

---

## Micro Focus Visual COBOL チュートリアル

---

### COBOL 開発 : Visual Studio – ネイティブ COBOL の単体テスト

#### 1. 目的

本チュートリアルでは、ネイティブ COBOL プログラムに対するテスト作成、実行方法、および、テスト結果を表示させる方法の習得を目的としています。

MFUnit は、Visual COBOL に搭載された xUnit 系の単体テストフレームワークです。xUnit はオブジェクト指向型の単体テストフレームワーク SUnit に起源を持つ JUnit や RUnit 等の単体テストフレームワークの総称です。MFUnit は xUnit の設計アーキテクチャーや仕組みは取り入れつつも COBOL 開発者にとって扱いやすい手続き型の COBOL を対象とした単体テストフレームワークという設計思想の下、開発されました。

MFUnit は COBOL 開発作業に以下の利点を提供します。

- テストを繰り返し実行させることができるため、修正作業時などのテスト工数の削減が見込める
- Jenkins などの継続的インテグレーション (Continuous Integration) ツールと連携によりテストの自動化が行え、DevOps サイクルの導入が足がかりを作れる

#### 2. 前提

- 本チュートリアルで使用したマシン OS : Windows Server 2019
- Micro Focus Visual COBOL 7.0 for Visual Studio がインストール済みであること

本資料では、Visual COBOL 7.0 for Visual Studio 2019 を使用しています。開発環境については、ホームページをご覧ください。また、ネイティブ COBOL に対する単体テストフレームワークの利用方法を記載したチュートリアルです。.NET COBOL の単体テスト実現方法については、別チュートリアルを参照ください。

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

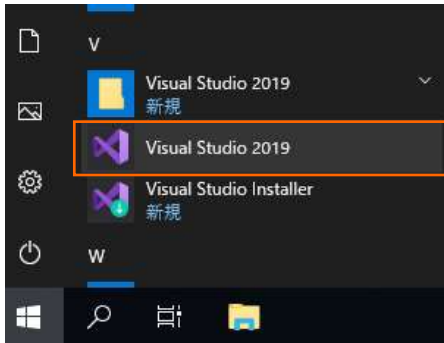
1. 目的
2. 前提
3. チュートリアル手順の概要
  - 3.1. IDE からの実行
    - 3.1.1. Visual Studio の起動
    - 3.1.2. チュートリアルプロジェクトの作成
    - 3.1.3. MFUnit テストの作成
    - 3.1.4. MFUnit テストの実行
  - 3.2. コマンドラインからの実行

### 3. チュートリアル手順の概要

#### 3.1. IDE からの実行

##### 3.1.1. Visual Studio の起動

- 1) スタートメニューより、Visual Studio 2019 を起動します。



##### 3.1.2. チュートリアルプロジェクトの作成

- 1) [新しいプロジェクトの作成] をクリックします。



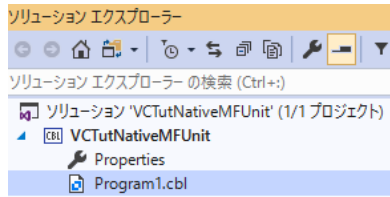
- 2) 言語に “COBOL”、プロジェクトタイプに “ネイティブ” を選択し、「コンソール アプリケーション」を選択した上で、[次へ(N)] ボタンをクリックします。



