Visual COBOL チュートリアル

ファイルハンドラーを使ったデータファイルアクセス

1　目的

メインフレームやオフコンからオープンプラットフォームに COBOL 資産を移行した時に必要になる処理は、やはりデータファイルアクセスではないでしょうか。Visual COBOL を使用しないでこれらのデータファイルにアクセスする場合、OS が提供するローレベルな API を使ってカーネルディスクドライバーを経由しファイルにアクセスします。ただし、この API が提供するのは低水準のバイトストリームファイルアクセスであり、COBOL アプリケーションから発行されるレコード単位による I/O 処理には対応できません。Visual COBOL は、 COBOL アプリケーションと OS の API の間にファイルハンドラーというインターフェースを介して COBOL アプリからの各種 I/O のハンドリングを行います。これにより、オープンプラットフォームにおいても、これまで同様、レコード単位によるデータファイルへのアクセスが可能になります。データファイルの種類には順編成ファイル、索引編成ファイルなどの種類が存在します。

順編成ファイルは、バッチ処理で扱うトランザクションファイルの操作に優れています。多くのバッチ処理は、夜間などのコンピュータ資源を占有できる環境で実行されるため、排他制御などのオーバーヘッドを受けることなく効率的に実行できる点にメリットがあります。一方で、夜間のバッチ処理は翌朝の業務開始時刻までには確実に処理が完了していることが必須であり、オンライン処理のレスポンスタイムと同様に、あるいは、それ以上に厳しい性能要求にさらされる処理となっています。たとえ 0.01 秒しかかからない処理でも 100 万回繰り返すことで 3 時間以上になりますので、大量バッチ処理を定時までに完了させるためには慎重なプログラミングが必要となります。トランザクションファイル 1 件の読み込みでも最も高速な手段が要求されるため、これを COBOL の順編成ファイルとして扱うことは最適な選択となります。
索引編成ファイルは、キー値で指定されたファイル内の固有のレコードにランダムにアクセスできる機能を持つファイル編成です。索引編成ファイルはマスターファイルとして使用されます。たとえば従業員マスターファイルは全従業員に関する様々な情報を保持していますが、従業員番号や従業員名などでランダムアクセスできなければなりません。
索引編成ファイルの概念は SQL を使い慣れた方には理解しやすいものです。実際、歴史的には索引編成ファイルは、データベースの概念に先立って登場しており、その後複数の索引編成ファイルの有機的な統合を実現するためにデータベース管理システムが考案された経緯があります。

このチュートリアルでは、ファイルハンドラーを使用したファイルのアクセス方法、ソートユーティリティの概要、rebuild ユーティリティの使用方法を学びます。

2　前提

本チュートリアルは、下記の環境を前提に作成されています。

OS	Windows 11
COBOL 製品	Visual COBOL 11.0 Patch Update 01 for Eclipse

- チュートリアル用プログラム

下記のリンクから事前にチュートリアル用のプログラムやデータ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。

プログラムおよびデータファイルのダウンロード

内容

3 チュートリアル手順

3.1 データファイルの準備

1) チュートリアル用データファイルの用意

① ダウンロードしたファイルを任意の場所に解凍します。work フォルダ、MFSORT フォルダ、REBUILD フォルダが入っていることを確認します。

3.2 Visual COBOL for Eclipse の起動とプロジェクト作成の準備

1) Visual COBOL for Eclipse の起動とプロジェクトの作成

① スタートメニューより [Rocket Visual COBOL] > [Visual COBOL for Eclipse] を選択します。

② ワークスペースの選択画面は任意のワークスペースを指定して、[起動(L)] をクリックします。

起動後は、ようこそ画面を閉じてください。

2) SJIS 資産利用の設定

① ワークスペース全体への設定

Eclipse メニューより [Window(W)] > [設定(P)] を選択し、ダイアログ上にて [一般] > [ワークスペース] をクリックしたのち、[テキスト・ファイル・エンコード] に “MS932” が設定されているかを確認します。異なる値が設定されている場合は、”MS932” を選択し、[適用して閉じる] をクリックします。

補足)

環境によっては、Windows-31J と表示されることがあります。その場合は、MS932 を Windows-31J に読み替えてください。

※Preference Recorder ダイアログが表示されたら、[キャンセル] を選択してください。

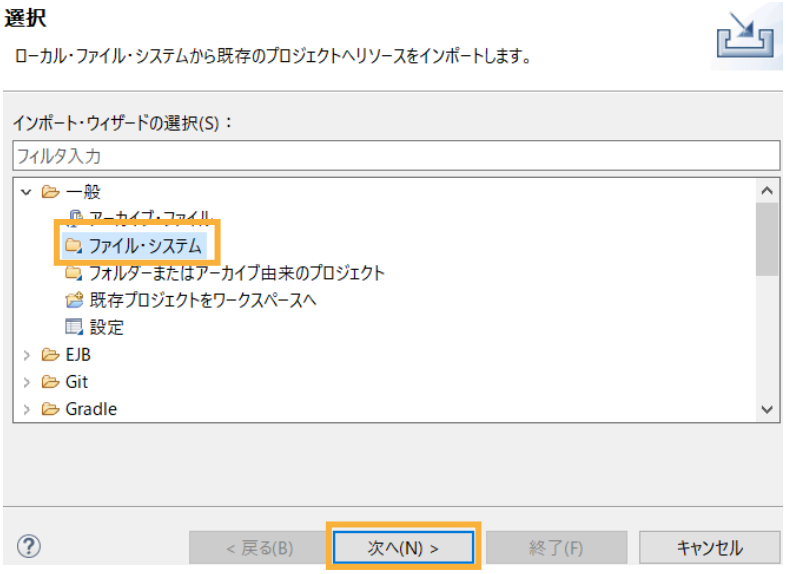
3.3 ネイティブ COBOL プログラムの作成と動作確認

1) ネイティブ COBOL プロジェクトの作成とプログラムのインポート

① Eclipse 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、プロジェクト名に “FileHandlerTutorial” を指定して、[終了(F)] をクリックします。

② COBOL エクスプローラーにて作成した「FileHandlerTutorial」プロジェクトを右クリックし、コンテクストメニューから [インポート(I)] > [インポート(I)] を選択します。

③ [一般] > [ファイル・システム] を選択し、[次へ(N)]をクリックします。



④ [参照(R)...] をクリックし、エクスプローラーから解凍したフォルダ配下の work フォルダを指定します。「FileDemo1.cbl」 にチェックを入れて [終了(F)] をクリックします。

2) プロジェクト設定

① COBOL エクスプローラーにて作成した「FileHandlerTutorial」プロジェクトを右クリックし、コンテクストメニューから [プロパティ(R)] を選択します。

