Pro*C/C++ for Windows プリコンパイラ

スタート・ガイド

リリース 9.2

2002 年 7 月

部品番号 : J06340-01

ORACLE®

Pro*C/C++ for Windows プリコンパイラ・スタート・ガイド, リリース 9.2

部品番号 : J06340-01

原本名 : Pro*C/C++ Precompiler Getting Started, Release 9.2 for Windows

原本部品番号 : A96111-01

原本協力者 : Subhranshu Banerjee, Eric Belden, Janis Greenberg, Ashish Saigal, Neelam Singh

目次

3 サンプル・プログラム

A Microsoft Visual C++ への Pro*C/C++ の統合

索引

はじめに

このマニュアルでは、Microsoft Windows オペレーティング・システムで実行する Pro*C/C++ プリコンパイラの概要について説明します。

このマニュアルでは、Windows NT、Windows 2000、Windows XP および Windows 98 オペレーティング・システムで使用される Oracle9*i* for Windows ソフトウェアの機能についてのみ説明します。

次の項目について説明します。

対象読者

『Pro*C/C++ for Windows プリコンパイラ・スタート・ガイド』は、Pro*C/C++ を使用して次のことを行うユーザーを対象としています。

- C または C++ プログラムへの Structured Query Language（SQL）文の埋込み
- Pro*C/C++ を使用した Oracle データベース・アプリケーションの作成

このマニュアルを使用するには、次の知識が必要です。

- ファイルの削除やコピーなどのコマンド、検索パス、サブディレクトリおよびパス名の概念
- Windows NT、Windows 98 または Windows 2000 オペレーティング・システム
- Visual C++ バージョン 5.0 以上

このマニュアルの構成

このマニュアルは、次のように構成されています。

第 1 章「Pro*C/C++ の概要」

Windows NT、Windows 98 または Windows 2000 オペレーティング・システムで実行される C および C++ 言語のための Oracle プログラム・インタフェースである、Pro*C/C++ について説明します。

第 2 章「Pro*C/C++ の使用方法」

プロジェクトの作成およびプリコンパイルの方法について説明します。Windows のメニューやアイコン、またはキーボート・ショートカットによりコマンドを実行できる Pro*C/C++ の Graphical User Interface（GUI）について、およびコマンド・プロンプトでの Pro*C/C++ の使用方法についても説明します。

第 3 章「サンプル・プログラム」

このリリースに含まれているサンプル・プログラムを使用し、Pro*C/C++ で Oracle データベース・アプリケーションを作成する方法を説明し、マルチスレッド・アプリケーションの作成方法の概要についても説明します。

付録 A「Microsoft Visual C++ への Pro*C/C++ の統合」

Pro*C/C++ を、Visual C++ 開発環境に統合する方法について説明します。

関連資料

詳細は、次の Oracle マニュアルを参照してください。

- 『Oracle9*i* Database for Windows インストレーション・ガイド』
- 『Oracle9*i* Database for Windows リリース・ノート』
- 『Pro*C/C++ Precompiler プログラマーズ・ガイド』
- 『Oracle9*i* Database for Windows 管理者ガイド』
- 『Oracle Enterprise Manager 管理者ガイド』
- 『Oracle9*i* Net Services 管理者ガイド』
- 『Oracle9*i* Real Application Clusters 概要』
- 『Oracle9*i* データベース新機能』
- 『Oracle9*i* データベース概要』
- 『Oracle9*i* データベース・リファレンス』
- 『Oracle9*i* データベース・エラー・メッセージ』

リリース・ノート、インストール・ドキュメント、ホワイト・ペーパー、またはその他の関連資料を無償でダウンロードするには、OTN-J（Oracle Technology Network Japan）にアクセスしてください。OTN-J を利用する前に、オンライン登録が必要です。次の URL で登録できます。

`http://otn.oracle.co.jp/membership/`

OTN-J のユーザー名およびパスワードをすでにお持ちの場合は、次の OTN-J の Web サイトのドキュメント・セクションに直接アクセスできます。

`http://otn.oracle.co.jp/document/`

表記規則

ここでは、このマニュアルの本文およびサンプル・コードで使用される表記規則について説明します。表記規則は次の 3 種類です。

- 本文の表記規則
- サンプル・コードの表記規則
- Windows オペレーティング・システムの表記規則

本文の表記規則

本文中では、特定の用語をより簡単に識別できるように、様々な表記規則を使用しています。次の表は、本文中で使用される表記規則とその使用例を説明したものです。

規則	意味	例
太字	太字は、本文中で定義されている用語、または用語集で説明されている用語、あるいはその両方を示します。	この句を指定する場合、**索引構成表**を作成します。
大文字（固定幅）フォント	大文字固定幅フォントは、システムによって指定される要素を示します。これらの要素には、パラメータ、権限、データ型、Recovery Manager のキーワード、SQL のキーワード、SQL*Plus またはユーティリティのコマンド、パッケージ、メソッドの他に、システムで表示される列名、データベースのオブジェクトおよび構造、ユーザー名およびロールがあります。	この句は NUMBER 列に対してのみ指定できます。 BACKUP コマンドを使用して、データベースをバックアップできます。 USER_TABLES データ・ディクショナリ・ビューの TABLE_NAME 列を問い合せます。 DBMS_STATS.GENERATE_STATS プロシージャを使用します。
小文字（固定幅）フォント	小文字固定幅フォントは、実行可能ファイル、ファイル名、ディレクトリ名、およびサンプルのユーザー指定要素を示します。これらの要素には、コンピュータ名およびデータベース名、ネット・サービス名、および接続識別子の他に、ユーザー指定のデータベースのオブジェクトおよび構造、列名、パッケージおよびクラス、ユーザー名およびロール、プログラム・ユニット、およびパラメータ値があります。 **注意：** 一部のプログラム要素には、大文字と小文字の両方が使用されます。これらの要素は、記載されているとおりに入力してください。	sqlplus を入力して、SQL*Plus を開きます。 パスワードは、orapwd ファイルで指定されます。 ¥disk1¥oracle¥dbs ディレクトリのデータ・ファイルと制御ファイルをバックアップします。 department_id、department_name および location_id 列は、hr.departments 表にあります。 QUERY_REWRITE_ENABLED 初期化パラメータを true に設定します。 oe ユーザーとして接続します。 JRepUtil クラスは、これらのメソッドを実装します。
小文字イタリック（固定幅）フォント	小文字イタリック固定幅フォントは、プレースホルダまたは変数を示します。	*parallel_clause* を指定できます。 U*old_release*.SQL を実行します。*old_release* は、アップグレード前にインストールしたリリースを表します。

サンプル・コードの表記規則

サンプル・コードは、SQL、PL/SQL、SQL*Plus またはその他のコマンドライン文を示します。これらは固定幅フォントで示され、次の例のように、通常の本文とは区別されています。

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

次の表は、サンプル・コードで使用される表記規則とそれらの使用例を説明したものです。

規則	意味	例
[ ]	大カッコは、1 つ以上のオプション項目を囲みます。大カッコは入力しないでください。	DECIMAL (*digits* [, *precision*])
{ }	中カッコは複数の項目を囲み、そのうちの 1 つが必要であることを示します。中カッコは入力しないでください。	{ENABLE | DISABLE}
|	縦線は、大カッコまたは中カッコ内にある複数のオプションの選択肢を区切るために使用します。オプションの 1 つを入力します。縦線は入力しないでください。	{ENABLE | DISABLE} [COMPRESS | NOCOMPRESS]
...	水平の省略記号は、次のいずれかを示します。	
	■ 例に直接関係のないコードの一部を省略	CREATE TABLE ... AS *subquery*;
	■ コードの一部の繰り返しが可能	SELECT *col1*, *col2*, ... , *coln* FROM employees;
. . .	垂直の省略記号は、例に直接関係のないコードの数行を省略したことを示します。	SQL> SELECT NAME FROM V$DATAFILE; NAME ---------- /fsl/dbs/tbs_01.dbf /fs1/dbs/tbs_02.dbf . . . /fsl/dbs/tbs_09.dbf 9 rows selected.
その他の表記規則	大カッコ、中カッコ、縦線および省略記号以外の記号は、示されているとおりに入力してください。	acctbal NUMBER(11,2); acct CONSTANT NUMBER(4) := 3;
イタリック	イタリックの文字は、特定の値を指定する必要のあるプレースホルダまたは変数を示します。	CONNECT SYSTEM/*system_password* DB_NAME = *database_name*

規則	意味	例

大文字	大文字は、システムによって指定される要素を示します。ユーザーが定義する語句と区別するために、大文字で示しています。語句が大カッコ内に表示されている場合を除き、記載されているとおりの順序とスペルで入力します。ただし、これらの語句には大文字と小文字の区別がないため、小文字で入力できます。	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

小文字	小文字は、ユーザーが指定するプログラム要素を示します。たとえば、小文字は表、列またはファイルの名前を示します。
	注意： 一部のプログラム要素には、大文字と小文字の両方が使用されます。これらの要素は、記載されているとおりに入力してください。
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Windows オペレーティング・システムの表記規則

次の表は、Windows オペレーティング・システムの表記規則とその使用例を説明したものです。

規則	意味	例
「スタート」→を選択	プログラムの起動方法。たとえば、Database Configuration Assistant を起動するには、タスクバーの「スタート」ボタンをクリックし、「プログラム」→「Oracle - *HOME_NAME*」→「Configuration and Migration Tools」→「Database Configuration Assistant」を選択します。	「スタート」→「プログラム」→「Oracle - *HOME_NAME*」→「Configuration and Migration Tools」→「Database Configuration Assistant」を選択します。
ファイル名およびディレクトリ名	ファイルおよびディレクトリ名には、大文字と小文字の区別がありません。<、>、:、"、/、|、および - の特殊文字は使用できません。特殊文字 ¥ は、引用符に囲まれている場合でも、要素の区切り文字として扱われます。ファイル名が ¥¥ で始まる場合、Windows では汎用命名規則を使用しているものと認識されます。	`c:¥winnt"¥"system32` は、`C:¥WINNT¥SYSTEM32` と同じです。