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

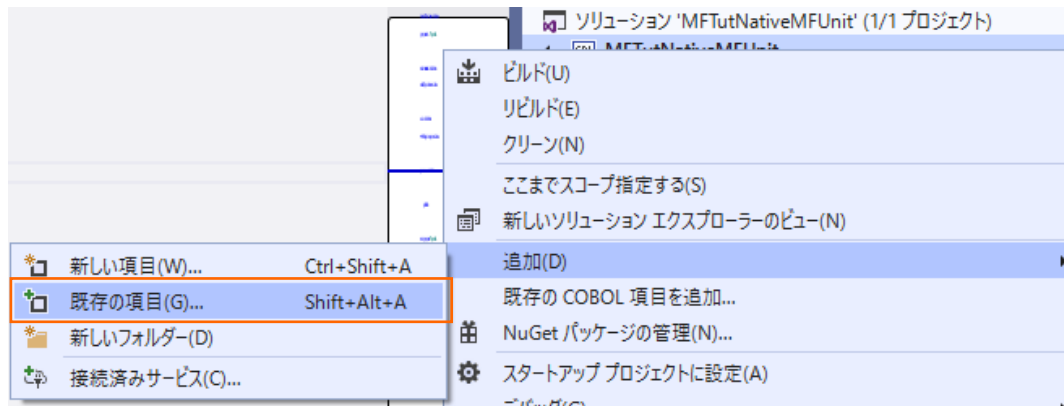


新規プロジェクトが作成されます。

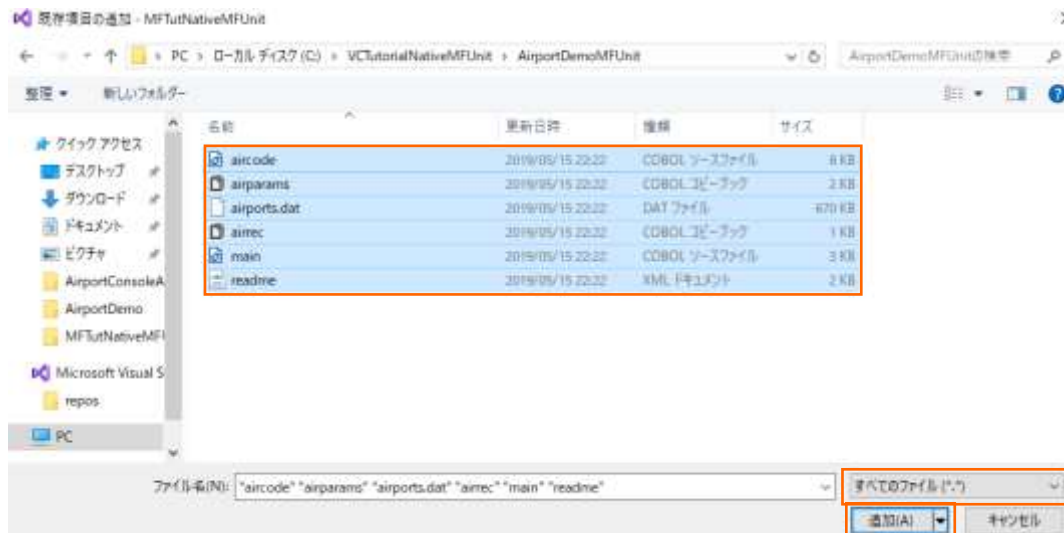


この Program1.cbl は不要なため、削除してください。

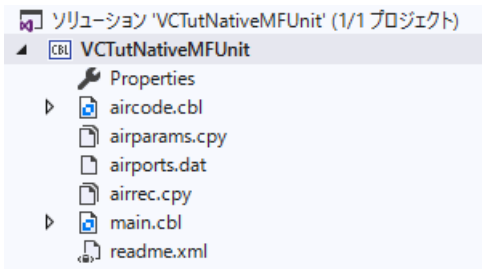
- 4) VCTutNativeMFUnit プロジェクト名を選択した状態で、マウスの右クリックにてコンテキストメニューを表示し、[追加(D)] > [既存の項目(G)] を選択します。



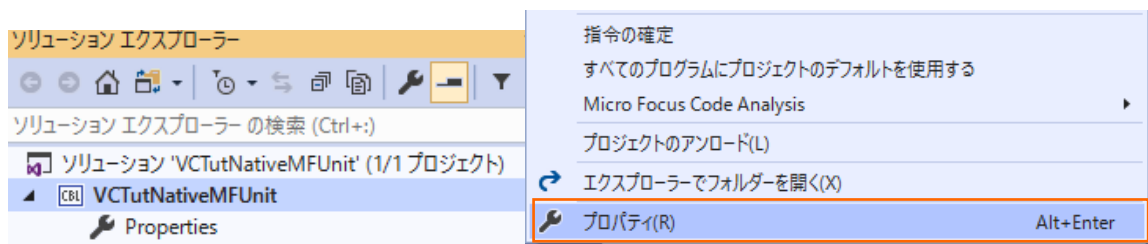
- 5) サンプルファイルを展開したフォルダ内の AirportDemoMFUnit フォルダ配下を選択し、“すべてのファイル(\*.\*)” を選択した結果、表示される全てのファイルを選択した上で、[追加(A)] ボタンをクリックします



