Studio 2017 で作成しましょう。

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。

第３章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、ソリューションを右クリックし、**追加(D)** > **新しいプロジェクト(N)...** へとナビゲートします。**インストールされたテンプレート**の一覧から **COBOL プロジェクト**、**Native**、**コンソールアプリケーション**を選択します。**名前(N)** に **LoadCSVFile** と入力し、 **ＯＫ** をクリックします。

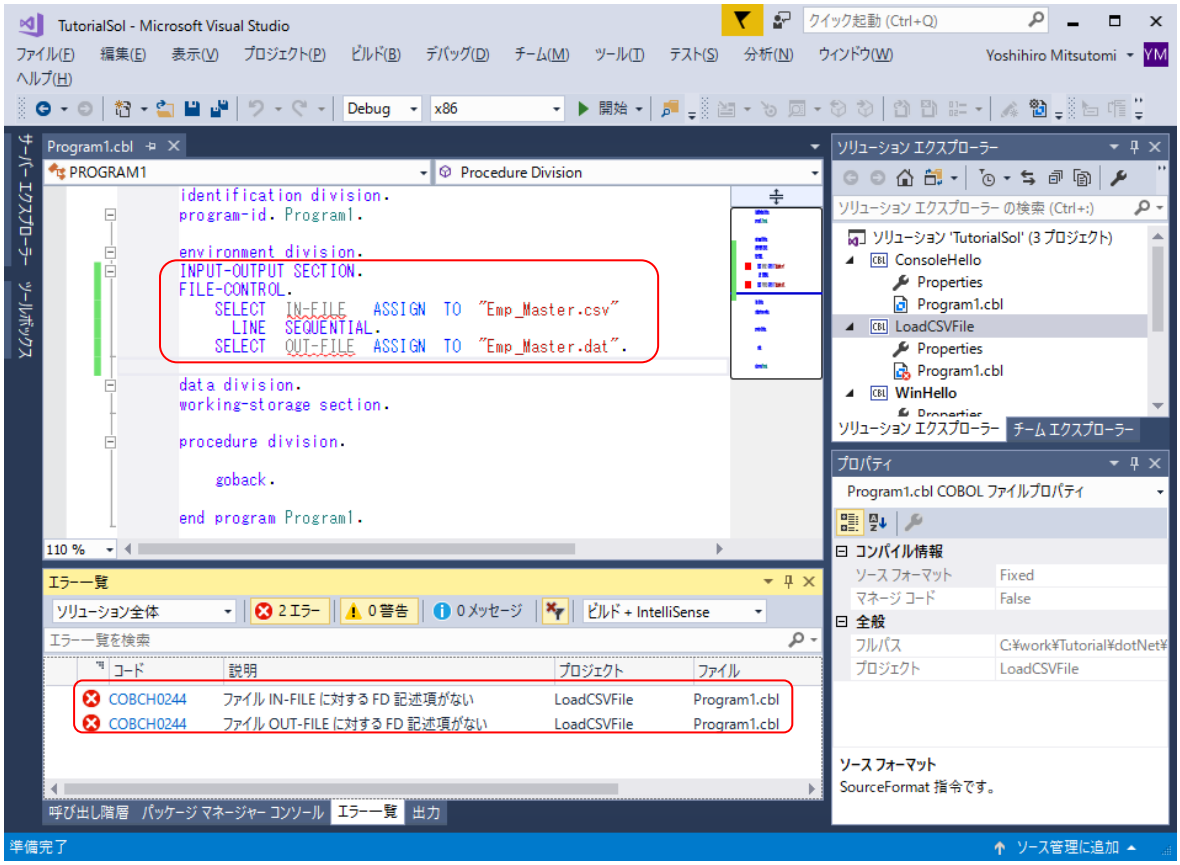


2　コードエディターで COBOL ソースコードを入力します。

プロジェクト「LoadCSVFile」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。

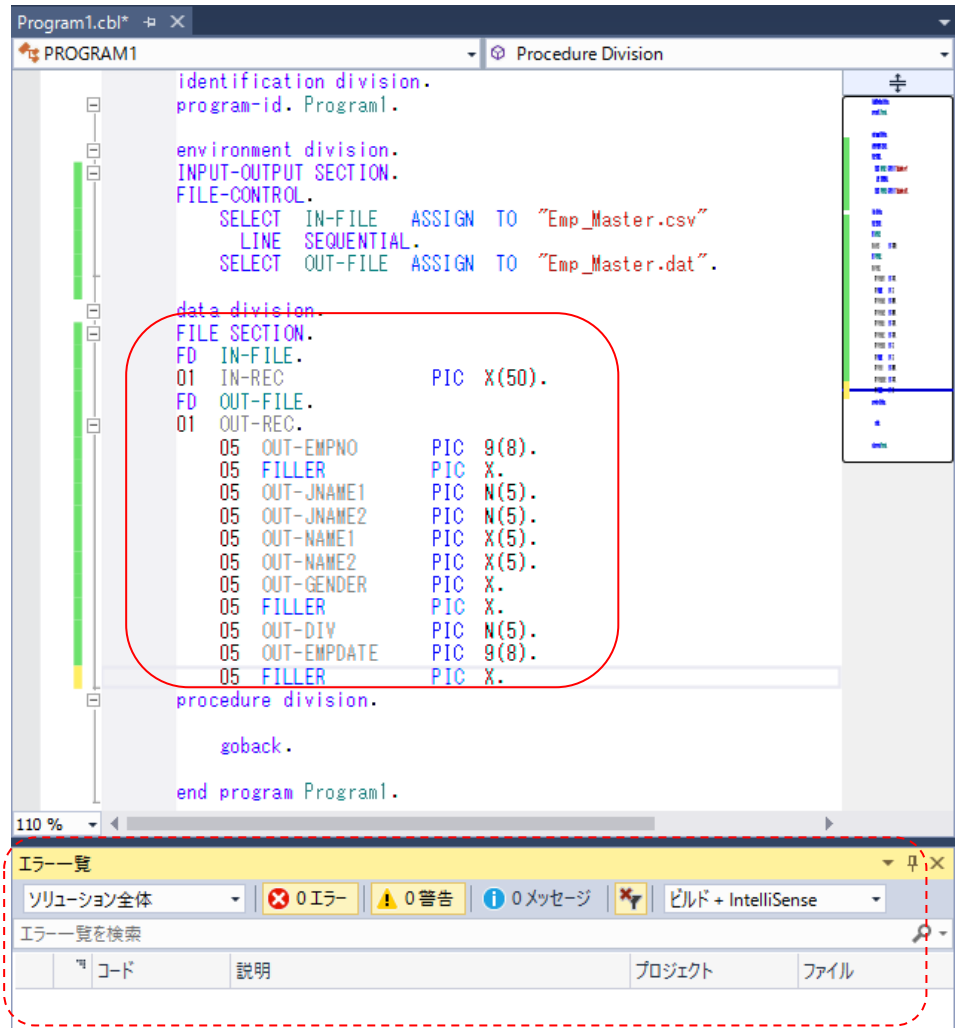
まず、環境部の構成節(configuration section) を削除し、以下の入出力節(input-output section) を追加します。 また、データ部のファイル定義が未入力なので IN-FILE と OUT-FILE がエラーとなりますが、ここでは無視して構いません。

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



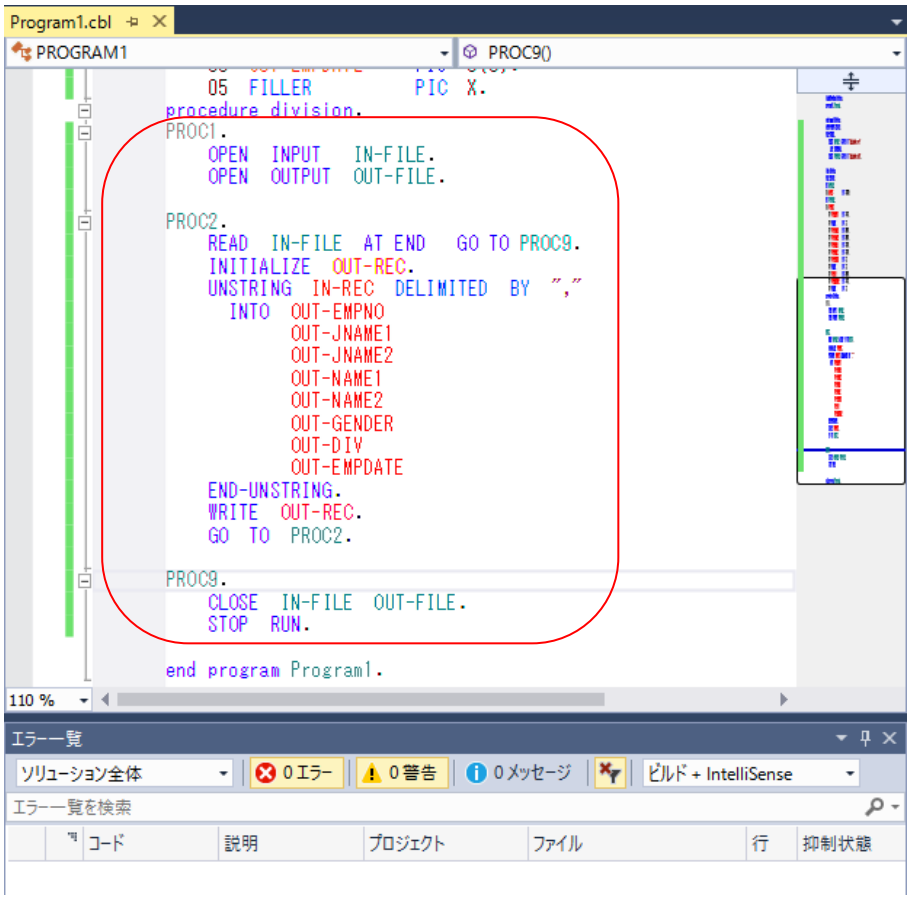
次に、データ部の作業場所節(working-storage section) を削除し、以下のファイル節(file section) を追加します。 なお、データ部のファイル定義を入力したので、環境部のエラーは無くなります。

```
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.
```



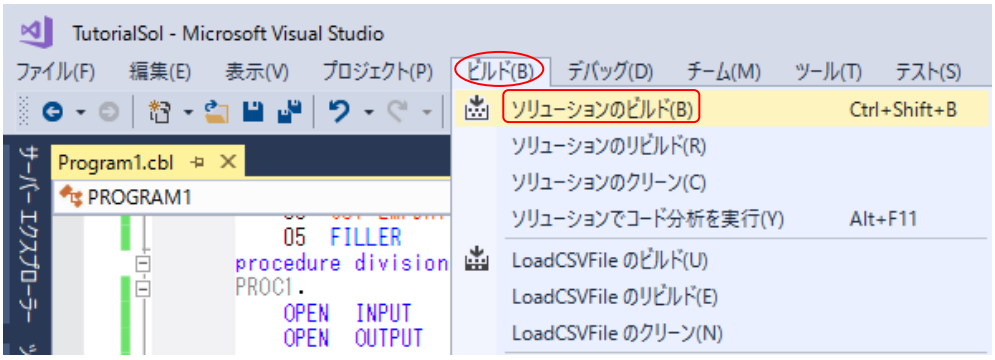
最後に、手続き部の goback 文を削除し、以下の 手続き文を追加します。

```
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.
```



3　COBOL アプリケーションをビルドします。

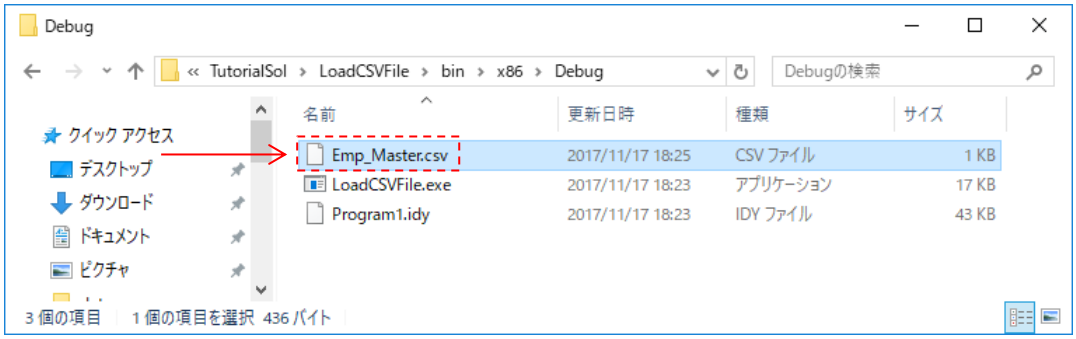
終止符(ピリオド)を含めてスペルミスがなければ、ソリューション構成が **Debug**、ソリューションプラットフォームが **x86** であることを確認して、**ビルド(B)**メニューから **ソリューションのビルド(B)** を選択します。 出力ウィンドウにビルド結果が表示されるので、すべてのビルドが正常終了したことを確認します。



4　CSV ファイルを作成します。

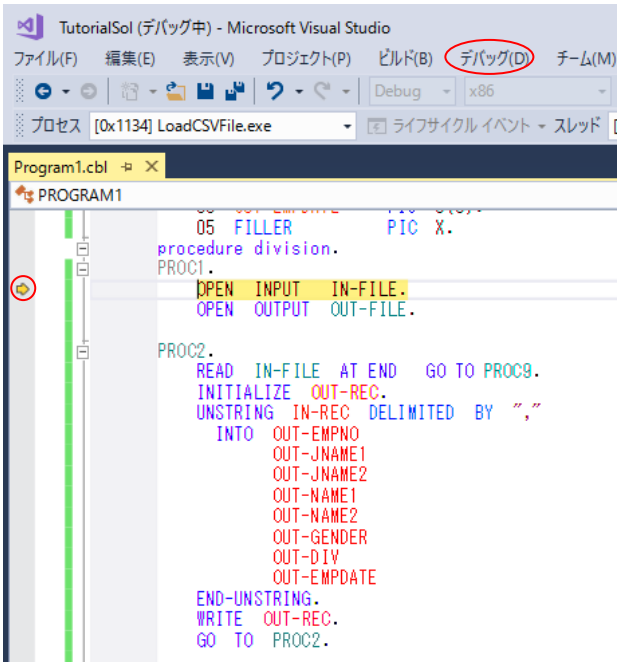
デバッグフォルダ(<第３章１で指定したフォルダ> ¥LoadCVSFile¥LoadCVSFile¥bin¥x86¥debug)にメモ帳などを利用して以下の **Emp_Master.csv** ファイルを作成します。

11111113,佐藤,隆,ｻﾄｳ,ﾀｶｼ,M,営業部,19980401,0
22222226,鈴木,尚之,ｽｽﾞｷ,ﾅｵﾕｷ,M,技術部,19981015,0
33333339,田中,直美,ﾀﾅｶ,ﾅｵﾐ,F,総務部,19990401,0
44444442,山田,洋一,ﾔﾏﾀﾞ,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,ｼﾝｼﾞ,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

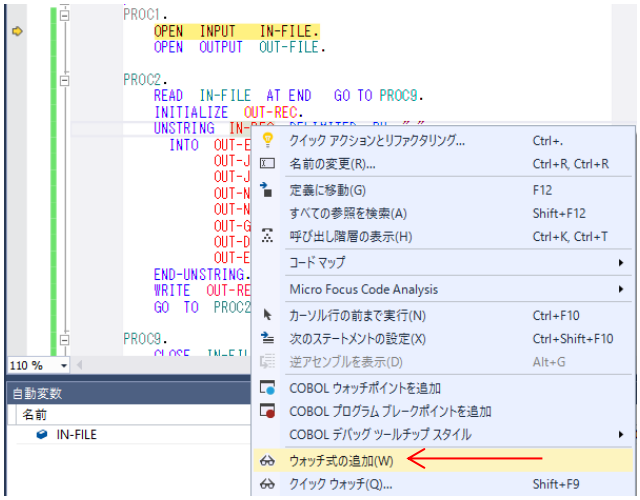


5　COBOL アプリケーションをデバッグ実行します。

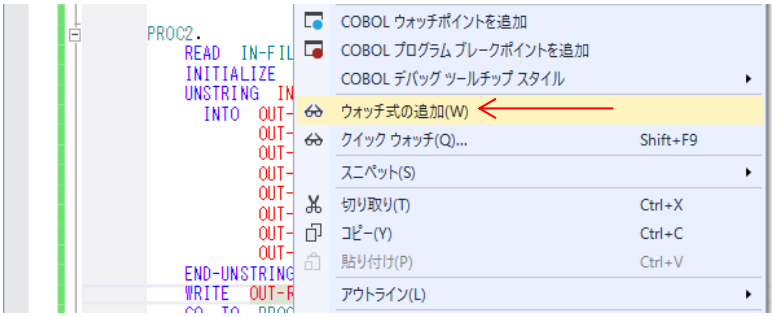
ソリューションエクスプローラーにて **LoadCSVFile** を右クリックから **スタートアッププロジェクトに設定(A)** を選択します。続いて、**デバッグ(D)**メニューから **ステップイン(I)** を選択するか **F11** キーを押すと、コマンドプロンプト画面が開き、デバッガーがステップ実行を開始します。 デバッガーは手続き部の最初の COBOL 文である open 文で実行を中断します。