規則	意味	例
C:¥>	現行のハード・ディスク・ドライブの Windows コマンド・プロンプトを示します。コマンド・プロンプトのエスケープ文字は、カレット（^）です。プロンプトは、現在作業中のサブディレクトリを示しています。このマニュアルでは、コマンド・プロンプトと呼びます。	C:¥oracle¥oradata>
特殊文字	特殊文字の円記号（¥）は、Windows コマンド・プロンプトで特殊文字の二重引用符（"）のエスケープ文字として必要な場合があります。カッコおよび特殊文字の一重引用符（'）は、エスケープ文字を必要としません。エスケープ文字および特殊文字の詳細は、Windows オペレーティング・システムのドキュメントを参照してください。	C:¥>exp scott/tiger TABLES=emp QUERY=¥"WHERE job='SALESMAN' and sal<1600¥" C:¥>imp SYSTEM/*password* FROMUSER=scott TABLES=(emp, dept)
HOME_NAME	Oracle ホーム名を示します。 ホーム名は、英数字 16 文字までです。ホーム名で使用できる特殊文字は、アンダースコアのみです。	C:¥> net start Oracle*HOME_NAME*TNSListener

規則	意味	例
ORACLE_HOME および ORACLE_BASE	Oracle8 リリース 8.0 以下のリリースでは、Oracle コンポーネントをインストールすると、サブディレクトリはすべて、最上位の ORACLE_HOME ディレクトリ（デフォルトでは次のとおり）の下に置かれました。 ■ Windows NT の場合は C:¥orant ■ Windows 98 の場合は C:¥orawin98 あるいは、Oracle ホームと呼ばれるディレクトリの下に置かれました。 今回のリリースは、Optimal Flexible Architecture（OFA）に準拠しています。すべてのサブディレクトリが最上位の ORACLE_HOME ディレクトリの下にあるわけではありません。ORACLE_BASE という最上位ディレクトリがあり、デフォルトは C:¥oracle です。コンピュータに最新の Oracle リリースをインストールし、他の Oracle ソフトウェアをインストールしない場合、最初の Oracle ホーム・ディレクトリのデフォルト設定は、C:¥oracle¥ora*nn* です。*nn* は、最新のリリース番号です。Oracle ホーム・ディレクトリは、ORACLE_BASE の直下に置かれます。 このマニュアルでは、ディレクトリ・パスの例は、すべて OFA 表記規則に準拠しています。	%ORACLE_HOME%¥rdbms¥admin ディレクトリに移動します。

Pro*C/C++ の新機能

この項では、Oracle9*i* の新機能について説明し、追加情報の参照先を示します。現在のリリースに移行するユーザーに役立つよう、前のリリースの新機能情報も記載しています。

次の項では、各リリースの Oracle Pro*C/C++ の新機能について説明します。

- Oracle9i リリース 2（9.2）の Pro*C/C++ における新機能
- Oracle9i リリース 1（9.0.1）の Pro*C/C++ における新機能
- Oracle8i リリース 8.1.6 の Pro*C/C++ における新機能

Oracle9*i* リリース 2（9.2）の Pro*C/C++ における新機能

このリリースの Pro*C/C++ には、Windows 固有の新機能はありません。

関連資料：『Pro*C/C++ Precompiler プログラマーズ・ガイド』の「新機能」

Oracle9*i* リリース 1（9.0.1）の Pro*C/C++ における新機能

Oracle9*i* リリース 1（9.0.1）では、Windows 2000 がサポートされるようになりました。

Windows 2000 での Oracle9*i* の使用

Pro*C/C++ は Windows 2000 でもサポートされています。Windows 2000 と Windows NT 4.0 では、Oracle9*i* の使用方法に一部異なる点があります。

関連資料：『Oracle9*i* Database for Windows スタート・ガイド』

Oracle8*i* リリース 8.1.6 の Pro*C/C++ における新機能

Oracle8*i* リリース 8.1.6 では、Pro*C/C++ アプリケーションの開発をより簡単に行えるよう、機能が追加および拡張されています。

完全に統合されたデバッグ機能

リリース 8.1.6 から、`LINES={YES|NO}` オプションの動作が変更されています。`LINES=YES` と指定すると、出力プログラムの各生成コード行の後ろに `#line` プリプロセッサ・ディレクティブが生成されます。これで、GDB などのデバッガ、または Microsoft Visual Studio for C++ などの IDE を使用する開発者は、生成されたコードを 1 つずつ確認せずに、Pro*C/C++ ソース・プログラムを表示してアプリケーション・プログラムをデバッグできます。

関連項目： A-2 ページの「Pro*C/C++ の Microsoft Visual C++ プロジェクトへの統合」

1

Pro*C/C++ の概要

この章では、Windows オペレーティング・システムで実行される C および C++ 言語のための Oracle のプログラム・インタフェースである Pro*C/C++ について説明します。Pro*C/C++ を使用して、Win32 環境の Oracle データベース・アプリケーションを作成できます。

この章の項目は次のとおりです。

> **関連資料：** 詳細は、『Pro*C/C++ Precompiler プログラマーズ・ガイド』を参照してください。

Pro*C/C++ の概要

Pro*C/C++ プリコンパイラを使用することにより、Oracle データベースにアクセスするアプリケーションを迅速に、また他のシステムとの互換性を考慮しながら開発できます。

Pro*C/C++ プログラミング・ツールにより、C または C++ プログラムに SQL 文を埋め込むことができます。Pro*C/C++ プリコンパイラは、埋込み SQL 文を標準の Oracle ランタイム・ライブラリ・コールに変換し、通常の方法でコンパイル、リンクおよび実行できるソース・プログラムに変更します。

機能

Pro*C/C++ では次の機能がサポートされます。

- Oracle データベースへのローカル・アクセスおよび Oracle Net Services によるリモート・アクセス
- 埋込み PL/SQL ブロック
- クライアント / サーバー環境で高いパフォーマンスを提供する、データベース・コールのまとまり
- 埋込み SQL プログラミングの ANSI 完全準拠
- PL/SQL リリース 9.0 および PL/SQL プロシージャ内のホスト言語配列
- マルチスレッド・アプリケーション
- ANSI C 完全準拠
- 32 ビット・アプリケーション用 Microsoft Visual C++ バージョン 6.0 のサポート

注意： Borland C++ はサポートされていません。

制限事項

Pro*C/C++ では、16 ビット・コードの生成はサポートされていません。

ディレクトリ構造

Oracle ソフトウェアをインストールすると、ハード・ディスク・ドライブ上に Oracle 製品のためのディレクトリ構造が作成されます。メインの Oracle ディレクトリには、Oracle サブディレクトリおよび Pro*C/C++ の実行に必要なファイルが含まれます。

Pro*C/C++ のインストール時に、Oracle Universal Installer によって ¥precomp というディレクトリが、ORACLE_HOME ディレクトリの下に作成されます。このサブディレクトリには、表 1-1 にリストする Pro*C/C++ 実行可能ファイル、ライブラリ・ファイルおよびサンプル・プログラムが含まれます。

表 1-1 precomp ディレクトリ構造

ディレクトリ名	内容
¥admin	構成ファイル
¥demo¥proc	Pro*C/C++ 用サンプル・プログラム
¥demo¥sql	サンプル・プログラム用 SQL スクリプト
¥doc¥proc	Pro*C/C++ の README ファイル
¥help¥proc	Pro*C/C++ のヘルプ・ファイル
¥lib¥msvc	Pro*C/C++ のライブラリ・ファイル
¥mesg	メッセージ・ファイル
¥misc¥proc	Pro*C/C++ 用の他のファイル
¥public	ヘッダー・ファイル

注意： ¥precomp ディレクトリには、Pro*COBOL など他の製品のファイルが格納されている場合があります。

既知の問題、制限および対処方法

Windows のすべてのオペレーティング・システムでは、ファイル名およびディレクトリ名にスペースを含めることができますが、Oracle Pro*C/C++ および Oracle Pro*COBOL プリコンパイラでは、ファイル名またはディレクトリ名にスペースを含むファイルはプリコンパイルされません。たとえば、次のような書式は使用しないでください。

- `proc iname=test one.pc`
- `proc iname=d:¥dir1¥second dir¥sample1.pc`

2

Pro*C/C++ の使用方法

この章では、プロジェクトの作成およびプリコンパイル方法について説明します。また、Windows のメニューやアイコン、またはキーボート・ショートカットによってコマンドを実行できる Pro*C/C++ の GUI の説明、およびコマンド・プロンプトでの Pro*C/C++ の使用方法も説明します。