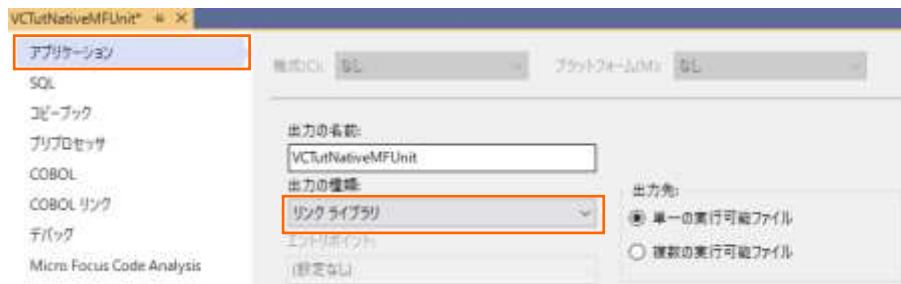
プロジェクトが、以下のようになります。



- 6) VCutNativeMfUnit 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。



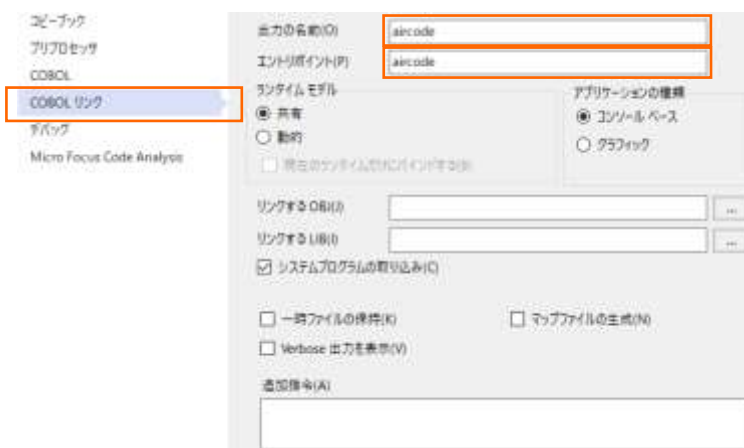
- 7) アプリケーションタブを選択し、「出力の種類」に「リンク ライブラリ」を選択し、保存します。



- 8) COBOL リンクタブを選択し、以下の入力を行った後、保存します。

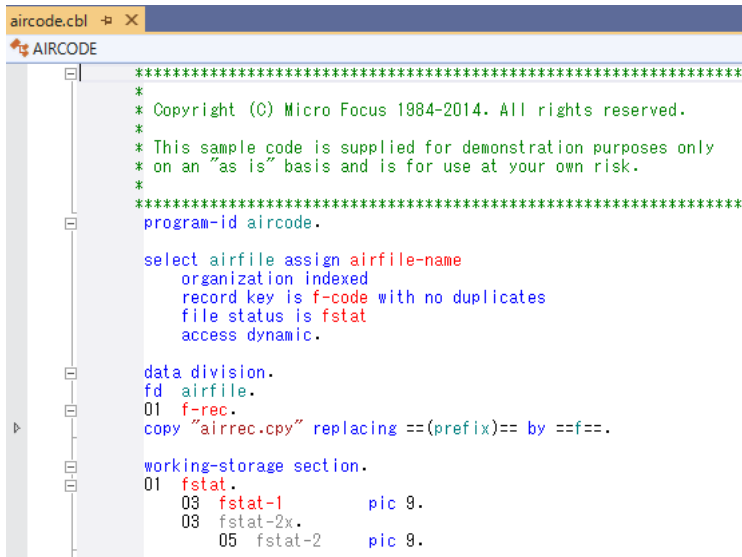
出力の名前： “aircode”

エントリーポイント： “aircode”