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 上で右クリックして **ウォッチ式の追加(W)** を選択します。

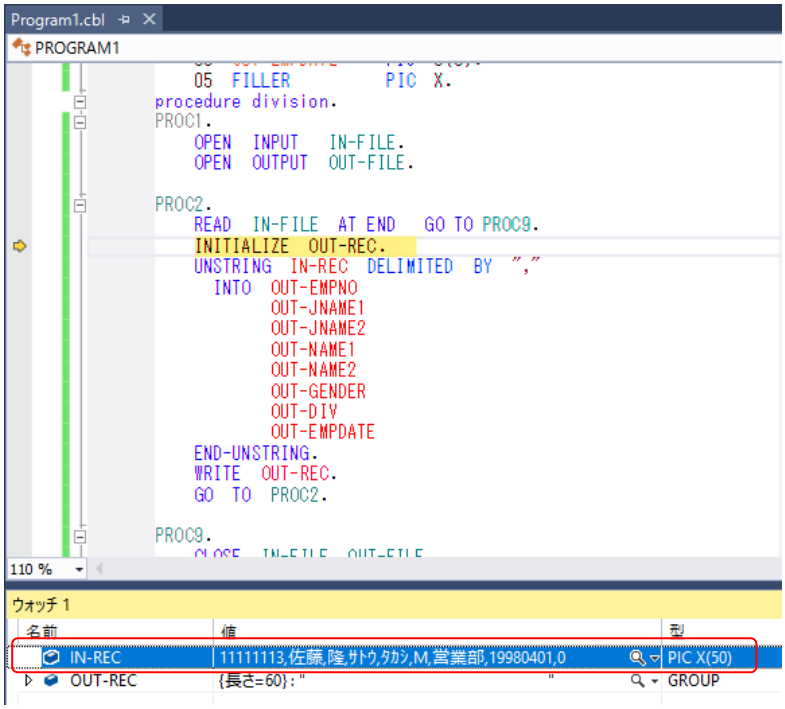


同様に出力ファイルに書き出すレコードの内容を確認するため、initialize 文の out-rec 上で右クリックして **ウォッチ式の追加(W)** を選択します。



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。

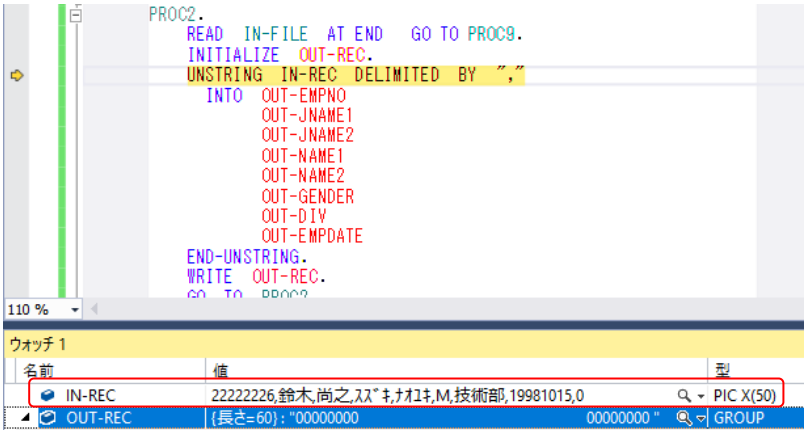


さらに **F11** キーを 2 回押すと、デバッガーは unstring 文を実行後、処理を中断します。

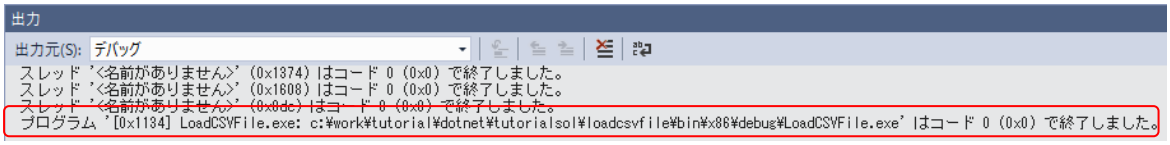
ウォッチ式の out-rec の値には出力ファイルへ書き出す最初のレコードが表示されます。

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。

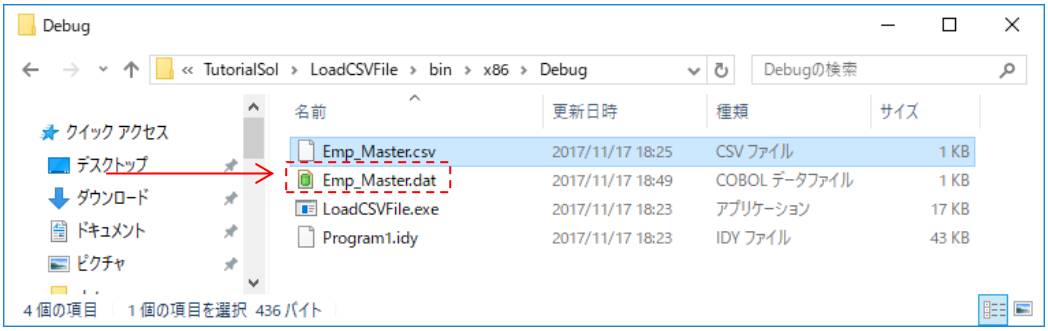
ウォッチ式の in-rec の値には CSV ファイルから読み込んだ 2 番目のレコードが表示され、out-rec の値は initialize 文で初期化されています。



デバッグ(D)メニューから **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。



デバッグフォルダ(<第 5 章**エラー! 参照元が見つかりません。**で指定したフォルダ>¥LoadCVSFile¥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。下図は、Tera Pad を使って 60 桁で折り返し表示した例です。

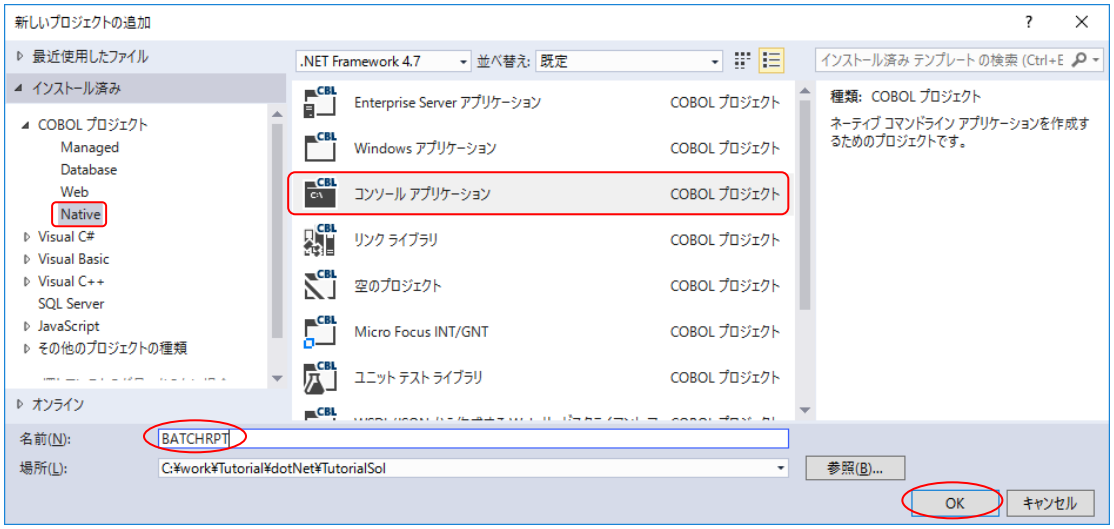


第6章　Visual COBOL のバッチアプリケーション

本章では、第 5 章で作成した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 Visual COBOL for Visual Studio 2017 で作成します。

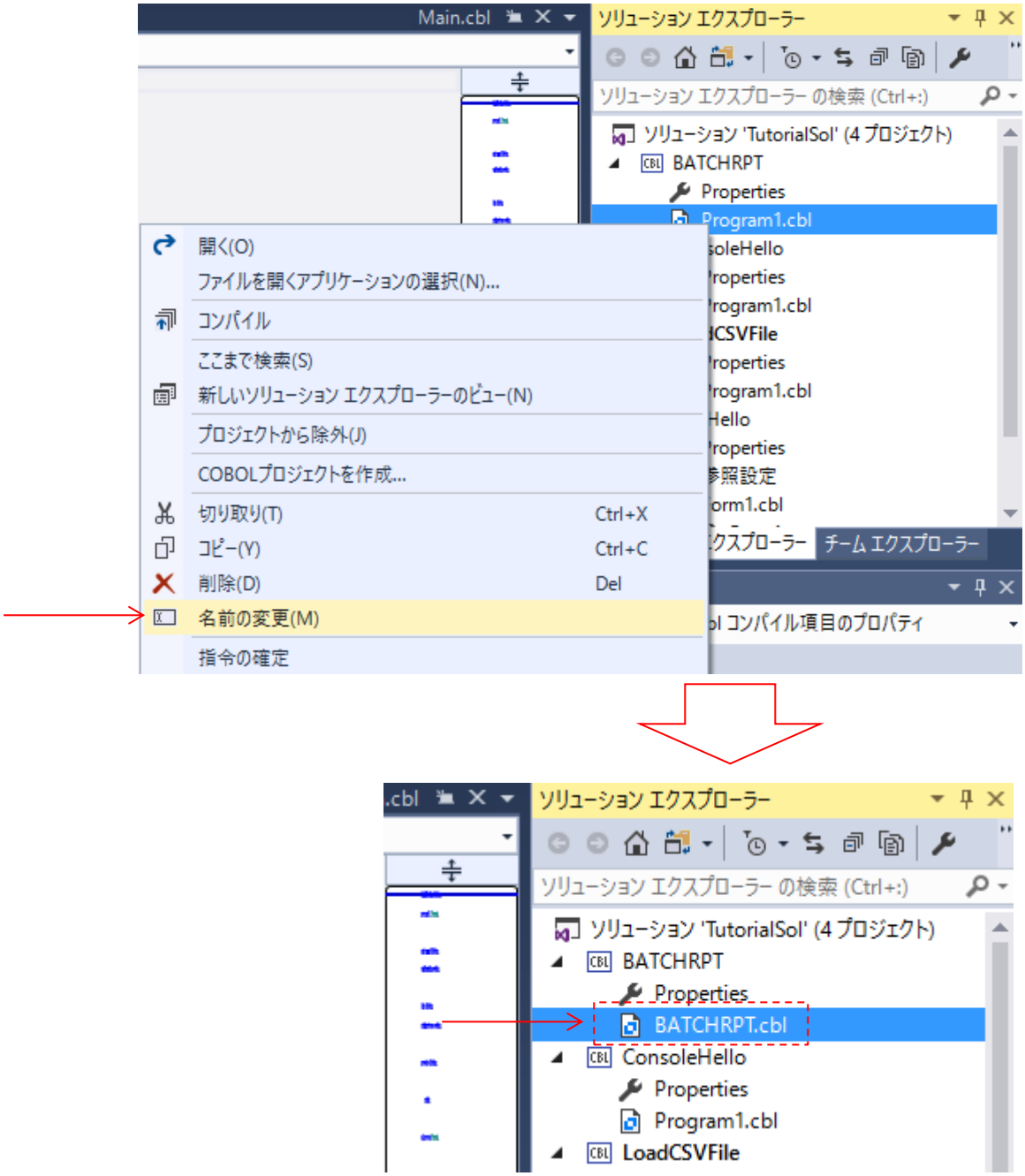
1　作成したソリューションへプロジェクトを追加します。

第 3 章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、ソリューションを右クリックし、**追加(D)** > **新しいプロジェクト(N)...** へとナビゲートします。 **インストールされたテンプレート**の一覧から **COBOL プロジェクト**、**Native**、**コンソールアプリケーション**を選択します。名前(N)に **BATCHRPT** と入力し、**ＯＫ** をクリックします。



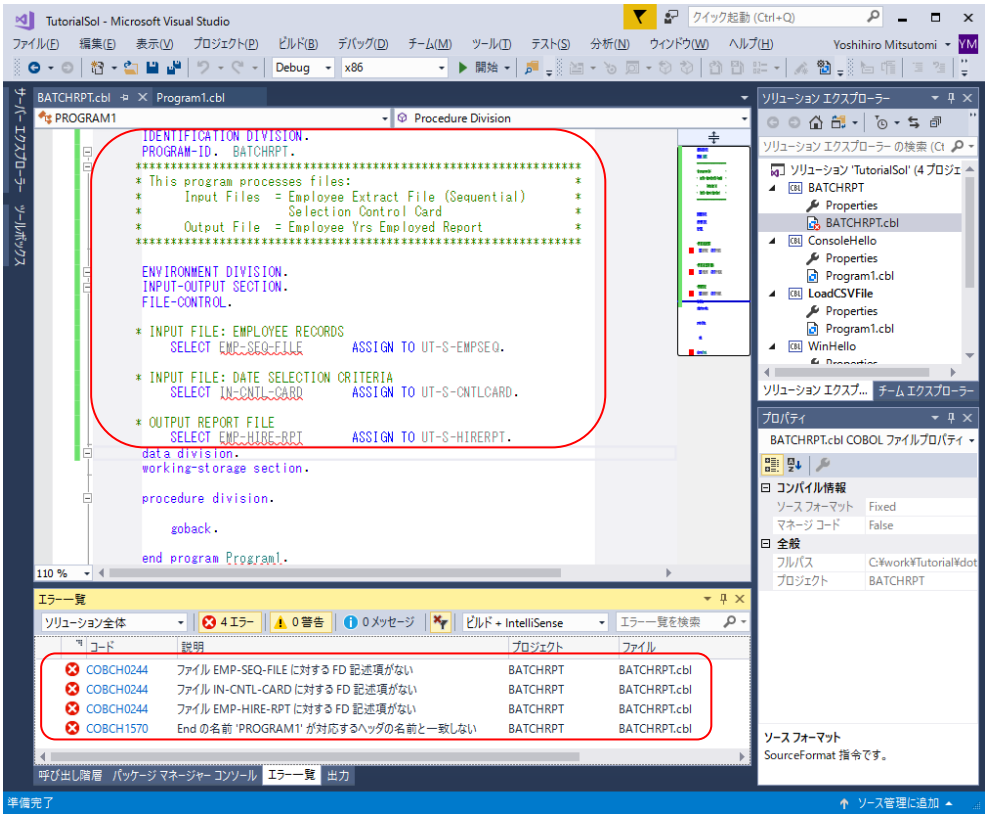
2　コードエディターで COBOL ソースコードを入力します。

プロジェクト「BATCHRPT」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ソースプログラム「Program1.cbl」を右クリックして **名前の変更(M)** を選択し、プログラム名を「BATCHRPT.cbl」に書き換えます。



