Micro Focus Visual COBOL for Visual Studio 2017







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。

Micro Focus Visual COBO	L for Visual Studio 2017 セットアップ - 🗆 🗙
	Micro Focus Visual COBOL for Visual Studio 2017
LFOCUS	バージョン 3.0.00206
	製品インストール場所:
	C:\Program Files (x86)\Micro Focus\Visual COBOL 参照(B)
	注意: Visual Studio のヘルプビューアがインストールされていないと、この製品のヘルプ(英語版)が正常 にインストールされません。 ヘルプビューアをインストールしていない場合は、このセットアップを実行する 前に ReadMe のシステム要件を参照してください。
	<u>ことでパリック</u> してリースノートを表示 □ 同意する(A) エンドユーザー使用許諾契約書 ● 「同意する(A) エンドユーザー使用許諾契約書



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。

Micro Focus Visual COBO	for Visual Studio 2017 セットアップ ー		×
	Micro Focus Visual COBOL for Visual Studio 2017		
L FOCÚS	パージョン 3.0.00206		
	製品インストール場所:		
	C:\Program Files (x86)\Micro Focus\Visual COBOL	参照(<u>B</u>)
	注意: Visual Studio のヘルプビューアがインストールされていないと、この製品のヘルプ(にインストールされません。ヘルプビューアをインストールしていない場合は、このセットアッ 前に ReadMe のシステム要件を参照してください。	英語版)カ プを実行	「正常 する
	<u>ここをクリック</u> してリリースノートを表示	_	_
	□□「同意する(A) エンドユーザー使用許諾契約書	シストール	0



5	セッ	トア	ップ	を終了	します。	þ
---	----	----	----	-----	------	---

[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。

 2ダートページ - Microsoft Visual Studio ファイル(F) 編集(E) 表示(V) プロジェクト(P) デパッグ(D) チーム(M) ○ • ○ 沿 • ○ □ □ □ □ □ □ □ · ○ • □ 	ツール(T) テスト(S) 分析(N) ウィンドウ(W - ▶ アタッチ 声 ₌	▼ 2 クイック起動 (Ctrl+Q) 0 ヘルプ(H)
「 ^ナ スタート ページ キ ×		-
PEDDO Latubal Enterprise 2017 の新機能の詳細 .NET Framework の新機能を確認する Visual Studio Team Services の新機能を確認する 最近	 (用く) リモトトバージョンコントロールからコード を取得するか、ローカルドライブで向か を開きます。 次からチェックアウト: Visual Studio Team Services プロジェクト / ソリューションを聞く コカルダーを聞く 	開発者向け情報 Hands on with Visual Studio for Mac Visual Studio for Mac was released just under two months ago at Build 2017, and already we've seen tremendous growth 新規 2017年6月23日

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。

ussteel.cbl* 🕂 🗡	READCUST.cbl	READAII.cbl	About.aspx.cs	COBOLClassLibs		
🔩 US-STEEL				TIME-TEST05-END()		
<u></u>	140-TIME-TESTO	4.				+
070600	IF FIELD65	= FIELD66				PERSONAL PROPERTY AND INC.
070700	GO TO 15	O-TIME-TESTO4-END).			※ 12000年20日間 111日の日本
070800	ADD 1 TO F	IELD65				P TREAMPLE RE P. ARM 200 LARKY
070900	MOVE FIELD	/1 TO FIELD/U.				ing Man
071000	GU TU 140-	IIME-IESIU4.				
07100	IDU-IIME-IESIU	4-ENU.				
071200	MOVE " ETC	OON ETLL 100" TO				
071300	MOVE FIG	-CON FILL TOU IN	J COMMENT-LINE.			I II IIII
071500	"NEVT A	100 DOSITION FIEL	D WITH BE ZEDOED	BY MOVING THE EIG		Bertabler.
071600	- "URATIVE	CONSTANT ZERO."	TO COMMENT.	bi moving the fig		
071700	WRITE LINE	-OUT FROM COMMENT	I-LINE AFTER ADVA	NCING 3 LINES.		E CERT
071800	MOVE SPACE	S TO COMMENT-LINE				De la la constante de la consta
071900	MOVE 0 TO	FIELD65 TEST-COL	DE .			C SCOTLANDS IN
072000	PERFORM 57	0-CLOCK-IN				STOCEMEN'S
	160-TIME-TESTO	5.				
072200	IF FIELD65	= FIELD66				
072300	GO TO 17	O-TIME-TESTO5-END).			
072400	ADD 1 TO F	IELD65				Paste in me
072500	MOVE ZERO	TO FIELD70.				B
072600	GO TO 160-	TIME-TESTO5				
⊟072700	170-TIME-TESTU	5-END.	004000	THIO TO HE INDEX	D ITENHITYE MO	
072800	PERFORM 58	U-CLUCK-OUT.	095000-	"E MOVED" TO COMM	ENT	
072800	MOVE U TO	PIELUOD CODDI MOUE DICDI	AV" TO 095100	WRITE LINE-OUT FROM	COMMENT-LINE AF	
073000	MOVE SUD	SURPI MUVE FUISPL TEMS WILL RE STOR	AT 10 0095200	MOVE SPACES TO COMM	ENT-LINE	E_AND NOT
073200		NT	CD 001MG095300	MOVE ZEROS TO FIELD	j5.	
073300	WRITE LINE	-OUT FROM COMMENT	U954UU	SET INDO L TO INDO		
073400	MOVE SPACE	S TO COMMENT-LINE	005000	SET IND2-L TO IND2.		
073500	MOVE CST7	TO TS-ADDR.	000000	SET INDE TO INDI-L.		
073600	MOVE 0 TO	TEST-CODE				1 0100-01-01
100 % 🔹 🖣	0000000000				• •	

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。

00100	SORCDI + ×	NativeConsole						
📬 UD	T0036P							♥ MAIN-S
	Ė015400∗ 015500∗ 015600	PERFORM	<< 車輌の減価償却調整 CAR-DEPRECIATION-RTN・	<2.0>	>>			
		PERFORM	<< 車輛種別調整 CAR-REG-TYPE-RTN.	<3.0>	>>			
		PERFORM	<< 契約距離区分調整 KEIYAKU-KYORI-KUBUN-RTN。	<4.0>	>>			
	016500* 016500	PERFORM	<< 記名被保険者の年齢調整 HIHOKENSHA-NENREI-RTN・	鉴 <5.0>	>>			
	016700*	PERFORM	<< 記名被保険者の免許証(MENKYOSHOU-IRO-RTN・	の色調整 〈	(6.0>			
	017000*	PERFORM	<< 運転者の範囲調整 <7. UNTENSHA-HAN-I-RTN・	0> >>				
•	017300	MOVE	HOKEN-RYOUKIN TO LK-HOKEN-	RYOUKIN.				
	017400*	TH OVODI 000	CK-HOI	KEN-RYOUKIN	000429	00 <u></u> ⊨ _		
۵	017600 MA	GOBACK.	·			- D	コピー(Y)	Ctrl+C
	017700						式のコピー(X)	
	Ė017800/ 017900≭	MODULE-NO.	<1.0>		**		値のコピー(L)	
100 %	-						値の編集(E)	
白動衣	5 #Kr					↔	ウォッチ式の追加(W)	
包刻炎			//r			⇔	並列ウォッチの追加(P)	
1480	HOKEN-RVO	UKIN	00042900				16 准数で表示(山)	
-	LK-HOKEN-R	YOUKIN	00042900					
-							COBOL テバック ツールナッノ スタイル	•

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。

新しいプロジェクトの追加			? ×
▶ 最近使用したファイル	.NET Framework 4.7 👻 並べ替え: 既定	- # E	インストール済み テンプレート の検索 (Ctrl+E 🔎 -
 ▲ インストール済み ▲ conol オロジークト 	Setup Project	Visual Studio Installer	種類: Visual Studio Installer
Visual C# Visual Basic	Web Setup Project	Visual Studio Installer	which files can be added
 Visual C++ SQL Server 	Merge Module Project	Visual Studio Installer	
▶ JavaScript ▲ その他のプロジェクトの種類	Setup Wizard	Visual Studio Installer	
Visual Studio Installer	CAB Project	Visual Studio Installer	

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。

新しいプロジェクト						?	×
▶ 最近使用したファイル		.NET Fr	amework 4.7 👻 並べ替え: 既定	• # E	インストール済み テンプレート の検索	툕 (Ctrl+E	ρ-
▲ インストール済み		CBL	Enterprise Server アプリケーション	COBOL プロジェクト	▲ 種類: COBOL プロジェクト		
⊿ テンプレート					ネーティブ コマンドライン アプリケー	ションを作り	成す
▲ COBOLプロジェ Managed	クト		Windows アプリケーション	COBOL プロジェクト	ったののノロシェクトです。		
Database		CBL C:\	コンソール アプリケーション	COBOL プロジェクト			
Native		CBL AC3	リンク ライブラリ	COBOL プロジェクト			
 Visual Basic Visual C++ 			空のプロジェクト	COBOL プロジェクト			
COL Conver		CBL	Micro Focus INT/GNT	COBOL プロジェクト			
▶ オンライン		- (8)			*		
名前(<u>N</u>):	ConsoleHello						
場所(<u>L</u>):	C:¥work¥Tutorial¥	dotNet		•	参照(<u>B</u>)		
ソリューション名(<u>M</u>):	TutorialSol			(<u>D</u>)	
					リーノ管理に追加(U)		
					ОК	キャンセ	Jル