### 3.1.3. MFUnit テストの作成

- 1) ソリューションエクスプローラーより、aircode.cbl をダブルクリックして、エディターで開きます。



```

aircode.cbl
AIRCODE
*****
* Copyright (C) Micro Focus 1984-2014. All rights reserved.
*
* This sample code is supplied for demonstration purposes only
* on an "as is" basis and is for use at your own risk.
*
*****
program-id airdcode.

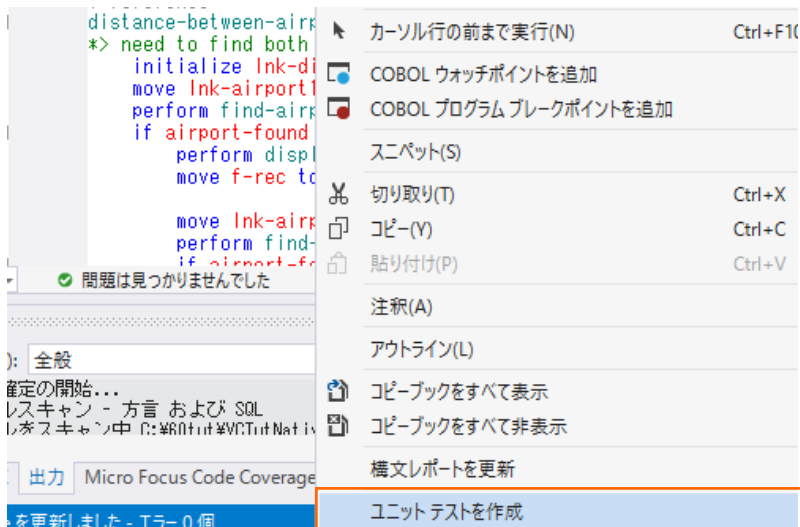
select airfile assign airfile-name
organization indexed
record key is f-code with no duplicates
file status is fstat
access dynamic.

data division.
fd airfile.
01 f-rec.
copy "airrec.cpy" replacing ==(prefix)== by ==f==.

working-storage section.
01 fstat.
03 fstat-1 pic 9.
03 fstat-2x.
05 fstat-2 pic 9.

```

- 2) エディター上にて、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、[ユニットテストを作成] を選択します。



- 3) [次へ >] ボタンをクリックします。

ユニットテストを作成 ×

テストプロジェクト: <新規のテストプロジェクト> ▾

新規のテストプロジェクト名: TestVCTutNativeMUnit

プロジェクトの場所: TestVCTutNativeMUnit

新規のテストプログラム名: TestAIRCODE

Test Type: Unit Test ▾

< 前へ 次へ > 完了 キャンセル

4) そのまま [完了] ボタンをクリックします。

ユニットテストを作成 ×

テストケースを生成するエントリーポイントを選択してください...

エントリーポイント名	テストケース名		Add New
AIRCODE ▾	TestAIRCODE	削除	

< 前へ 次へ > 完了 キャンセル

単体テストプログラムの雛型が作成されます。



```

TestAIRCODE.cbl  → ×  aircode.cbl
TESTAIRCODE
  entry MFU-TC-PREFIX & TEST-TESTAIRCODE.
    call "AIRCODE" using
      by value Ink-function
      by value Ink-airport1
      by value Ink-airport2
      by value Ink-prefix-text
      by reference Ink-rec
      by reference Ink-distance-result
      by reference Ink-matched-codes-array

    *> Verify the outputs here
    goback returning MFU-PASS-RETURN-CODE
  .
  $region TestCase Configuration
  entry MFU-TC-SETUP-PREFIX & TEST-TESTAIRCODE.
  perform InitializeLinkageData
  *> Add any other test setup code here
  goback returning 0
  .
  InitializeLinkageData section.
  *> Load the library that is being tested
  set pp to entry "VCTutNativeMFUnit"

  initialize Ink-function

```

5) 新規のテストケース（羽田・ロンドンヒースロー空間間の距離 (km) のテスト）を追加した上で、実行を行いません。サンプルファイルを展開したフォルダ内の TestAIRCODE.cbl で、現在の TestAIRCODE.cbl を上書き保存します。これは、テストケース “testDistance” を途中まで作成したものになります。プログラムを確認すると、MFU-TC-SETUP-PREFIX, MFU-TC-PREFIX, MFU-TC-TEARDOWN-PREFIX から始まる “testDistance” の 3 entry が定義されていることが分かります。MUnit では、テストを下記のように決められた手順で実行しています。

- ① entry MFU-TC-SETUP-PREFIX & “testDistance”
- ② entry MFU-TC-PREFIX & “testDistance”
- ③ entry MFU-TC-TEARDOWN-PREFIX & “testDistance”