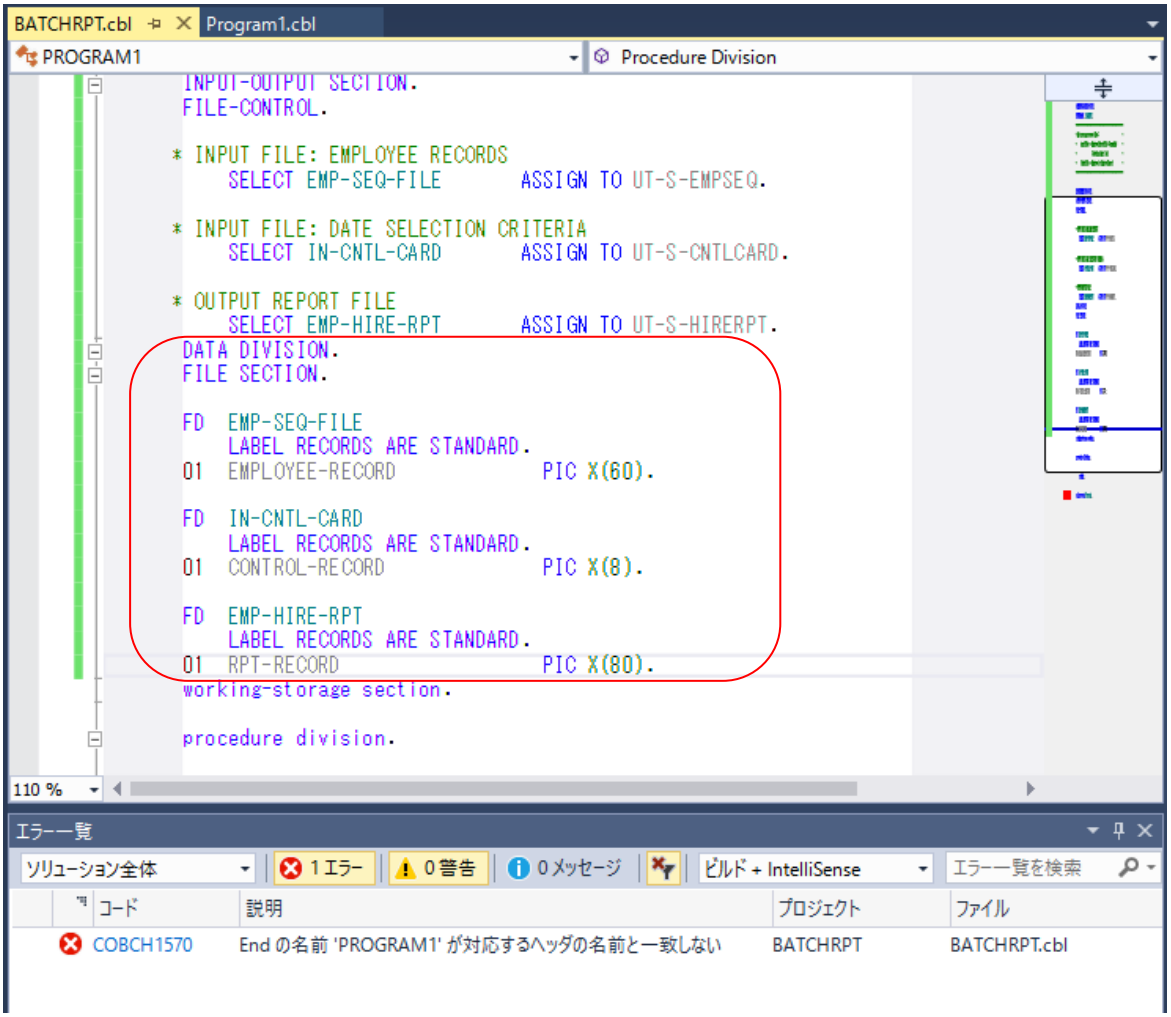
本章では既存資産の流用を想定して COBOL 正書法に従った伝統的スタイルのソースコードを入力しますので、アスタリスクで始まるコメント行が 7 列目(エディター画面左側のグレー領域の右端)から始まるよう注意して、以下の見出し部と環境部を入力します。 この時点では、データ部のファイル定義未入力によるエラーとなりますが、ここでは無視して構いません。

```
 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.
```



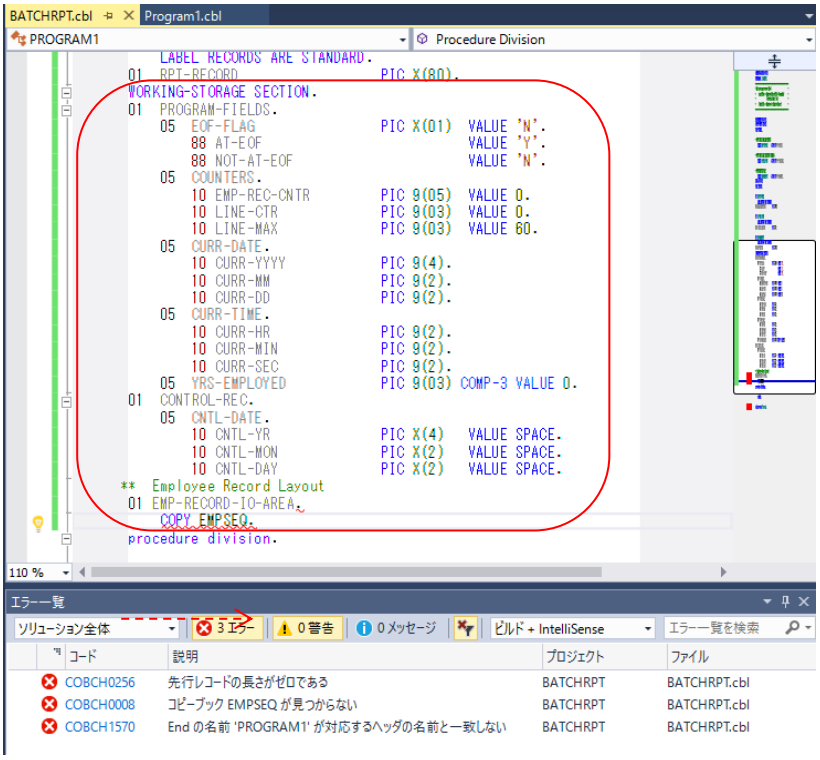
データ部のファイル節を入力します。 なお、データ部のファイル定義を入力したので、環境部のエラーは無くなります。

```
DATA DIVISION.
FILE SECTION.

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

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).
```



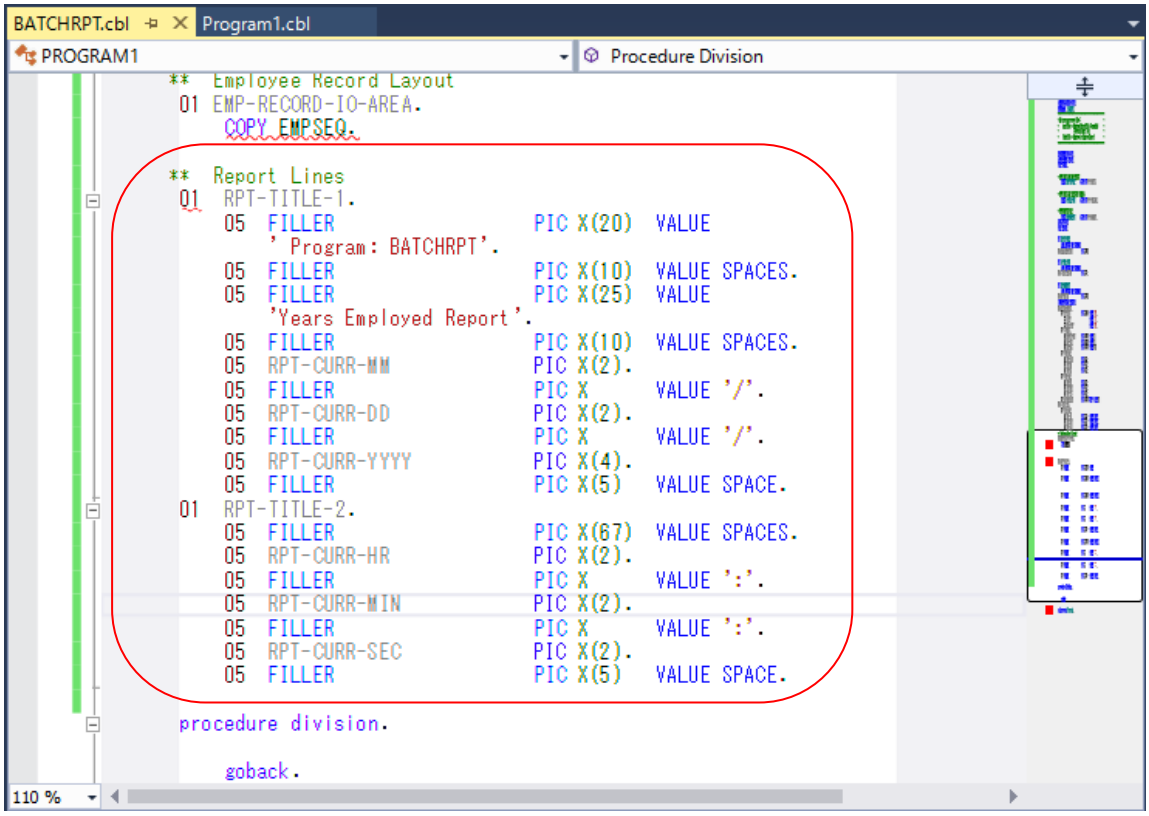
データ部の作業場所節で PROGRAM-FIELDS、CONTROL-REC データ項目を入力します。 COPY 文で外部参照する EMP-RECORD-IO-AREA データ項目はエラーとなりますが、無視して構いません。

```
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'.
    05  COUNTERS.
        10 EMP-REC-CNTR          PIC 9(05)   VALUE 0.
        10 LINE-CTR              PIC 9(03)   VALUE 0.
        10 LINE-MAX              PIC 9(03)   VALUE 60.
    05  CURR-DATE.
        10 CURR-YYYY             PIC 9(4).
        10 CURR-MM               PIC 9(2).
        10 CURR-DD               PIC 9(2).
    05  CURR-TIME.
        10 CURR-HR               PIC 9(2).
        10 CURR-MIN              PIC 9(2).
        10 CURR-SEC              PIC 9(2).
    05  YRS-EMPLOYED            PIC 9(03) COMP-3 VALUE 0.
01  CONTROL-REC.
    05  CNTL-DATE.
        10 CNTL-YR               PIC X(4)    VALUE SPACE.
        10 CNTL-MON              PIC X(2)    VALUE SPACE.
        10 CNTL-DAY              PIC X(2)    VALUE SPACE.
**  Employee Record Layout
 01 EMP-RECORD-IO-AREA.
    COPY EMPSEQ.
```



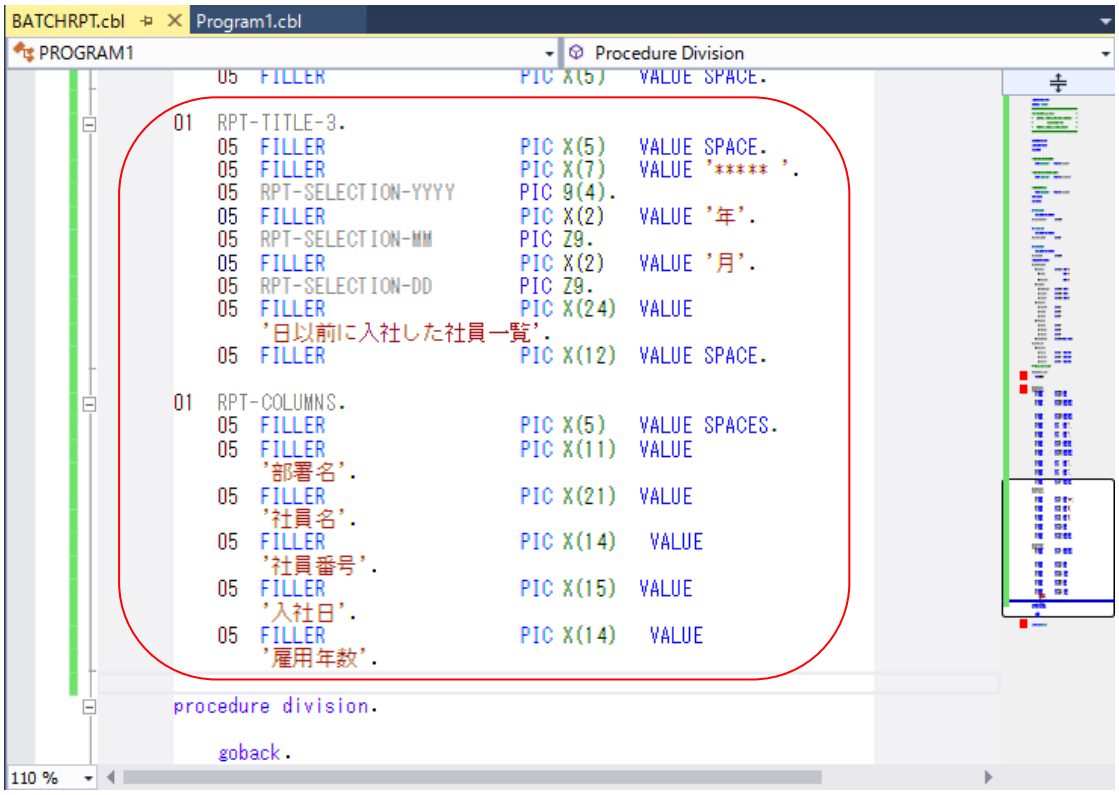
データ部の作業場所節で RPT-TITLE-1 と RPT-TITLE-2 データ項目を入力します。

```
**  Report Lines
 01  RPT-TITLE-1.
     05  FILLER                  PIC X(20)  VALUE
         ' Program: BATCHRPT'.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  FILLER                  PIC X(25)  VALUE
         'Years Employed Report'.
     05  FILLER                  PIC X(10)  VALUE SPACES.
     05  RPT-CURR-MM             PIC X(2).
     05  FILLER                  PIC X      VALUE '/'.
     05  RPT-CURR-DD             PIC X(2).
     05  FILLER                  PIC X      VALUE '/'.
     05  RPT-CURR-YYYY           PIC X(4).
     05  FILLER                  PIC X(5)   VALUE SPACE.
 01  RPT-TITLE-2.
     05  FILLER                  PIC X(67)  VALUE SPACES.
     05  RPT-CURR-HR             PIC X(2).
     05  FILLER                  PIC X      VALUE ':'.
     05  RPT-CURR-MIN            PIC X(2).
     05  FILLER                  PIC X      VALUE ':'.
     05  RPT-CURR-SEC            PIC X(2).
     05  FILLER                  PIC X(5)   VALUE SPACE.
```



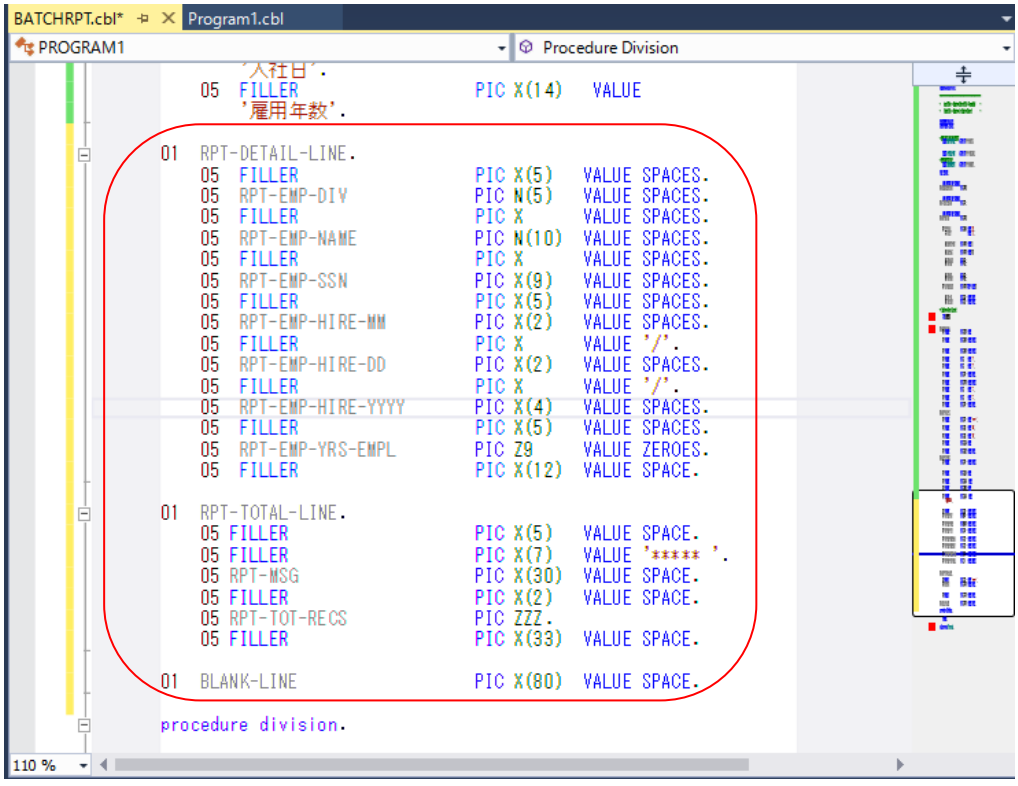
作業場所節で RPT-TITLE-3 と RPT-COLUMNS データ項目を入力します。

```
01  RPT-TITLE-3.
    05  FILLER                 PIC X(5)   VALUE SPACE.
    05  FILLER                 PIC X(7)   VALUE '***** '.
    05  RPT-SELECTION-YYYY     PIC 9(4).
    05  FILLER                 PIC X(2)   VALUE '年'.
    05  RPT-SELECTION-MM       PIC Z9.
    05  FILLER                 PIC X(2)   VALUE '月'.
    05  RPT-SELECTION-DD       PIC Z9.
    05  FILLER                 PIC X(24)  VALUE
        '日以前に入社した社員一覧'.
    05  FILLER                 PIC X(12)  VALUE SPACE.

01  RPT-COLUMNS.
    05  FILLER                 PIC X(5)   VALUE SPACES.
    05  FILLER                 PIC X(11)  VALUE
        '部署名'.
    05  FILLER                 PIC X(21)  VALUE
        '社員名'.
    05  FILLER                 PIC X(14)   VALUE
        '社員番号'.
    05  FILLER                 PIC X(15)  VALUE
        '入社日'.
    05  FILLER                 PIC X(14)   VALUE
        '雇用年数'.
```