② [Rocket Software] > [プロジェクト設定] > [COBOL] をクリックし、以下の設定を行ったうえで、[適用(L)] をクリックします。

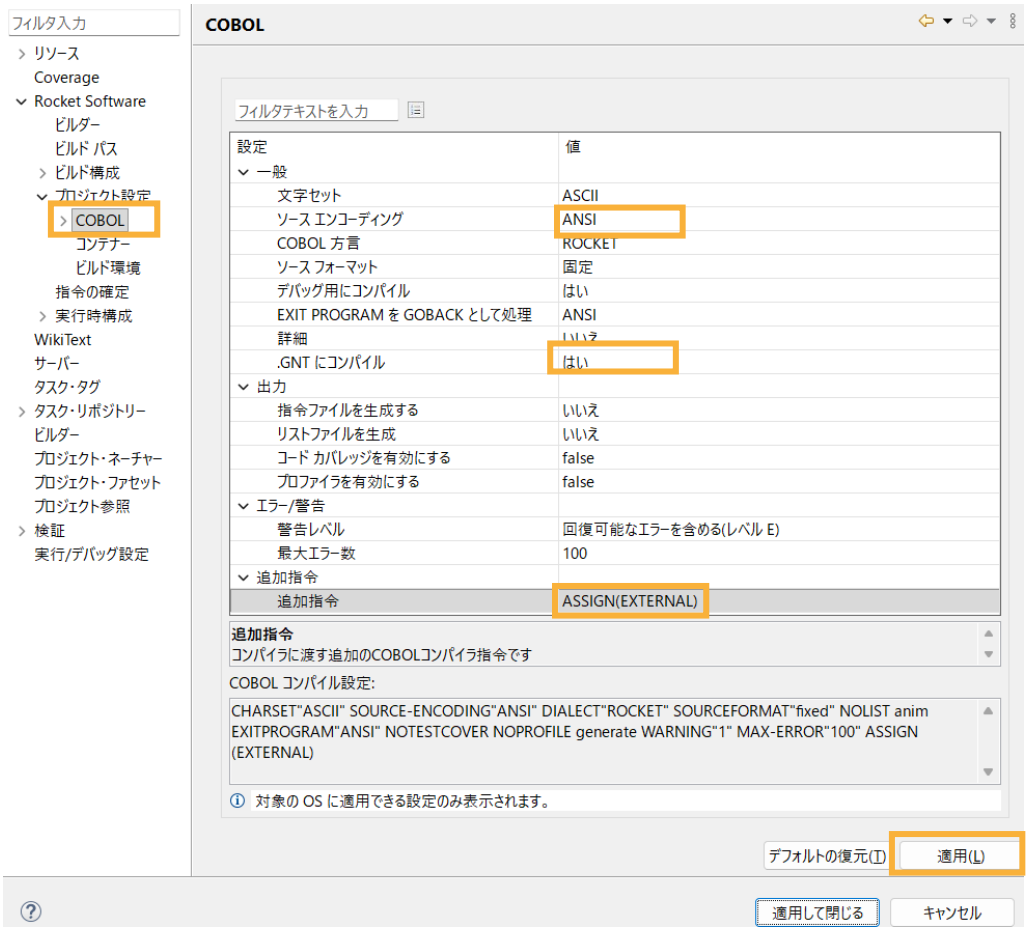
ソースエンコーディング： “ANSI”

.GNT にコンパイル： “はい”

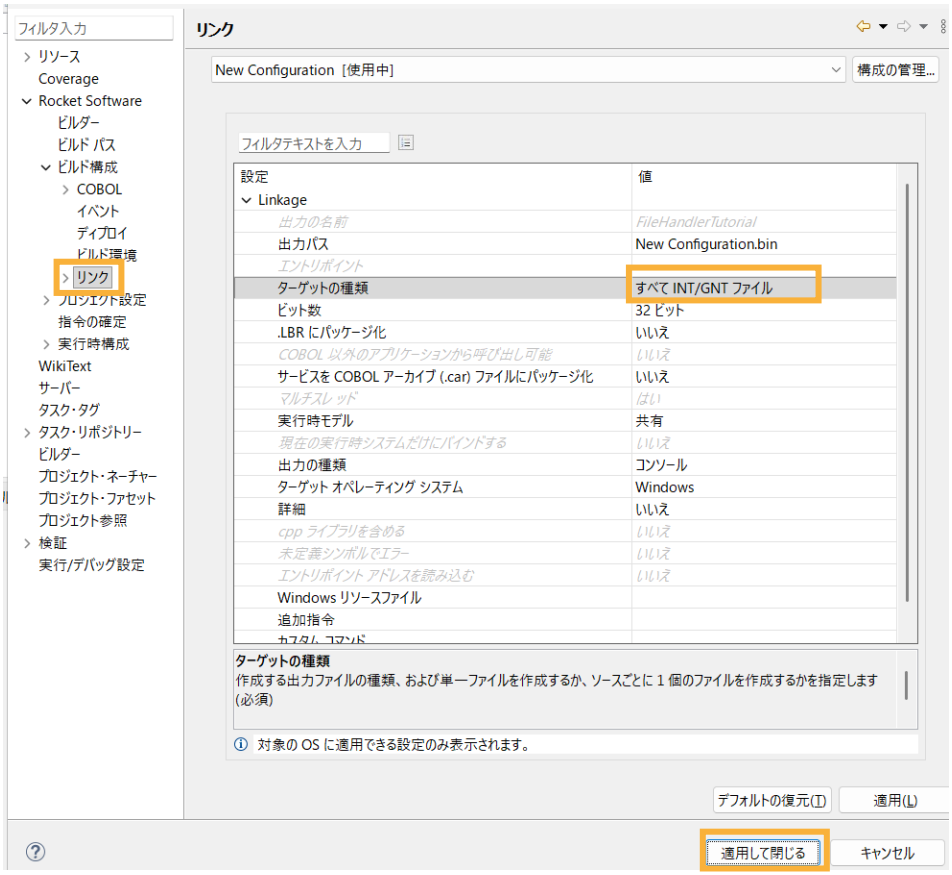
追加指令： “ASSIGN(EXTERNAL)”

補足）

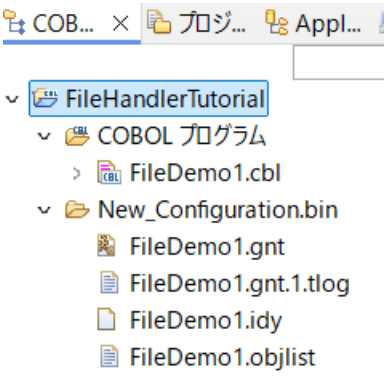
ソースエンコーディング “ANSI” によって、プロジェクトで SJIS 資産の利用が行えるようになります。



③ [ビルド構成] > [COBOL] > [リンク] をクリックし、 [ターゲットの種類] から「すべて INT/GNT ファイル」を選択し、[適用して閉じる] をクリックします。

