Visual COBOL チュートリアル

Eclipse – 静的コード解析

1 目的

本チュートリアルでは、静的コード解析ツールの利用方法の習得を目的としています。

静的コード解析ツールは、プログラムに内在する下記のような問題を検出することができます。

- 潜在バグになりえるようなコーディング
  未初期化変数の利用、動作が規定されない命令の利用
- 性能の観点上、非効率なコーディング
  ゾーン十進変数を使った算術演算、リトルエンディアン環境におけるビックエンディアン変数の使用
- ソースの可読性・保守性を低下させるようなコーディング
  デッドコード、GO TO 句の多用

一般的な開発では、コーディングルールの中で上記のような問題となる記述をしないように規約として定義します。Visual COBOL は、これらの規約をルールとして定義し、定義に基づいた問題点の検出を実施できます。Visual COBOL ではビルドインルールが利用できますが、 弊社別製品である Enterprise Analyzer と連携することで、独自のカスタムルールを追加することができるようになります。

なお、本機能は、ネイティブ COBOL のみに有効です。また、JVM COBOL プロジェクトに追加されたネイティブ COBOL プログラムには利用できません。

2 前提

- 本チュートリアルで使用したマシン OS ： Windows Server 10
- Visual COBOL 10.0 for Eclipse がインストール済みであること

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

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

内容

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

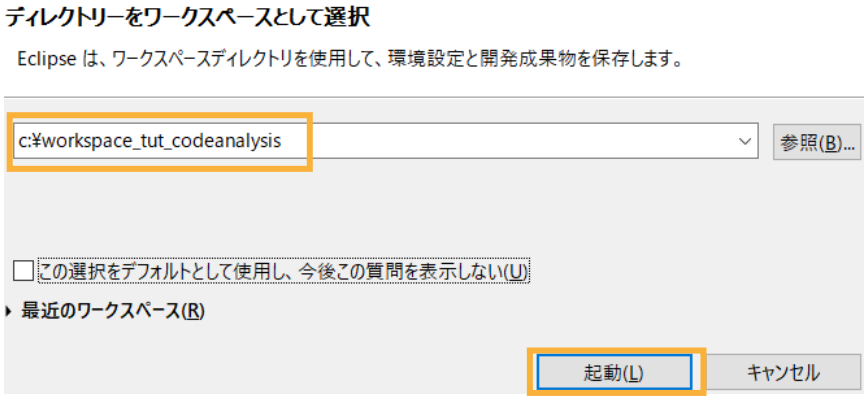
3 チュートリアルについて

本チュートリアルでは、いくつかのルールに違反したコードを内在する短いプログラムを利用して機能の確認を行います。

3.1 Eclipse の起動

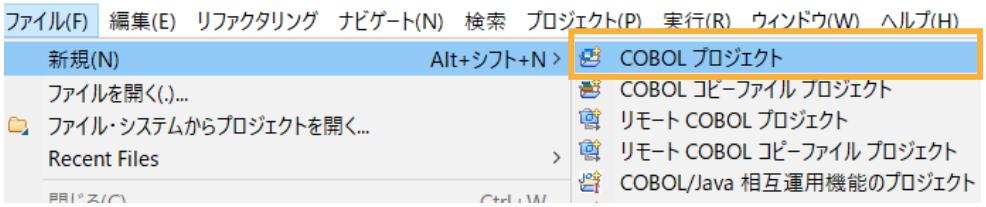
1) スタートメニューより、Visual COBOL for Eclipse を起動します。