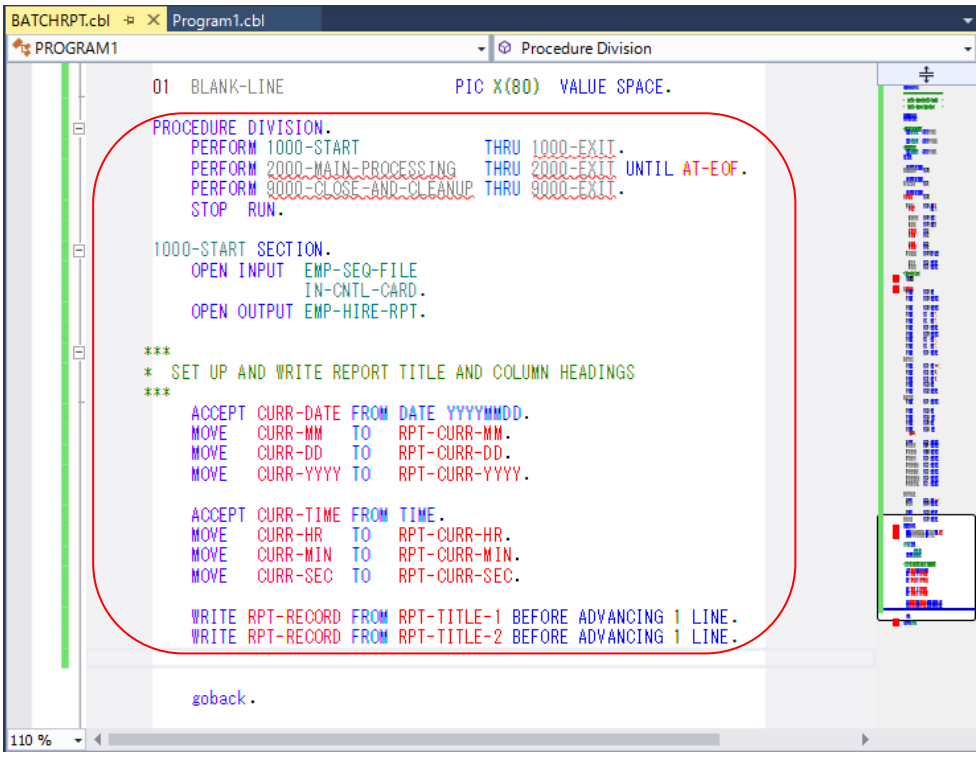
作業場所節で RPT-DETAIL-LINE、RPT-TOTAL-LINE と BLANK-LINE データ項目を入力します。

```
01  RPT-DETAIL-LINE.
    05  FILLER                  PIC X(5)    VALUE SPACES.
    05  RPT-EMP-DIV             PIC N(5)    VALUE SPACES.
    05  FILLER                  PIC X       VALUE SPACES.
    05  RPT-EMP-NAME            PIC N(10)   VALUE SPACES.
    05  FILLER                  PIC X       VALUE SPACES.
    05  RPT-EMP-SSN             PIC X(9)    VALUE SPACES.
    05  FILLER                  PIC X(5)    VALUE SPACES.
    05  RPT-EMP-HIRE-MM         PIC X(2)    VALUE SPACES.
    05  FILLER                  PIC X       VALUE '/'.
    05  RPT-EMP-HIRE-DD         PIC X(2)    VALUE SPACES.
    05  FILLER                  PIC X       VALUE '/'.
    05  RPT-EMP-HIRE-YYYY       PIC X(4)    VALUE SPACES.
    05  FILLER                  PIC X(5)    VALUE SPACES.
    05  RPT-EMP-YRS-EMPL        PIC Z9      VALUE ZEROES.
    05  FILLER                  PIC X(12)   VALUE SPACE.

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.
```



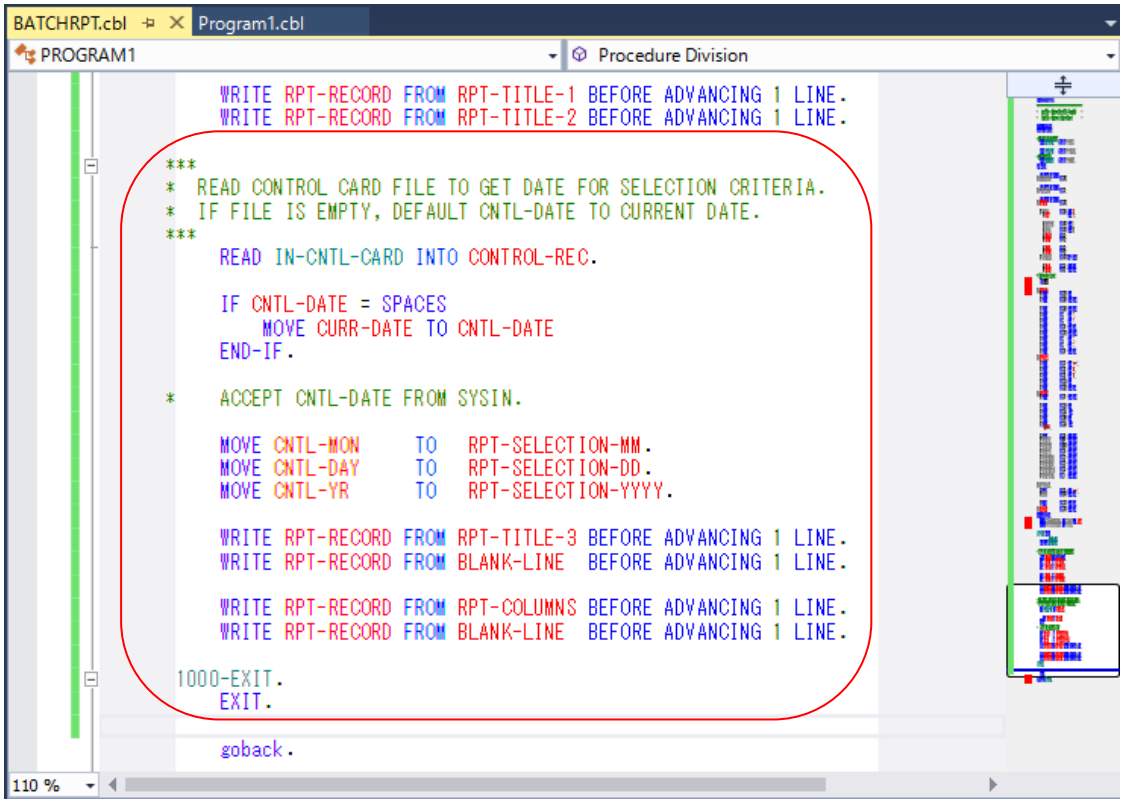
最後に、手続き部の 1000-START 節の前半部分を入力します。PERFORM 文で参照する手続き名が未定義なのでエラーが 5 件増えますが、気にせず先に進んでください。

```
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.
    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.
```



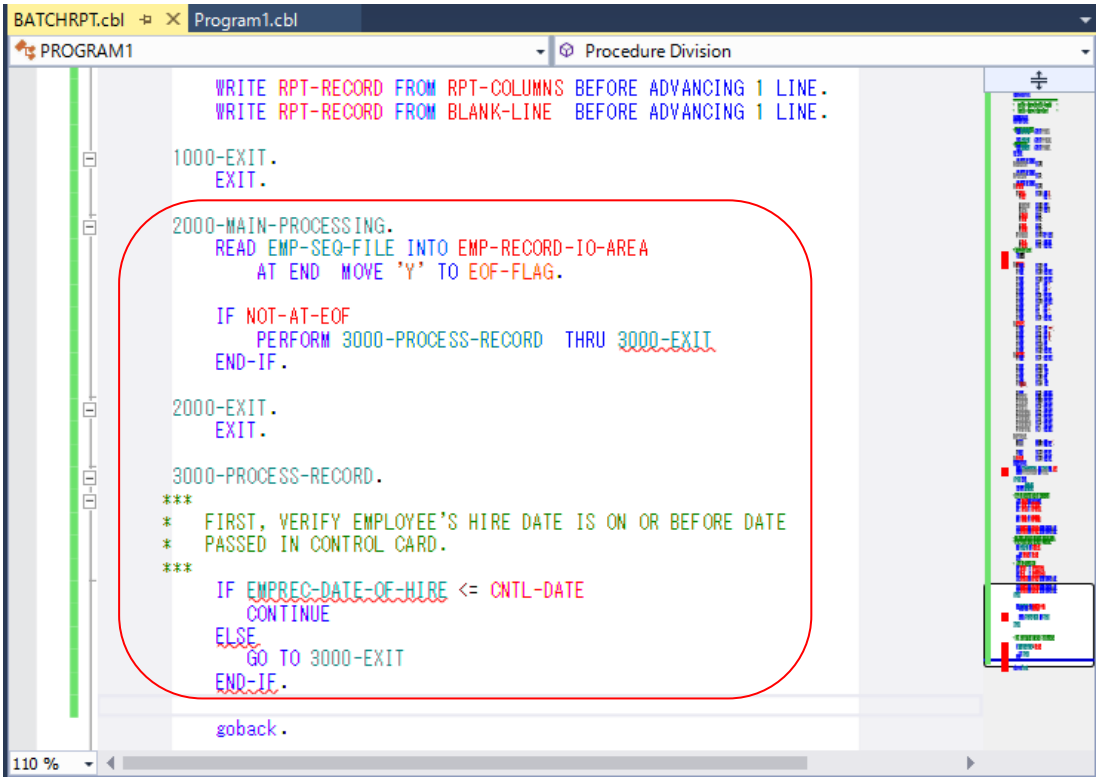
手続き部の 1000-START 節の後半部分を入力します。

```
***
*  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.
```



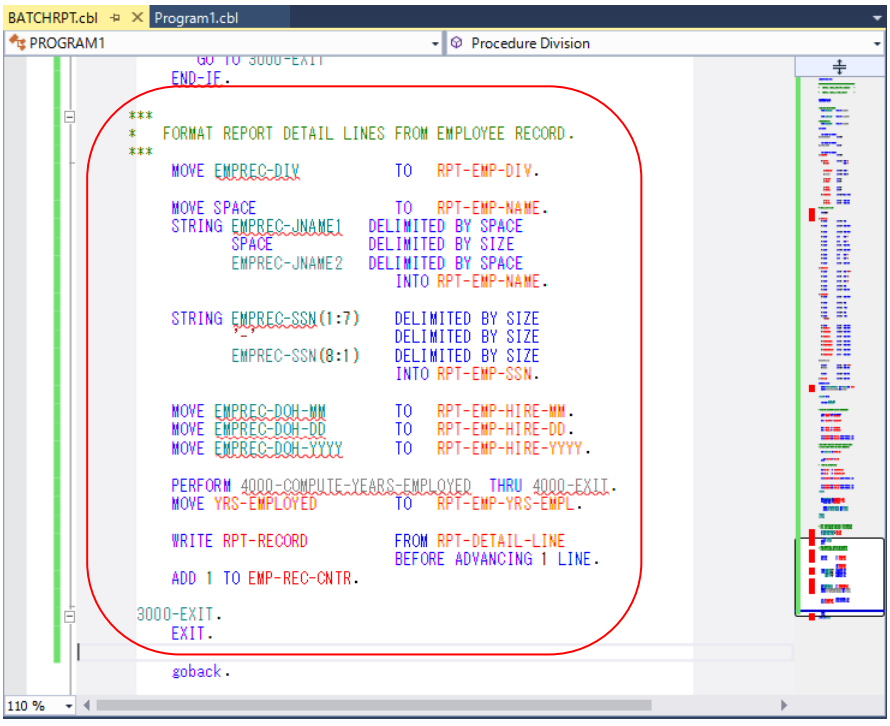
手続き部の 2000-MAIN-PROCESSING 段落と 3000-PROCESS-RECORD 段落の前半部分を入力します。

```
 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.
```



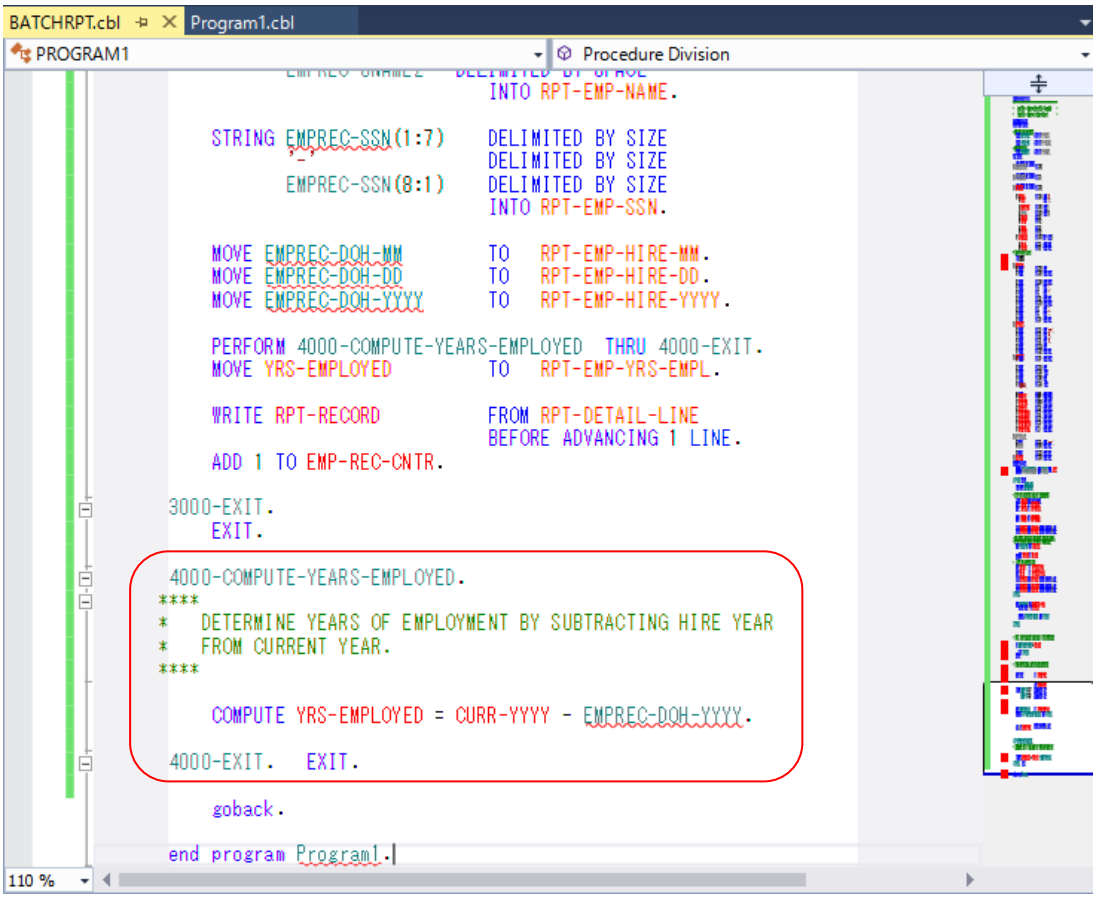
手続き部の 3000-PROCESS-RECORD 段落の後半部分を入力します。

```
***
*   FORMAT REPORT DETAIL LINES FROM EMPLOYEE RECORD.
***
     MOVE EMPREC-DIV            TO   RPT-EMP-DIV.

     MOVE SPACE                 TO   RPT-EMP-NAME.
     STRING EMPREC-JNAME1   DELIMITED BY SPACE
            SPACE           DELIMITED BY SIZE
            EMPREC-JNAME2   DELIMITED BY SPACE
                               INTO RPT-EMP-NAME.

     STRING EMPREC-SSN(1:7)     DELIMITED BY SIZE
            '-'                 DELIMITED BY SIZE
            EMPREC-SSN(8:1)     DELIMITED BY SIZE
                                INTO RPT-EMP-SSN.

     MOVE EMPREC-DOH-MM         TO   RPT-EMP-HIRE-MM.
     MOVE EMPREC-DOH-DD         TO   RPT-EMP-HIRE-DD.
     MOVE EMPREC-DOH-YYYY       TO   RPT-EMP-HIRE-YYYY.

     PERFORM 4000-COMPUTE-YEARS-EMPLOYED  THRU 4000-EXIT.
     MOVE YRS-EMPLOYED          TO   RPT-EMP-YRS-EMPL.

     WRITE RPT-RECORD           FROM RPT-DETAIL-LINE
                                BEFORE ADVANCING 1 LINE.
     ADD 1 TO EMP-REC-CNTR.

 3000-EXIT.
     EXIT.
```