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。

刘 TutorialSol - Microsoft Visu	al Studio							▼ 🖌	クイック起動 (Ctrl+Q)	₽ = ¤ ×
ファイル(F) 編集(E) 表示(V)	プロジェクト(P) k	ビルド(B) デバッグ(D)	チーム(M)	ツール(T)	テスト(S)	分析(N)	ウィンドウ(W)	ヘルプ(ト	H)	Yoshihiro Mitsutomi 👻 ٧
G - O 者 - 🍟 💾 🚰	9 - C - D	ebug - x86	-	▶開始▼	🎜 🛯 🖉	- 🏷 🗖	- 9 0	00:	- 🖌 🛍 🚽 🖿 🖷	표 2월 및 위 위 위 위
Program1.cbl + X			_	1	• //				 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>	• # X
PROGRAM1			• @ Proce	dure Divisio	n					ኤ.ኤ.መ.Թ. 🖌 "
5 identi	fication divis	ion.						+		
C Progra	n-id. Programl							and the second sec		ninisol' (1 Tustatus)
🔶 🖻 enviro	nment division	•						2	▲ CBL ConsoleHelle	
config	uration section	n.							🔑 Propertie	s
🚊 😐 data d	ivision.	1							Program'	l.cbl
wurkin.	g-storage sect	run -						÷		
E proced	ure division. SPLAY "Hello W	or Id"								
go	back.	0110								
end pr	ogram Program1						l			
	Ser am Proeram									
									991-9391920	
									プロパティ	▼ ₽ ×
110 % 👻 🖣									Program1.cbl COBO	L ファイルプロパティ 🔹
 出力								- ↓ ;	_ ₽ ₽	
出力元(S): 全般		•	2 4	놀 🚈 광	þ				□ コンパイル情報	
			1 - 1						ソース ノオーマット ▲ マネージ コード	Fixed
									曰 全般	T GID C
									フルパス	C:¥work¥Tutorial¥dotN
									- V-7 7+-7wh	ConcoloHollo
•								Þ	SourceFormat 指令で	す。
呼び出し階層 パッケージ マネー	ジャー コンソール エラ	覧 出力								
IntelliSense を更新しました - エラー 0 個	8	11行	34 列	34 文字		挿入				↑ ソース管理に追加 🔺

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。

M TutorialSol - M	icrosoft Visual Stud	dio				▼ 🖌	クイック起動 (Ctrl+Q)	P _ 0	x
ファイル(F) 福集(F)	表示(V) プロミ	2775(P) PILE(B)	デバッグ(D) そ	F-//M)) 7 75(S)	分析(N)	ว่าวหว่าหว	Voshihiro Mitsutomi	- VM
ヘルプ(日)	2010 - 101				J //((2/	55 01 (<u>EB</u>)	5151 5 CE	TO SHITTO THILSOLOTH	
G • O 13 •	🖕 🖬 🔐 🛛 🤊	- 연 - Debug -	×86	- ▶ 開始	- 🎜 🚽 🛍	- 70 🖂	• 0 0 0 0 0 0	- 🔺 🛍 🚚 🖢 🖷	i i i
Program1.cbl +							 ソリューション エクスプロ]-7- 🔻	
🕈 🏘 PROGRAM1			Procedu	e Division			- 000 M-	10-5-00 J	с ^и
19	identificat	ion division.	N I			÷			0 -
	program-id.	Program1.	N	N		-	791-93719871	コーノー ()(使業 (Cul+:)	p .
	environment	division.				2		itorialSol' (1 ノロシエクト)	_
	configurat i	on section.				1.2	Propert	ties	
Le la	data divisi	on.				-	Program	m1.cbl	
1	working-sto	orage section.							
	a second use of	liulatan				-			
1	DISPLAY	"Hello World"							
	goback -						-		
	end program	Program1.							
L .	ond program								
							ソリューション エクスプロ	コーラー チーム エクスプローラ	
110.96							プロパティ		
110 /8 •							Program1.cbl COB	OL ファイルプロパティ	•
出力	-					• 4 .			
出力元(S): ビル			- 1	눈 알 알 🎽	52				
・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	開始: フロシェク utorial¥dotNet¥Ti	r: consoleHello, ∦ utorialSol¥ConsoleH	⊧b%: Debug x86 ello¥Program1.0	 地 のコンバイル・	Þ		1	Fixed	10
* Generatin * Data:	g C:obj¥x86¥Debu 432 Cod	s¥Program1 e: 336	iterals.	140			マネージ コード	False	- 11
COBOL コン/	イル:1個正常	終了または最新の状態	0 個 失敗。				□ 全般		- 11
	o ノし:#Work¥luti ルド:正常終了ま:	orial#dotNet#lutori; たは最新の状態 1、共	ilSol¥Uonsolene 敗 0、スキッゴ	/ 10/0 ======	ig#Conso i eHe i	lo.exe	フルパス	C:¥work¥Tutorial¥c	lotN _
								ConsoleHalle	
							 ソースノオーマット SourceFormat 指令 	7.4	
呼び出し階層 パ	ッケージ マネージャー	コンソール エラー一覧	出力			-	sourcer official jairs	× 7 0	
	15	4 70	1						
华调元丁	117	199	TX7	坤.	<			↑ ソース管理に追加	- 4

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。

d TutorialSol (デ	パッグ中) - Microsoft Visual Studio	▼ 🗗 クイック起動 (Ctrl+Q)	₽ = ¤ ×
ファイル(F) 編集(E) ソール(T) テスト(S)	表示(V) プロジェクト(P) ビルド(B) 分析(N) ウィンドウ(W) ヘルプ(H)	デパッグ(D) チーム(M)	Yoshihiro Mitsutomi 👻 YM
G - O 👌 ·	- 🖆 💾 🚰 💙 - 🤊 - 🗎 Debug	→ x86 → ▶ 続	行(C) • 声 📮 🐺 🐺 🐺
プロセス [0x3B8](ConsoleHello.exe 🔹 💽 5475	サイクル イベント 👻 スレッド [0x1A1	0] <名前がありません> 👻 🍹
Program1.cbl ↔ ×	<mark>د</mark> .		<u>ب</u> ک
🕏 PROGRAM1		Procedure Division	<u>ー</u> シ
	data division. working-storage section.	デバッグ開	また サレンシン サレート
•	procedure division. DISPLAY "Hello World". goback.	如但1交	1 1 1 1 1 1
	end program Program1.	ステ	ップイン
10 % -			
	Animator application		- 🗆 ×
l	Hello World		^
			\checkmark

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ヘ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。

		ソリューション エクスプローラー マ 単 ×
edure Division		<u>- </u> © ○ ☆ ☆ - 「 o - ち @ ি 🗡 "
	÷	────────────────────────────────────
		Im ハリューション 'TutorialSol' (1 プロジェクト)
	 ソリューションのどルド(B) ソリューションのリビルド(R) ソリューションのクリーン(C) 解析(A) パッチビルド(T) 構成マネージャー(O) ソリューションの NuGet パッケージの管理(N) NuGet パッケージの復元(G) 	Ctrl+Shift+B soleHello Properties Program1.cbl
	 	
#41 10 TD21+71 (N)	追加(D)	•
既存のプロジェクト(E)	スタートアッププロジェクトの設定(A)	
新しい Web サイト(V)	Q、 プロジェクト詳細	
既存の Web サイト(B)	🔽 ソリューションをソース管理に追加(D)	エクスプローラー チール エクスプローラー

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。

プ	プロパティ ▼ ₽ ×							
b	utton1 System.Windov	vs.Forms.Button						
0	🗄 💱 🖗 🗲							
	FlatStyle	Standard 🔺						
ŧ	Font	MS UI Gothic, 9pt						
	ForeColor	ControlText						
	Image	(なし)						
	ImageAlign	MiddleCenter						
	ImageIndex	(なし)						
	ImageKey	(なし)						
	ImageList	(なし)						
	RightToLeft	No						
	Text	button1						
	TextAlign	MiddleCenter						
	TextImageRelation	Overlay						
	UseMnemonic	True						
	UseVisualStyleBackCc	True						
	UseWaitCursor	False 🗸						

Text

コントロールに関連付けられたテキストです。



テキストの値を「Button1」から「Say Hello」に 変更します。

プ	ロパティ	- ₽	х
b	utton1 System.Windov	ws.Forms.Button	•
	🗄 💱 📢 🦻 🗲		
	FlatStyle	Standard	٠
Ð	Font	MS UI Gothic, 9pt	
	ForeColor	ControlText	
	Image	(なし)	
	ImageAlign	MiddleCenter	
	ImageIndex	(なし)	
	ImageKey	(なし)	
	ImageList	(なし)	
	RightToLeft	No K	
	Text	Say Hello 🗸 🗸 🗸	

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。

プロパティ ・ 부 ×							
la	bel1 System.Windows	.Forms.Label	•				
	🗄 💱 🔽 🗲 🔎						
	BorderStyle	None	٠				
	Cursor	Default					
	FlatStyle	Standard	4				
Ð	Font	MS UI Gothic, 9pt					
	ForeColor	ControlText					
	Image	(なし)					
	ImageAlign	MiddleCenter	Ŵ				
	ImageIndex	(なし)					
	ImageKey	(なし)					
	ImageList	(なし)					
	RightToLeft	No	×.				
	Text C	\checkmark					

以上でウィンドウ画面の作成は終了です。



Form1.cbl [デザイン]	→ × Program1.cbl	
E Form1	Say Hello	
	】 設定後のフォーム イメージ	

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.