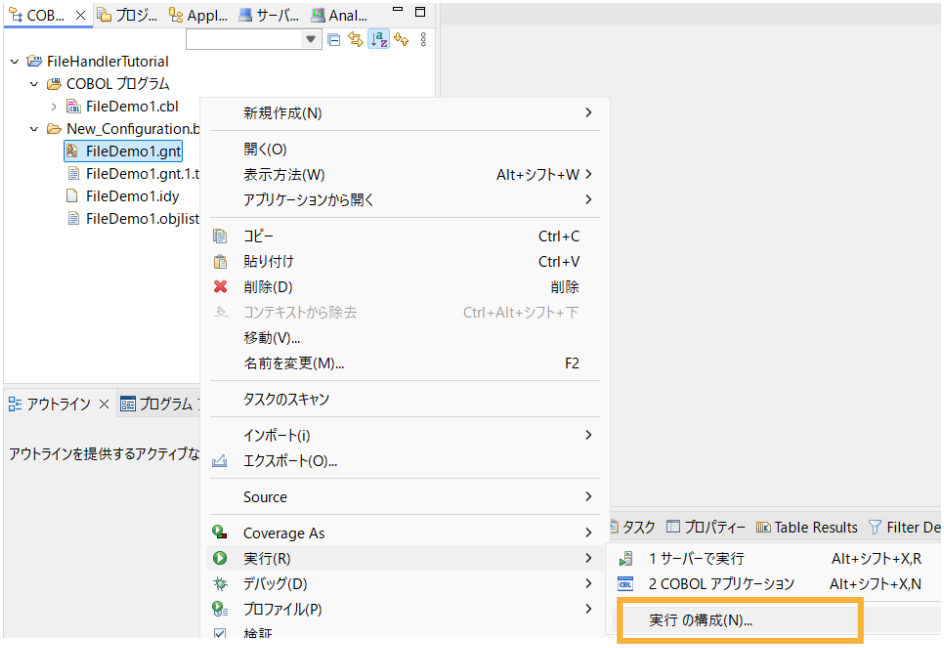


④　自動的にビルドが行われます。実行ファイルが作成されていることを確認します。

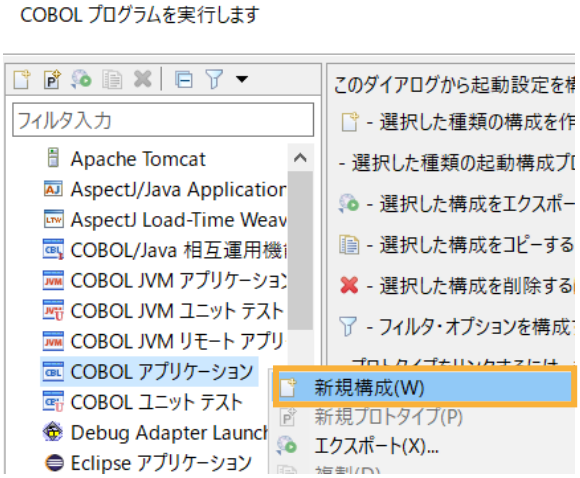


3) プログラムの実行

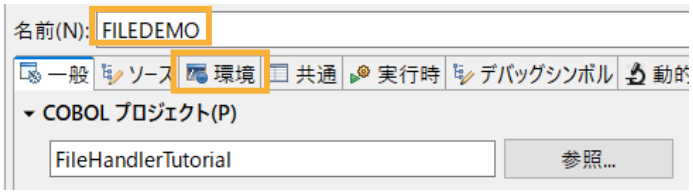
① 「FileDemo1.gnt」上で右クリックし、コンテクストメニューから [実行(R)] > [実行の構成(N)] を選択します。



② [COBOL アプリケーション] を選択し、マウスの右クリックにてコンテクストメニューを開き、[新規構成(W)] をクリックします。



③ [名前] に “FILEDEMO” を入力し、[環境] タブを選択します。



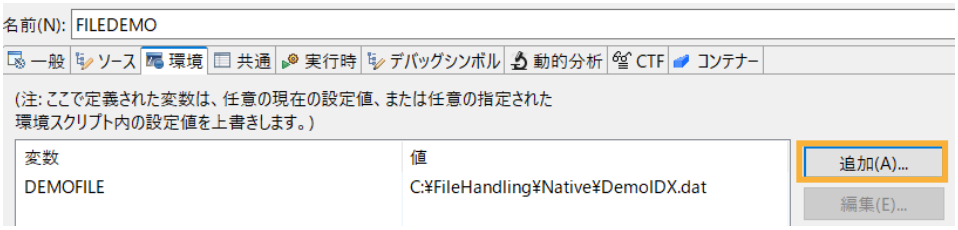
④　[追加] をクリックして、以下の情報を入力して、[OK] をクリックします。

変数：“DEMOFILE”

値：”C:¥FileHandling¥Native¥DemoIDX.dat”

補足）

値には任意のパスを指定できます。変更した場合は、以降の手順でも同様の値を使用してください。



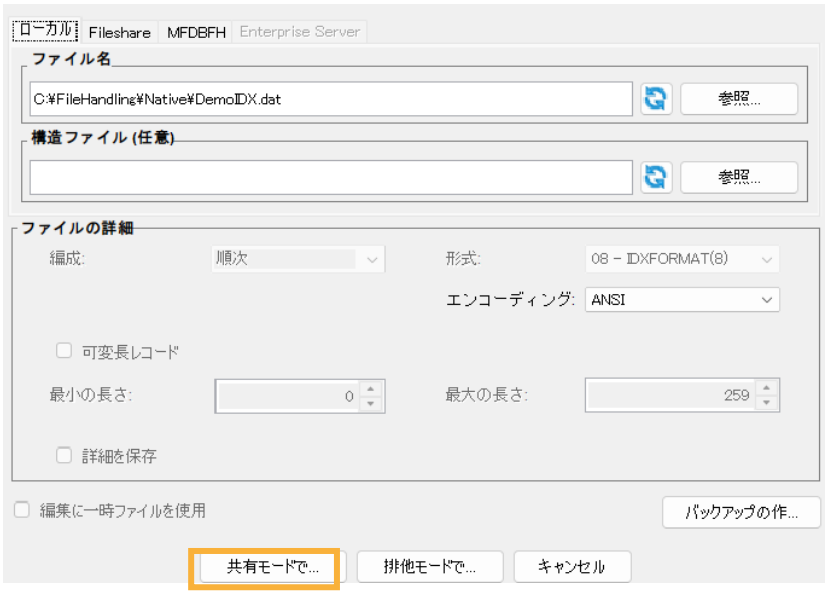
⑤　エクスプローラーより「C:¥FileHandling¥Native」フォルダを作成します。

⑥　[実行] をクリックすると、DOS プロンプトが表示されプログラムが実行され、ファイル作成が行われます。「C:¥FileHandling¥Native¥DemoIDX.dat」が作成されます。