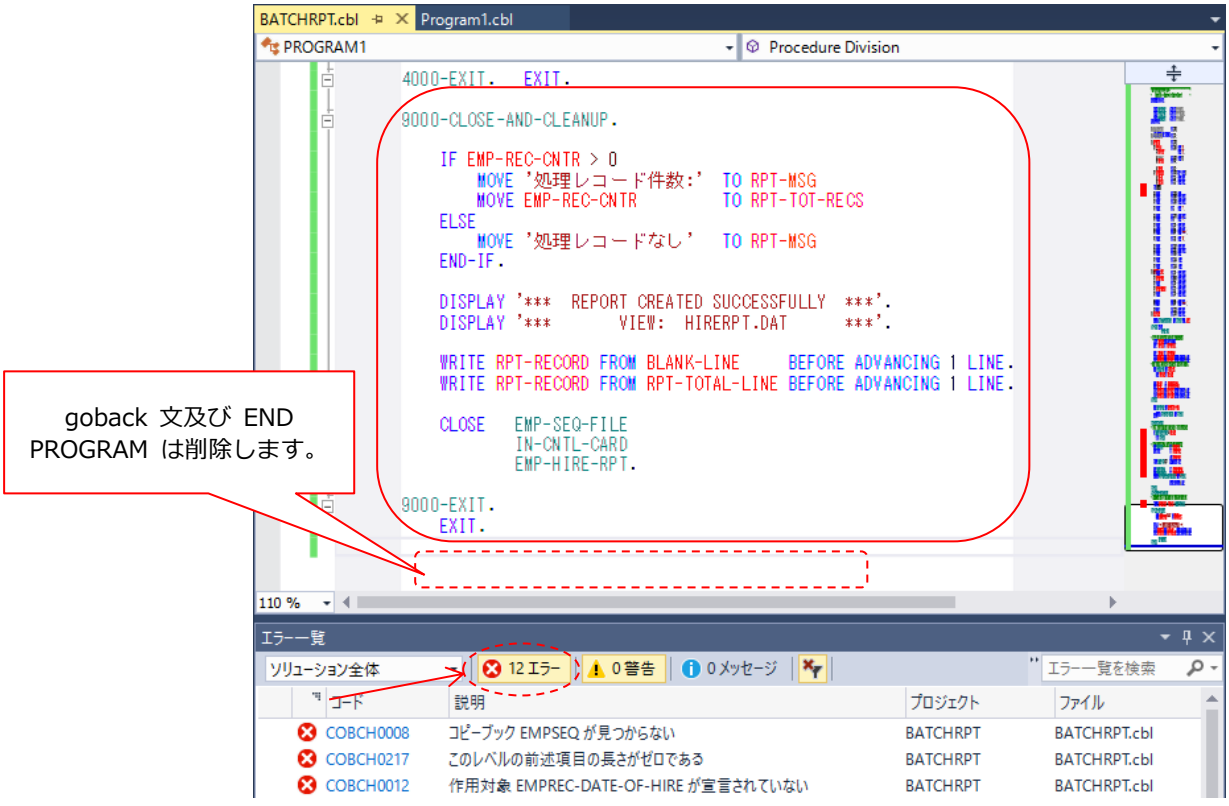
手続き部の 4000-COMPUTE-YEARS-EMPLOYED 段落を入力します。

```
 4000-COMPUTE-YEARS-EMPLOYED.
****
*   DETERMINE YEARS OF EMPLOYMENT BY SUBTRACTING HIRE YEAR
*   FROM CURRENT YEAR.
****

     COMPUTE YRS-EMPLOYED = CURR-YYYY - EMPREC-DOH-YYYY.

 4000-EXIT.    EXIT.
```



手続き部の 9000-CLOSE-AND-CLEANUP 段落を入力します。

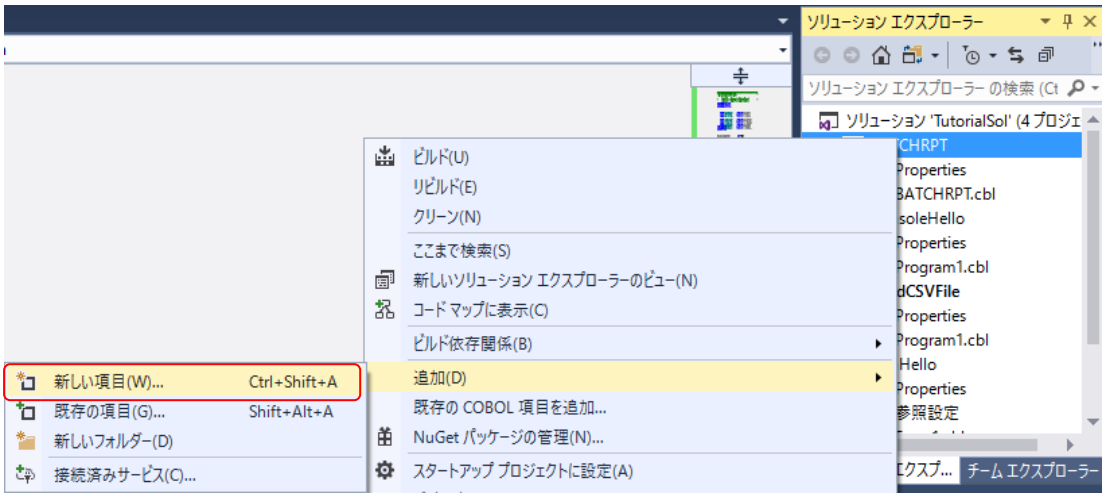
```
9000-CLOSE-AND-CLEANUP.

    IF EMP-REC-CNTR > 0
        MOVE '処理レコード件数:'  TO RPT-MSG
        MOVE EMP-REC-CNTR         TO RPT-TOT-RECS
    ELSE
        MOVE '処理レコードなし'    TO RPT-MSG
    END-IF.

    DISPLAY '***  REPORT CREATED SUCCESSFULLY  ***'.
    DISPLAY '***       VIEW:  HIRERPT.DAT      ***'.

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

    CLOSE   EMP-SEQ-FILE
            IN-CNTL-CARD
            EMP-HIRE-RPT.

9000-EXIT.
    EXIT.
```



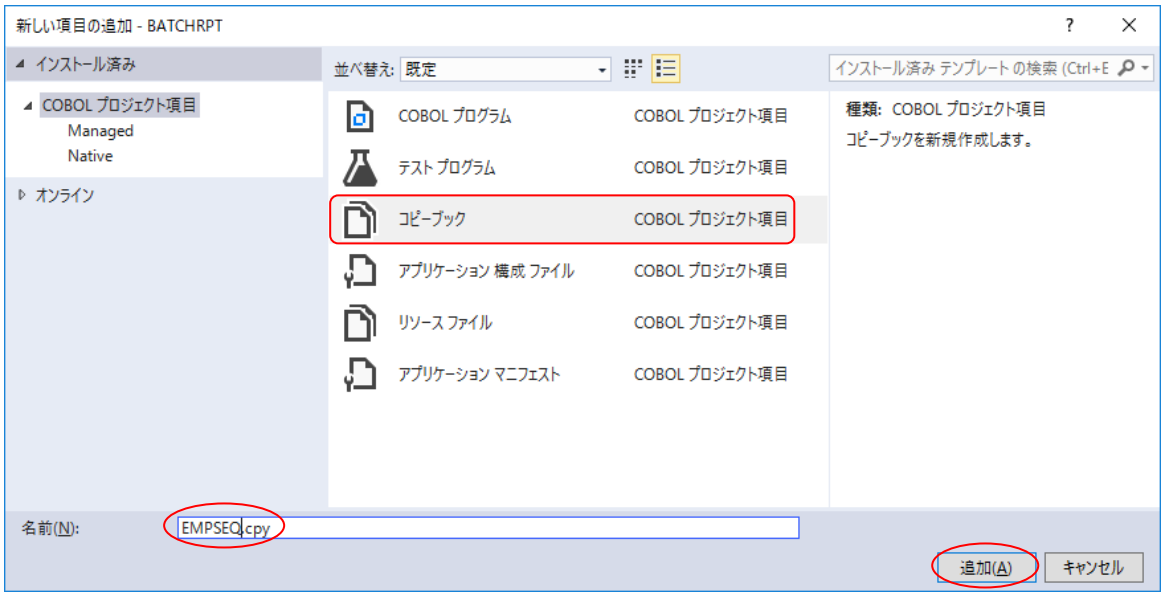
以上で BATCHRPT.cbl ソースプログラムの入力は終了です。 ここでエラーが 12 件であれば、先に進んでください。

3　コードエディターで COBOL コピーファイルを入力します。

ソリューションエクスプローラーでプロジェクト「BATCHRPT」を右クリックして **追加(D)、新しい項目(W)** を選択します。

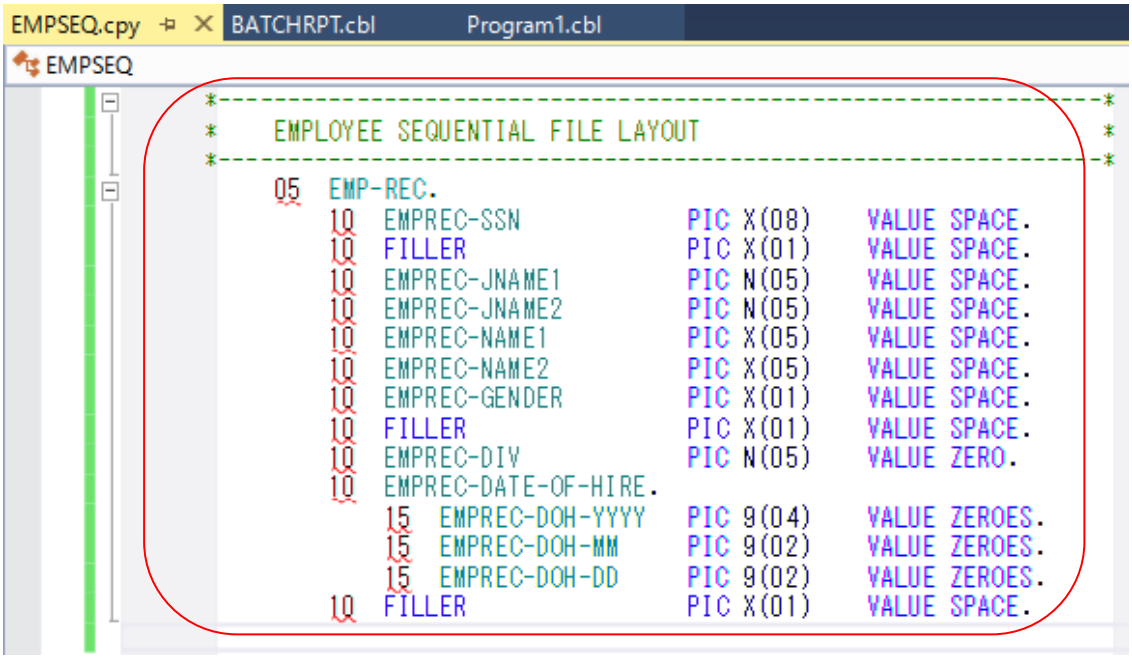


インストールされたテンプレートの一覧から **COBOL プロジェクト項目**、**コピーブック**を選択します。 名前(N)に **EMPSEQ.cpy** と入力し、**追加(A)** をクリックします。