Form1.cbl → ×	Form1.cbl [デザイン] Program1.cbl
🔩 WinHello.Form	1 *
	class-id WinHello.Form1 is partial inherits type System.Windows.Forms.Form.
	working-storage section.
	method-id NEW. procedure division. invoke self::InitializeComponent goback. end method.
	method-id button1_Click final private. procedure division using by value sender as object e as type System.EventArgs. move "Hello World!" to self::label1::Text. end method.



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。

M	TutorialSol - Micro	osoft Visual Studio					
771	イル(F) 編集(E) #	表示(V) プロジェクト(P)	Ľ)V	ド(B) デバッグ(D)	チーム(M)	ツール(T)	テスト(S)
	3 • 🗇 🏠 • 😩	🗎 🖉 🦻 - C -	*	ソリューションのビルド	(B)	Ctrl	+Shift+B
4	Form1.cbl + × F	- orm1.cbl [デザイン]		ソリューションのリビル	ド(R)		
1	🔩 WinHello.Form1			ソリューションのクリーン	7(C)		
5	Ę	class-id ∛inHello.		ソリューションでコード	分析を実行(Y)	Alt+	-F11
۲ ۲		inherits	×.	WinHello のビルド(U	Ŋ		
-j-		working-storage se		WinHello のリビルド	(E)		
				WinHello のクリーン	(N)		
у–Ји		method-id NEW. procedure division		バッチ ビルド(T)…			
নি খ		invoke self::I		構成マネージャー(O).			

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。

		•	-בעע	・ション エクスプロ・	-ラ-	v .	η×
		-	G	ə 🔂 🗂 -	™ 5	a 🖻 🌶	**
		++ relition and and relian relian t	ען - עם עם	-ション エクスプロ・ ソリューション 'Tut ⑩ ConsoleHell	ーラーの検索 orialSol' (2 o es 1.cbl	= + + + + + + + + + + + + + + + + + + +	ρ-
	ษัแสดก			BL WinHello			
				Propertie 未限設定	25		
	クロット(E) /7川ーン/ND			Form1.cb	- ol		
	クリーフ(N) ハキニ(7)			The Form	1.resx		
	分析(乙)		•	Porm	1.Designer	.cbl	
<u> </u>	HockeyApp (昭布する			👌 Main.cbl	l		
	ここまで検索(S)						
	新しいソリューション エクスプローラーのビュー	(N)					
乙	コードマップに表示(C)			シーン・エクスプロ			
	ビルド依存関係(B)		•	-937 19X70-	-7- 7-1	<u>, エクスノローラー</u>	1
	追加(D)		•	71		-	Ψ×
	既存の COBOL 項目を追加			lello プロジェクト	プロパティ		-
Ħ	NuGet パッケージの管理(N)			4 1			
Ø	スタートアップ プロジェクトに設定(A))他			
	デバッグ(G)		•	ジェクトパス	C:¥wo	rk¥Tutorial¥do	tNet¥

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。

💀 Form1	_	×
Say Hello		
> Hello World!		

[×] 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ヘ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。 まだ、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。

FIL	e se	CTION.		
FD	IN-	FILE.		
01	IN-	REC	PIC	X (50).
FD	0UT	-FILE.		
01	0UT	-REC.		
	05	OUT-EMPNO	PIC	9(8).
	05	FILLER	PIC	Х.
	05	OUT-JNAME1	PIC	N(5).
	05	OUT-JNAME2	PIC	N(5).
	05	OUT-NAME1	PIC	Х(5).
	05	OUT-NAME2	PIC	Х(5).
	05	OUT-GENDER	PIC	Х.
	05	FILLER	PIC	Х.
	05	OUT-DIV	PIC	N(5).
	05	OUT-EMPDATE	PIC	9(8).
	05	FILLER	PIC	Х.





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	JutorialSol - Microsoft Visual Studio
	ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S)
ドウにビルド結果	🔹 🕒 🗸 😂 🔛 🔐 🎐 - 🤍 - 📩 <u>ソリューションのビルド(B)</u> Ctrl+Shift+B
が表示されるの	ソリューションのリビルド(R)
	ソリューションのクリーン(C)
で、すべてのビル	T PROGRAM I
いおて豊物フィモ	O5 FILLER
トか正吊絵」した	📮 📄 procedure division 🔛 LoadCSVFileのビルド(U)
ことを確認しま	E PROC1. LoadCSVFileのリビルド(E)
	OPEN INPUT LoadCSVFileのクリーン(N)
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 2222226,鈴木,尚之,スズキ,ナオユキ,M,技術部,19981015,0 3333339,田中,直美,タナカ,ナオミ,F,総務部,19990401,0 44444442,山田,洋一,ヤマダ,ヨウイチ,M,営業部,20000701,0 55555555,伊藤,弘子,イトウ,ヒロコ,F,技術部,20010401,0 666666668,木村,貴弘,キムラ,タカヒロ,M,営業部,20021220,0 77777771,中村,慎司,ナカムラ,シンジ,M,技術部,20030401,0 88888884,橋本,悦子,ハシモト,エツコ,F,総務部,20040805,0 99999997,三井,薫,ミツイ,カオル,F,営業部,20050401,0

Debug				- 🗆	×
← → • ↑ <mark>-</mark> «	TutorialSol > LoadCSVFile > bin > x8	6 → Debug 🛛 🗸	・ ひ Debugの検索		Q
뢒 ባイック アクヤス	^ 名前 ^	更新日時	種類	サイズ	
= = 7/1 km 1	Emp_Master.csv	2017/11/17 18:25	CSV ファイル	1 KB	
	LoadCSVFile.exe	2017/11/17 18:23	アプリケーション	17 KB	
♣ 9720-F	Program1.idy	2017/11/17 18:23	IDY ファイル	43 KB	
🛗 ドキュメント	*				
📰 ピクチャ	*				
	✓ を選択 436 バイト				



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。

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

COBOL ウォッチポイントを追加 PROC2. READ IN-FIL COBOL ウォッチボイントを追加 COBOL プログラム ブレークポイントを追加 INITIALIZE UNSTRING 1 COBOL デバッグ ツールチップ スタイル ٠ TN INTO OUT-↔ ウォッチ式の追加(W) 🗲 OUT-🔂 クイック ウォッチ(Q).. Shift+F9 ŎŬŤ-スニペット(S) ۲ OUT OIITж 切り取り(T) Ctrl+X OUT-OUT- D コピー(Y) Ctrl+C OUT-貼り付け(P) Ctrl+V END-UNSTRING WRITE OUT-F アウトライン(L) ٠



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 は出カファイルへ書き出す最初 のレコードが表示されます。

\$		EN WR GO PROC9.	D-UNSTRING. ITE OUT-REC. TO PROC2.	
110 %	0	•		
ウォッ	£1			
名前	Ú		值	型
	9	IN-REC	11111113,佐藤、隆, サトウ, タカシ, M, 営業部, 19980401,0 Q、	PIC X(50)
-	9	OUT-REC	{長さ=60}: "11111113 佐藤 隆 サトウ タカシ M 営きQ •	GROUP
		OUT-EMPNO	1111113	PIC 9(8)
		FILLER	٩	PIC X
		OUT-JNAME1	佐藤 Q -	PIC N(5)
		OUT-JNAME2	隆 Q.▼	PIC N(5)
		OUT-NAME1	<u> </u>	PIC X(5)
		OUT-NAME2	<u>ዓ</u> ታን ር	PIC X(5)
		OUT-GENDER	M Q.+	PIC X
		FILLER	Q	PIC X
		OUT-DIV	営業部 Q マ	PIC N(5)
		OUT-EMPDATE	19980401	PIC 9(8)
\searrow		FILLER	Q.+	PIC X