MFU-TC-SETUP-PREFIX で始まる entry にて、テストの前処理を定義できます。前処理の代表例としては、ファイルをあらかじめオープンしておくなどが考えられます。一方、MFU-TC-TEARDOWN-PREFIX で始まる entry では、テスト実行後の処理を定義できます。前処理でオープンしたファイルをクローズするような処理が該当します。前処理、後処理ともに省略可能です。

テスト本体である MFU-TC-PREFIX を確認すると、下記のように結果検証コードが実装されていません。

```
entry MFU-TC-PREFIX & "testDistance"
  set get-distance to true
  move "HND" to lnk-airport1
  move "LHR" to lnk-airport2
  call "AIRCODE" using by value lnk-function
                    by value lnk-airport1
                    by value lnk-airport2
                    by value lnk-prefix-text
                    by reference lnk-rec
                    by reference lnk-distance-result
                    by reference lnk-matched-codes-array

  move 9591 to wk-distance-km
  display wk-distance-km " / " distance-km.

$region TestCase Configuration
entry MFU-TC-SETUP-PREFIX & "testDistance".
```

このテストを実装するため、以下のコードを 7 8 行目に挿入してください。

```
if wk-distance-km = distance-km
then
  goback returning MFU-PASS-RETURN-CODE
else
  string
    "expected "
    wk-distance-km
    ", but "
    distance-km
    z""
  into err-msg
end-string
call MFU-ASSERT-FAIL-Z using err-msg
end-if.
```



補足)

テスト失敗時の記述方法として、成功時同様に、戻り値で返す方法は、以下の通りです。

```
if wk-distance-km = distance-km
then
  goback returning MFU-PASS-RETURN-CODE
else
  string
    "expected "
    wk-distance-km
    ", but "
    distance-km
    into err-msg
  end-string
  display err-msg
  goback returning MFU-FAIL-RETURN-CODE
end-if.
```

戻り値 MFU-FAIL-RETURN-CODE を利用する場合、テスト結果を確認するためにエラー情報を、display などで出力する必要があります。

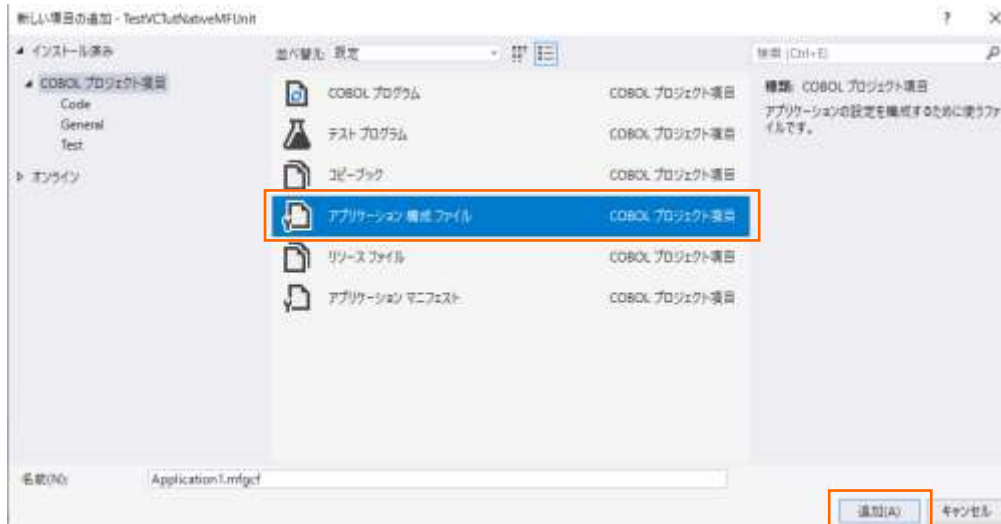
### 3.1.4. MJUnit テストの実行

本テスト対象のプログラムは、環境変数で設定された空港情報が保存されたデータファイルを参照するため、手順内で設定を行います。

- 1) TestVCTutNativeMJUnit プロジェクトを選択し、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)] > [新規の項目(W)] を選択します。