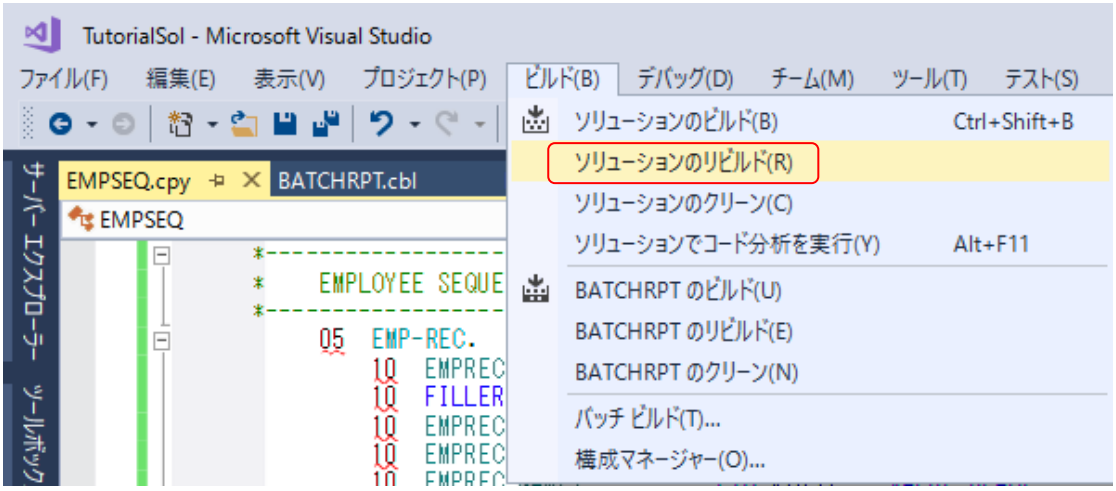


EMPSEQ.cpy へ EMP-RECORD-IO-AREA データ項目のレコード記述を入力します。

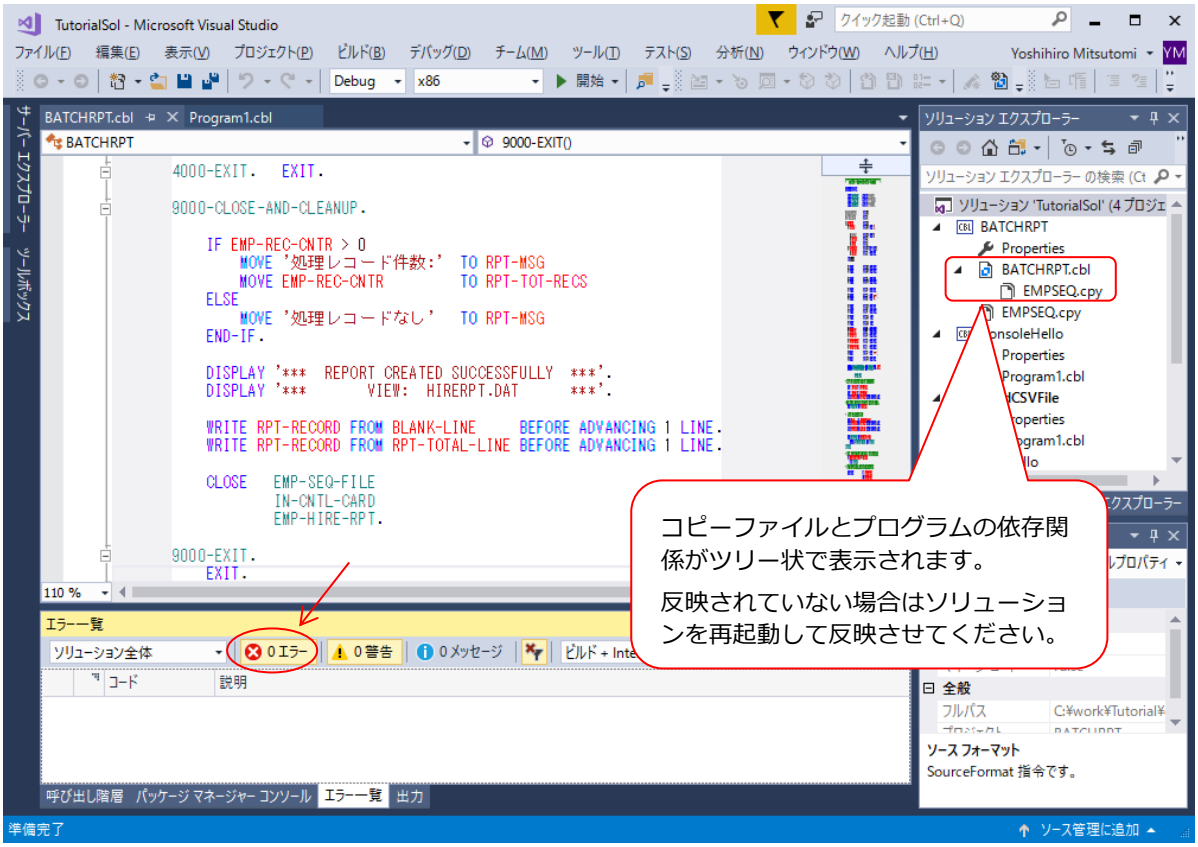
```
*-----------------------------------------------------------------*
*    EMPLOYEE SEQUENTIAL FILE LAYOUT                              *
*-----------------------------------------------------------------*
     05  EMP-REC.
         10  EMPREC-SSN             PIC X(08)    VALUE SPACE.
         10  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.
```



ビルド(B) メニューから **ソリューションのリビルド(R)** を選択し、一度コンパイルします。

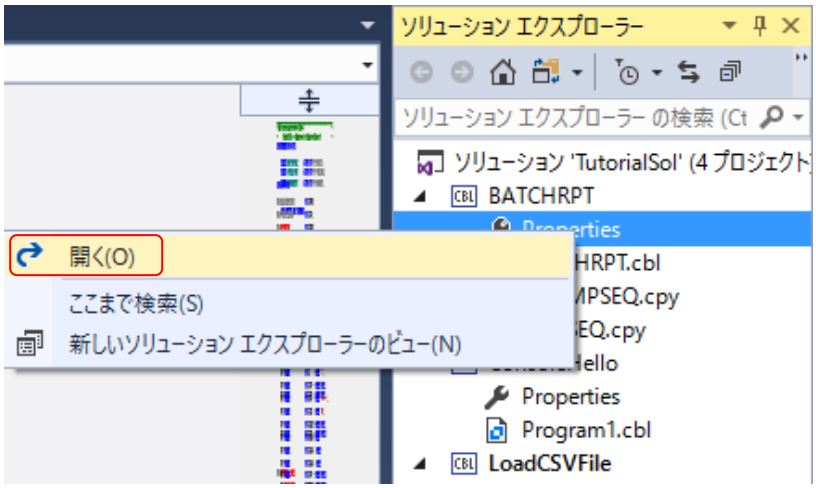


エディター画面の **BATCHRPT.cbl [コード]**タブをクリックして、**表示(V)**メニューから **エラー一覧(I)** を選択します。 エラーが 0 件であることを確認して、次に進んでください。

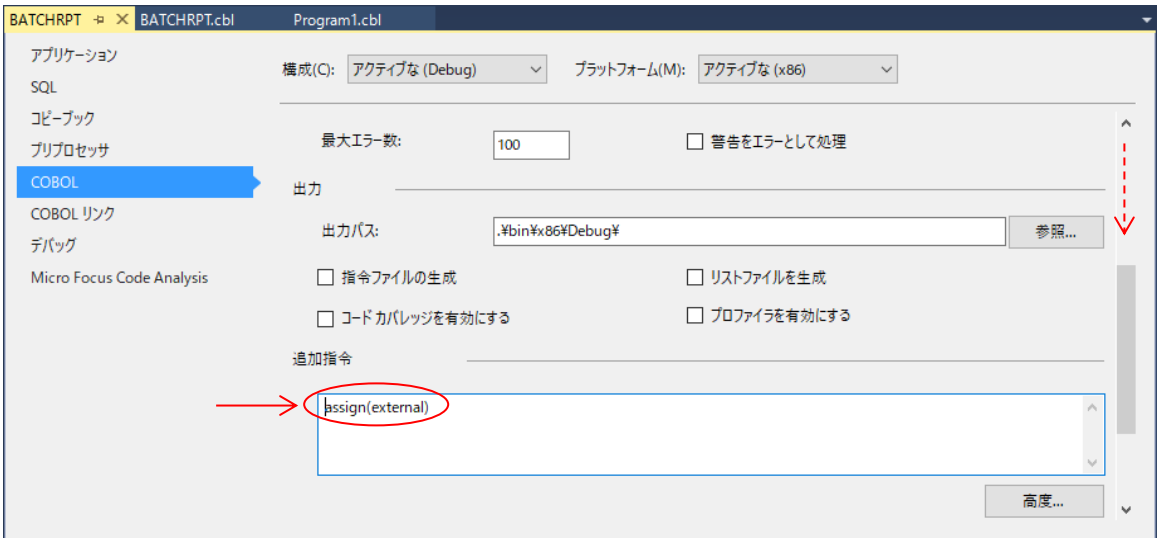


4　COBOL コンパイル指令を追加します。

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「BATCHPRT」プロジェクト配下の **Properties** を右クリックし **開く(O)** を選択します。

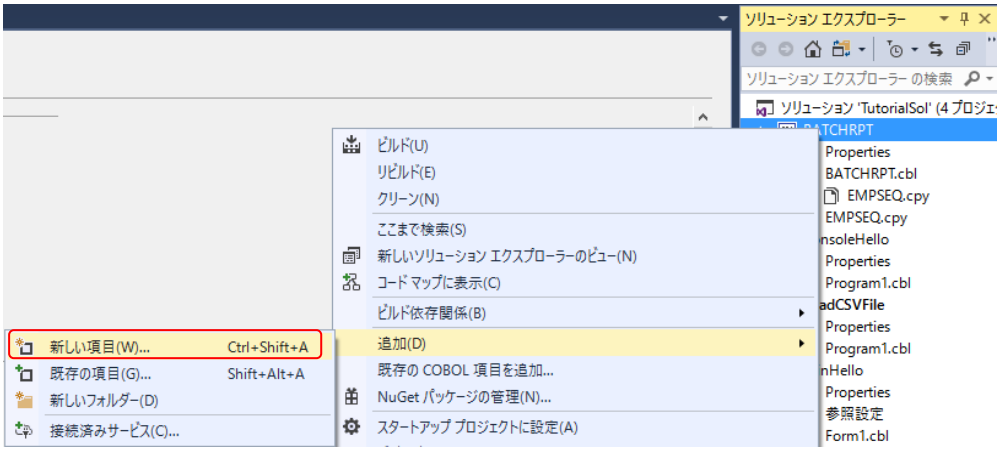


COBOL タブを選択し 追加指令に **assign(external)** を入力し、プロパティファイルを保存します。

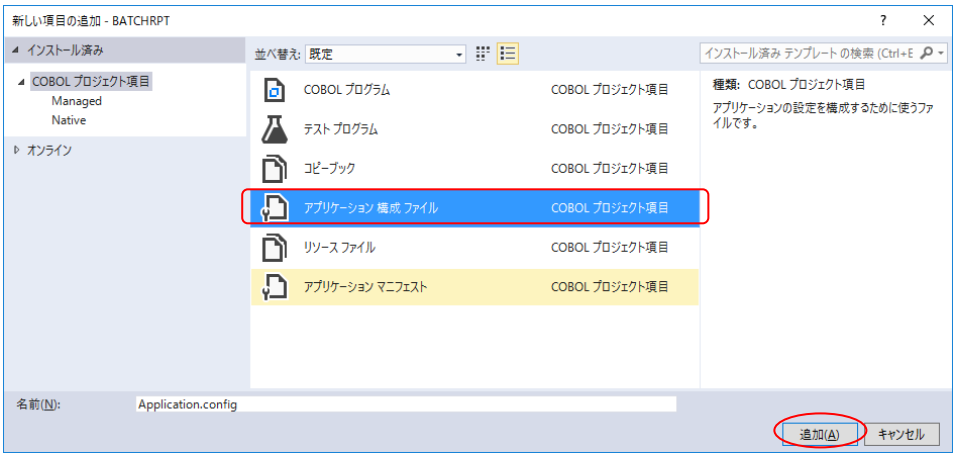


5　アプリケーション構成ファイルを作成します。

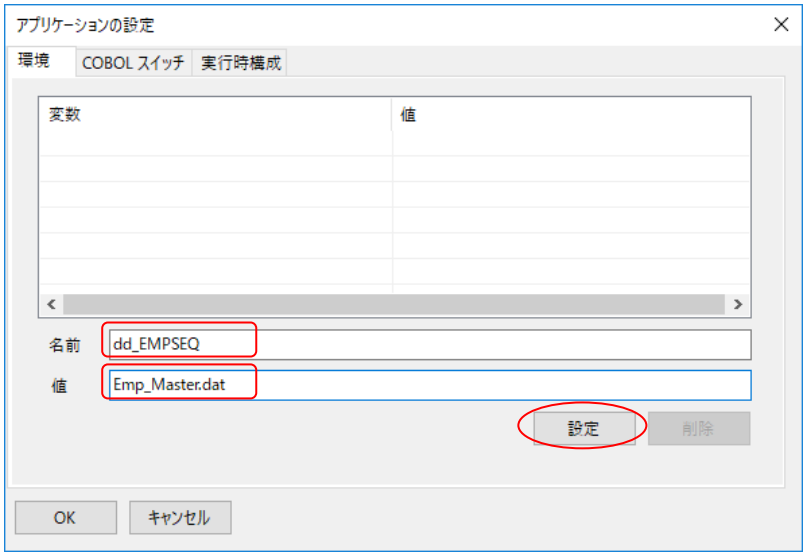
ソリューションエクスプローラーでプロジェクト「BATCHRPT」を右クリックして **追加(D)、新しい項目(W)** を選択します。



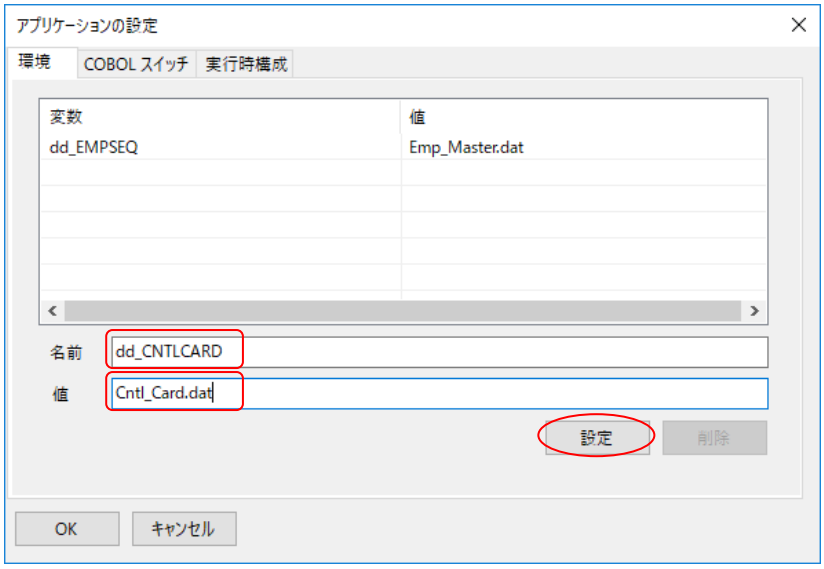
インストールされたテンプレートの一覧から **COBOL プロジェクト項目**、**アプリケーション構成ファイル**を選択し、**追加(A)** をクリックします。ファイル名はデフォルトのままで構いません。



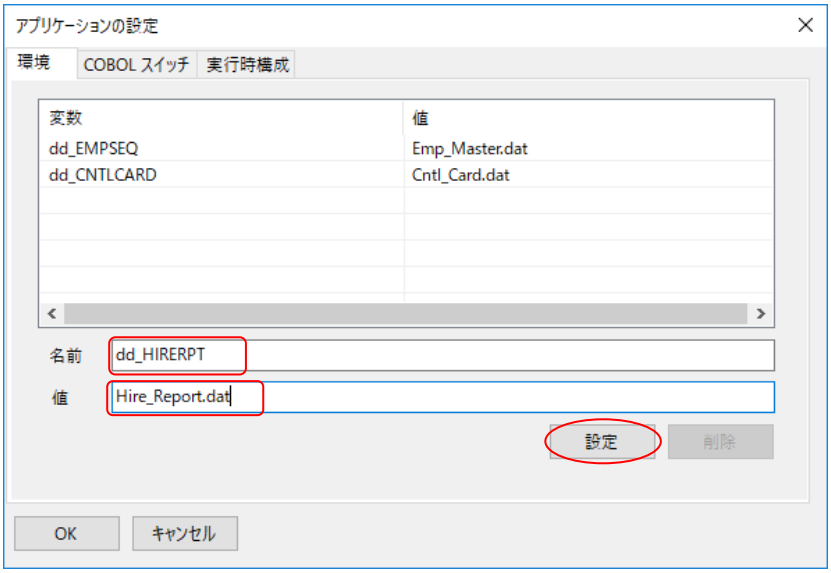
生成されたファイルをダブルクリックします。**アプリケーションの設定**で名前に **dd_EMPSEQ**、値に **Emp_Master.dat** を入力し、**設定**をクリックします。



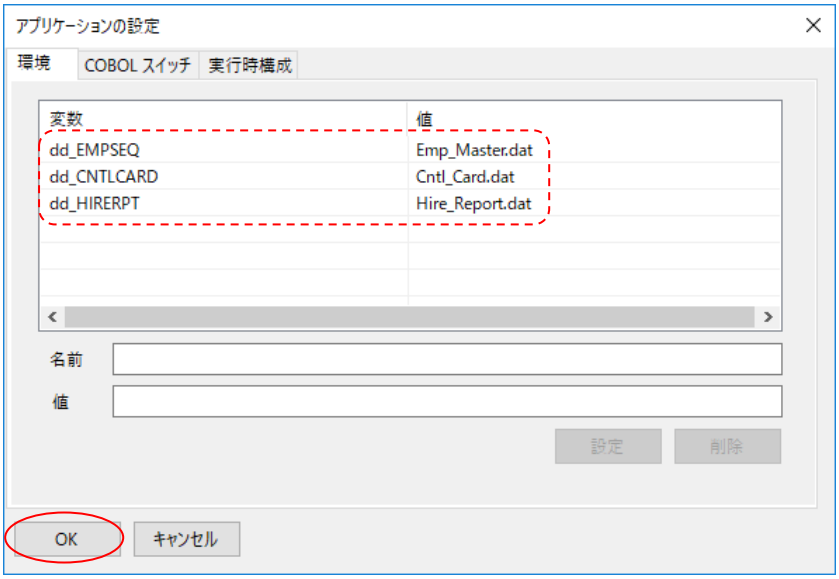
アプリケーションの設定で名前に **dd_CNTLCARD**、値に **Cntl_Card.dat** を入力し、**設定**をクリックします。