さらに F11 キーを 4 回押す と、デバッガーは initialize 文を 実行後、処理を中断します。

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





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。

出力	
出力元(S): デバッグ	- <u>2</u> <u>2</u> <u>2</u> <u>2</u>
スレッド '<名前がありません>' スレッド '<名前がありません>'	(0x1374) はコード 0 (0x0) で終了しました。 (0x1508) はコード 0 (0x0) で終了しました。
プログラム '[0×1134] LoadCSVF	(axadd) Tali To (axa) Curr Dok Ore ile.exe: c:¥work¥tutorial¥dotnet¥tutorialsol¥loadcsvfile¥bin¥x86¥debug¥LoadCSVFile.exe'はコード 0 (0x0) で終了しました。

デバッグフォルダ(<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。

Rebug					- 0	×
← → * ↑ 📙 «	TutorialSol	\rightarrow LoadCSVFile \rightarrow bin \rightarrow x86	> Debug	・ ひ Debugの検索		Q
🛃 ካイック アクセス	^	名前 ^	更新日時	種類	サイズ	
= = -7.0 km ⁻¹		Emp_Master.csv	2017/11/17 18:25	CSV ファイル	1 K	В
	\rightarrow	📋 Emp_Master.dat	2017/11/17 18:49	COBOL データファイル	1 K	В
	R	LoadCSVFile.exe	2017/11/17 18:23	アプリケーション	17 K	В
≝ ドキュメント	*	Program1.idy	2017/11/17 18:23	IDY ファイル	43 K	В
📰 ピクチャ	*					
 4 個の項目 1 個の項目	✔ 目を選択 436	5 パイト				

🗃 Emp_Master.dat - TeraPad	—		×
ファイル(<u>E</u>) 編集(<u>E</u>) 検索(<u>S</u>) 表示(<u>V</u>) ウィンドウ(<u>W</u>) ツール(<u>T</u>) ヘルプ(<u>H</u>)			
🚨 😂 🗳 👗 🐚 💼 🖉 🗠 의 🔎 🚱			
■ Io Io	998040 998101! 999040 000070 001040 002122! 003040 004080! 005040	, 160 1 1 1 1 1 1 1 1 1 5 1 1 1 1 1 1 1 1 1	
<			>
1行: 1桁 標準 [60] SJIS	CRLF	挿入	



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ヘ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.	
**************************************	*****
* Input Files = Employee Extract File (Sequential)	*
* Selection Control Card	*
* Output Filo - Employee Vre Employed Penert	*
* Output File – Employee IIS 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.	
<pre>During two two two two two two two two two two</pre>	
ファイルに 日本に いるい いちゅう studio ファイル (人) 福集(E) 表示(L) ガジェクト(P) ビルド(B) デパッグ(D) チーム(M) ツール(D) テスト(S) 分析(M) ウインドウ	(<u>₩)</u> ∧ <i>I</i> \ <i>I</i> (<u>H</u>) Yoshihiro Mitsutomi • <mark>YM</mark>
S S S S S S S S S S S S S S S S S S S	▼ 10 0 0 - 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
PROGRAM1	<u>-</u> 00台台- 10-50 "
PROGRAM-ID. BATCHRPT.	★ ソリューション エクスプローラー の検索 (Ct クマ
* This program processes files: * Input Files = Employee Extract File (Sequential) *	ig_ 791-937 lutonalSol (4 7091 ▲
* Selection Control Card * * Output File = Employee Yrs Employed Report *	BATCHRPT.cbl
	ConsoleHello Properties
ENVIRONMENT DIVISION.	Program1.cbl
	A CN LoadCSVEile
FILE-CONTROL.	CIL LoadCSVFile Properties Description
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EMPLOYEE ASSIGN TO UT-S-EMPSEQ.	CoadCSVFile Properties Program1.cbl City Winhello
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT ENC_SEQ_FILE * INPUT FILE: DATE SELECTION CRITERIA SELECT INCOTL-CARD SELECT INCOTL-CARD SELECT INCOTL-CARD	CloadCSVFile Properties ProgramLcbl WinHello WinHello VIII-5/32/T0.72.1
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EUE-SECUEILE * INPUT FILE: DATE SELECTION CRITERIA SELECT IN-CULL-CARD SELECT IN-CULL-CARD ASSIGN TO UT-S-CNTLCARD. * ONTPUT REPORT FILE	
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EMPLOYEE RECORDS * INPUT FILE: DATE SELECTION CRITERIA SELECT EMPLOYEE ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT EMPLOYEE ASSIGN TO UT-S-HIRERPT. data division	
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EMPLOYEE ASSIGN TO UT-S-EMPSEO. * INPUT FILE: DATE SELECTION CRITERIA SELECT IU-CHIL-CARD ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT	▲ 団 LoadCSVFile ● ProgramLcbl ● ProgramLcbl ● Denorative ● Denora
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EMPLOYEE RECORDS * INPUT FILE: DATE SELECTION CRITERIA SELECT IN-CALL-CARD * OUTPUT REPORT FILE SELECT EMPLANE-RET data division. Procedure division.	▲ CadCSVFile
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EMPLOYEE RECORDS * INPUT FILE: DATE SELECTION CRITERIA SELECT ENC.CULL-CARD ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT ENC.MILEE-RET ASSIGN TO UT-S-HIRERPT. data division. working=storage section. procedure division. goback.	▲ 団 LoadCSVFile ● Program Lob ● Program Lob ▲ 団 WinHello ● Demontion ● Demo
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EUC_SEQUENCE ASSIGN TO UT-S-EMPSEO. * INPUT FILE: DATE SELECTION CRITERIA SELECT IN-COLLI-CARD ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT EMP-HIRE-BEI ASSIGN TO UT-S-HIRERPT. data division. working=storage section. procedure division. goback. end program Program.	
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EMPLOYEE RECORDS SELECT EMPLOYEE RECORDS * INPUT FILE: DATE SELECTION CRITERIA SELECT EMPLOATE ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT EMPLANEERET data division. goback. end program Program1	▲ ④ LoadCSVFile
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EMPLOYEE RECORDS SELECT EMPLOYEE RECORDS * INPUT FILE: DATE SELECTION CRITERIA SELECT EMPLANEL.CARD. ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT EMPLANEL.BET ordet division. goback. end program Program!. 110 % Source YUJ1-YB2Y2f# YUJ1-YB2Y2f# YU1-YB YU1-YB </td <td>▲ ○ LoadCSVFile ● ProgramLcbl ● ProgramLcbl ●</td>	▲ ○ LoadCSVFile ● ProgramLcbl ●
FILE-CONTROL. * INPUT FILE: ENPLOYEE RECORDS SELECT ENPLOYEE RECORDS SELECT ENPLOYEE RECORDS SELECT ENPLOYEE RECORDS SELECT ENPLOYEE RECORDS SELECT ENPLOYEE ASSIGN TO UT-S-ENPSED. * INPUT FILE: SELECTION CRITERIA SELECT ENPLOYEE ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT ENPLOYEE ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT ENPLOYEE ASSIGN TO UT-S-HIRERPT. • OUTPUT REPORT FILE SELECT ENPLOYEE SECTION. • OUTPUT REPORT FILE SELECT ENPLOYEE ASSIGN TO UT-S-HIRERPT. • OUTPUT REPORT FILE SELECT ENPLOYEE SECTION. • OUTPUT REPORT FILE SELECT ENPLOYEE ASSIGN TO UT-S-HIRERPT. • OUTPUT REPORT FILE SELECT ENPLOYEE SECTION. • OUTPUT REPORT FILE SELECT ENPLOYEE ASSIGN TO UT-S-HIRERPT. • OUTPUT REPORT FILE SELECT ENPLOYEE SECTION. • OUTPUT REPORT FILE	▲ ○ LoadCSVFile ● Properties ● ProgramLcbl ▲ ○ WinHello ▲ ○ WinHello ▲ ○ WinHello ▲ ○ ProgramLcbl ▲ ○ ProgramLcbl ■ ○ ProgramLcbl
FILE-CONTROL. * INPUT FILE: EMPLOYEE RECORDS SELECT EMELSEQETLIE ASSIGN TO UT-S-EMPSEO. * INPUT FILE: DATE SELECTION CRITERIA SELECT EMELSEQET ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT EMELSERT ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT EMELSERT ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT EMELSERT ASSIGN TO UT-S-HIRERPT. data division. goback. end program Ptogram!. 100 % YUZ->izyzk* Query YUZ->izyzk* Query SCOBCH0244 7/1/WEMP-SEQ-FILE [2対するFD 認識類が知/ BATCHRPT BATCHRPT BATCHRPT BATCHRPT BATCHRPT BATCHRPT BATCHRPT	▲ ④ LoadCSVFile ● ProgramLcbl ● ProgramLcbl ▲ ④ WinHello ● ProgramLcbl ▲ ⑤ WinHello ● ProgramLcbl ▲ ⑤ WinHello ● ProgramLcbl ● Pr
FILE-CONTROL. * INPUT FILE: EMPLOYFE RECORDS SELECT EMPLOYFE RECORDS SELECT EMPLOYFE RECORDS SELECT EMPLOYFE RECORDS ASSIGN TO UT-S-EMPSEO. * INPUT FILE: DATE SELECTION CRITERIA SELECT EMPLANDICABLE ASSIGN TO UT-S-CNTLCARD. * OUTPUT REPORT FILE SELECT EMPLANEL.BET ASSIGN TO UT-S-HIRERPT. data division. goback. end program Program!. 100% ID-% VUI-vsvyzkf Q 4 IJ- A 0086 0 0087 VJz-vsvyzkf 2 -F EMM 2 -F EMM 2 -F 2 -F <td></td>	
FILE-CONTROL. * INPUT FILE: ENPLOYEE RECORDS SELECT END: SELECTION CRITERIA SELECT UN: CALL CARDAR ASSIGN TO UT-S-EMPSEO. * INPUT FILE: SELECTION CRITERIA SELECT UN: CALL CARDAR ASSIGN TO UT-S-CHILCARD. * OUTPUT REPORT FILE SELECT UN: CALL CARDAR ASSIGN TO UT-S-CHILCARD. * OUTPUT REPORT FILE SELECT UN: CALL CARDAR ASSIGN TO UT-S-CHILCARD. * OUTPUT REPORT FILE SELECT UN: CALL CARDAR ASSIGN TO UT-S-CHILCARD. * OUTPUT REPORT FILE SELECT END: CALL CARDAR ASSIGN TO UT-S-HIRERPT. data division. goback. end program @rogram. 110% * UTP-TR VUI>292/2/k • Q 4 ID-1 © COBCH0244 ??/// EMP-SEQ-FILE EXTROL © COBCH0244 ??/// EMP-HIRE-RPT EXTROL © COBCH0244 ??/// EMP-HIRE-RPT EXTROL FOR EXTROL & BATCHRPT BATCHRPTL & COBCH0244 © COBCH0244 ??/// EMP-HIRE-RPT EXTROL FOR EXTROL & BATCHRPT BATCHRPTL & COBCH0244 © COBCH0244 ??/// EMP-HIRE-RPT EXTROL * SED EXTROL & BATCHRPT BATCHRPTL & COBCH1570 © COBCH1570 End 0.9.4 TPACEMAM'' * MARTA/>* YARD * AXAV	▲ ④ LoadCSVFile ● ProgramLcbl ● ProgramLcbl ▲ ④ WinHello ● ProgramLcbl ▲ ④ WinHello ● ProgramLcbl ▲ ⑤ WinHello ● ProgramLcbl ● Pr