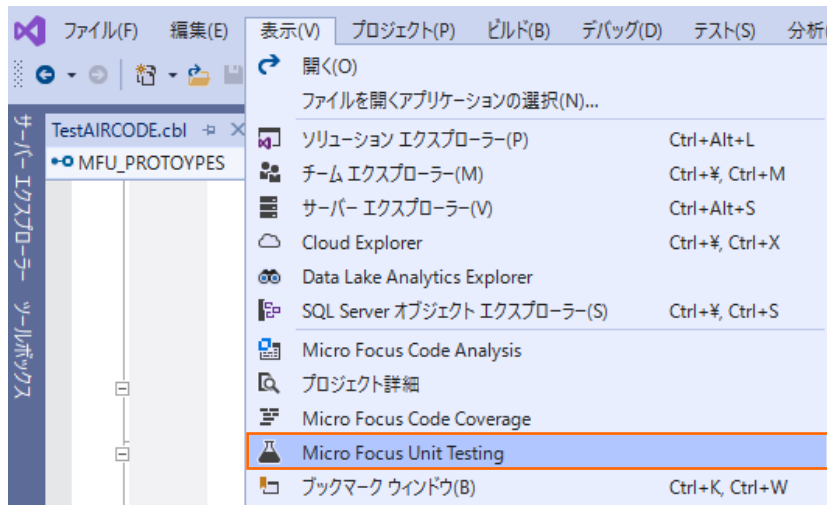


- 2) 「アプリケーション構成ファイル」を選択し、[追加(A)] ボタンをクリックします。





- 4) Visual Studio IDE メニューより、[表示(V)] > [Micro Focus Unit Testing] を選択します。



- 5) Micro Focus Unit Testing ビューより、[全て実行] ボタンをクリックします。



以下のように全て緑色のマークが設定されます。緑色は、テストに成功したことを示します。



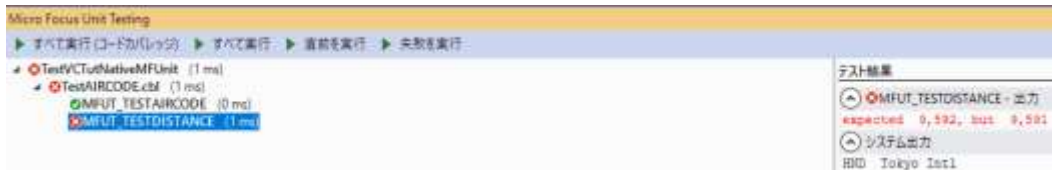
- 6) エラーケースを確認します。「TestAIRCODE.cbl」をエラーとなるように修正した上で、再度、Micro Focus Unit Testing ビューより、[全て実行] ボタンをクリックします。

なお、本例では、3.1.3 で作成したテストプログラム内に記載されていた期待値 9591 を 9592 に修正しています。

```
move 9592 to wk-distance-km
display wk-distance-km " / " distance-km.
if wk-distance-km = distance-km
then
goback returning MFU-PASS-RETURN-CODE
else
string
"expected "
wk-distance-km
", but "
distance-km
into err-msg
end-string
call MFU-ASSERT-FAIL-2 using err-msg
end-if.
```



MFUT\_TESTDISTANCE のテストで、エラーが発生したことが一覧から判断できます。



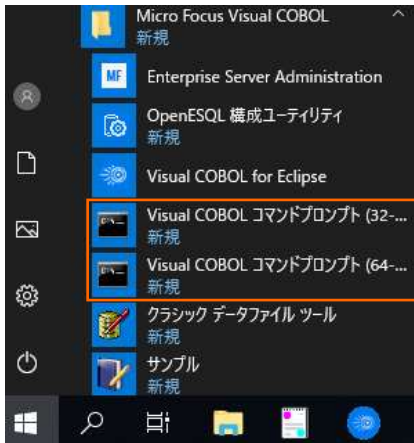


### 3.2. コマンドラインからの実行

MFUnit によるテストは、Visual Studio 上の画面からではなく、コマンドライン上からも行なうことができます。従来のスタイルでのテスト作業の効率化を図ることができ、Jenkins などの CI ツールと連携する事で、テストの自動実行を行なえるため、品質担保や作業工数の削減が見込めます。

ここでは、3.1.1 で作成したテストプログラムをコマンドラインから実行する方法について学びます。

- 1) スタートメニューより、実行環境に合わせた [Visual COBOL コマンドプロンプト] をクリックします。



- 1) 作業フォルダを作成し、作成したフォルダに移動します。

```
C:¥>mkdir VC50CommandTutorial
C:¥>cd VC50CommandTutorial
C:¥VC50CommandTutorial>
```