この章の項目は次のとおりです。

> **関連資料：** 詳細は、『Pro*C/C++ Precompiler プログラマーズ・ガイド』を参照してください。

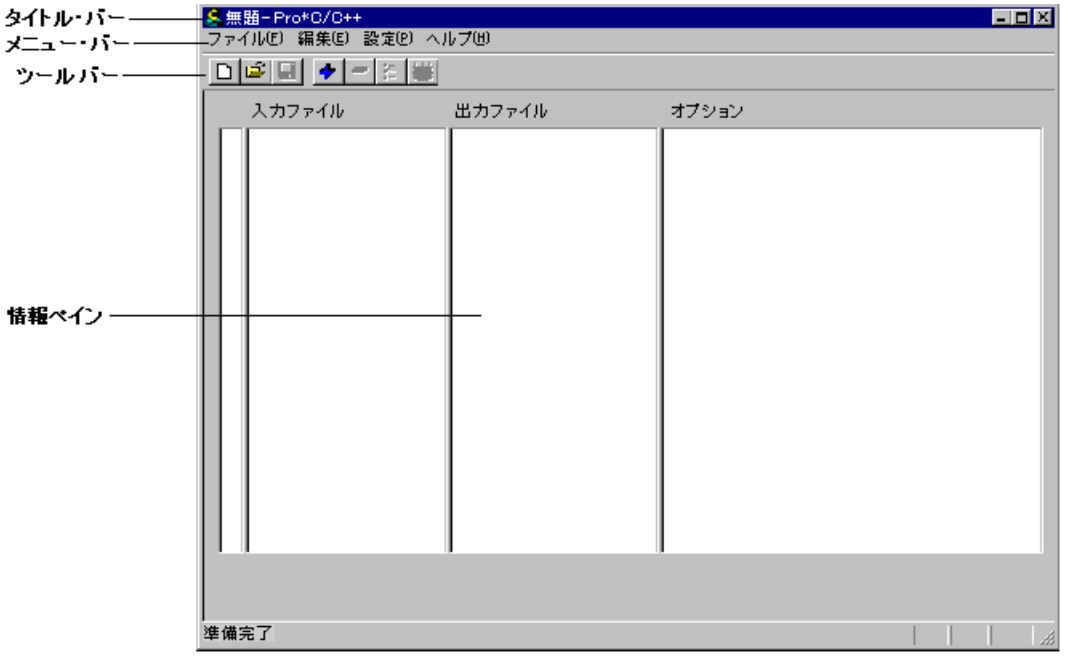
GUI の使用方法

Pro*C/C++ プロジェクトの作成およびプリコンパイルの指示に従って作業を開始する前に、Pro*C/C++ の GUI の基本的なコマンド、ダイアログ・ボックス、メニューおよびボタンを知っておく必要があります。

Pro*C/C++ の GUI の起動

GUI を起動するには、「スタート」→「プログラム」→「Oracle - *HOME_NAME*」→「Application Development」→「Pro C-C++」を選択します。図 2-1 に、Pro*C/C++ のプリコンパイル環境の 4 つの要素を示します。

図 2-1 Pro*C/C++ のプリコンパイル環境の要素



タイトル・バー

タイトル・バーには、Pro*C/C++ プロジェクトの名前が表示されます。現行のプロジェクトにまだ名前を付けていない場合は、「無題」と表示されます。

メニュー・バー

メニュー・バーには、次のメニューがあります。表 2-1 では、メニューについて説明します。

表 2-1 メニュー・バーのメニュー

メニュー	説明
ファイル	新規 Pro*C/C++ プロジェクトを作成、既存の Pro*C/C++ プロジェクトを開く、アクティブな Pro*C/C++ プロジェクトを同じ名前または別名で保存、Oracle データベースへの接続文字列を指定、Pro*C/C++ プロジェクトをプリコンパイル、およびアプリケーションを終了するコマンドが含まれます。
編集	Pro*C/C++ プロジェクトにファイルを追加、Pro*C/C++ プロジェクトからファイルを削除、プリコンパイラのオプションを表示または変更するコマンドが含まれます。
設定	出力ファイルのデフォルト・ファイル拡張子を設定するコマンドが含まれます。
ヘルプ	Pro*C/C++ アプリケーションのバージョン番号および著作権情報を表示する「Pro*C/C++ のバージョン情報」が含まれます。

ツールバー

ツールバーの次のボタンを選択すると、対応するコマンドを実行できます。図 2-2 に、ツールバーのボタンを示します。

図 2-2 ツールバー・ボタン

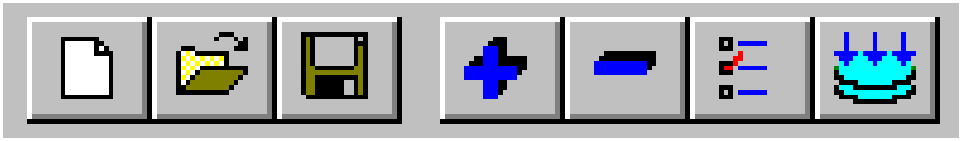


表 2-2 では、ツールバーのボタンについて左から右の順に説明します。

表 2-2 ツールバー・ボタン

ボタン	説明
新規作成	新規 Pro*C/C++ プロジェクトを作成します。
開く	既存の Pro*C/C++ プロジェクトを開きます。
上書き保存	アクティブな Pro*C/C++ プロジェクトを同じ名前で保存します。
追加	Pro*C/C++ プロジェクトにファイルを追加します。
削除	Pro*C/C++ プロジェクトからファイルを削除します。
オプション	プリコンパイラのオプションを表示または変更します。
プリコンパイル	Pro*C/C++ プロジェクトをプリコンパイルします。

情報ペイン

図 2-3 に、情報ペインの 4 つの要素を示します。

図 2-3 情報ペインの要素

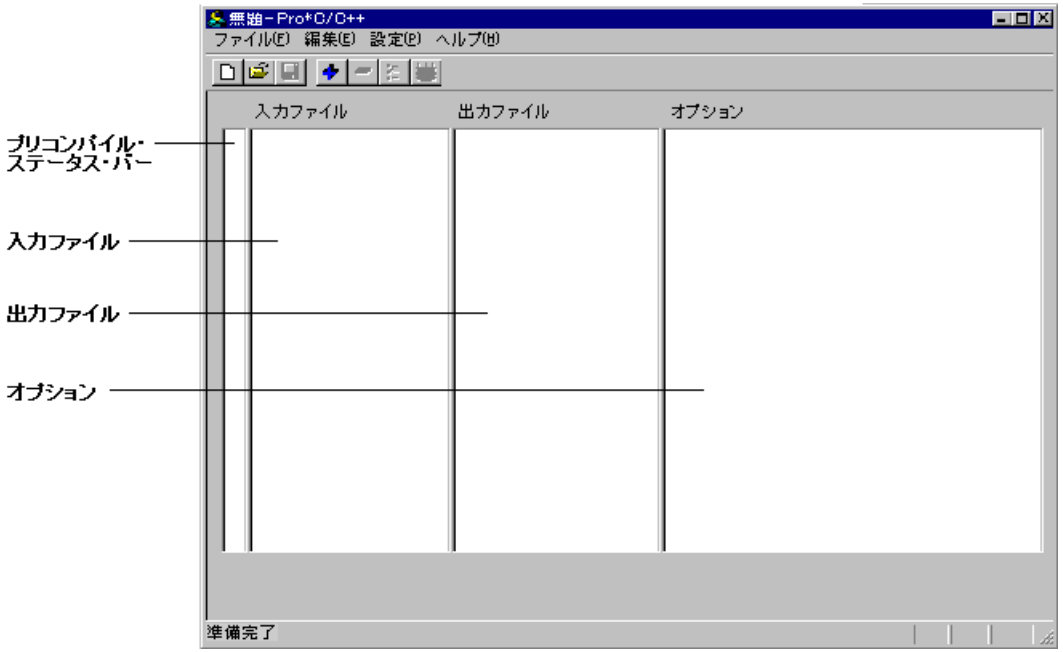


表 2-3 では、情報ペインの要素について説明します。

表 2-3 情報ペインの要素

要素	説明
プリコンパイル・ステータス・バー	ファイルのプリコンパイルの成功または失敗を表示
入力ファイル	プリコンパイルされる Pro*C/C++ プロジェクトのファイルを表示
出力ファイル	プリコンパイルされた後の Pro*C/C++ プロジェクトの出力ファイルを表示
オプション	デフォルトのオプションとは異なるプリコンパイル・オプションを表示

プリコンパイルが完了した後、プリコンパイル・ステータス・バーに次の 3 つのアイコンのいずれかが表示されます。

- 緑色のチェック・マークは、ファイルのプリコンパイルが成功したことを示します。
- 黄色のチェック・マークは、ファイルのプリコンパイルは成功したが 1 つ以上の警告メッセージがあることを示します。
- 赤色の×マークは、ファイルのプリコンパイルが失敗したことを示します。

ステータス・アイコンをダブルクリックすると、「プリコンパイル状態」ダイアログ・ボックスが表示されます。このダイアログ・ボックスには、警告または失敗理由の詳細な情報が表示されます。

ステータス・バー

ウィンドウの一番下のステータス・バーに、プリコンパイルの進捗状況に関する情報が表示されます。ツールバー・ボタンまたはメニュー・コマンドにマウス・ポインタを合せると、そのボタンやコマンドの用途がステータス・バーに表示されます。

Pro*C/C++ プロジェクトの作成およびプリコンパイル

この項では、Pro*C/C++ プロジェクトの作成およびプリコンパイルの手順について説明します。Pro*C/C++ アプリケーションを起動した後、次の手順を実行します。

- プロジェクトを開く
- 出力ファイルのデフォルト拡張子の設定
- 既存の入力ファイル名または出力ファイル名の変更
- プロジェクトへのファイルの追加
- プロジェクトからのファイルの削除
- プリコンパイラのオプションの設定
- データベース接続情報の指定
- Pro*C/C++ プロジェクトのプリコンパイル
- 結果の確認
- エラーの修正
- Pro*C/C++ の終了

プロジェクトを開く

Pro*C/C++ では、一度に開けるのは 1 つのプロジェクトのみです。プロジェクトは、1 つ以上のプリコンパイル可能ファイルで構成されます。プロジェクト・ファイルには、拡張子 .pre が付いています。

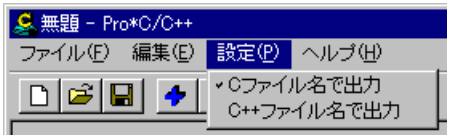
- 新規プロジェクトを作成するには、「ファイル」→「新規プロジェクト」を選択します。
- 既存のプロジェクトを開くには、「ファイル」→「プロジェクトを開く」を選択します。

注意： 以前のリリースで作成されたプロジェクトは、Oracle9*i* では開けません。「予期しないファイル形式です。」のエラーになります。プロジェクトを再作成する必要があります。

出力ファイルのデフォルト拡張子の設定

出力ファイルのデフォルト拡張子を設定するには、「設定」メニューを使用します。図 2-4 に、「設定」メニューを示します。

図 2-4「設定」メニュー



この設定は、この後追加する入力ファイルにのみ適用されます。既存の出力ファイル名は変更されません。ただし、既存の出力ファイル名は、出力ファイルをダブルクリックして新しい名前を入力すると変更できます。