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).
- FDIN-CNTL-CARD
LABEL RECORDS ARE STANDARD.01CONTROL-RECORDPIC 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。

<ing-< th=""><th>-STORAGE SECTION.</th><th></th><th></th><th></th><th></th></ing-<>	-STORAGE SECTION.				
PROC	GRAM-FIELDS.				
05	EOF-FLAG	PIC	X(01)	VALUE	'N'.
	88 AT-EOF			VALUE	Ϋ́.
	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.
CONT	TROL-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.
	(ING- PROC 05 05 05 05 05 05 05 05	<pre><ing-storage 05="" 10="" 88="" at-eof="" cntl-date.="" cntl-day<="" cntl-mon="" control-rec.="" counters.="" curr-date.="" curr-dd="" curr-hr="" curr-min="" curr-mm="" curr-sec="" curr-time.="" emp-rec-cntr="" eof-flag="" line-ctr="" line-max="" not-at-eof="" pre="" program-fields.="" section.="" yrs-employed=""></ing-storage></pre>	KING-STORAGE SECTION.PROGRAM-FIELDS.05EOF-FLAG91088AT-EOF88NOT-AT-EOF05COUNTERS.10EMP-REC-CNTR10LINE-CTR10LINE-MAX910URR-DATE.10CURR-DATE.10CURR-MM10CURR-DD10CURR-MM10CURR-DD10CURR-TIME.10CURR-SEC10CURR-SEC10CURR-SEC10CNTL-DATE.10CNTL-DATE.10CNTL-MON10CNTL-DAY10CNTL-DAY10CNTL-DAY	(ING-STORAGE SECTION. PROGRAM-FIELDS. 05 EOF-FLAG PIC X (01) 88 AT-EOF 88 NOT-AT-EOF 05 COUNTERS. 10 EMP-REC-CNTR PIC 9 (05) 10 LINE-CTR PIC 9 (03) 10 LINE-MAX PIC 9 (03) 05 CURR-DATE. 10 10 CURR-MM PIC 9 (2). 10 CURR-TIME. 10 10 CURR-HR PIC 9 (2). 10 CURR-SEC PIC 9 (2). 05 CNTL-DATE. 10 10 CNTL-REC. 05 05 CNTL-DATE. 10 10 CNTL-MON PIC X (2) 10 CNTL-DAY PIC X (2)	(ING-STORAGE SECTION. PROGRAM-FIELDS. 05 EOF-FLAG PIC X (01) VALUE 88 AT-EOF VALUE 88 NOT-AT-EOF VALUE 05 COUNTERS. 10 EMP-REC-CNTR PIC 9 (05) VALUE 10 LINE-CTR PIC 9 (03) VALUE 10 LINE-MAX 10 LINE-MAX PIC 9 (03) VALUE 05 05 CURR-DATE. 10 CURR-MM PIC 9 (2). 10 CURR-MM PIC 9 (2). 10 CURR-DD 10 CURR-MM PIC 9 (2). 10 CURR-MIN 10 CURR-MIN PIC 9 (2). 10 CURR-SEC 10 CURR-SEC PIC 9 (2). 10 COMP-3 05 YRS-EMPLOYED PIC 9 (2). 03 COMP-3 05 CNTL-DATE. 10 CNTL-YR PIC X (4) VALUE 10 CNTL-MON PIC X (2) VALUE 10 10 CNTL-MON PIC X (2) VALUE 10

- ** 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。

**	Repo	rt 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
		'入社曰'.			
	05	FILLER	PIC	X (14)	VALUE
		'雇用年数'.			

BATCHRPT.cbl 👳 🗙 P	Program1.cbl	-
4 PROGRAM1	✓ Ø Procedure Division	-
-	US FILLER PIC X(5) VALUE SPACE.	÷
	US FILLER PIC X(5) VALUE SPACE. I RPT-TITIE-3. OS FILLER PIC X(5) VALUE SPACE. OS FILLER PIC X(7) VALUE '#***** '. OS RPT-SELECTION-YYYY PIC 9(4). OS FILLER PIC X(2) VALUE '#'. OS RPT-SELECTION-MM PIC Z9. OS FILLER PIC X(2) VALUE '月'. OS RPT-SELECTION-DD PIC Z9. OS FILLER PIC X(24) VALUE '1'. OS FILLER PIC X(12) VALUE SPACE. I RPT-COLUMNS. OS FILLER PIC X(12) VALUE SPACES. OS FILLER PIC X(11) VALUE '2'. OS FILLER PIC X(14) VALUE '2'. OS FILLER PIC X(15) VALUE '2'. OS FILLER PIC X(14) VALUE '2'. OS FILLER PIC X(15) VALUE '2'. OS FILLER PIC X(14) VALUE '2'. OS FILLER PIC X(14) VALUE '2'. OS FILLER PIC X(15) VALUE '2'. OS FILLER PIC X(14) VALUE '2'. OS FILLER PIC X(15) VALUE '2'. OS FILLER PIC X(14) VALUE '2'. OS FILLER PIC X(15) VALUE '2'. OS FILLER PIC X(14) VALUE '2'. OS FILLER PIC X(15) VALUE '2'. OS FILLER PIC X(15) VALUE '2'. OS FILLER PIC X(15) VALU	
110 % 👻 🖣	4	



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	Х	VALUE	SPACES.
	05	RPT-EMP-NAME	PIC	N(10)	VALUE	SPACES.
	05	FILLER	PIC	Х	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	Х	VALUE	'/'.
	05	RPT-EMP-HIRE-DD	PIC	X (2)	VALUE	SPACES.
	05	FILLER	PIC	Х	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	ррт					
01		-IUIAL-LINE.	DIC	V (E)		SDACE
	05			X (3) X (7)		SFAUE.
	05	FILLER DDT MCC		$\Lambda(I)$ $\chi(20)$		ጥጥጥጥ . የDACE
	05	RPI-MOG		X(30)	VALUE	SPACE.
	05	FILLER		Λ(Z) 777	VALUE	SPAGE.
	05	KF1-101-KEC2	PIC	LLL.		

UI BLANK-LINE	01	BL	ANK-	-L I	NE
---------------	----	----	------	------	----

05 FILLER

PIC X(80) VALUE SPACE.

PIC X (33) VALUE SPACE.