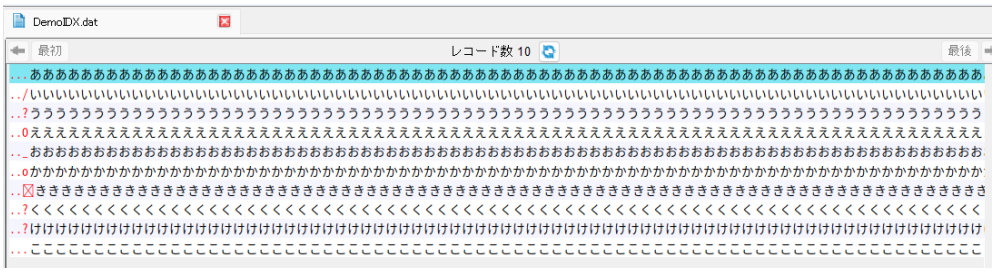
4)　実行結果の確認

①　Windows のスタートメニューから [Rocket Visual COBOL] >　[Data File Tools] を選択します。

②　[ファイル(F)] メニュー から [開く(O)] を選択し、「C:¥FileHandling¥Native¥DemoIDX.dat」を [共有モードで...] をクリックして開きます。



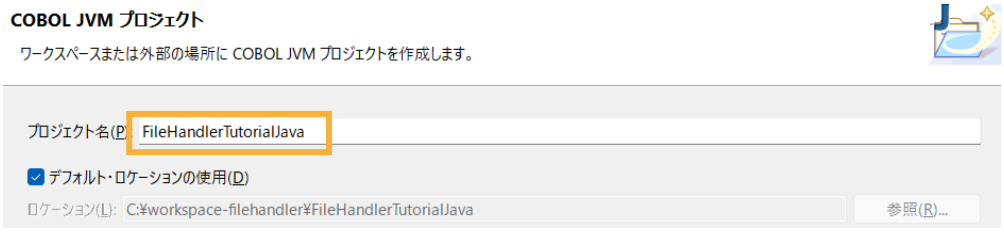
③　以下のようにファイルの中身が表示されます。



3.4 JVM COBOL プログラムの作成と動作確認

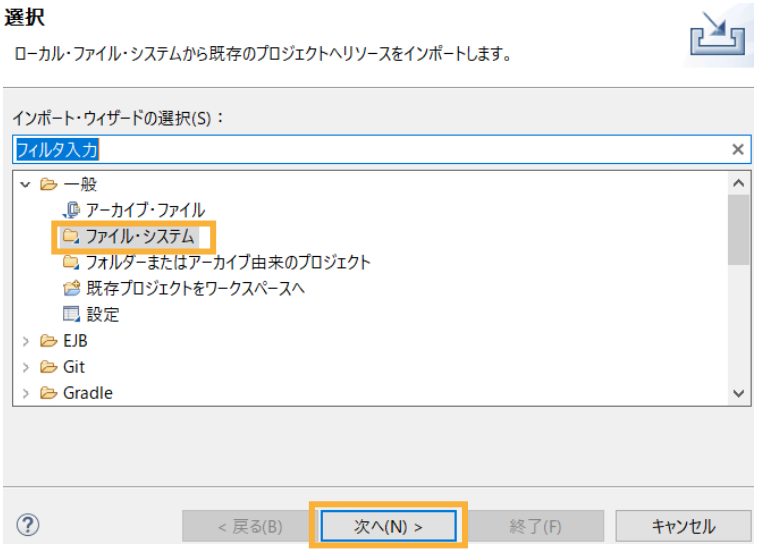
1) JVM COBOL プロジェクトの作成とプログラムのインポート

① Eclipse メニューより、[ファイル(F)] > [新規(N)] > [COBOL JVM プロジェクト] を選択し、プロジェクト名に “FileHandlerTutorialJava” を指定して、[終了(F)] をクリックします。



② COBOL エクスプローラーにて作成した「FileHandlerTutorialJava」プロジェクト配下の src フォルダを右クリックし、コンテクストメニューから [インポート(i)] > [インポート(I)] を選択します。

③ [一般] > [ファイル・システム] を選択し、[次へ(N)] をクリックします。



④ [参照(R)] をクリックし、エクスプローラーから解凍したフォルダ配下の work フォルダを指定します。「FileDemo1.cbl」にチェックを入れて [終了(F)] をクリックします。

この時点ではエラーが報告されますが、次の手順で修正するので無視してください。

2) プロジェクト設定

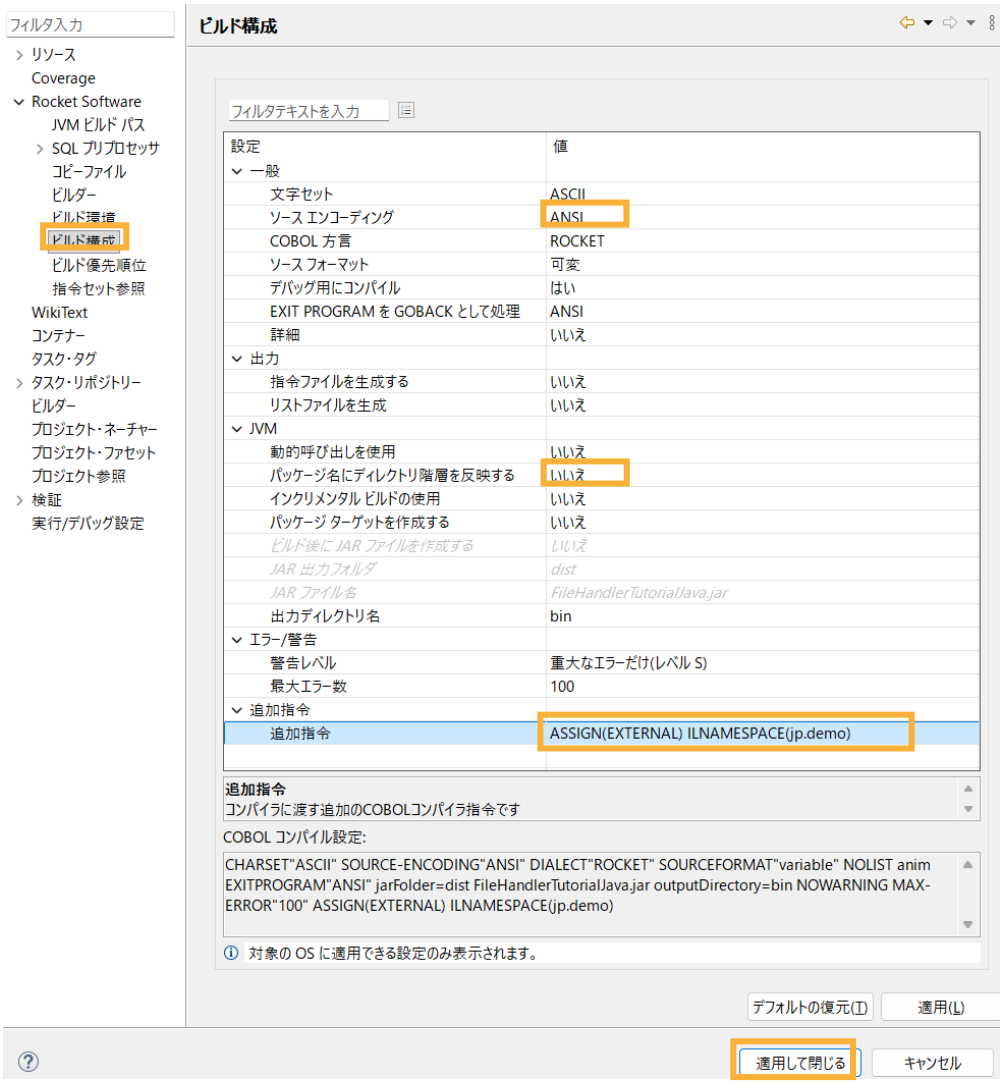
① プロジェクトの構成を変更します。COBOL エクスプローラーにて作成した「FileHandlerTutorialJava」プロジェクトを右クリックし、コンテクストメニューから [プロパティ(R)] を選択します。