- 「C ファイル名で出力」を選択した場合、出力ファイルのデフォルト拡張子は .c になります。
- 「C++ ファイル名で出力」を選択した場合、出力ファイルのデフォルト拡張子は .cpp になります。
- 「C ファイル名で出力」および「C++ ファイル名で出力」両方を選択解除した場合は、出力ファイルを追加するときに「Output Filename」ダイアログ・ボックスが表示されます。
- 選択したファイルの出力ファイル名を入力します。ファイル名を選択または入力すると、名前が情報ペインの「出力ファイル」領域に表示されます。

既存の入力ファイル名または出力ファイル名の変更

既存の入力ファイルまたは出力ファイルの名前を変更するには、次のようにします。

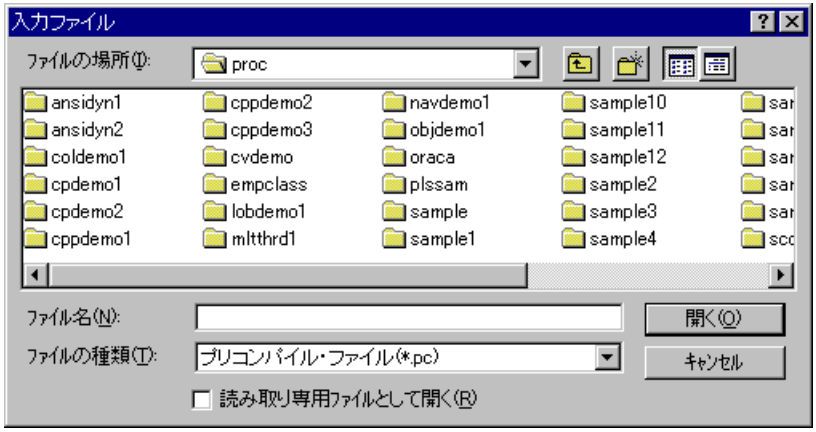
1. 情報ペインの「入力ファイル」領域または「出力ファイル」領域に表示されているファイル名をダブルクリックします。「入力ファイル」または「出力ファイル」ダイアログ・ボックスが表示されます。

2. 古いファイル名を新しいファイル名に置き換えます。
3. 「開く」を選択します。

プロジェクトへのファイルの追加

プロジェクトにファイルを追加するには、次のようにします。

1. 「編集」→「追加」を選択します。「入力ファイル」ダイアログ・ボックスが表示されます。



2. .pc ファイルを 1 つ以上選択します。連続していないファイルを複数選択するには、[Ctrl] キーとマウスを使用します。
3. 「開く」を選択します。選択したファイルが情報ペインに表示されます。

プロジェクトからのファイルの削除

必要に応じて、プロジェクトから 1 つ以上のファイルを簡単に削除できます。

プロジェクトからファイルを削除するには、次のようにします。

1. 情報ペインでファイルを選択します。
2. 「編集」→「削除」を選択します。
3. 「はい」を選択します。

プリコンパイラのオプションの設定

プリコンパイラのオプションを使用し、リソースの使用方法、エラーのレポート方法、入力ファイルおよび出力ファイルの形式設定方法、カーソルの管理方法を制御できます。

プリコンパイルのオプションを設定するには、次のようにします。

1. 「入力ファイル」リストからファイルを 1 つ以上選択します。
2. 「編集」→「オプション」を選択します。「オプション」ダイアログ・ボックスが表示されます。

デフォルト・オプションは、新たに追加されたすべてのファイルに適用されます。オプションのデフォルト設定を変更すると、「オプション」ダイアログ・ボックスの一番下の「オプション文字列」編集フィールドおよび情報ペインの「オプション」領域に変更内容が表示されます。オプションの詳細は、2-17 ページの「プリコンパイラのオプション」を参照してください。

3. プリコンパイラがディスクに書き込む出力リスト・ファイルの形式を変更するには、「リスト / エラー」ボタンをクリックします。「リスト / エラー」ダイアログ・ボックスが表示されます。

設定には、生成されるエラー情報の種類およびリスト・ファイルの名前が含まれます。

4. 「オプション」ダイアログ・ボックスでオプションを設定した後、「OK」を選択します。

データベース接続情報の指定

「オプション」ダイアログ・ボックスの「SQL チェック」オプションで「semantics」または「full」を選択した場合、Oracle データベースに対するデータベース接続情報の指定が必要になることがあります。データ操作文または PL/SQL ブロックで参照されるすべての表が DECLARE TABLE 文で定義されている場合には、Oracle データベースに接続する必要はありません。

データベース接続情報を指定するには、次のようにします。

1. 「ファイル」→「接続」を選択します。「接続」ダイアログ・ボックスが表示されます。

2. このダイアログ・ボックスを使用して、プリコンパイルを行う前にデータベース接続情報を指定します。この時点ではデータベース接続は行われません。SQLCHECK の SEMANTICS または FULL のチェックが必要なすべてのファイルに対して指定できるデータベース接続情報は、1 セットのみです。
3. 指定していない場合、プリコンパイル時に「接続」ダイアログ・ボックスが自動的に表示されます。ユーザー名、パスワードおよびネットワーク・サービス名（データベース別名）を入力します。ネットワーク・サービス名は、ローカル・データベースには必要ありません。
4. 次の Pro*C/C++ セッションで使用するために接続情報を保存する場合は、「接続文字列をディスクに保存」チェックボックスを選択します。このチェックボックスを選択しておかないと、プリコンパイルするたびにこの情報の入力が必要になります。
5. 「OK」を選択します。

Pro*C/C++ プロジェクトのプリコンパイル

「入力ファイル」リスト内のファイルはいくつでもプリコンパイルできます。

プリコンパイルするには、次のようにします。

1. 「入力ファイル」リストからファイルを 1 つ以上選択します。[Ctrl] キーとマウスを使用すると、連続していない複数のファイル（たとえば、リストの 1 番目と 3 番目のファイル）を選択できます。
2. 「ファイル」→「プリコンパイル」を選択します。

   プリコンパイルが完了すると、ダイアログ・ボックスに「プリコンパイルが終了しました。」というメッセージが表示され、「取消」ボタンが「OK」に変わります。
3. 「OK」を選択します。

注意： 「取消」をクリックしてもプリコンパイル中のファイルの処理は中断できませんが、残りのファイルのプリコンパイルは停止できます。

結果の確認

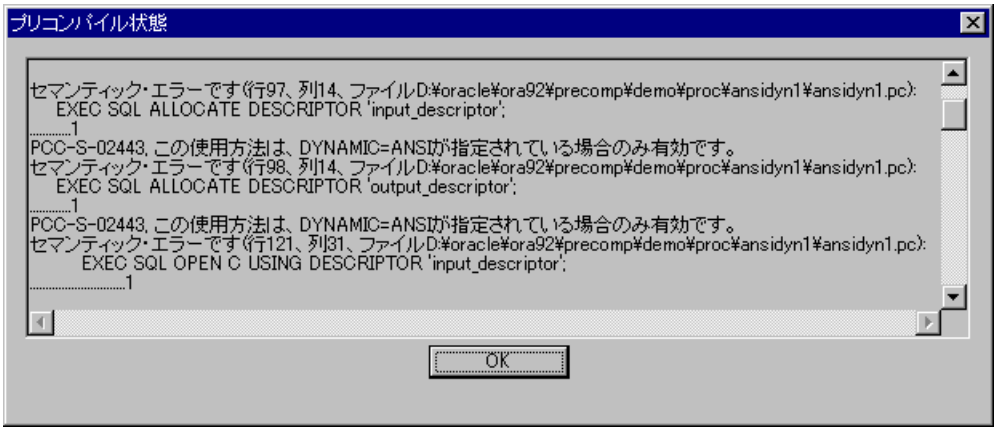
プリコンパイルの結果は成功、警告付きの成功または失敗です。プリコンパイルが終了したときに、プリコンパイル・ステータス・バーで確認します。

- 緑色のチェック・マークは、ファイルのコンパイルが成功したことを示します。
- 黄色のチェック・マークは、ファイルのコンパイルは成功したが 1 つ以上の警告メッセージがあることを示します。
- 赤色の×マークは、ファイルのコンパイルが失敗したことを示します。

エラーの修正

ステータス・バーに黄色のチェック・マークまたは赤色の×マークが表示された場合は、アイコンをダブルクリックします。「プリコンパイル状態」ダイアログ・ボックスが表示されます。このダイアログ・ボックスには、警告メッセージまたはプリコンパイルが失敗した理由が表示されます。次に例を示します。

図 2-5 プリコンパイル状態



開発環境に切り替えて問題を修正します。エラーを修正した後、再度プリコンパイルします。

注意： PCC-S-02014 エラー（行 *num*、列 *colnam*、ファイル *name* での構文エラー）が発生した場合は、次の操作を行います。

- 問題が発生した INCLUDE ファイルがあるディレクトリに、%ORACLE_HOME%¥precomp¥misc¥proc ディレクトリからバッチ・ファイル mod_incl.bat および add_newl.bat をコピーします。
- mod_incl.bat を実行します。

Pro*C/C++ の終了

Pro*C/C++ を終了するには、「ファイル」→「終了」を選択します。プロジェクトが変更されている場合、保存するよう要求されます。

> **注意：** 変更されたファイルのみではなく、元のファイルもそのまま保持する場合は、「別名保存」コマンドを選択します。「保存」コマンドは、前の状態を上書きします。

コマンド・プロンプトでの Pro*C/C++ の使用方法

コマンド・プロンプトでファイルをプリコンパイルするには、次のコマンドを入力します。

```
C:¥> proc iname=filename.pc
```

filename.pc はファイルの名前です。ファイルが現在の作業ディレクトリ内に存在しない場合は、そのファイルのフルパスを INAME 引数の後に指定します。

Pro*C/C++ は、*filename*.c を生成します。生成されたファイルを C コンパイラでコンパイルします。

ヘッダー・ファイル

%ORACLE_HOME%¥precomp¥public ディレクトリには、Pro*C/C++ ヘッダー・ファイルが格納されています。表 2-4 では、ヘッダー・ファイルについて説明します。