アプリケーションの設定で名前に **dd_HIRERPT**、値に **Hire_Report.dat** を入力し、**設定**をクリックします。

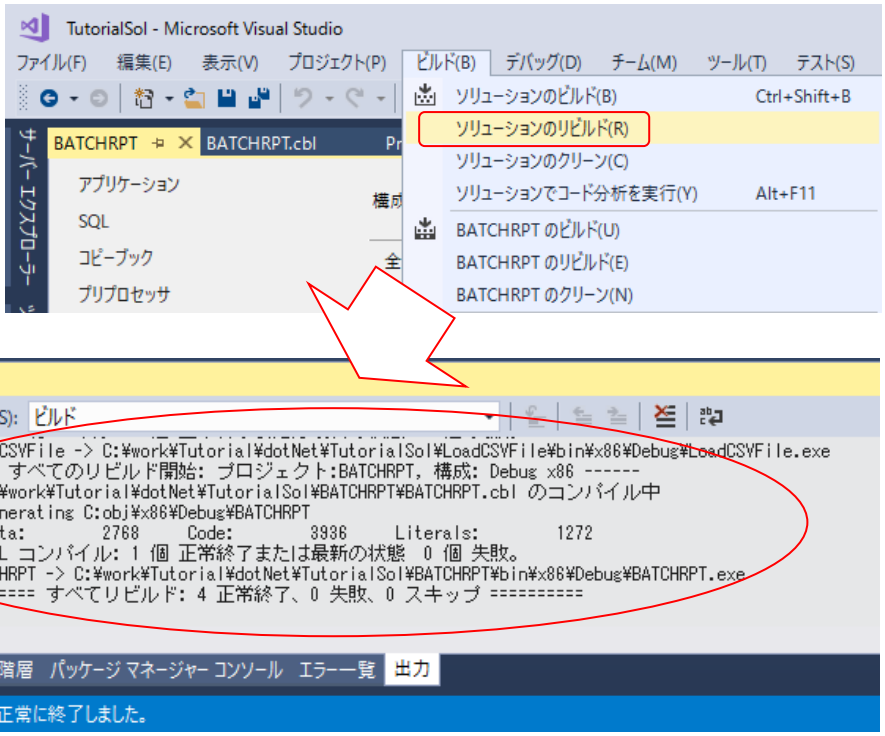


アプリケーションの設定で **OK** をクリックします。



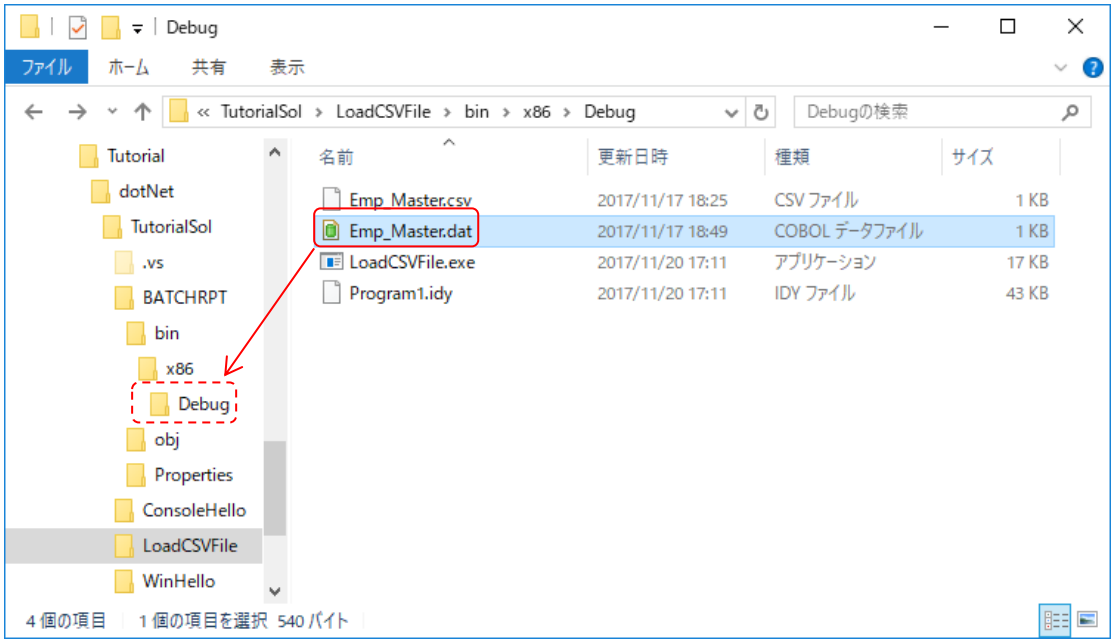
6　COBOL アプリケーションをビルドします。

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、**ビルド(B)**メニューから **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。



7　入力ファイルをコピーします。

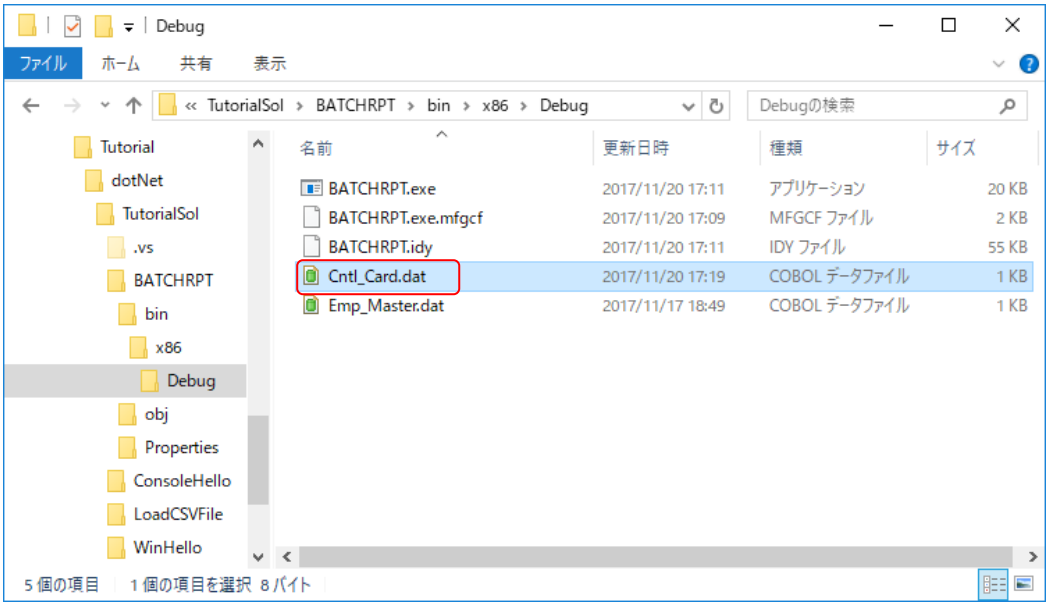
第５章４で作成した **Emp_Master.dat** ファイルをデバッグフォルダ(＜ソリューションが格納されたフォルダ＞¥BATCHRPT¥BATCHRPT¥bin¥x86¥debug)にコピーします。

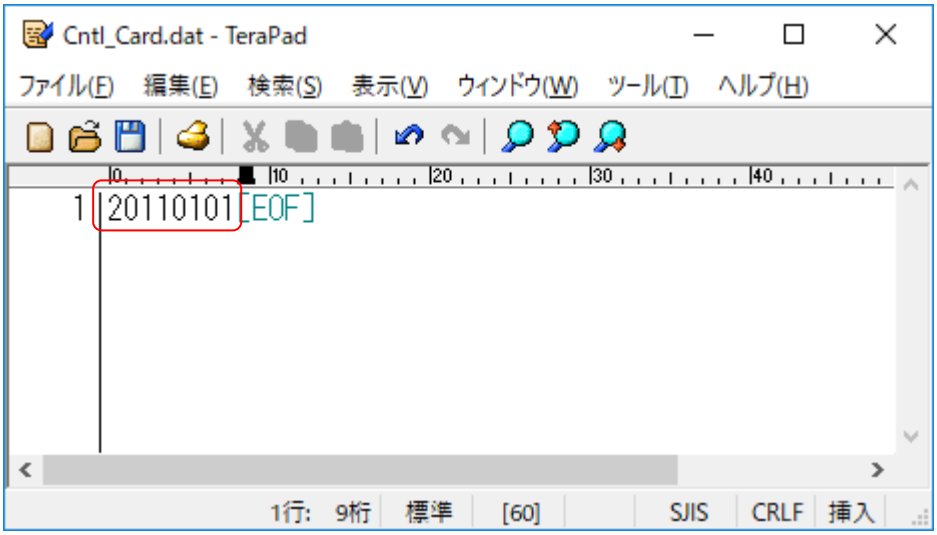


8　制御ファイルを作成します。

デバッグフォルダ(<ソリューションが格納されたフォルダ> ¥BATCHRPT¥BATCHRPT¥bin¥x86¥debug)にメモ帳などを利用して以下のデータが記述された **Cntl_Card.dat** ファイルを作成します。

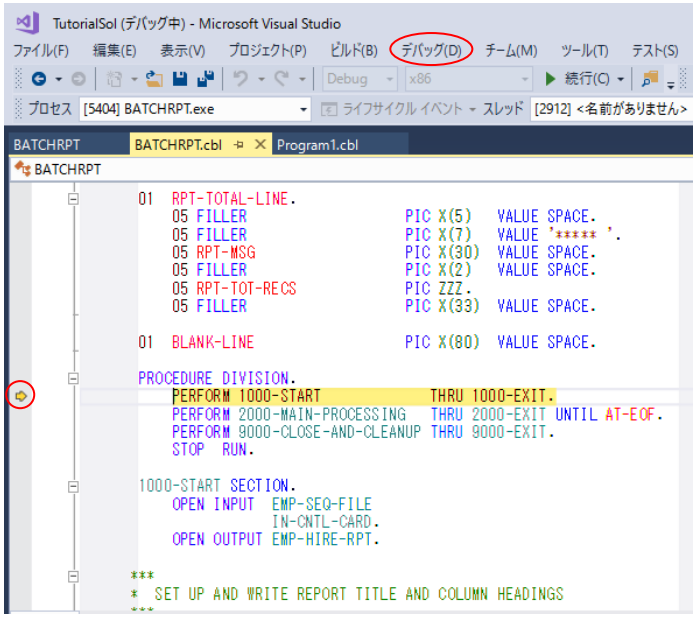
20110101



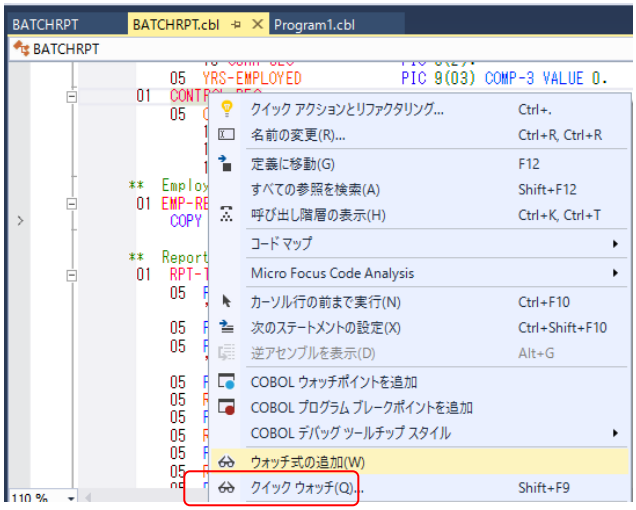


9　COBOL アプリケーションをデバッグ実行します。

ソリューションエクスプローラーにて **BATCHRPT** を右クリックから **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、**デバッグ(D)**メニューから **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。

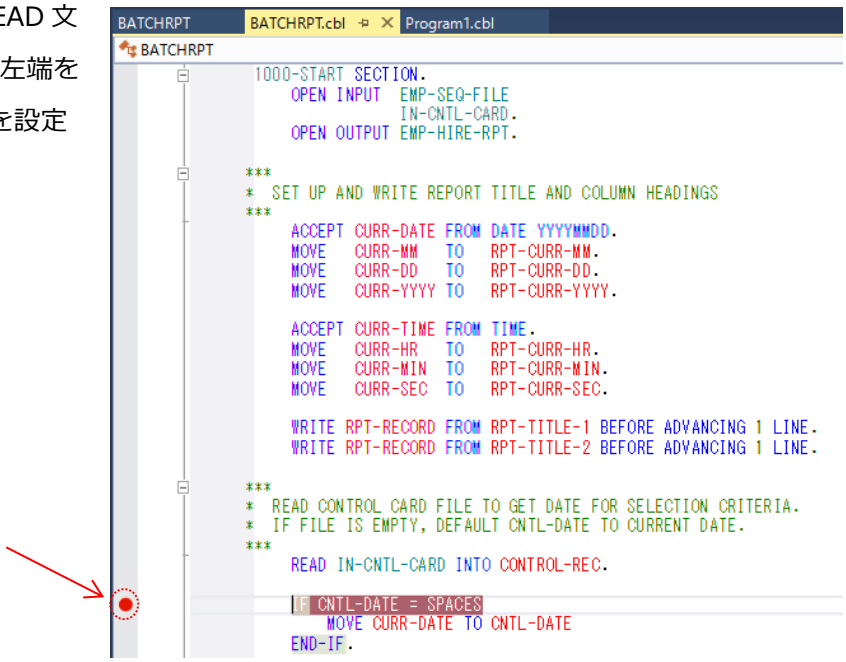


制御ファイルから読み込んだレコードの内容を確認するため、データ部の **CONTROL-REC** 上で右クリックして **ウォッチ式の追加(W)** を選択します。

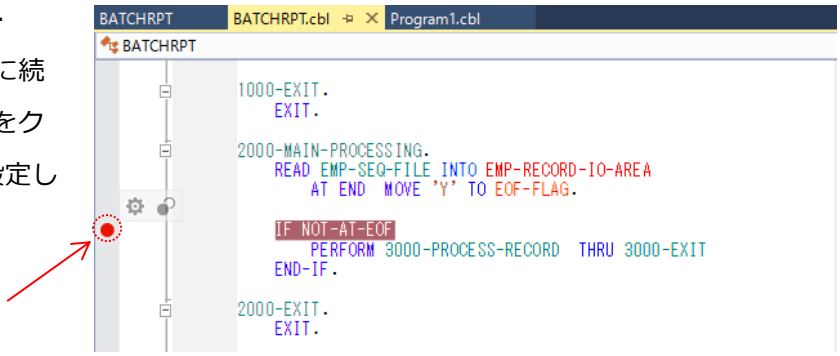


同様に入力ファイルから読み込んだレコードの内容を確認するため、データ部の **EMP-RECORD-IO-AREA** 上で右クリックして **ウォッチ式の追加(W)** を選択します。

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。

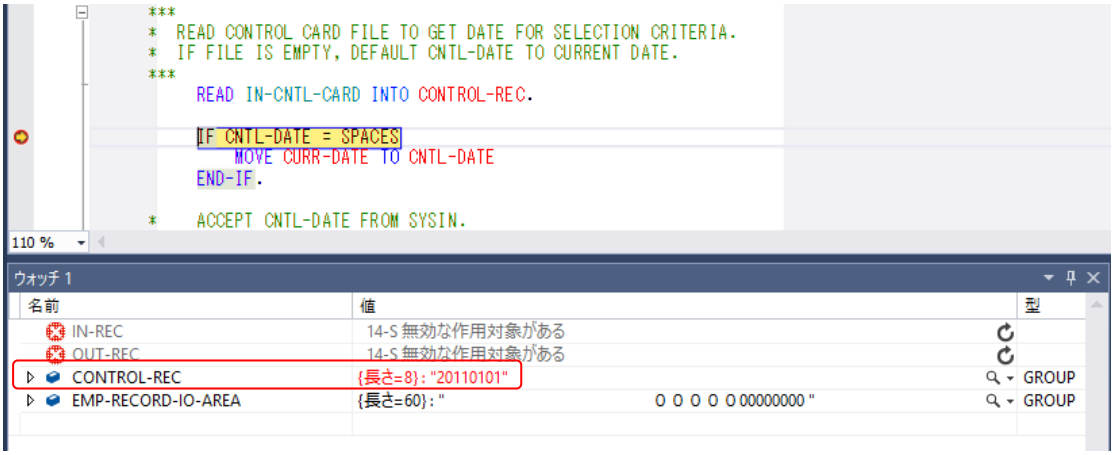


同様に手続き部 **2000-MAIN-PROCESSING 段落**の READ 文に続く IF 文でエディター画面の左端をクリックし、ブレークポイントを設定します。