2) ワークスペースを指定し、[起動(L)] ボタンをクリックします。

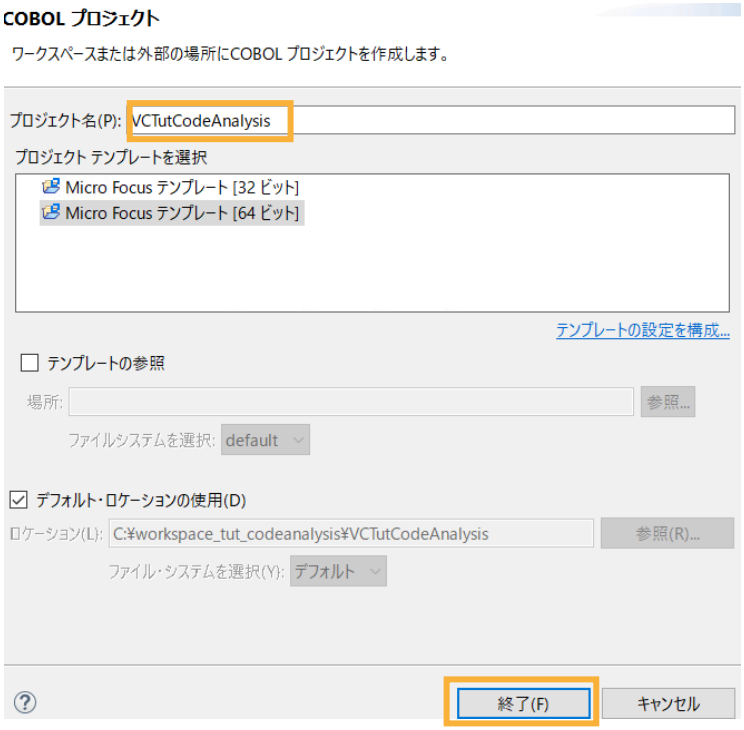


3.2 静的解析の実施

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

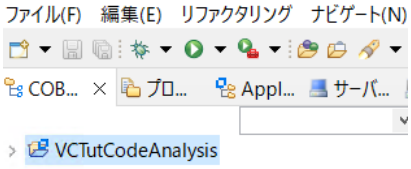


2) プロジェクト名に “VCTutCodeAnalysis” を入力し、[終了(F)] ボタンをクリックします。

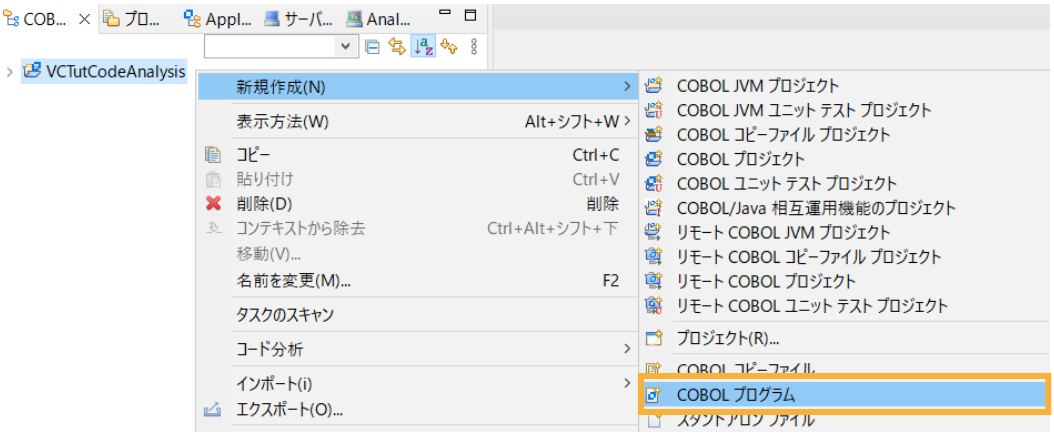


© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

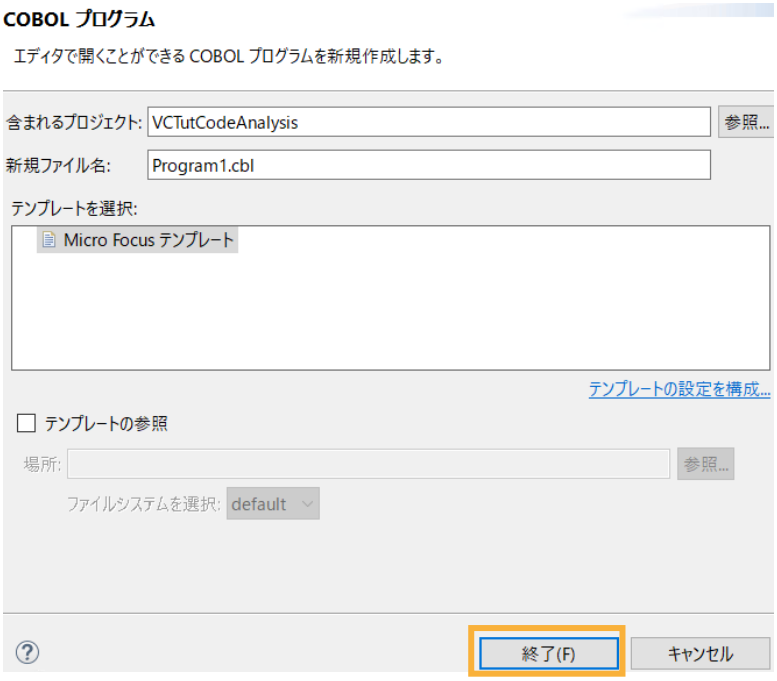
VCTutCodeAnalysis プロジェクトが作成されます。