BATCHRPT.cbl* ↔ ×	Program1.cbl		•
1 PROGRAM1		 Procedure Division 	•
	「人社日」。 O5 FILLER 「雇用年数」。	PIC X(14) VALUE	
C 01	RPT-DETAIL-LINE. 05 FILLER 05 RT-EMP-DIV 05 FILLER 05 RT-EMP-NAME 05 FILLER 05 RT-EMP-NAME 05 FILLER 05 RT-EMP-SSN 05 FILLER 05 RT-EMP-HIRE-MM 05 FILLER 05 RT-EMP-HIRE-YYYY 05 FILLER 05 FILLER 05 FILLER 05 FILLER	PIC X(5)VALUE SPACES.PIC N(5)VALUE SPACES.PIC XVALUE SPACES.PIC N(10)VALUE SPACES.PIC X(9)VALUE SPACES.PIC X(9)VALUE SPACES.PIC X(2)VALUE SPACES.PIC X(4)VALUE SPACES.PIC X(5)VALUE SPACES.PIC X(12)VALUE SPACE.	
E 01	RPT-TOTAL-LINE. OS FILLER OS FILLER OS RPT-MSG OS FILLER OS RPT-TOT-RECS OS FILLER BLANK-LINE	PIC X(5) VALUE SPACE. PIC X(7) VALUE '***** '. PIC X(30) VALUE SPACE. PIC X(2) VALUE SPACE. PIC ZZZ. PIC X(33) VALUE SPACE. PIC X(80) VALUE SPACE.	1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
Pro	cedure division.		
110 % 🔹 🖣			



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.

MOVECURR-HRTORPT-CURR-HR.MOVECURR-MINTORPT-CURR-MIN.MOVECURR-SECTORPT-CURR-SEC.

WRITE RPT-RECORD FROM RPT-TITLE-1 BEFORE ADVANCING 1 LINE. WRITE RPT-RECORD FROM RPT-TITLE-2 BEFORE ADVANCING 1 LINE.

PROGRAM1	- 🛛 Procedure Division	
-	01 BLANK-LINE PIC X(80) VALUE SPACE.	
	PROCEDURE DIVISION. PERFORM 1000-START THRU 1000-EXIL. PERFORM 2000-NAIN-PROCESSING THRU 2000-EXIL UNTIL AT-EOF. PERFORM 9000-CLOSE-AND-CLEANUP THRU 9000-EXIL. 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 ***	
Ē	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.	
	goback -	



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 T0 RPT-SELECTION-MM. MOVE CNTL-DAY Τ0 RPT-SELECTION-DD. MOVE CNTL-YR T0 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	Т0	RPT-EMP-DIV.
	MOVE SPACE STRING EMPREC-JNAME1 DEI SPACE DEI EMPREC-JNAME2 DEI	TO LIMITI LIMITI LIMITI INTO	RPT-EMP-NAME. ED BY SPACE ED BY SIZE ED BY SPACE RPT-EMP-NAME.
	STRING EMPREC-SSN (1:7) '_' EMPREC-SSN (8:1)	DEL II DEL II DEL II INTO	MITED BY SIZE MITED BY SIZE MITED BY SIZE RPT-EMP-SSN.
	MOVE EMPREC-DOH-MM MOVE EMPREC-DOH-DD MOVE EMPREC-DOH-YYYY PERFORM 4000-COMPUTE-YEAR	TO TO TO S-EMPI	RPT-EMP-HIRE-MM. RPT-EMP-HIRE-DD. RPT-EMP-HIRE-YYYY. LOYED THRU 4000-EXIT.
	MOVE YRS-EMPLOYED WRITE RPT-RECORD ADD 1 TO EMP-REC-CNTR.	TO FROM BEFOI	RPT-EMP-YRS-EMPL. RPT-DETAIL-LINE RE ADVANCING 1 LINE.

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.

BATCHRPT.cbl 😕 🗙 Program1.cbl	•
◆g PROGRAM1	-
INTO RPT-EMP-NAME.	+++++++++++++++++++++++++++++++++++++++
STRING <u>ENPREC-SSN(1:7)</u> DELIMITED BY SIZE DELIMITED BY SIZE EMPREC-SSN(8:1) DELIMITED BY SIZE INTO RPT-EMP-SSN.	
MOVE ENPREC-DOH-NM 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.	
↓ 4000-COMPUTE-YEARS-EMPLOYED.	
<pre>**** * DETERMINE YEARS OF EMPLOYMENT BY SUBTRACTING HIRE YEAR * FROM CURRENT YEAR. ****</pre>	
COMPUTE YRS-EMPLOYED = CURR-YYYY - EMPREC-DOH-YYYY.	
4000-EXIT. EXIT.	AL.
goback -	
end program Program1.	•



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 > 0MOVE '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. BATCHRPT.cbl 😐 🗙 Program1.cbl 🔩 PROGRAM1 4000-EXIT. EXIT. 9000-CLOSE-AND-CLEANUP. IF EMP-REC-CNTR > 0 MOVE '処理レコード件数:' MOVE EMP-REC-CNTR TO RPT-MSG 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. goback 文及び END EMP-SEQ-EILE CL OSE IN-CNTL-CARD EMP-HIRE-RPT. PROGRAM は削除します。 9000-EXIT. EXIT. _____ -110 % エラー一覧 • ¤ × 🗲 🚺 12 エラー 🔪 🔥 0 警告 📔 🛈 0 メッセージ 🛛 🌾 - 覧を検索 Q ソリューション全体 エラー ----" J-F 詳明 プロジェクト ファイル COBCH0008 コピーブック EMPSEQ が見つからない BATCHRPT BATCHRPT.cbl 8 COBCH0217 このレベルの前述項目の長さがゼロである BATCHRPT.cbl BATCHRPT

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。

作用対象 EMPREC-DATE-OF-HIRE が宣言されていない

BATCHRPT

BATCHRPT.cbl

S COBCH0012



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。

1					 עבעיבעע עביבעע עבעע ₪	<mark>ビクスプローラー</mark> 聞 ・ で) ・ ち ビクスプローラー の検索 ンョン 'TutorialSol' (4	▼早× 『『 (Ct ク ・ プロジェ
			1 1 1 1 1 1 1 1	ビルド(U) リビルド(E) クリーン(N) ここまで検索(S) 新しいソリューション エクスプローラーのビュー(N) コード マップに表示(C) ビルド依存関係(B)		CHRPT Properties BATCHRPT.cbl soleHello Properties Program1.cbl dCSVFile Properties Program1.cbl	
*1	新しい項目(W)	Ctrl+Shift+A		追加(D)	•	Properties	
to.	既存の項目(G)	Shift+Alt+A		既存の COBOL 項目を追加		参照設定	-
*	新しいフォルダー(D)		ĺ₿.	NuGet バッケージの管理(N)			•
t₽	接続済みサービス(C)		Ф	スタートアップ プロジェクトに設定(A)	[<mark>「クスプ…</mark> 」チームエク	スプローラー

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。

新しい項目の追加 - BATCHRPT					?	×
▲ インストール済み	並べ替え	既定	• # E	インストール済み テンプレートの検索	(Ctrl+E	- م
▲ COBOL プロジェクト項目 Managed	٥	COBOL プログラム	COBOL プロジェクト項目	種類: COBOL プロジェクト項目 コピーブックを新規作成します。		
Native	Ж	テストプログラム	COBOL プロジェクト項目			
▶ オンライン	D	コピーブック	COBOL プロジェクト項目			
	Ð	アプリケーション 構成 ファイル	COBOL プロジェクト項目			
	D	リソース ファイル	COBOL プロジェクト項目			
	Ð	アプリケーション マニフェスト	COBOL プロジェクト項目			
名前(<u>N</u>): EMPSEQ.cpy				き加山	キャンヤ	IL



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。

05	EMP	-REC.					
	10	EMPRE	C-SSN	PIC	X (08)	VALUE	SPACE.
	10	FILLE	R	PIC	X(01)	VALUE	SPACE.
	10	EMPRE	C-JNAME1	PIC	N(05)	VALUE	SPACE.
	10	EMPRE	C-JNAME2	PIC	N(05)	VALUE	SPACE.
	10	EMPRE	C-NAME1	PIC	X (05)	VALUE	SPACE.
	10	EMPRE	C-NAME2	PIC	X (05)	VALUE	SPACE.
	10	EMPRE	C-GENDER	PIC	X(01)	VALUE	SPACE.
	10	FILLE	R	PIC	X(01)	VALUE	SPACE.
	10	EMPRE	C-DIV	PIC	N(05)	VALUE	ZERO.
	10	EMPRE	C-DATE-OF-HIRE.				
		15 E	MPREC-DOH-YYYY	PIC	9 (04)	VALUE	ZEROES.
		15 E	MPREC-DOH-MM	PIC	9 (02)	VALUE	ZEROES.
		15 E	MPREC-DOH-DD	PIC	9 (02)	VALUE	ZEROES.
	10	FILLE	R	PIC	X(01)	VALUE	SPACE.