- 2) 3.1 で作成したソリューションが保存されているフォルダを VS\_SOLUTION\_PATH に指定した上で、下記コマンドを実行します。

- set VS\_SOLUTION\_PATH=c:¥vs\_solution\_path
- cobol %VS\_SOLUTION\_PATH%¥VCTutNativeMFUnit¥aircode.cbl;
- cobol %VS\_SOLUTION\_PATH%¥TestVCTutNativeMFUnit¥TestAIRCODE.cbl  
sourceformat(variable);
- cbllink -D aircode.obj
- cbllink -D TestAIRCODE.obj

注意)

VS\_SOLUTION\_PATH は、各環境に合わせて修正してください。

```
C:¥VC50CommandTutorial> set VS_SOLUTION_PATH=c:¥vs_solution_path
C:¥VC50CommandTutorial>cobol %VS_SOLUTION_PATH%¥VCTutNativeMFUnit¥aircode.cbl;
(コマンド実行中の出力内容を省略)
C:¥VC50CommandTutorial>cobol %VS_SOLUTION_PATH%¥TestVCTutNativeMFUnit¥TestAIRC
```

```
ODE.cbl sourceformat(variable);
(コマンド実行中の出力内容を省略)
C:¥VC50CommandTutorial>cbllink -D aircode.obj
(コマンド実行中の出力内容を省略)
C:¥VC50CommandTutorial>cbllink -D TestAIRCODE.obj
(コマンド実行中の出力内容を省略)
```

以下のファイルが作成されていることを確認します。

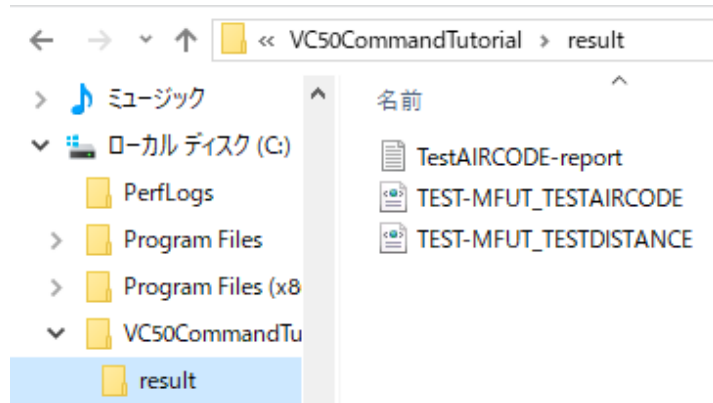
```
C:¥VC50CommandTutorial>dir
ドライブ C のボリューム ラベルがありません。
ボリューム シリアル番号は 969A-9F98 です
C:¥VC50CommandTutorial のディレクトリ
2019/06/27  11:33    <DIR>          .
2019/06/27  11:33    <DIR>          ..
2019/06/27  11:33                17,408 aircode.dll
2019/06/27  11:22                10,121 aircode.obj
2019/06/27  11:33                15,360 TestAIRCODE.dll
2019/06/27  11:32                 7,503 TestAIRCODE.obj
                4 個のファイル             50,392 バイト
                2 個のディレクトリ 37,157,203,968 バイトの空き領域
```

3) プロンプト上で下記コマンドを実行し、MJUnit を実行します。

- set dd\_airports=%VS\_SOLUTION\_PATH%¥VCTutNativeMJUnit¥airports.dat
- mfunrun -report:junit -outdir:result TestAIRCODE.dll

```
C:¥VC50CommandTutorial>set
dd_airports=%VS_SOLUTION_PATH%¥VCTutNativeMJUnit¥airports.dat
C:¥VC50CommandTutorial>mfunrun -report:junit -outdir:result TestAIRCODE.dll
Micro Focus COBOL - mfunrun Utility
Unit Testing Framework for Windows/Native/64
Fixture : TestAIRCODE
Test Run Summary
Overall Result      Passed
Tests run           2
Tests passed        2
Tests failed        0
Total execution time 0
```

outdir オプションにより、出力結果を result フォルダに保存されていることを確認してください。



## WHAT'S NEXT

- 本チュートリアルで学習した技術の詳細については製品マニュアルをご参照ください。

## 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、著作権法に基づき、適切な扱いを行ってください。