3) VCTutCodeAnalysis 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クストメニューを表示し、[新規作成(N)] > [COBOL プログラム] を選択します。

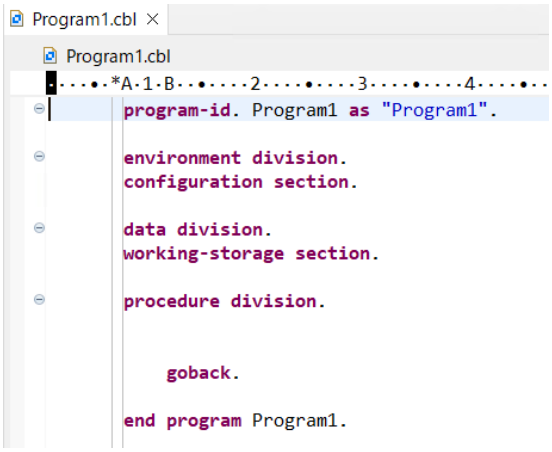


4) そのまま [終了(F)] ボタンをクリックします。



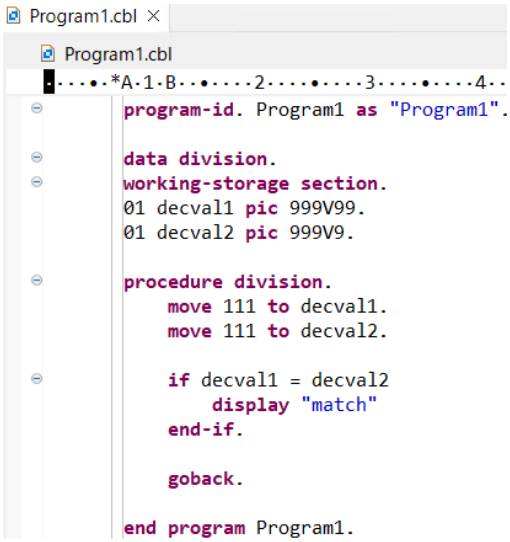
エディターが自動的に開きます。

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

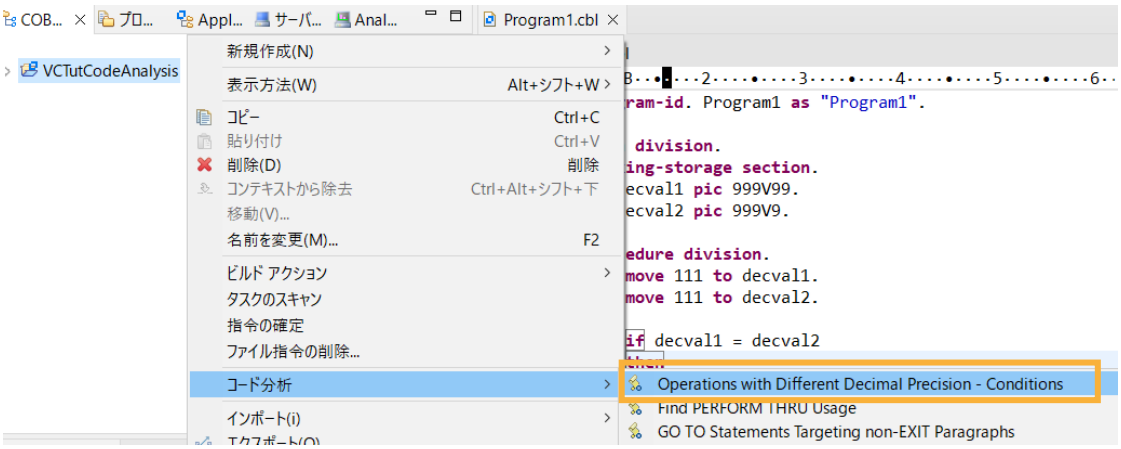


5) サンプルプログラム Program1.cbl をメモ帳などで開き、すべての内容をコピーし、Eclipse エディター上の Program1.cbl のコードをすべて選択して、貼り付けてください。その後、[ファイル(F)] > [保存(S)] で上書き保存してください。