② [Rocket Software] > [ビルド構成] をクリックしたうえで以下の変更を行い、[適用して閉じる] をクリックします。

ソースエンコーディング： “ANSI”

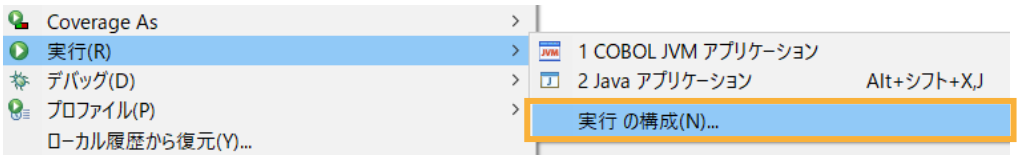
パッケージ名にディレクトリ階層を反映する： “いいえ”

追加指令： “ASSIGN(EXTERNAL) ILNAMESPACE(jp.demo)”



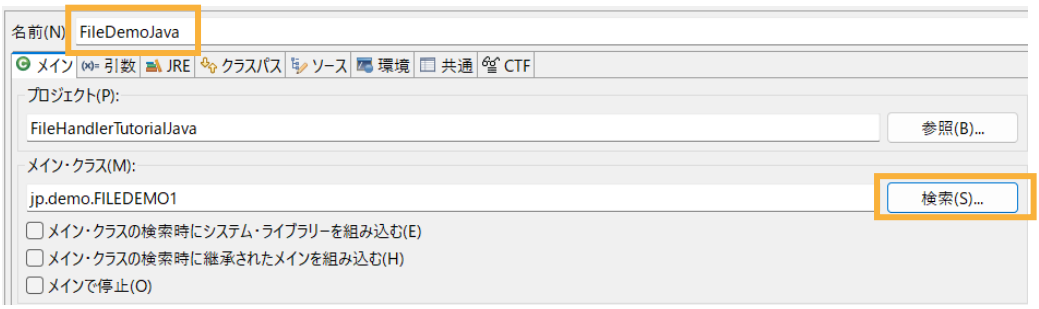
3) プログラムの実行

① 「FileHandlerTutorialJava」プロジェクト上で右クリックし、コンテクストメニューから[実行] > [実行の構成(N)]を選択します。

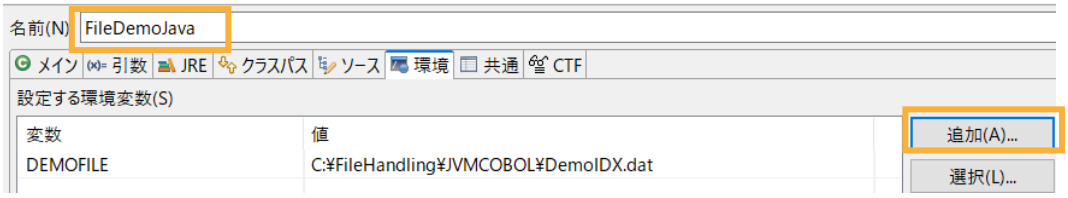


② 左側の [COBOL JVM アプリケーション] 上で右クリックし、コンテクストメニューから [新規構成(W)] を選択します。名前は、”FileDemoJava” と入力します。

③ [メイン] タブの [メイン・クラス] 横にある [検索(S)] をクリックし、「FILEDEMO1 – jp.demo」を選択し、[OK] をクリックします。



④ [環境] タブを選択し、 [新規(E)] をクリックします。[変数(N)] に "DEMOFILE"、[値(V)] には "C:¥FileHandling¥JVMCOBOL¥DemoIDX.dat" を入力し、[OK] をクリックします。



⑤ エクスプローラーより「C:¥FileHandling¥JVMCOBOL」ディレクトリを作成します。

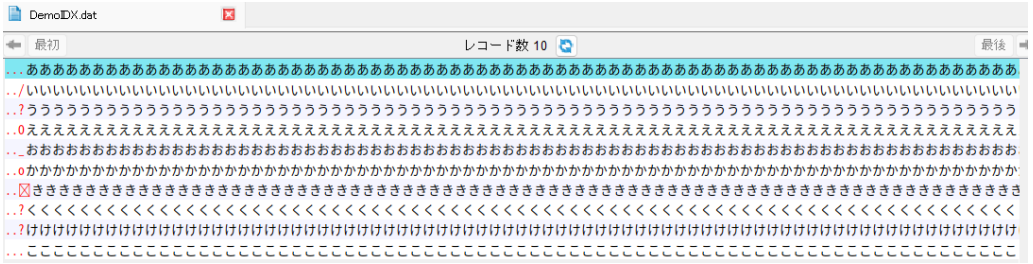
⑥ [実行(R)] をクリックすると、プログラムが実行され、「C:¥FileHandling¥JVMCOBOL¥DemoIDX.dat」が作成されます。

4) 実行結果の確認

① Windows のスタートメニューから [Rocket Visual COBOL] > [Data File Tools] を選択します。

② [ファイル(F)] メニュー から [開く(O)] を選択し、「C:¥FileHandling¥JVMCOBOL¥DemoIDX.dat」を共有モードで開きます。

③ 以下のようにファイルの中身が表示されます。



3.5 データファイルツール

データファイルツールは COBOL ファイル編成の変換、ファイルのブラウズ、レコードの編集等の機能を持った GUI ツールです。

1) データファイルツールの起動とファイルの読み込み

① データファイルツールを終了した場合は、Windows のスタートメニューから[Rocket Visual COBOL] > [Data File Tools] を選択し、起動します。

② [ファイル(F)] メニュー から [開く(O)] を選択し、「C:¥FileHandling¥Native¥DemoIDX.dat」を共有モードで開きます。