> **関連資料：** oraca.h、sqlca.h および sqlda.h の詳細は、『Pro*C/C++ Precompiler プログラマーズ・ガイド』を参照してください。

表 2-4 ヘッダー・ファイル

ヘッダー・ファイル	説明
oraca.h	ランタイム・エラーの診断、およびプログラムによる様々な Oracle9*i* リソースの使用状況の監視に役立つ oracle 通信領域（ORACA）が含まれています。
sql2oci.h	Oracle Call Interface（OCI）環境ハンドルおよび OCI サービス・コンテキストを Pro*C/C++ アプリケーション内から取得できるようにする SQLLIB ファンクションが含まれています。
sqlapr.h	OCI とともに使用できる外部関数の ANSI プロトタイプが含まれています。
sqlca.h	ランタイム・エラーの診断に役立つ、SQL コミュニケーション領域（SQLCA）が含まれています。SQLCA は、実行可能な SQL 文が実行されるたびに更新されます。

表 2-4 ヘッダー・ファイル（続き）

ヘッダー・ファイル	説明
sqlcpr.h	Pro*C/C++ によって生成される SQLLIB ファンクションのプラットフォーム固有の ANSI プロトタイプが含まれています。デフォルトでは、Pro*C/C++ は SQL プログラミング・コールの全ファンクションのプロトタイピングをサポートしません。この機能が必要な場合は、アプリケーションのソース・ファイル内のどの EXEC SQL 文よりも前に sqlcpr.h をインクルードします。
oraca.h	ランタイム・エラーの診断、およびプログラムによる様々な Oracle9*i* リソースの使用状況の監視に役立つ oracle 通信領域（ORACA）が含まれています。
sql2oci.h	Oracle Call Interface（OCI）環境ハンドルおよび OCI サービス・コンテキストを Pro*C/C++ アプリケーション内から取得できるようにする SQLLIB ファンクションが含まれています。
sqlapr.h	OCI とともに使用できる外部関数の ANSI プロトタイプが含まれています。

ライブラリ・ファイル

%ORACLE_HOME%¥precomp¥lib¥msvc ディレクトリには、Pro*C/C++ アプリケーションにリンクするときに使用するライブラリ・ファイルが格納されています。このライブラリ・ファイルの名前は orasql9.lib です。

Pro*C/C++ の Application Program Interface（API）コールは、使用している Pro*C/C++ ソフトウェアに付属の DLL ファイル内で実装されています。その DLL を使用するには、Pro*C/C++ の DLL に対応するインポート・ライブラリ（.lib ファイル）とアプリケーションをリンクする必要があります。さらに、Pro*C/C++ アプリケーションを実行するコンピュータに DLL ファイルがインストールされている必要があります。

Microsoft 社は、libc.lib、libcmt.lib および msvcrt.lib の 3 つのライブラリを提供しています。Oracle の DLL では、msvcrt.lib ランタイム・ライブラリを使用します。他の 2 つの Microsoft ライブラリではなく、必ず msvcrt.lib を使用してリンクしてください。

マルチスレッド・アプリケーション

同時データベース操作を実行する場合、マルチスレッド・アプリケーションを作成します。

Windows NT、Windows 2000 および Windows 98 では、プロセスに含めるスレッドをスケジューリングして割り当てます。スレッドはプログラム実行パスです。これはカーネル・スタック、CPU レジスタの状態、スレッド環境ブロックおよびユーザー・スタックで構成されます。各スレッドはプロセスのリソースを共有します。マルチスレッド・アプリケーションは、プロセスのリソースを使用して個々のスレッドのアクティビティを調整します。

マルチスレッド・アプリケーションを作成する場合、C/C++ コードがリエントラントであることが必要です。つまり、静的データまたはグローバル・データへのアクセスは、一度に

1 つのスレッドに制限される必要があります。マルチスレッドと非リエントラント関数が混在すると、スレッドは別のスレッドが必要とする情報を変更する可能性があります。

Pro*C/C++ プリコンパイラは、スレッドのローカル・スタックに変数を自動作成します。これにより、確実に Pro*C/C++ 関数を使用するスレッドが変数の一意の集合にアクセスし、リエントラントであることが保証されます。

> **関連資料：** Pro*C/C++ でのマルチスレッド・アプリケーション作成方法の追加情報は、『Pro*C/C++ Precompiler プログラマーズ・ガイド』を参照してください。

プリコンパイラのオプション

この項では、Windows プラットフォーム用の Pro*C/C++ に関連した内容について説明します。

> **関連資料：** 詳細は、『Pro*C/C++ Precompiler プログラマーズ・ガイド』の「プリコンパイラのオプション」を参照してください。

構成ファイル

構成ファイルは、プリコンパイラのオプションを含むテキスト・ファイルです。

このリリースでは、システム構成ファイルの名前は pcscfg.cfg です。このファイルは、%ORACLE_HOME%¥precomp¥admin ディレクトリにあります。

CODE

CODE オプションのデフォルト設定は、ANSI_C です。他のオペレーティング・システム用の Pro*C/C++ では、デフォルト設定が KR_C の場合もあります。

DBMS

CHAR_MAP=VARCHAR2 を使用している場合、DBMS=V6_CHAR はサポートされません。その場合は、DBMS=V7 を使用してください。

INCLUDE

Pro*C/C++ の GUI の場合、INCLUDE パス・ディレクトリを入力するには「オプション」ダイアログ・ボックスの「インクルード・ディレクトリ」フィールドを使用します。複数のパスを入力する場合は、それぞれのパスをセミコロンで区切ります。セミコロンの後にスペースを入れないでください。各ディレクトリの前に個別の INCLUDE= 文字列が挿入される原因になります。

PARSE=PARTIAL または PARSE=FULL を指定してサンプル・プログラムをプリコンパイルした場合、c:¥program files¥devstudio¥vc¥include のインクルード・パスが追加されます。Microsoft Visual C++ が別の場所にインストールされている場合は、「インクルード・ディレクトリ」フィールドをそれに応じて変更します。これによりサンプル・プログラムを正しくプリコンパイルできます。

PARSE

PARSE オプションのデフォルト設定は、NONE です。他のオペレーティング・システム用の Pro*C/C++ では、デフォルト設定が FULL の場合もあります。

Oracle XA ライブラリと Pro*C/C++ の使用方法

Oracle データベースが次のトランザクション処理（TP）モニターと対話できるようにするには通常、XA API を使用します。

- BEA Tuxedo
- IBM Transarc Encina
- IBM CICS

クライアント・プログラムで TP モニターの文を使用することもできます。また、XA API の使用は Pro*C/C++ および OCI の両方でサポートされています。

Oracle XA ライブラリは、Oracle9*i* Enterprise Edition の一部として自動的にインストールされます。Oracle ホーム・ディレクトリに、次のコンポーネントが作成されます。

表 2-5 Oracle XA ライブラリのコンポーネントおよび場所

コンポーネント	場所
oraxa9.lib	%ORACLE_HOME%¥rdbms¥xa
xa.h	%ORACLE_HOME%¥rdbms¥demo

XA を使用した Pro*C/C++ プログラムのコンパイルおよびリンク

XA を使用して Pro*C/C++ プログラムをコンパイルおよびリンクするには、次のようにします。

1. Pro*C/C++ を使用して *filename*.pc をプリコンパイルして *filename*.c を生成します。
2. %ORACLE_HOME%¥rdbms¥xa がパスに含まれていることを確認し、*filename*.c をコンパイルします。
3. *filename*.obj と次のライブラリをリンクします。

ライブラリ	場所
oraxa9.lib	%ORACLE_HOME%¥rdbms¥xa
oci.lib	%ORACLE_HOME%¥oci¥lib¥msvc
orasql9.lib	%ORACLE_HOME%¥precomp¥lib¥msvc

4. *filename*.exe を実行します。

XA 動的登録

Oracle は、XA 動的登録の使用をサポートしています。XA 動的登録により、XA 準拠 TP モニターとのインタフェースであるアプリケーションのパフォーマンスが向上します。

Windows NT の Oracle データベースで TP モニターが XA 動的登録を使用するには、環境変数またはレジストリ変数のどちらかを、TP モニターが実行されている Windows NT コンピュータに追加する必要があります。手順は、次のいずれかを参照してください。

- 現行のセッションに対する環境変数の追加
- すべてのセッションに対するレジストリ変数の追加

現行のセッションに対する環境変数の追加

コマンド・プロンプトで環境変数を追加すると、現行のセッションのみが適用対象となります。

現行のセッションに環境変数を追加するには、次のようにします。

1. TP モニターがインストールされているコンピュータに移動します。
2. コマンド・プロンプトで次のように入力します。

```
C:¥> set ORA_XA_REG_DLL = vendor.dll
```

vendor.dll は、ベンダーから提供された TP モニターの DLL です。

すべてのセッションに対するレジストリ変数の追加

レジストリ変数を追加すると、使用している Windows NT コンピュータのすべてのセッションが適用対象となります。TP モニターが 1 つだけ実行中のコンピュータでは、この方法が便利です。

すべてのセッションに対してレジストリ変数を追加するには、次のようにします。

1. TP モニターがインストールされているコンピュータに移動します。
2. コマンド・プロンプトで次のように入力します。

   ```
   C:¥> regedt32
   ```

   「レジストリ エディタ」ウィンドウが表示されます。
3. ¥HKEY_LOCAL_MACHINE¥SOFTWARE¥ORACLE¥HOME*ID* に移動します。
4. 「編集」メニューから「値の追加」を選択します。

   「値の追加」ダイアログ・ボックスが表示されます。
5. 「値の名前」フィールドに ORA_XA_REG_DLL と入力します。
6. 「データ タイプ」ドロップダウン・リスト・ボックスから「REG_EXPAND_SZ」を選択します。
7. 「OK」を選択します。