これは、静的コード解析ルールに違反するコードを含めた簡単なプログラムです。



6) VCTutCodeAnalysis 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クストメニューを表示し、[コード分析] > [Operations with Different Decimal Precision – Conditions] を選択します。

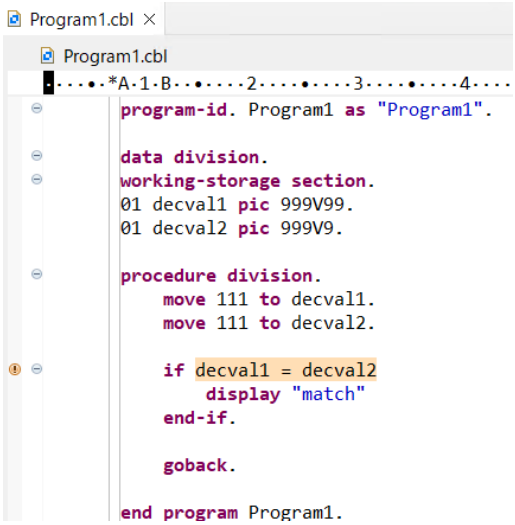


© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

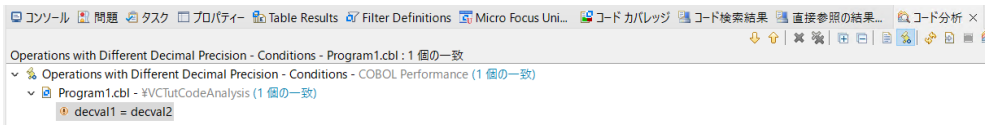
補足）

本チュートリアルでは、プロジェクト名を選択しているため、プロジェクトに含まれる全ての COBOL プログラムが静的コード解析対象となっています。任意の COBOL プログラム名を選択して実行することで、解析対象を限定することができます。