2) デバッグ情報ファイルからレコードレイアウトを作成

① Windows のスタートメニューから [Rocket Visual COBOL] > [Visual COBOL Command Prompt (64-bit)] を選択します。

② 「Data File Structure Command Line ユーティリティ」を使用してデバッグ情報ファイルからレコードレイアウトを作成します。下記のコマンドを一行で入力・実行し、レコードレイアウトを生成します。レイアウトファイルは、指定した idy ファイルと同じフォルダに出力されます。

- dfstrcl <FileHandlerTutorial ワークスペースフォルダへのパス>¥New_Configuration.bin¥FileDemo1.idy /d F-REC

```
C:¥>dfstrcl c:¥workspace_filehandler¥FileHandlerTutorial¥New_Configuration.bin¥FileDemo1.idy /d F-REC

Layout File Creation - DFSTRCL V20240423
Loading file - FileDemo1.idy
Added default structure - F-REC
Structure file created - FileDemo1.str
Processing complete

C:¥>
```

補足）

パスは環境に合わせて修正してください。

なお、Eclipse IDE 上で FileHandlerTutorial ワークスペースへのフォルダパスを、以下の手順で確認できます。

① FileHandlerTutorial プロジェクトを選択し、マウスの右クリックにて、[プロパティ(R)] をクリックします。

② 画面左側より [リソース] を選択すると、右側の [ロケーション(L)] にワークスペースへのフォルダパスが表示されます。

3) レコードレイアウトを適用し可視化できないデータを確認

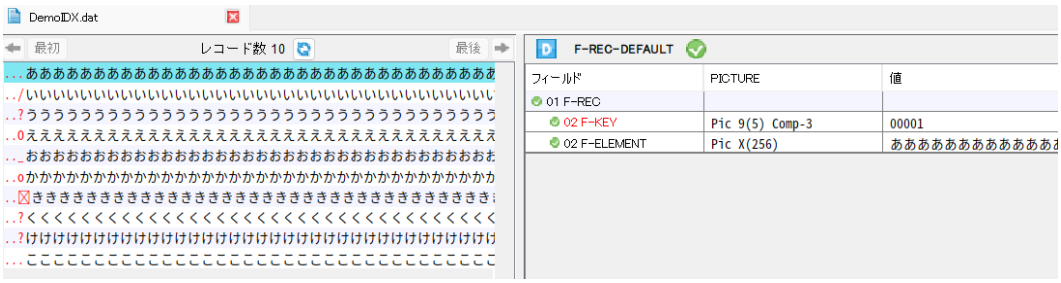
① [構造] > [構造をロード] を選択します。

② 前のステップで作成した「FileDemo1.str」ファイルを指定します。

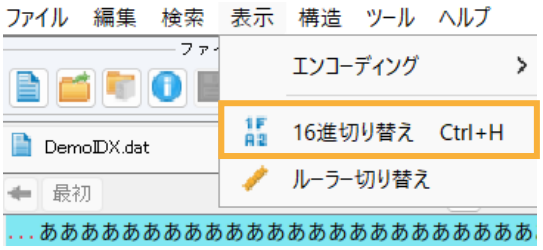
注意）

このファイルは、前手順にて dfstrcl コマンドで指定した idy ファイルと同じフォルダにあります。

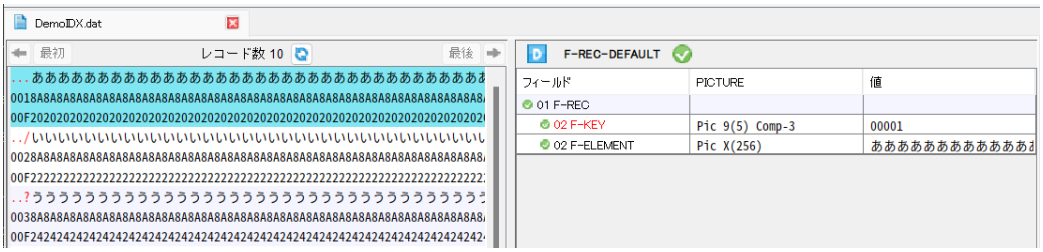
③ ファイルの中身がレコード単位で確認できるようになります。



④ [表示] > [16 進の切り替え] を選択します。



データが１６進数表記で表示されます。

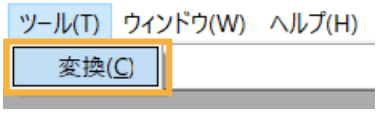


4) 他のファイル編成に変換

ファイル編成の変換には、クラシックデータファイルツールを使う必要があります。

① Windows メニューより [Rocket Visual COBOL] > [Classic Data File Tools] をクリックします。