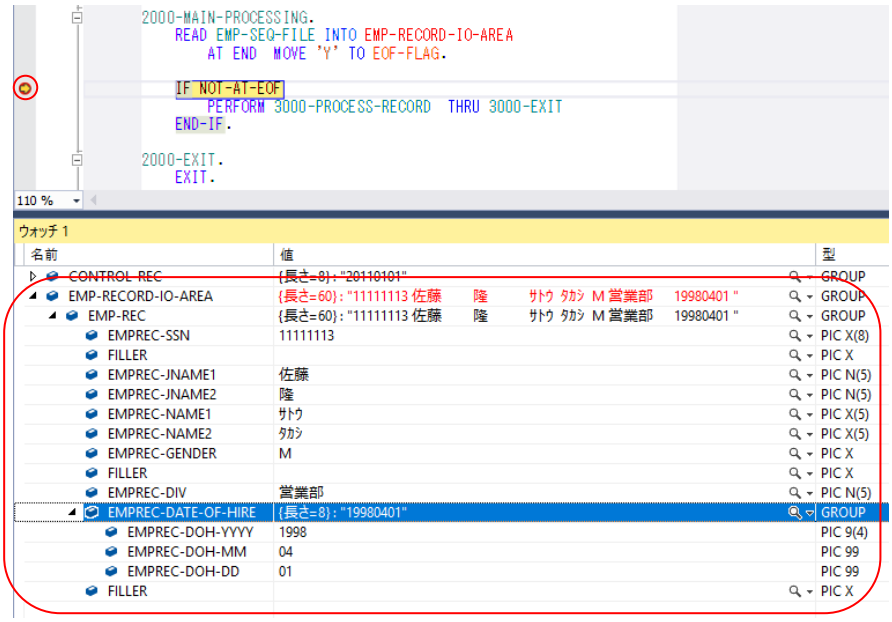
デバッグ(D)メニューから **続行(C)** を選択するか **F5** キーを押すと、デバッガーは最初のブレークポイントで実行を中断します。

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。



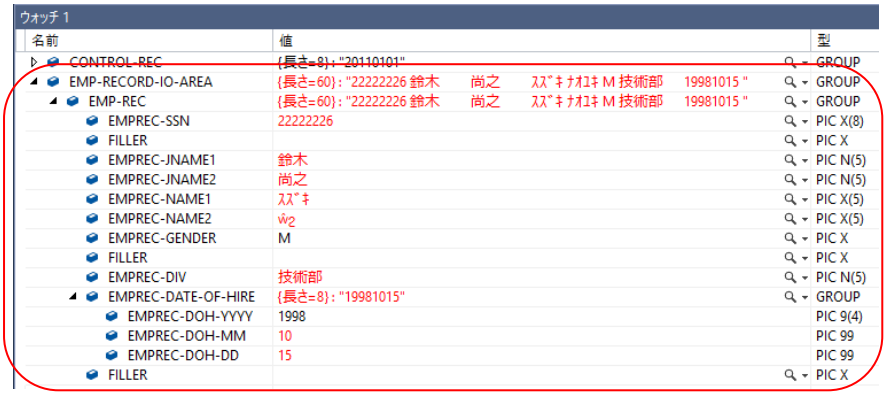
デバッグ(D)メニューから **続行(C)** を選択するか **F5** キーを押すと、デバッガーは 2 番目のブレークポイントで実行を中断します。

ウォッチ式の 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。



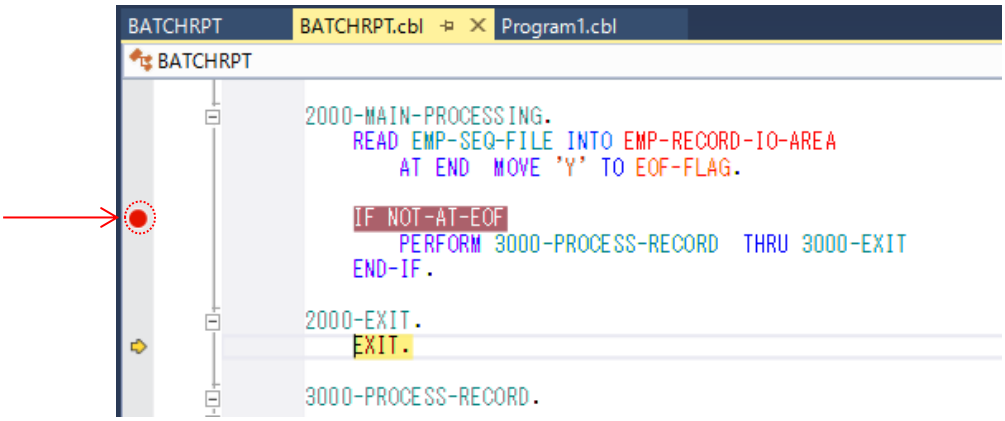
同様に **デバッグ(D)**メニューから **続行(C)** を選択するか **F5** キーを押すと、デバッガーは 2 番目のブレークポイントで実行を中断します。

ウォッチ式の EMP-RECORD-IO-AREA の値に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。

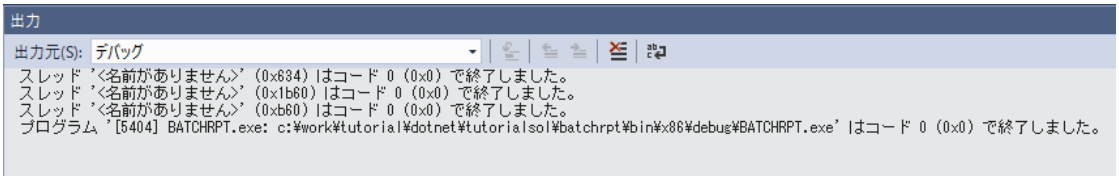


さらに **F5** キーを 8 回、 **F11** キーを 1 回押すと、デバッガーは 2 番目のブレークポイントに続く EXIT 文で実行を中断します。

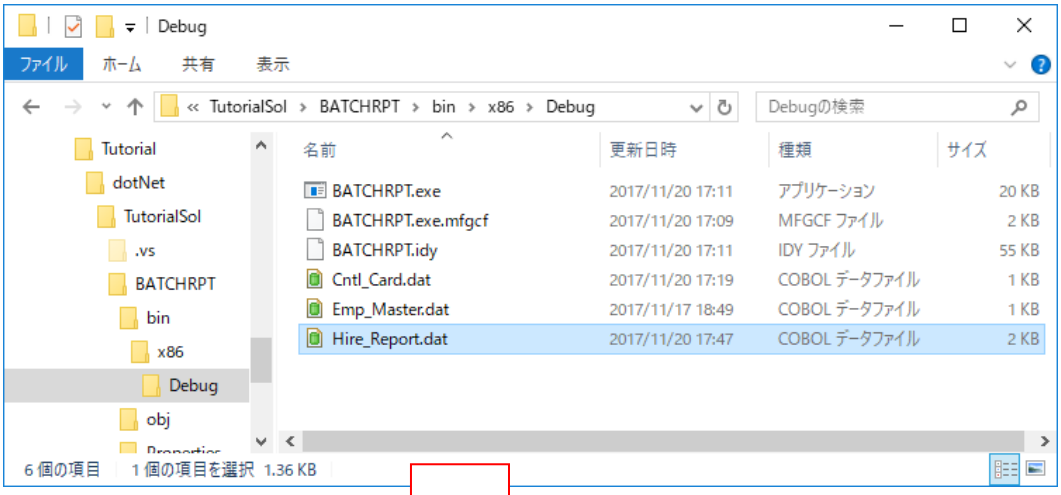
IF 文の条件式は、入力ファイルがファイル終了状態であることを示しています。

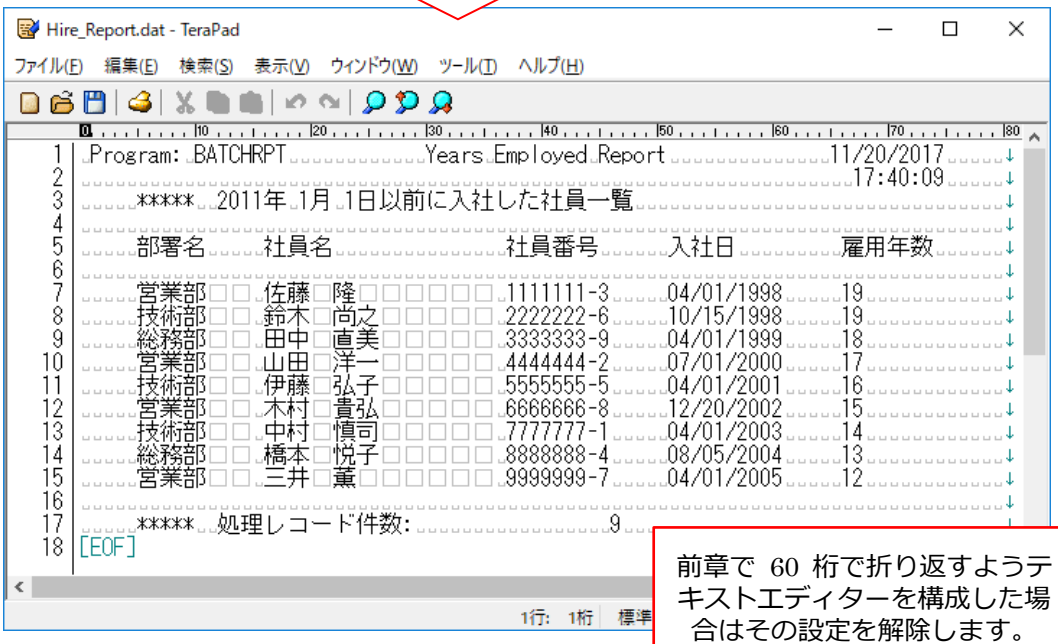


デバッグ(D)メニューから **続行(C)** を選択するか STOP 文を実行するまで **F11** キーを押すと、デバッガーは終了します。



デバッグフォルダ(<ソリューションが格納されたフォルダ> ¥BATCHRPT¥BATCHRPT¥bin¥x86¥debug)に **Hire_Report.dat** ファイルが作成されるので、メモ帳などテキスト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。

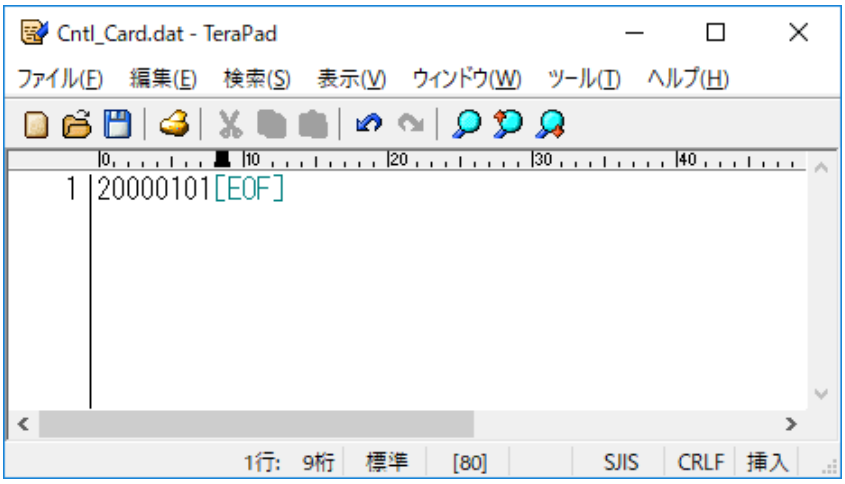




前章で 60 桁で折り返すようテキストエディターを構成した場合はその設定を解除します。

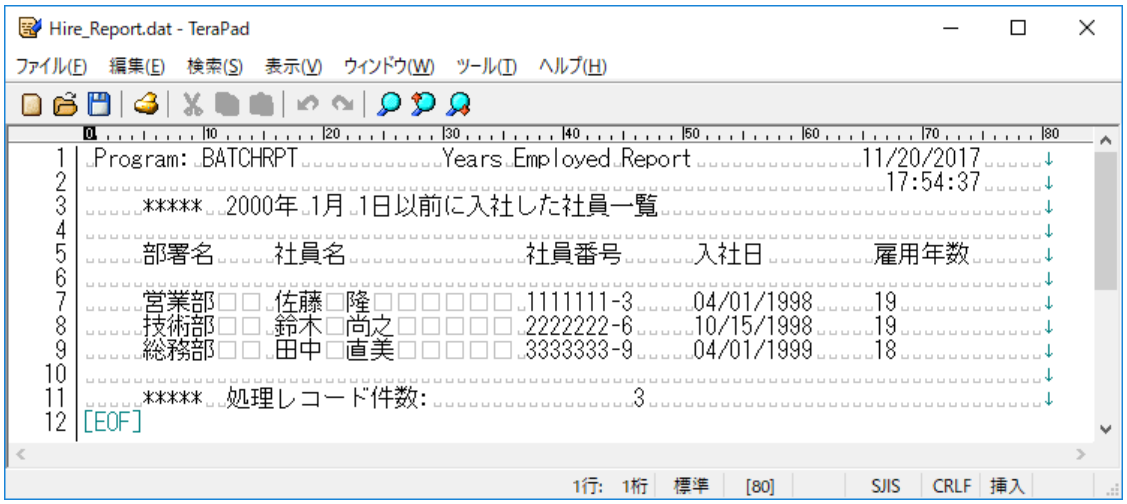
デバッグフォルダ(<ソリューションが格納されたフォルダ> ¥BATCHRPT¥BATCHRPT¥bin¥x86¥debug)の **Cntl_Card.dat** ファイルを以下の値に更新します。

20000101



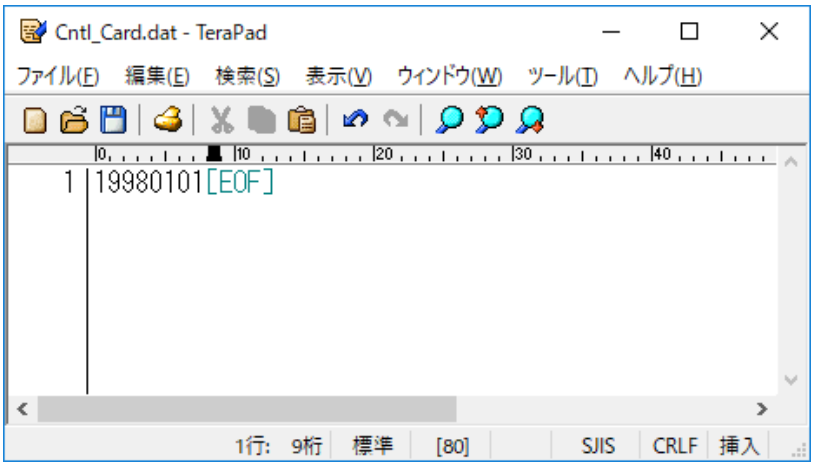
デバッグ(D)メニューから **デバッグなしで開始(H)** を選択するか **Ctrl+F5** キーを押すと、コマンドプロンプト画面が開くので、任意のキーを押してアプリケーションを実行します。

デバッグフォルダ(<ソリューションが格納されたフォルダ> ¥BATCHRPT¥BATCHRPT¥bin¥x86¥debug)の **Hire_Report.dat** ファイルを開いて、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。



デバッグフォルダ(<ソリューションが格納されたフォルダ> ¥BATCHRPT¥BATCHRPT¥bin¥x86¥debug)の **Cntl_Card.dat** ファイルを以下の値に更新します。

19980101



デバッグ(D)メニューから **デバッグなしで開始(H)** を選択するか **Ctrl+F5** キーを押すと、コマンドプロンプト画面が開くので、任意のキーを押してアプリケーションを実行します。

デバッグフォルダ(<ソリューションが格納されたフォルダ> ¥BATCHRPT¥BATCHRPT¥bin¥x86¥debug)の **Hire_Report.dat** ファイルを開いて、処理レコードなしが表示されることを確認します。

2017 年 11 月 21 日　　第 4 版

マイクロフォーカス株式会社

〒106-0032 東京都港区六本木 7-18-18
住友不動産六本木通ビル 9F

電話　03-5413-4800
URL　http://www.microfocus.co.jp/