7) コード解析ルールに違反している個所が、エディター上でハイライト表示されます。

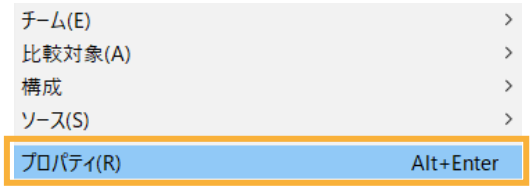


また、コード分析ビューでは結果の一覧を確認することができます。



3.3 プログラムビルド時の自動実行

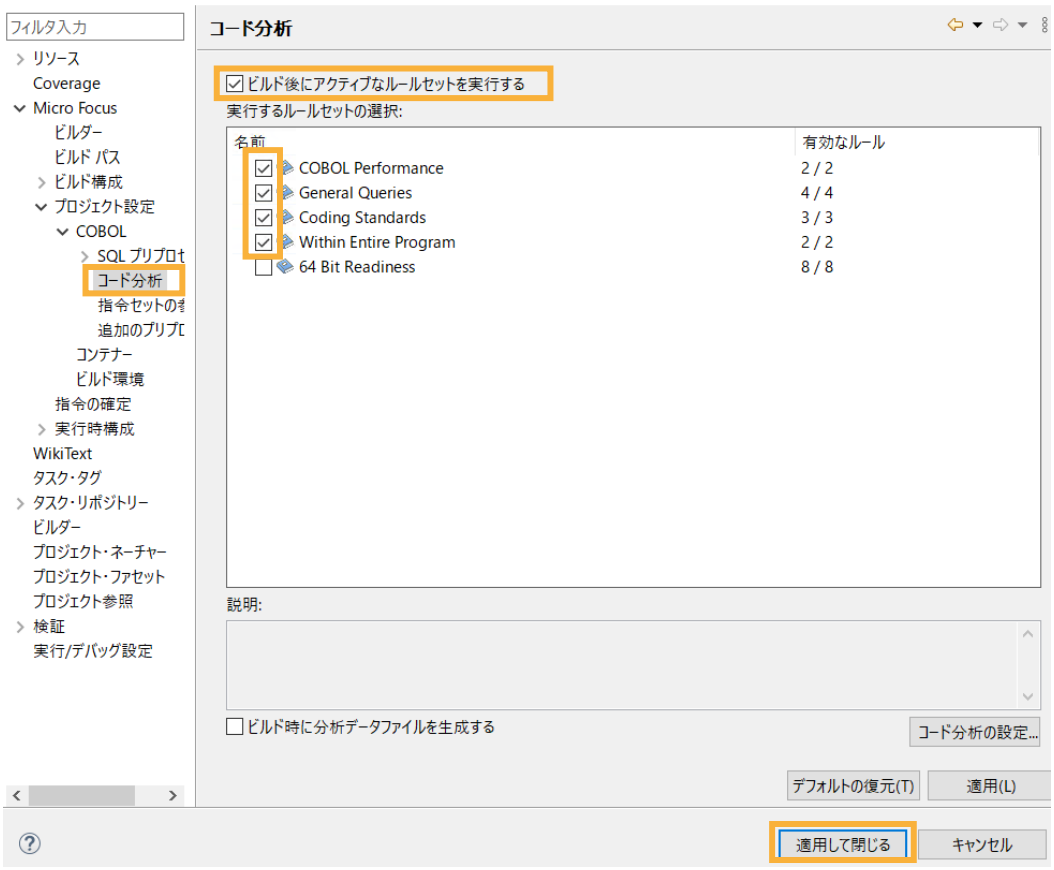
1) VCTutCodeAnalysis プロジェクト名を選択した状態で、マウスの右クリックにてコンテクストメニューを表示し、[プロパティ(R)] ボタンをクリックします。



2) 左側のツリーメニューより、[Micro Focus] > [プロジェクト設定] > [COBOL] > [コード分析] を選択します。

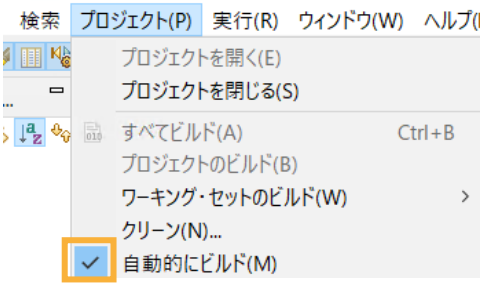
[ビルド後にアクティブなルールセットを実行する] 項目をチェックして、「実行するルールセットの選択」項目にある以下のルールセットにチェックを行った後、[適用して閉じる]をクリックします。

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

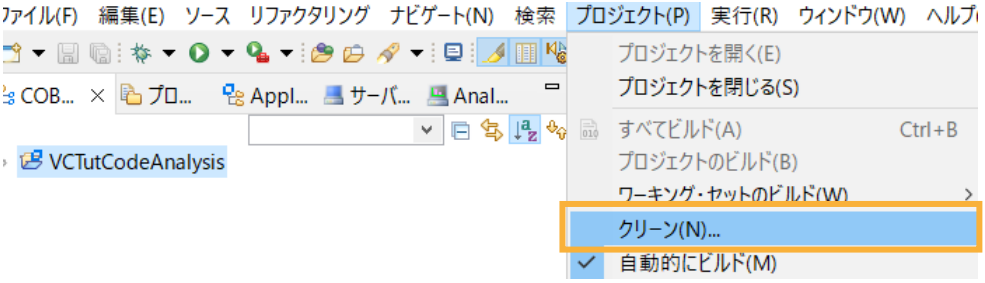


[適用して閉じる] ボタンクリック後にコード分析が実行されますが、ビルド後の自動コード解析を確認するため、ここでは結果を無視してください。

3) Eclipse IDE メニューより、[プロジェクト(P)] > [自動的にビルド] にチェックされていることを確認してください。されていない場合は、[自動的にビルド] を選択します。