   「文字列エディタ」ダイアログ・ボックスが表示されます。
8. 「文字列」フィールドに *vendor*.dll と入力します。*vendor*.dll はベンダーから提供された TP モニターの DLL です。
9. 「OK」を選択します。

   レジストリ エディタでパラメータが追加されます。
10. 「レジストリ」メニューから「レジストリ エディタの終了」を選択します。

    レジストリ エディタが終了します。

XA および TP モニターの情報

XA および TP モニターの詳細は、次の情報を参照してください。

- 使用している TP モニター固有のドキュメント

  関連資料： Oracle XA ライブラリおよび XA 動的登録の使用方法の詳細は、『Oracle9*i* アプリケーション開発者ガイド - 基礎編』を参照してください。

3

サンプル・プログラム

この章では、このリリースに含まれているサンプル・プログラムを使用した Pro*C/C++ での Oracle データベース・アプリケーションの作成方法について説明します。

この章の項目は次のとおりです。

サンプル・プログラムの説明

Pro*C/C++ をインストールするときに、Oracle Universal Installer によって、Pro*C/C++ の一連のサンプル・プログラムが %ORACLE_HOME%¥precomp¥demo¥proc ディレクトリにコピーされます。表 3-1「サンプル・プログラム」にこれらのサンプル・プログラムをリストし、後に続く項で説明します。

Oracle から提供されたサンプル・プログラムをビルドすると、.exe 実行ファイルが生成されます。

表の「注意」列に記載されているように、一部のサンプル・プログラムについては、プリコンパイルして実行する前に、サンプル・ディレクトリにある SQL スクリプトを実行する必要があります。SQL スクリプトは、サンプル・プログラムが正常に実行されるよう適切な表およびデータを設定します。これらの SQL スクリプトは、%ORACLE_HOME%¥precomp¥demo¥sql ディレクトリにあります。

オラクル社では、これらのサンプル・プログラムをビルドおよび実行し、Pro*C/C++ が正しくインストールされ正常に動作することを確認するようお薦めします。使用後は、プログラムを削除してもかまいません。

サンプル・プログラムは、pcmake.bat というバッチ・ファイル、または Visual C++ 6.0 を使用してビルドできます。

関連資料： 3-8 ページの「サンプル・プログラムのビルド」

表 3-1 サンプル・プログラム

サンプル・プログラム	ソース・ファイル	Pro*C/C++ GUI プロジェクト・ファイル	MSVC コンパイラ プロジェクト・ファイル	注意
ANSIDYN1	ansidyn1.pc	ansidyn1.pre	ansidyn1.dsp	-
ANSIDYN2	ansidyn2.pc	ansidyn2.pre	ansidyn2.dsp	-
COLDEMO1	coldemo1.h coldemo1.pc coldemo1.sql coldemo1.typ	coldemo1.pre	coldemo1.dsp	coldemo1 をビルドする前に coldemo1.sql および Object Type Translator（OTT）を実行します。
CPDEMO1	cpdemo1.pc	cpdemo1.pre	cpdemo1.dsp	-
CPDEMO2	cpdemo2.pc	cpdemo2.pre	cpdemo2.dsp	-
CPPDEMO1	cppdemo1.pc	cppdemo1.pre	cppdemo1.dsp	-
CPPDEMO2	cppdemo2.pc empclass.pc cppdemo2.sql empclass.h	cppdemo2.pre	cppdemo2.dsp	cppdemo2 をビルドする前に cppdemo2.sql を実行します。

表 3-1 サンプル・プログラム（続き）

サンプル・プログラム	ソース・ファイル	Pro*C/C++ GUI プロジェクト・ファイル	MSVC コンパイラ プロジェクト・ファイル	注意
CPPDEMO3	cppdemo3.pc	cppdemo3.pre	cppdemo3.dsp	-
CVDEMO	cv_demo.pc cv_demo.sql	cv_demo.pre	cv_demo.dsp	cv_demo をビルドする前に cv_demo.sql を実行します。
EMPCLASS	cppdemo2.pc empclass.pc cppdemo2.sql empclass.h	empclass.pre	empclass.dsp	empclass をビルドする前に cppdemo2.sql を実行します。
LOBDEMO1	lobdemo1.h lobdemo1.pc lobdemo1.sql	lobdemo1.pre	lobdemo1.dsp	lobdemo1 をビルドする前に lobdemo1.sql を実行します。
MLTTHRD1	mltthrd1.pc mltthrd1.sql	mltthrd1.pre	mltthrd1.dsp	mltthrd1 をビルドする前に mltthrd1.sql を実行します。
NAVDEMO1	navdemo1.h navdemo1.pc navdemo1.sql navdemo1.typ	navdemo1.pre	navdemo1.dsp	navdemo1 をビルドする前に navdemo1.sql および OTT を実行します。
OBJDEMO1	objdemo1.h objdemo1.pc objdemo1.sql objdemo1.typ	objdemo1.pre	objdemo1.dsp	objdemo1 をビルドする前に objdemo1.sql および OTT を実行します。
ORACA	oraca.pc oracatst.sql	oraca.pre	oraca.dsp	oraca をビルドする前に oracatst.sql を実行します。
PLSSAM	plssam.pc	plssam.pre	plssam.dsp	-
SAMPLE	sample.pc	sample.pre	sample.dsp	-
SAMPLE1	sample1.pc	sample1.pre	sample1.dsp	-
SAMPLE2	sample2.pc	sample2.pre	sample2.dsp	-
SAMPLE3	sample3.pc	sample3.pre	sample3.dsp	-
SAMPLE4	sample4.pc	sample4.pre	sample4.dsp	-
SAMPLE5	sample5.pc exampbld.sql examplod.sql	sample5.pre	sample5.dsp	sample5 をビルドする前に exampbld.sql、続いて examplod.sql を実行します。
SAMPLE6	sample6.pc	sample6.pre	sample6.dsp	-
SAMPLE7	sample7.pc	sample7.pre	sample7.dsp	-

表 3-1 サンプル・プログラム（続き）

サンプル・プログラム	ソース・ファイル	Pro*C/C++ GUI プロジェクト・ファイル	MSVC コンパイラ プロジェクト・ファイル	注意
SAMPLE8	sample8.pc	sample8.pre	sample8.dsp	-
SAMPLE9	sample9.pc calldemo.sql	sample9.pre	sample9.dsp	sample9 をビルドする前に calldemo.sql を実行します。
SAMPLE10	sample10.pc	sample10.pre	sample10.dsp	-
SAMPLE11	sample11.pc sample11.sql	sample11.pre	sample11.dsp	sample11 をビルドする前に sample11.sql を実行します。
SAMPLE12	sample12.pc	sample12.pre	sample12.dsp	-
SCDEMO1	scdemo1.pc	scdemo1.pre	scdemo1.dsp	-
SCDEMO2	scdemo2.pc	scdemo2.pre	scdemo2.dsp	-
SQLVCP	sqlvcp.pc	sqlvcp.pre	sqlvcp.dsp	-
WINSAM	resource.h winsam.h winsam.ico winsam.pc winsam.rc	winsam.pre	winsam.dsp	-

ここでは、サンプル・プログラムの機能について説明します。

ANSIDYN1
実行時まで認識されない SQL 文を、ANSI 動的 SQL を使用して処理する方法を示します。このプログラムは、ANSI 動的 SQL を使用するための最も簡単な（ただし最も効率的というわけではない）方法です。

ANSIDYN2
実行時まで認識されない SQL 文を、ANSI 動的 SQL を使用して処理する方法を示します。このプログラムでは、バッチ処理および参照セマンティクスで Oracle の拡張機能が使用されています。

COLDEMO1
カリフォルニア州の複数の郡についての調査情報をフェッチします。このプログラムは、コレクション型のデータベース列をナビゲートする様々な方法を示します。

CPDEMO1
接続プーリング機能の使用方法を示します。各接続プーリング・オプションを使用して、パフォーマンスを最適化する方法も示します。

CPDEMO2
比較的複雑な SQL 文のセットでの接続プーリング機能について示し、パフォーマンスの向上がプログラムで使用される SQL 文の種類によって決まることを示します。

CPPDEMO1
従業員番号を指定して、emp 表に従業員の名前、給与および歩合を問い合せます。このプログラムでは、（インジケータ構造体内の）インジケータ変数を使用して、歩合が NULL でないかどうかを判別します。

CPPDEMO2
指定した部門の全従業員の名前を emp 表から取得します（動的 SQL 方法 3）。

CPPDEMO3
すべての営業担当者を検索して名前と総所得（歩合を含む）を出力します。このプログラムは C++ の継承の例です。

CVDEMO
REF カーソルを宣言し、オープンします。

EMPCLASS
EMPCLASS および CPPDEMO2 ファイルは、C++ フレームワーク内での Pro*C/C++ プログラムの記述方法の例を提供するためのものです。EMPCLASS は、emp 表に対する特定の問合せをカプセル化しており、カーソル変数を使用して実装されています。EMPCLASS は、問合せのインスタンスをインスタンス化し、emp クラスに含まれる C++ メンバー関数によりカーソル変数の機能（open、fetch および close）を提供します。empclass.pc ファイルはスタンドアロンのデモ・プログラムではありません。このファイルは、cppdemo2 デモ・プログラムによって使用されます。emp クラスを使用するには、emp クラスのインスタンスを宣言し、このクラスのメンバー関数をコールするドライバ（cppdemo2.pc）を作成する必要があります。

LOBDEMO1
個人の社会保障番号に基づいて、データベースに対して犯罪記録のフェッチおよび追加を行います。このプログラムは、表にアクセスしてラージ・オブジェクト（LOB）を格納するメカニズム、および DBMS_LOB パッケージにより使用可能となるストアド・プロシージャで LOB を操作するメカニズムの例です。