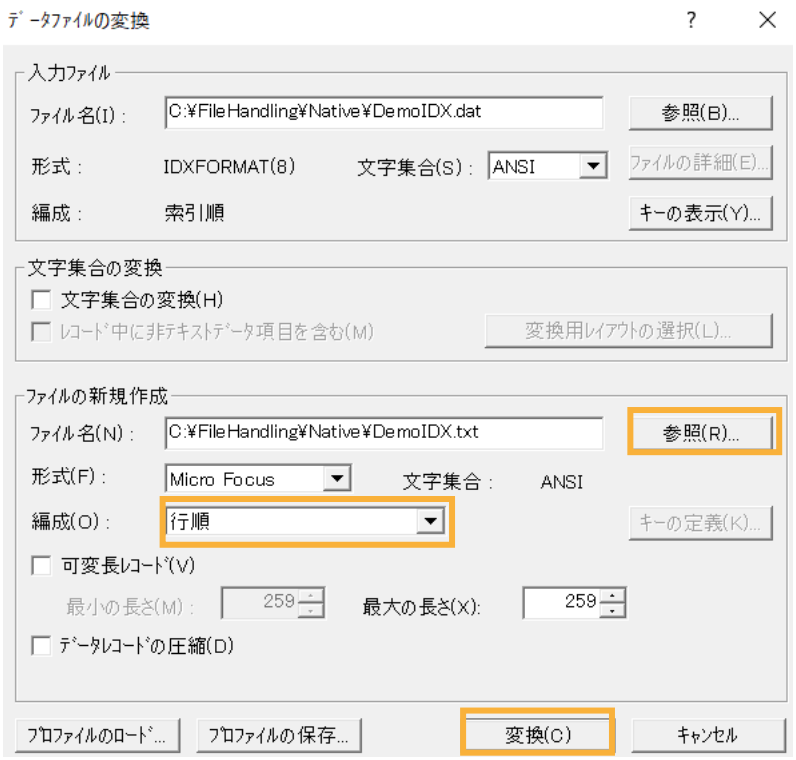
② [ツール(T)] メニュー > [変換(C)] を選択します。



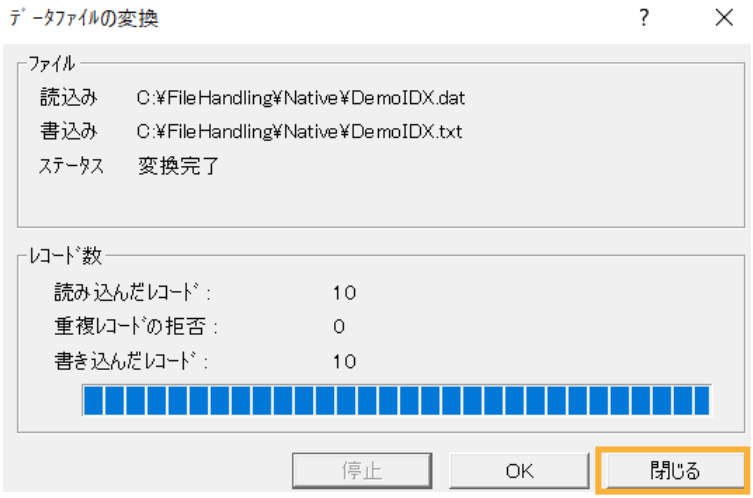
③ [データファイルの変換] ウィンドウが表示されます。[入力ファイル] セクションの [参照(B)] をクリックし、エクスプローラーより「C:¥FileHandling¥Native¥DemoIDX.dat」をオープンします。

④ 以下の設定を行い、[変換(C)] をクリックします。

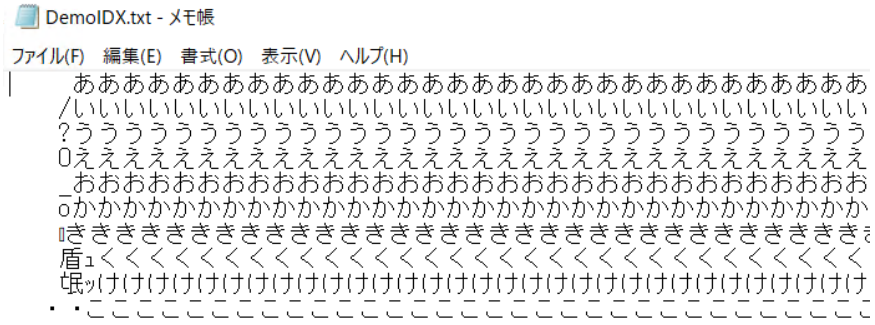
- [ファイルの新規作成] の[参照(R)]をクリックし、エクスプローラーより「C:¥FileHandling¥Native¥DemoIDX.txt」を指定
- [編成(O)] を「行順」に変更



変換結果が表示されますので、[閉じる] をクリックします。



⑤ メモ帳で作成したファイルを開きます。各レコードが改行で区切られた行順ファイルになっていることを確認します。



⑥ クラシックデータファイルツールを終了します。

3.6 ソートユーティリティ MFSOFT の確認

ソートユーティリティ MFSORT は、IBM メインフレームの DFSORT とコマンド互換のあるソート・マージユーティリティです。

1) ソート用ファイルの用意

① ダウンロードしたサンプルファイル内の MFSORT フォルダを任意のフォルダにコピーします。本チュートリアルでは、C ドライブ直下にコピーします。

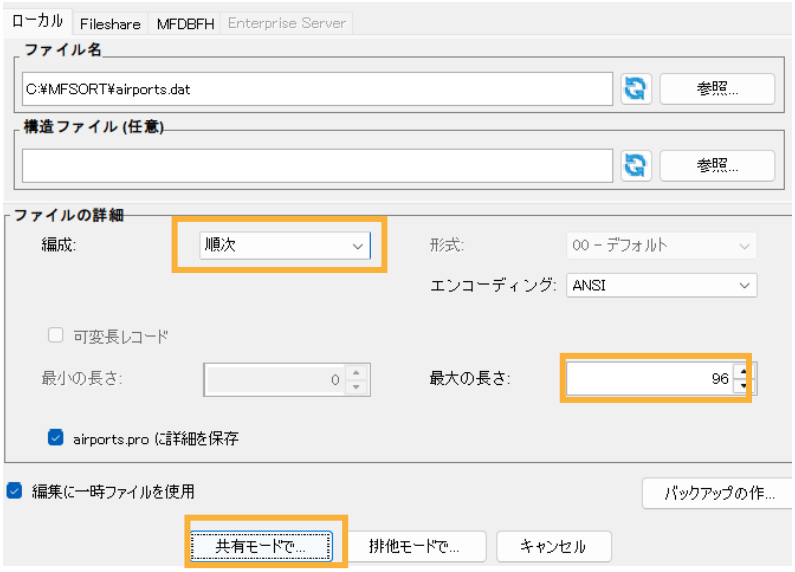
2) データファイルツールの起動と読み込み

① データファイルツールを起動して MFSORT フォルダ配下の「airports.dat」を開きます。

② 以下の設定を行い、[OK] をクリックします。

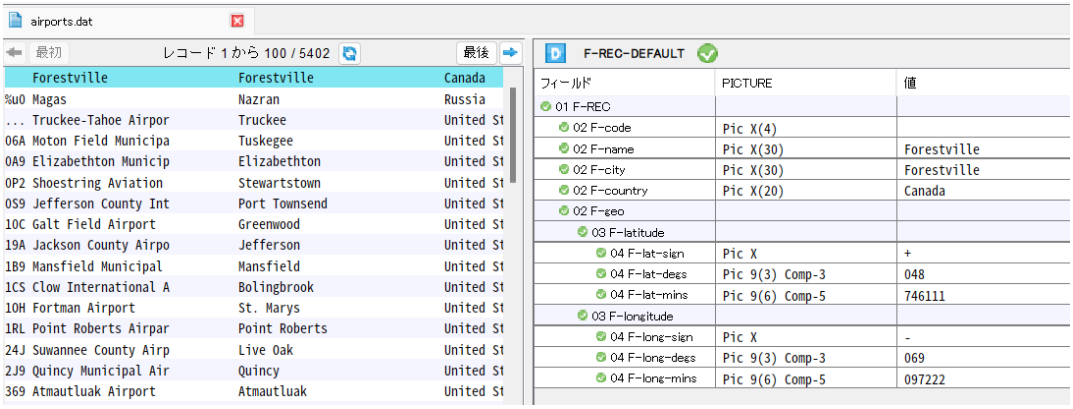
ファイル編成： “順次”

最大の長さ： “96”



③ [構造] > [構造をロード] を選択し、MFSORT フォルダ配下の「AIRPORTSEQ.STR」ファイルを指定して、[開く] をクリックします。

ファイルの中身がレコード単位で確認できるようになります。



3) ソート順の変更

このファイルは「F-code」と呼ばれる空港コードの昇順でソートされています。これを「F-name」という空港名でソートしてみます。

① ソート用のコマンドファイルを用意します。

１行目はインプットするデータファイルの指定、２行目はアウトプットするデータファイルの指定、３行目がソートするフィールド名の指定です。ここでは「F-name」を昇順で指定するので 5 バイト目から 30 バイト、昇順（ascending）に並べ替えを指定しています。

```
use airports.dat record (f, 96)
give sorted.dat ORG SQ
sort fields (5, 30 CH, a)
```