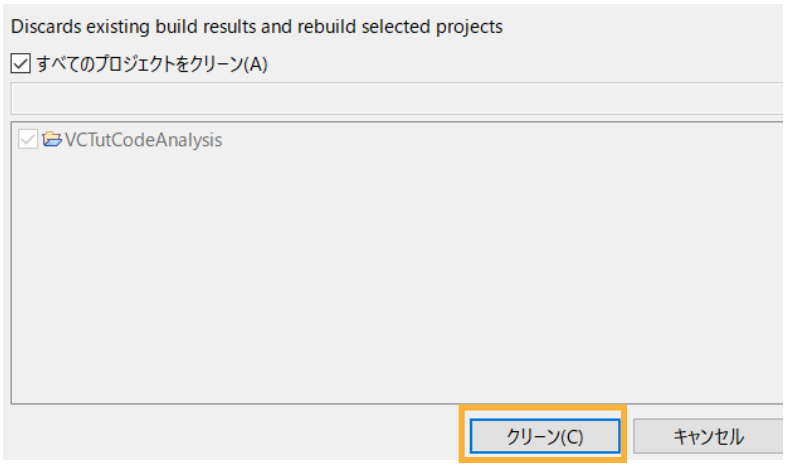


4) VCTutCodeAnalysis プロジェクトを選択し、Eclipse IDE メニューより [プロジェクト(P)] > [クリーン(N)] を選択します。

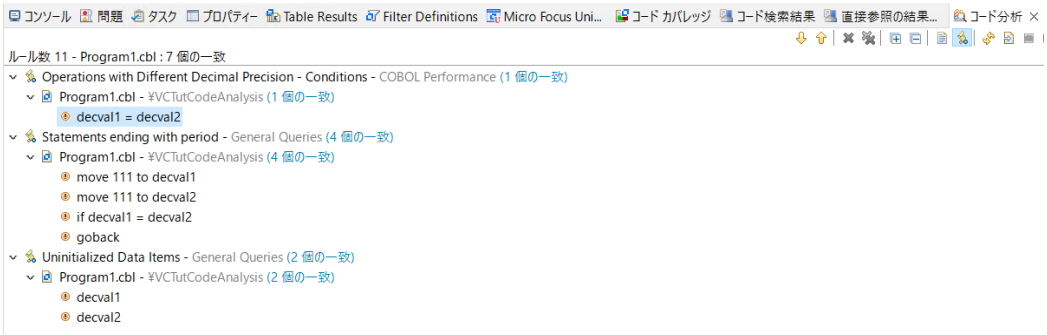


© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

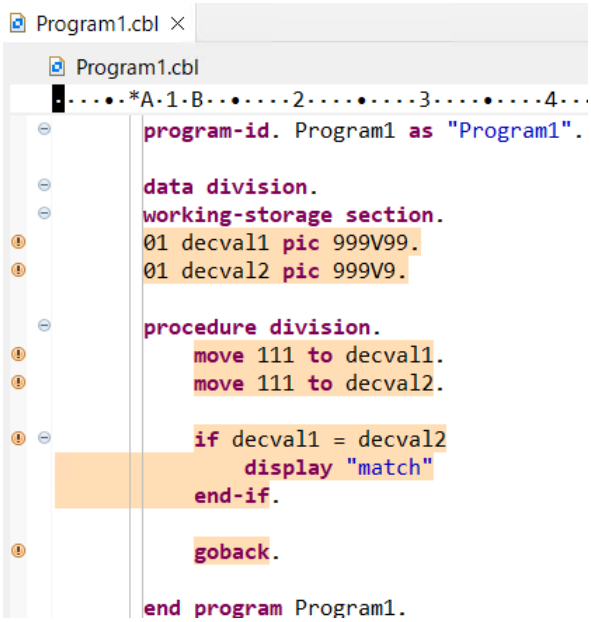
そのまま［クリーン(C)］ボタンをクリックします。



コード分析ビューより、前手順で確認した際と異なるルール違反が検出されていることを確認します。



エディター上にも、各違反箇所がハイライトされます。



© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

3.4 カスタムルールの作成

弊社別製品である Enterprise Analyzer をルールエディタとして利用することで独自ルールの作成することができ、作成されたルールを Visual COBOL にインポートすることで実現できます。本手順については、Enterprise Analyzer 製品マニュアルを参照ください。