MLTTHRD1
スレッドをプリコンパイラとともに使用する方法を示します。このプログラムでは、スレッドと同じ数のセッションが作成されます。

関連資料： 2-16 ページの「マルチスレッド・アプリケーション」

NAVDEMO1
オブジェクト・キャッシュ内のオブジェクトへのナビゲーショナル・アクセスを示します。

OBJDEMO1
オブジェクトの使用方法を示します。このプログラムでは、オブジェクト型の person および address を操作します。

ORACA
実行時に様々なパフォーマンス・パラメータを判断するために ORACA を使用する方法を示します。

PLSSAM
埋込み PL/SQL ブロックの使用方法を示します。このプログラムでは、データベースに登録されている従業員名を入力するよう要求されます。名前を入力すると PL/SQL ブロックが実行され、4 つの SELECT 文の結果が返されます。

SAMPLE
人事データベースに新しい従業員レコードを追加し、データベースの整合性をチェックします。データベース内の従業員番号には、現在の従業員番号の最大値 +10 の値が自動的に選択されます。

SAMPLE1
Oracle データベースにログオンし、ユーザーに従業員番号の入力を要求し、データベースに従業員の名前、給与および歩合を問い合せて結果を表示します。ユーザーが従業員番号として 0 を入力するまでこの処理を繰り返します。

SAMPLE2
Oracle データベースにログオンし、カーソルを宣言してオープンし、すべての営業担当者の名前、給与および歩合をフェッチして結果を表示します。最後にカーソルをクローズします。

SAMPLE3
Oracle データベースにログオンし、カーソルを宣言してオープンし、配列を使用してバッチでフェッチを実行し、print_rows() 関数を使用して結果を出力します。

SAMPLE4
LONG VARRAW 外部データ型を使用する型同値化の使用方法を示します。

SAMPLE5
口座番号と引出金額を入力するようにユーザーに要求します。プログラムは口座から金額を引き出す前に、口座番号が正しいことおよび引出金額を取り出せるだけの預金があることを確認します。このプログラムは埋込み SQL の使用例です。

SAMPLE6
表を作成して行を挿入し、挿入をコミットして表を削除します（動的 SQL 方法 1）。

SAMPLE7
emp 表に 2 つの行を挿入し、挿入した行を削除します（動的 SQL 方法 2）。

SAMPLE8
指定した部門の全従業員の名前を emp 表から取得します（動的 SQL 方法 3）。

SAMPLE9
scott/tiger アカウントを使用して Oracle データベースに接続します。いくつかのホスト配列を宣言し、PL/SQL ストアド・プロシージャ（CALLDEMO パッケージの GET_EMPLOYEES）をコールします。PL/SQL プロシージャは、最大で ASIZE の値を返します。このプログラムは、すべての行を取得するまで GET_EMPLOYEES をコールし、毎回 ASIZE の配列を取得して、値を出力し続けます。

SAMPLE10
ユーザー名とパスワードを使用して Oracle データベースに接続し、SQL 文を入力するよう要求します。有効な任意の SQL 文を入力できますが、埋込み SQL ではなく通常の SQL 構文を使用する必要があります。入力した文が処理されます。文が問合せであれば、フェッチされた行が表示されます（動的 SQL 方法 4）。

SAMPLE11
カーソル変数を使用して、emp 表からデータをフェッチします。カーソルをオープンする操作には、EMP_DEMO_PKG パッケージ内の PL/SQL ストアド・プロシージャ open_cur を使用しています。

SAMPLE12
動的 SQL 方法 4 を使用して配列をフェッチする方法を示します。

SCDEMO1
Oracle の動的 SQL 方法 4 でスクロール可能カーソルを使用する方法を示します。スクロール可能カーソルは、ANSI の動的 SQL 方法 4 でも使用できます。

SCDEMO2
ホスト配列でスクロール可能カーソルを使用する方法を示します。

SQLVCP
sqlvcp() ファンクションを使用して VARCHAR 構造体の実際のサイズを判断する方法を示します。サイズは、VARCHAR の配列を移動するポインタに加算するオフセットとして使用されています。

このプログラムでは、SQLStmtGetText() ファンクションを使用して、最後に実行された SQL 文のテキストを取得する方法も示しています。

WINSAM
人事データベースに新しい従業員レコードを追加し、データベースの整合性をチェックします。必要な数の従業員名を入力できます。「Employee Record」ダイアログ・ボックスの適切なボタンを選択して SQL コマンドを実行できます。これはサンプル・プログラムの GUI バージョンです。

サンプル表の作成

サンプル・プログラムを実行するには、ユーザー名が `scott` でパスワードが `tiger` のデータベース・アカウントが必要です。サンプル表 `emp` および `dept` が入ったデータベースも必要です。このアカウントは、Oracle9*i* サーバーの初期データベースに含まれています。このアカウントがデータベースにない場合は、サンプル・プログラムを実行する前に作成します。データベースに `emp` 表および `dept` 表がない場合は、`demobld.sql` スクリプトを使用して作成します。

関連資料：『Oracle9*i* Database for Windows 管理者ガイド』

サンプル表を作成するには、次のようにします。

1. SQL*Plus を起動します。
2. ユーザー名 `scott`、パスワード `tiger` で接続します。
3. `demobld.sql` スクリプトを実行します。

```
SQL> @%ORACLE_HOME%¥sqlplus¥demo¥demobld.sql;
```

サンプル・プログラムのビルド

サンプル・プログラムは、次の 2 つの方法でビルドできます。

- 用意された `pcmake.bat` ファイルを使用する方法
- Microsoft Visual C++ 6.0 を使用する方法

pcmake.bat を使用する方法

Pro*C/C++ のデモをコンパイルするための pcmake.bat ファイルは、次の場所にあります。

```
%ORACLE_HOME%¥precomp¥demo¥proc
```

このバッチ・ファイルは、コマンド・プロンプトで Pro*C/C++ アプリケーションを作成する方法を示すよう設計されています。

このバッチ・ファイルを使用するには、Microsoft Visual Studio がインストールされている必要があります。環境変数 MSVCDir を設定する必要があります。Pro*C/C++ コマンドライン・オプションおよびリンカー・オプションは、アプリケーションによって異なります。

このファイルを使用してデモをビルドできます。たとえば、sample1 をビルドするには、次のようにします。

1. デモ・ファイルの場所に移動します。コマンド・プロンプトで次のように入力します。

   ```
   C:¥> CD %ORACLE_HOME%¥precomp¥demo¥proc¥sample1
   ```

2. 次のように入力します。

   ```
   C:¥> %ORACLE_HOME%¥precomp¥demo¥proc¥pcmake sample1
   ```

Microsoft Visual C++ を使用する方法

Microsoft Visual C++ 6.0 のプロジェクト・ファイルには、拡張子 .dsp が付いています。%ORACLE_HOME%¥precomp¥demo¥proc ディレクトリの .dsp ファイルにより、サンプル・プログラムをプリコンパイル、コンパイルおよびリンクするために必要な手順が示され、制御されます。

Pro*C/C++、SQL*Plus および OTT は、Microsoft Visual C++ のサンプル・プロジェクト・ファイルに統合されています。コンパイル前に、Pro*C/C++、SQL*Plus および OTT を個別に実行する必要はありません。

関連資料：

- 2-10 ページの「プリコンパイラのオプションの設定」
- 3-10 ページの「サンプルの .pre ファイルのパスの設定」
- 付録 A「Microsoft Visual C++ への Pro*C/C++ の統合」
- OTT の詳細は、『Pro*C/C++ Precompiler プログラマーズ・ガイド』

サンプル・プログラムをビルドするには、次のようにします。

1. sample1.dsp などの Visual C++ のプロジェクト・ファイルを開きます。
2. プロジェクト・ファイルに指定されているパスを調べ、使用しているシステムの構成に対応していることを確認します。対応していない場合は正しくパスを変更します。コンポーネントへのパスに間違いがあるとシステムはエラー・メッセージを生成します。

> **注意：** サンプル・プログラムはすべて、C:¥oracle¥ora92 をデフォルトのドライブとして作成されています。

3. 「ビルド」→「リビルド」を選択します。Visual C++ によって実行ファイルが作成されます。

サンプルの .pre ファイルのパスの設定

デフォルトでは、サンプルの .pre ファイルは対応する .pc ファイルを C:¥oracle¥ora92 ディレクトリ内で検索します。C:¥ は使用しているドライブで、oracle¥ora92 は Oracle ホームの場所を表します。使用しているコンピュータで Oracle ベース・ディレクトリと Oracle ホーム・ディレクトリが異なる場合は、ディレクトリ・パスを正しいパスに変更する必要があります。

サンプルの .pre ファイルのディレクトリ・パスを変更するには、次のようにします。

1. Pro*C/C++ で .pre ファイルを開きます。
2. 「入力ファイル」領域でファイル名をダブルクリックし、「入力ファイル」ダイアログ・ボックスを表示します。
3. ディレクトリ・パスを正しいパスに変更します。
4. 「開く」をクリックします。

A

Microsoft Visual C++ への Pro*C/C++ の統合

この付録では、Pro*C/C++ を Microsoft Visual C++ 開発環境に統合する方法について説明します。

この付録の項目は次のとおりです。

Pro*C/C++ の Microsoft Visual C++ プロジェクトへの統合

この項では、Microsoft Visual C++ プロジェクトに Pro*C/C++ を完全に統合する方法について説明します。

プリコンパイラのすべてのエラーと警告は、Microsoft Visual C++ がコンパイラおよびリンカーのメッセージを表示する出力ボックスに表示されます。Microsoft Visual C++ のビルド環境とは別にファイルをプリコンパイルする必要はありません。さらに重要なことは、Microsoft Visual C++ により、.c ファイルと .pc ファイルの間の依存性がメンテナンスされることです。必要であれば、Microsoft Visual C++ によって、依存性とプリコンパイル・ファイルがメンテナンスされます。