EMPSEQ.cpy 😐 🗙 🛛	BATCHRPT.cbl	Program1.cbl			
🔩 EMPSEQ					
- *	EMPLOYEE SEG	QUENTIAL FILE LAYOU	JT		*
	TO EMPF IQ FILL IQ EMPF IQ EMPF	REC-SSN LER REC-JNAME1 REC-JNAME2 REC-NAME1 REC-NAME2 REC-GENDER LER REC-DIV REC-DIV REC-DATE-OF-HIRE. EMPREC-DOH-MM EMPREC-DOH-DD LER	PIC X(08) PIC X(01) PIC N(05) PIC X(05) PIC X(05) PIC X(01) PIC X(01) PIC X(01) PIC N(05) PIC 9(04) PIC 9(02) PIC 9(02) PIC X(01)	VALUE SPACE. VALUE SPACE. VALUE SPACE. VALUE SPACE. VALUE SPACE. VALUE SPACE. VALUE SPACE. VALUE SPACE. VALUE SPACE. VALUE ZEROES VALUE ZEROES VALUE ZEROES 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。

刘 🛛 TutorialSol - Microsoft Visual Studio	
ファイル(F) 編集(E) 表示(V) プロジェクト(P)	ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S)
G - O 🔠 - 🖆 💾 🌮 🤊 - 🤆 -	📩 ソリューションのビルド(B) Ctrl+Shift+B
	ソリューションのリビルド(R)
	ソリューションのクリーン(C)
€ *	ソリューションでコード分析を実行(Y) Alt+F11
· EMPLOYEE SEQUE	🛗 BATCHRPT のビルド(U)
□ *	BATCHRPT のリビルド(E)
10 EMPREC	BATCHRPT のクリーン(N)
	バッチ ビルド(T)
	構成マネ−ジャ−(O)

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 エラーー 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。

BATCHRPT 🗢 🗙 BATCHRPT.cbl	Program1.cbl			-
アプリケーション	進成(の)。 マクティブな (Debue)			
SQL	構成(C): アクティアな (Debug)			
コピーブック				^
プリプロセッサ	最大エラー数:	100 曾告をエラーとして処理		1
COBOL	出力			
COBOL リンク	山 力 パフ.	VL:-V. 8CVD-LV	* 07	$\frac{1}{\sqrt{2}}$
デバッグ	四川八入:	.+bin+x00+Debug+	参照	
Micro Focus Code Analysis	□ 指令ファイルの生成	□ リストファイルを生成		
	🔲 コードカバレッジを有効に	する コプロファイラを有効にする		
	追加指令 ——			
	assign(external)		^	I.
			\sim	
			高度	¥



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 アイルを選択し、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。

新しい項目の追加 - BATCHRPT			? ×
▲ インストール済み	並べ替え: 既定 -	# E	インストール済み テンプレートの検索 (Ctrl+E 🔎 🗸
▲ COBOL プロジェクト項目 Managed	COBOL プログラム	COBOL プロジェクト項目	種類: COBOL プロジェクト項目 アプリケーションの設定を構成するために使うファ
Native	テスト プログラム	COBOL プロジェクト項目	イルです。
▶ オンライン		COBOL プロジェクト項目	
	アプリケーション 構成 ファイル	COBOL プロジェクト項目]
	אראד ג-עע 🚹	COBOL プロジェクト項目	
	アプリケーション マニフェスト	COBOL プロジェクト項目	
名前(N): Application.config			1 H H (A)
			1年かりセル

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。

マプリケ-	ションの設定	>
镜	COBOL スイッチ 実行時構成	
変	故值	
<	>	
名	ti dd_EMPSEQ	
値	Emp_Master.dat	
	設定創除	
O	く キャンセル	



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 ×								
環境	COBOL スイッチ 実行時構成							
ま 	变数 Id_EMPSEQ 名前 dd_CNTLCARD 值 Cntl_Card.dat	値 Emp_Master.dat >						
(OK キャンセル							

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									×	
環境 COBOL スイッチ 実行時構成										
	変数 dd_ dd_	ά EMPSEQ CNTLCARD				値 Emp_Master.d Cntl_Card.dat	lat			
	、名前	t dd_HIRERP1	r						/	
	値	Hire_Report	.dat							
								設定		
	OK	キャンセ	JL							

アプリケーションの設定で

OK をクリックします。

ブリケー	・ションの設定				
瞶	COBOL スイッチ	実行時構成			
কা	ST .		店		
1 dd	EMPSEO		Emp Masterdat		
dd	CNTLCARD		Cntl_Card.dat		
dd	HIRERPT		Hire_Report.dat		
<u>`</u>					
-					
-					
<					>
名i	iti				
値					
				設定	
_	<u> </u>				
O	() ++>>t	.JL			



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 Microsoft Visual Studio ソリューションのリビル ファイル(F) 編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 📩 ソリューションのビルド(B) G - O | 🎦 - 🖕 💾 🥐 | 🏸 - 🤆 - | Ctrl+Shift+B ド(R) を選択します。 ソリューションのリビルド(R) BATCHRPT 🕆 🗙 BATCHRPT.cbl サーバー エクスプローラー ソリューションのクリーン(C) 出力ウィンドウにビルド アプリケーション ソリューションでコード分析を実行(Y) Alt+F11 構成 結果が表示されるので、 SQL BATCHRPT のビルド(U) コピーブック ŧ BATCHRPT のリビルド(E) すべてのビルドが正常終 プリプロセッサ BATCHRPT のクリーン(N) 了したことを確認しま す。 出力 🖉 🖆 🎽 🖉 🐉 出力元(S): ビルド ŁoadCSVFile -> C:¥work¥Tutorial¥dotNet¥TutorialSol¥LoadCSVFile¥bin¥x86¥Debug¥LoadCSVFile.exe ----- すべてのリビルド開始: プロジェクト:BATCHRPT, 構成: Debug x86 ------* C:¥work¥Tutorial¥dotNet¥TutorialSol¥BATCHRPT¥BATCHRPT.cbl のコンパイル中 * C:¥work¥Tutorial¥dotNet+Iutorial60... * Generating C:obj¥x86¥Debug¥BATCHRPT 2700 Code: 3936 * Data: 2768 Code: 3936 Literals: COBOL コンパイル:1個正常終了または最新の状態 0個 失敗。 1272 BATCHRPT -> C: #work #Tutorial #dot Net #Tutorial Sol #BATCHRPT #bin #x86 #Debug #BATCHRPT.exe ====== すべてリビルド: 4 正常終了、0 失敗、0 スキップ ======== 呼び出し階層 パッケージマネージャー コンソール エラー一覧 出力 リビルドがすべて正常に終了しました。

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



🗃 Cntl_Card.dat - TeraPad	- 0	Х
ファイル(E) 編集(E) 検索(<u>S</u>) 表示(<u>V</u>) ウィンドウ(<u>W</u>) ツール(<u>T</u>)	ヘルプ(<u>H</u>)
🖸 🛱 💾 🍊 🗶 🐚 💼 🖉 🗠 🔎 💭 💭		
		<u></u> .
<		>
1行: 9桁 標準 [60] SJ	IS CRLF	挿入 .:



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。



BATCHRPT	BAT	CHRPT.cl	bl -¤	× Program1.cbl	
🔩 BATCHRPT					
-		05 Ý	RS-E	MPLOYED PIC 9(03) C	OMP-3 VALUE D.
	01	05 0	9	クイック アクションとリファクタリング	Ctrl+.
		1	X	名前の変更(R)	Ctrl+R, Ctrl+R
		i	•	定義に移動(G)	F12
	**	Employ		すべての参照を検索(A)	Shift+F12
>	01	COPY	Ζ	呼び出し階層の表示(H)	Ctrl+K, Ctrl+T
		0		コードマップ	•
ļ.	01	RPT-1		Micro Focus Code Analysis	•
		05 F	ŀ.	カーソル行の前まで実行(N)	Ctrl+F10
		05 F	2	次のステートメントの設定(X)	Ctrl+Shift+F10
		05 6	G.:	逆アセンブルを表示(D)	Alt+G
		05 F		COBOL ウォッチポイントを追加	
		05 6		COBOL プログラム ブレークポイントを追加	
		05 F		COBOL デバッグ ツールチップ スタイル	•
		05 F	⇔	ウォッチ式の追加(W)	
110 % 👻 🖣		- ^K	⇔	クイック ウォッチ(Q)	Shift+F9

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 オッチ式の追加(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。

-	BATCHRPT	BATCHRPT.cbl ⊉ 🗙 Program1.cbl
	🔩 BATCHRPT	
に続		1000-EXIT.
をク		EXIT.
设定し	□ 	2000-MAIN-PROCESSING. READ EMP-SEQ-FILE INTO EMP-RECORD-IO-AREA AT END MOVE 'Y' TO EOF-FLAG.
7	•	IF NOT-AT-EOF PERFORM 3000-PROCESS-RECORD THRU 3000-EXIT END-IF.
		2000-EXIT. EXIT.