4 ビルドインルール一覧

4.1 Operations with Different Decimal Precision – Conditions

小数情報の精度が異なる変数に対する操作を検出するルールです。

```
       01 decval1 pic 999V99.
       01 decval2 pic 999V9.

        procedure division.
       *> (中略)
           if decval1 = decval2
           then
                display "match"
           end-if.
```

4.2 Find PERFORM THRU Usage

PERFORM THRU 句を用いて、複数回同じ section 句の実行を検出するルールです。

```
       procedure division.
           perform para1 thru para3.
           display "between performs".
           perform para2 thru para3.
           goback.

       para1.
           display "para1".

       para2.
           display "para2".

       para3.
           display "para3".
```

4.3 GO TO Statements Targeting non-EXIT Paragraphs

EXIT 句が存在しない段落に対して GO TO 句を使用した遷移を検出するルールです。

```
       procedure division.
           go to sec2.
           goback.

       sec1.
           display "sec1".
           goback.
```

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

```
       sec2.
            display "sec2".

       end program Program4.
```

4.4 Uninitialized Data Items

値の初期化をすることなく、プログラム内で使用している変数を検出するルールです。

```
       data division.
       working-storage section.
       01 val pic x(6).

       procedure division.
            initialize val.
            display val.
            goback.
```

4.5 GO TO statement outside of PERFORMed section

PERFORM 句により実行された Section 節内で、Section 外への遷移を検出するルールです。

```
       procedure division.
      * GO TO statement outside of PERFORMed section
            perform sec1.
            goback.

       sec1 section.
            go to para1.

       sec2 section.

       para1.

       end program
```

4.6 Look for MOVE statements with loss of sign

符号なしの数値定義に対し、符号つきの値を設定するため、符号情報が欠落する箇所検出するルールです。

```
       data division.
       working-storage section.
       01 val pic 9(4).

       procedure division.
            move -123 to val.
```

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

4.7 Missing Explicit Scope Terminators

範囲符が未記載のため、暗黙的にスコープが終了している箇所を検出するルールです。その他範囲符については、Visual COBOL の製品マニュアルから以下のページを参照ください。

[リファレンス] > [COBOL 言語リファレンス] > [第 1 部： 言語の概念] > [COBOL 言語の概念] > [データの字類および項類] > [明示指定および暗示指定] > [明示範囲符および暗示範囲符]

```
       procedure division.

     * Explicit Scope Terminators
           if 1 = 2
               display "1 won't match with 2"
           else
               display " not match"
           .
```

4.8 ALTER Statements

ALTER 句の利用により、動的にプログラムフローを書き換えている箇所を検出するルールです。

```
       procedure division.

       alter main to proceed to sec2
       perform main.
       goback.
       main.
       display "tgt".
       go to sec1.

       sec1.
           display "sec1".

       sec2.
           display "sec2".
```

4.9 Dead Statements

プログラムの記述上、実行されない不要な記述を検出するルールです。

```
       procedure division.
           goback.
           display "DEAD CODE".
```

4.10 Unused Data

定義はあるものの、プログラム内で使用していない変数を検出するルールです。

```
       data division.
       working-storage section.
       01 unused pic x.
       01 used pic x.
       procedure division.
```

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

```
            move "X" to used.
            goback.
```

5 代表的なビルドインルールセット一覧

ルールセットとは、複数の静的コード解析ルールをまとめたもので、セット内のルールを一括で解析実行することができます。それぞれ、以下のビルドインルールが含まれています。

5.1 COBOL Performance

- Find PERFORM THRU Usage
- Operations with Different Decimal Precision – Conditions

5.2 General Queries

- GO TO Statements Targeting non-EXIT Paragraphs
- GO TO statement outside of PERFORMed section
- Uninitialized Data Items

5.3 Coding Standards

- ALTER Statements
- Look for MOVE statements with loss of sign

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.

- Missing Explicit Scope Terminators

5.4 Within Entire Program

- Dead Statements
- Unused Data

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。

© Rocket Software, Inc. or its affiliates 1990–2024. All rights reserved. Rocket and the Rocket Software logos are registered trademarks of Rocket Software, Inc. Other product and service names might be trademarks of Rocket Software or its affiliates.