補足）

上記ファイルは、MFSORT フォルダ配下内の airport.srt ファイルとして保存されています。

② Windows のスタートメニューから [Rocket Visual COBOL] > [Visual COBOL Command Prompt (64-bit)] を選択します。

③ 以下のコマンドを実行します。

- cd ¥mfsort
- mfsort take airport.srt

```
C:¥FileHandling>cd ¥mfsort
C:¥MFSORT>mfsort take airport.srt
C:¥MFSORT>
```

注意）

データファイルツールなどで airports.dat を開いている場合、上記コマンドを実行する前に閉じてください。

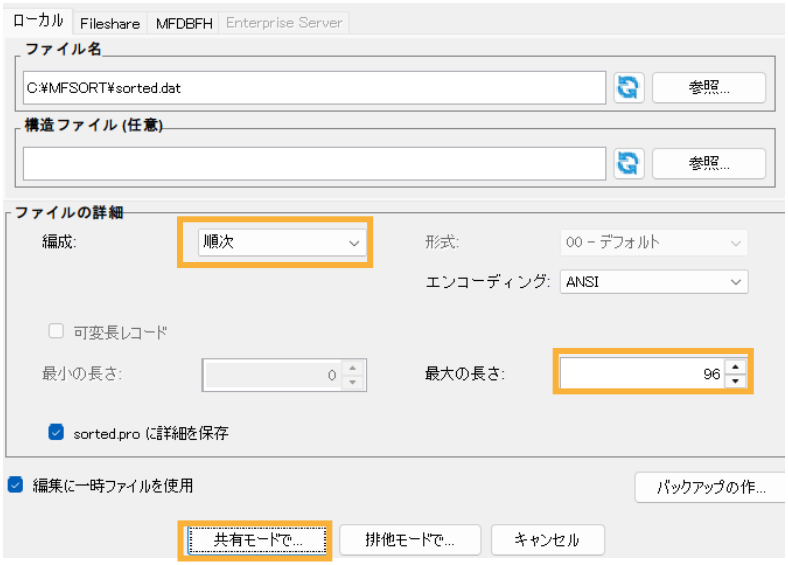
4) 実行結果の確認

① SYSOUT ファイルに実行結果ログが入っているのでメモ帳を起動して内容を確認します。

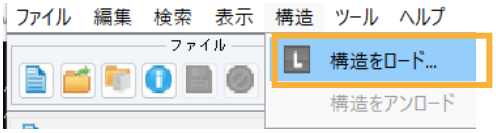
```
          Rocket Software MFJSORT ユーティリティ 3.0.00

          take airport.srt
          use airports.dat record (f, 96)
          give sorted.dat ORG SQ
          sort fields (5, 30 CH, a)
SORT204I: ********************* ソート結果 *********************
SORT205I: INPUT  ファイル 'airports.dat'
                入力レコード                5402 件
                使用レコード                5402 件
SORT206I: OUTPUT ファイル 'sorted.dat'
                使用レコード                5402 件
                出力レコード                5402 件
SORT399I: Rocket Software MFJSORT ユーティリティ終了
```

② クラシックデータファイルツールを起動して「sorted.dat」を開きます。

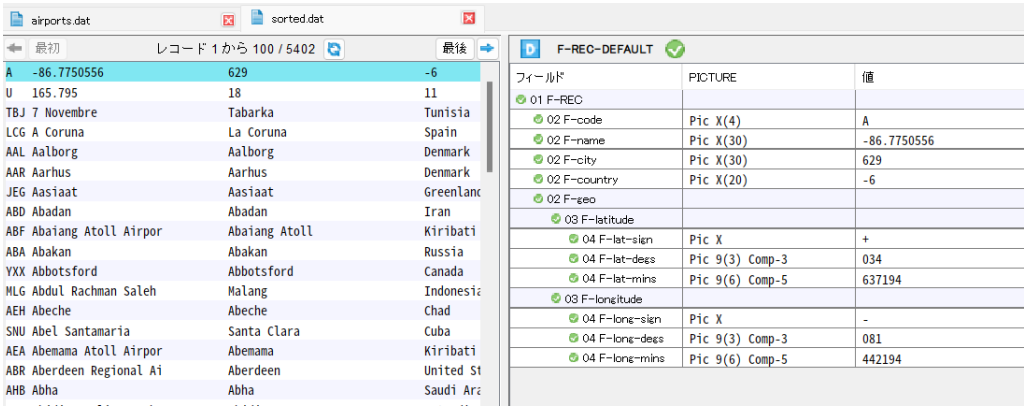
さきほど同様、[ファイル編成] には “順次”、[最大の長さ] に “96” を指定して共有モードで開きます。



③ [構造] > [構造のロード] を選択し、MFSORT フォルダ配下の「AIRPORTSEQ.STR」ファイルを指定します。



空港名でソートされていることを確認します。



3.7 ファイル管理ユーティリティ REBUILD の確認

ファイル管理ユーティリティ REBUILD は、COBOL のファイル編成の変換、索引ファイルの索引再編成、破損した索引ファイルのリビルド機能を提供するコマンドラインのユーティリティです。ここでは索引ファイルの再編成を行います。

1) サンプルファイルの用意

① ダウンロードしたサンプルファイル内の REBUILD フォルダを任意のフォルダにコピーします。本チュートリアルでは、C ドライブ直下にコピーします。

2) rebuild コマンドによるインデックスの再編

① Windows のスタートメニューから [Rocket Visual COBOL] > [Visual COBOL Command Prompt (64-bit)] を選択します。

② 以下のコマンドを実行します。

- cd ¥REBUILD
- rebuild airportsIdxOld.dat,airportsIdxNew.dat

```
C:¥MFSORT>cd ¥REBUILD
C:¥REBUILD>rebuild airportsIdxOld.dat,airportsIdxNew.dat
airportsIdxOld.dat,airportsIdxNew.dat
再構成が完了しました - レコード読み込み =       5383
C:¥REBUILD>
```

再編成前のファイル airportsIdxOld.dat はレコードの追加、更新、削除を繰り返していたため、ファイルサイズが大きくなっています。rebuild コマンドにて、再編成が行われ、不要領域が解放されファイルサイズが縮小されたことが確認できます。

```
C:¥REBUILD>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 3837-A4A9 です

 C:¥REBUILD のディレクトリ

2025/11/17  17:05    <DIR>          .
2025/11/17  17:05           618,200 airportsIdxNew.dat
2016/01/14  16:47           621,200 airportsIdxOld.dat
               2 個のファイル           1,239,400 バイト
               1 個のディレクトリ  82,505,326,592 バイトの空き領域

C:¥REBUILD>
```

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。