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。

•	*** * READ CONTROL CAT * IF FILE IS EMPTY *** READ IN-CNTL-(IF CNTL-DATE : MOVE CURR- END-IF. * ACCEPT CNTL-DA	RD FILE TO GET DATE FOR Y, DEFAULT CNTL-DATE TO CARD INTO CONTROL-REC. SPACES DATE TO CNTL-DATE ATE FROM SYSIN.	SELECTION CRITERIA. CURRENT DATE.		
ウォッチ 1					- ₽ ×
名前		値			型 🔺
IN-REC		14-S 無効な作用対象があ	53	Ċ	
OUT-R	EC	<u>14-S 無効な作用対象</u> があ	53	Ō	
D 🥥 CONTR	OL-REC	{長さ=8}: "20110101"		٩	GROUP
🕨 🥥 EMP-R	ECORD-IO-AREA	{長さ=60}: "	0 0 0 0 0 0000000	0" Q,+	GROUP



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。

0	D 2000-MAIN-PROCES READ EMP-SEC AT END IF NOT-AT-EC PERFORM END-IF. 2000-EXIT. EXIT.	SING. -FILE INTO EMP-RECORD-IO-AREA MOVE 'Y' TO EOF-FLAG. FI 3000-PROCESS-RECORD THRU 3000-EXIT		
11	10 % 👻 🖣			
<u>ウ</u> :	1オツチ1			
	名前	值		型
	CONTROL REC	(長さ=8): "20110101"	٩.	GROUP
1	EMP-RECORD-IO-AREA	{長さ=60}: "11111113 佐藤 隆 サトウ タカシ M 営業部 19980401 "	Q	GROUP
1	EMP-REC	{長さ=60}: "11111113 佐藤 隆 サトウ タカシ M 営業部 19980401 "	Q, -	GROUP
	EMPREC-SSN	11111113	Q, -	PIC X(8)
	FILLER		Q, -	PIC X
	EMPREC-JNAME1	佐藤	Q, 7	PIC N(5)
_	EMPREC-JNAME2	隆	Q, -	PIC N(5)
	EMPREC-NAME1	<u> </u>	Q, -	PIC X(5)
	EMPREC-NAME2	9 1 5	Q, 7	PIC X(5)
	EMPREC-GENDER	M	۹	PIC X
	FILLER		Q, +	PIC X
	EMPREC-DIV	営業部	Q, ,	PIC N(5)
l	EMPREC-DATE-OF-HIRE	{長さ=8}: "19980401"	_Q,⊽	GROUP
_	EMPREC-DOH-YYYY	1998		PIC 9(4)
-	EMPREC-DOH-MM	04		PIC 99
V -	EMPREC-DOH-DD	01		PIC 99
N	FILLER		۹, ۰	PIC X

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。

名前	値					型
CONTROL-REC	{長さ=8}: "20110101"				Q	GR
EMP-RECORD-IO-AREA	{長さ=60}: "22222226 鈴木	尚之	スズキ ナオユキ M 技術部	19981015 "	Q, +	GRO
🔺 🥥 EMP-REC	{長さ=60}: "22222226 鈴木	尚之	スズキ ナオユキ M 技術部	19981015 "	Q	GRO
EMPREC-SSN	22222226				Q, +	PIC
FILLER					Q	PIC
EMPREC-JNAME1	鈴木				Q, +	PIC
EMPREC-JNAME2	尚之				Q	PIC
EMPREC-NAME1	77. \$				Q	PIC
EMPREC-NAME2	ŵ2				Q	PIC
EMPREC-GENDER	M				Q	PIC
FILLER					Q	PIC
EMPREC-DIV	技術部				Q	PIC
EMPREC-DATE-OF-HIRE	{長さ=8}: "19981015"				Q	GRC
EMPREC-DOH-YYYY	1998					PIC
EMPREC-DOH-MM	10					PIC
EMPREC-DOH-DD	15					PIC
FILLER					Q	PIC

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。

	BATCHRPT	BATCHRPT.cbl
	🔩 BATCHRPT	
		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.



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。

出力		
出力元(S):	デバッグ	- 🏝 🛳 🐸 🐉
スレッド スレッド スレッド プログラム	、〈名前がありません〉、(0x634)はコード 0(0x0) 、〈名前がありません〉、(0x1660)はコード 0(0x0) 、〈名前がありません〉、(0xb60)はコード 0(0x0) _A 、[5404] BATCHRPT.exe: c:¥work¥tutorial¥dotna	で終了しました。)で終了しました。 で終了しました。 st¥tutorialsol¥batchrpt¥bin¥x86¥debug¥BATCHRPT.exe'はコード 0 (0x0) で終了しました。

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。

	🖌 🚽 🛛 Debug			_	\Box \times
ファイル	ホーム 共有	表示			~ 🕐
← -	🗧 👻 🛧 📙 « Tutoria	alSol > BATCHRPT > bin > x86 > Del	bug 🗸 Ö	Debugの検索	Q
	Tutorial	▲ へ	更新日時	種類	サイズ
	dotNet	E DATCUERT ave	2017/11/20 17:11		20 KB
	TutorialSol	BATCHRPI.exe	2017/11/20 17:11	MEGGE 77K	20 KB
	NC NC		2017/11/20 17:09		2 ND
			2017/11/20 17:19	COBOL データファイル	1 KB
	BAICHRPT	Emp Master.dat	2017/11/17 18:49	COBOL データファイル	1 KB
	bin	Hire_Report.dat	2017/11/20 17:47	COBOL データファイル	2 KB
	x86				
	Debug				
	obj				
6個の1	「	1.36 KB			===
o Hoy.					
E Lin	a Roport dat - TaraDad				
	e_neport.dat - terarad				
77170	E) 編果(E) 使衆(S)	表示(M) りインドウ(M) ツール(I) ヘルフ(<u>H</u>)		
			150	100 170	
1	L.Program: BATCH	RPT	ed Report		7
Ż					94
3		1年。1月。1日以前に入社した社	員一覧		
5	部署名	.社員名社員習	号入社日。	雇用年数	
<u>6</u>				10	
	□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□	□15膝□ 唯□□□□□□□□□□□11111 鈴木□尚之□□□□□□□ 22222'	22-6 10/15/1	998	+000000
) ğ	総務部□□	□田中□直美□□□□□□□□333333	33-904/01/1	999	
	□営業部□□	.山田口洋一口口口口口.44444	44-2	000	
	□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	- (ア膝山弘士山山山山山) - 20000(- 木村口舎弘口口口口口) - 66666	00-00000004/01/20 86-8 12/20/20	JUT16 102 15	
13	1	.中村口慎笥口口口口口	77-104/01/2	03	
14	総務部口口	_橋本□悦子□□□□□.88888	88-4	004	
15 16		□二开□重□□□□□□□.999999	99-704/01/20	JU5	
17	*****	flレコード件数:			1000000
18	[EOF]				
<	I		別早し	しい かいしかい しんしょう しんしょう しんしょう しんしょう しんしん しんしょう しんしん しんしょう しんしょ しんしょ	^と りよフナ ^東 卍」た担
		1行:	1桁 標準 ムノー	エナイターを慎 マの記宣も知吟	明水しに場 」 ナナ
				モの設正を解除	しまり。



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。

Wire_Report.dat - TeraPad −	×
ファイル(E) 編集(E) 検索(<u>S</u>) 表示(<u>V</u>) ウィンドウ(<u>W</u>) ツール(<u>T</u>) ヘルプ(<u>H</u>)	
😡 🛱 🗳 🐰 🐚 💼 🗠 🗠 💭 🥬	
■ 1 Program: BATCHRPT Program: BatchRPT Program: Program: BatchRPT Program: Program: BatchRPT Program: Program: BatchRPT Program: Pr	
	>
1行: 1桁 標準 [80] SJIS CRLF 挿入	



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

🗃 Cntl_Card.dat - TeraPad	_	
ファイル(<u>F</u>) 編集(<u>F</u>) 検索(<u>S</u>) 表示(<u>V</u>) ウィンドウ(<u>W</u>) ツ-	-ル(<u>T</u>) ヘルフ	f(<u>H</u>)
🗋 🛱 💾 🍛 🐰 🐚 🋍 🖉 🗠 💭 💭 🔎		
	40	<u> </u>
1 19980101[E0F]		
<		>
1行: 9桁 標準 [80]	SJIS C	RLF 挿入::

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。

₩ Hire_Report.dat - TeraPad	
ファイル(E) 編集(E) 検索(S) 表示(V) ウィンドウ(W) ツール(D) ヘルプ(H)	
🗋 🖆 🍊 🐰 💼 💼 🖉 🔍 🗩 🥬	
Comparison Comp	
	~
<	>
1行: 1桁 標準 [80] SJIS CRLF 挿	入



2017 4	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 URL	03-5413-4800 http://www.microfocus.co.	jp/