この項で説明する手順はすべて、Microsoft Visual C++ 内で実行します。

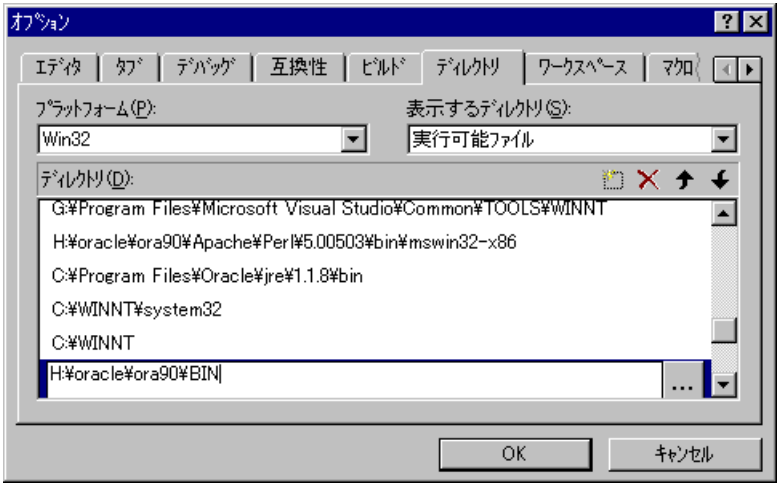
Pro*C/C++ 実行可能プログラムの場所の指定

Microsoft Visual C++ から Pro*C/C++ を実行するには、Microsoft Visual C++ によって Pro*C/C++ 実行可能プログラムの場所が認識されている必要があります。Microsoft Visual C++ をインストールした後でリリース 9.2 の Oracle 製品をインストールした場合は、ディレクトリ・パスを追加してください。

Pro*C/C++ 実行可能プログラムの場所を指定するには、次のようにします。

1. 「ツール」メニューから「オプション」を選択します。

   「オプション」ダイアログ・ボックスが表示されます。



2. 「ディレクトリ」タブをクリックします。
3. 「表示するディレクトリ」リスト・ボックスから「実行可能ファイル」を選択します。
4. 「ディレクトリ」ボックスの一番下までスクロールして、点線の長方形をクリックします。
5. %ORACLE_HOME%¥bin ディレクトリを入力します。次に例を示します。

   ```
   H:¥oracle¥ora92¥bin
   ```

6. 「OK」をクリックします。

Pro*C/C++ ヘッダー・ファイルの場所の指定

Pro*C/C++ ヘッダー・ファイルの場所を指定するには、次のようにします。

1. 「ツール」メニューから「オプション」を選択します。「オプション」ダイアログ・ボックスが表示されます。
2. 「ディレクトリ」タブをクリックします。
3. 「表示するディレクトリ」リスト・ボックスから「インクルード ファイル」を選択します。
4. 「ディレクトリ」ボックスの一番下までスクロールして、点線の長方形をクリックします。
5. %ORACLE_HOME%¥precomp¥public ディレクトリを入力します。次に例を示します。

   ```
   H:¥oracle¥ora92¥precomp¥public
   ```

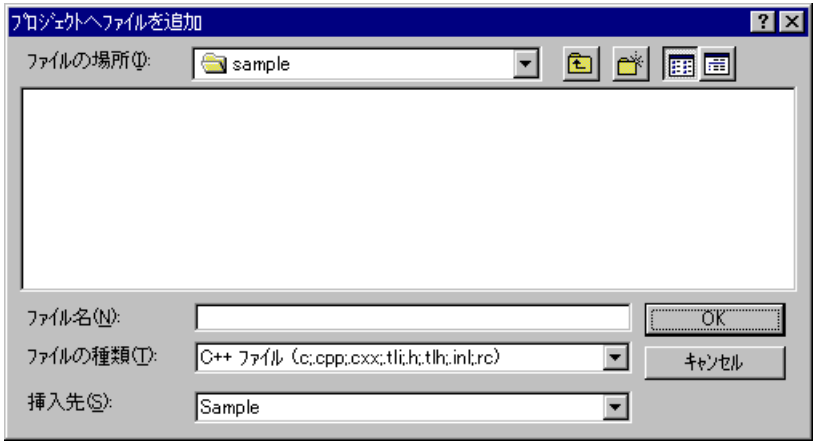
6. 「OK」をクリックします。

プロジェクトへの .pc ファイルの追加

プロジェクトの作成後、.pc ファイルを追加する必要があります。

プロジェクトに .pc ファイルを追加するには、次のようにします。

1. 「プロジェクト」メニューから「プロジェクトへ追加」→「ファイル」を選択します。「プロジェクトへファイルを追加」ダイアログ・ボックスが表示されます。



2. 「ファイルの種類」リスト・ボックスで「すべてのファイル」を選択します。
3. .pc ファイルを選択します。
4. 「OK」をクリックします。

プロジェクトへの .c ファイルの参照の追加

各 .pc ファイルに対して、プリコンパイルの結果作成される .c ファイルへの参照を追加する必要があります。

.c ファイルへの参照をプロジェクトに追加するには、次のようにします。

1. 「プロジェクト」メニューから「プロジェクトへ追加」→「ファイル」を選択します。「プロジェクトへファイルを追加」ダイアログ・ボックスが表示されます。
2. 「ファイル名」フィールドに、.c ファイルの名前を入力します。
3. 「OK」をクリックします。.c ファイルがまだ作成されていないため、Microsoft Visual C++ により、「指定されたファイルは存在しません。プロジェクトに対するファイルの参照を追加しますか?」というメッセージが表示されます。
4. 「はい」をクリックします。

プロジェクトへの Pro*C/C++ のライブラリの追加

Pro*C/C++ アプリケーションは、ライブラリ・ファイル orasql9.lib とリンクする必要があります。

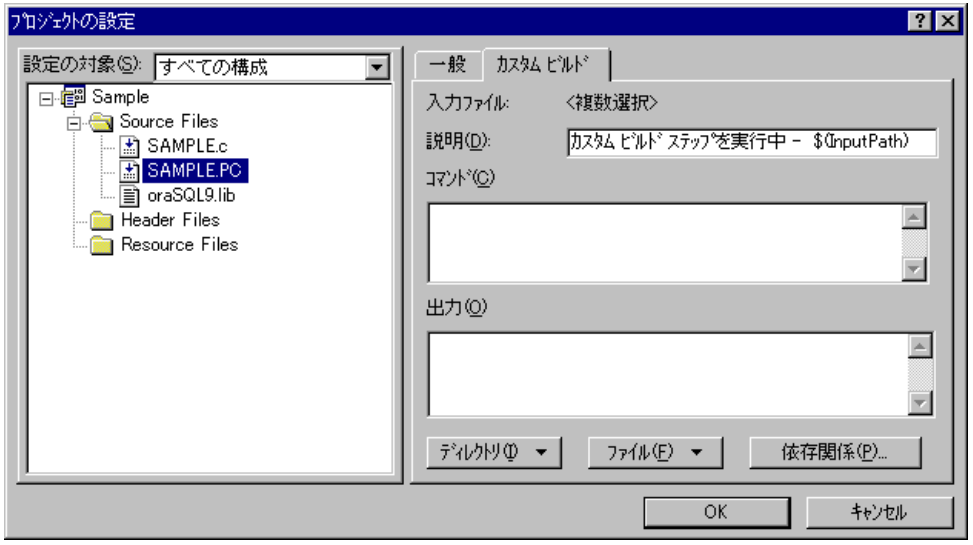
プロジェクトへ Pro*C/C++ のライブラリを追加するには、次のようにします。

1. 「プロジェクト」メニューから「プロジェクトへ追加」→「ファイル」を選択します。「プロジェクトへファイルを追加」ダイアログ・ボックスが表示されます。
2. 「ファイルの種類」リスト・ボックスで「すべてのファイル」を選択します。
3. %ORACLE_HOME%¥precomp¥lib¥msvc ディレクトリから orasql9.lib を選択します。
4. 「OK」をクリックします。

「カスタム ビルド」オプションの指定

「カスタム ビルド」オプションを指定するには、次のようにします。

1. FileView で .pc ファイルを右クリックして「設定」を選択します。「プロジェクト設定」ダイアログ・ボックスが、「カスタム ビルド」タブが表示された状態で表示されます。



2. 「コマンド」ボックスに、$ORACLE_HOME 設定と同一のハードコードされたパスを使用するビルドを 1 行で設定します。
3. 「出力」ボックスに次のいずれかを入力します。
   - .c ファイルを生成している場合は、$(ProjDir)¥$(InputName).c と入力します。
   - .cpp ファイルを生成している場合は、$(ProjDir)¥$(InputName).cpp と入力します。

   $(ProjDir) および $(InputName) は、Microsoft Visual C++ のカスタム・ビルド・コマンドのマクロです。プロジェクトの作成時、Microsoft Visual C++ では出力ファイルの日付をチェックして、ソース・コードへの新たな変更のためにプロジェクトを再作成する必要があるかどうかを判断します。
4. 「OK」をクリックします。

関連資料： Microsoft Visual C++ のドキュメント

「ツール」メニューへの Pro*C/C++ の追加

Microsoft Visual C++ の「ツール」メニューに選択項目として「Pro*C/C++」を追加できます。

「ツール」メニューに「Pro*C/C++」を追加するには、次のようにします。

1. Microsoft Visual C++ で、「ツール」メニューから「カスタマイズ」を選択します。「カスタマイズ」ダイアログ・ボックスが表示されます。

2. 「ツール」タブをクリックします。
3. 「メニュー項目」ボックスの一番下までスクロールして点線の長方形をクリックします。
4. 次のように入力します。

   ```
   Pro*C/C++
   ```

5. 「コマンド」ボックスに Pro*C/C++ のグラフィカルな実行可能ファイルのパスとファイル名を入力するか、ボックスの右側のブラウズ・ボタンを使用してファイル名を選択します。次に例を示します。

   ```
   H:¥oracle¥ora92¥bin¥procui.exe
   ```

6. 「引数」ボックスに、次のように入力します。

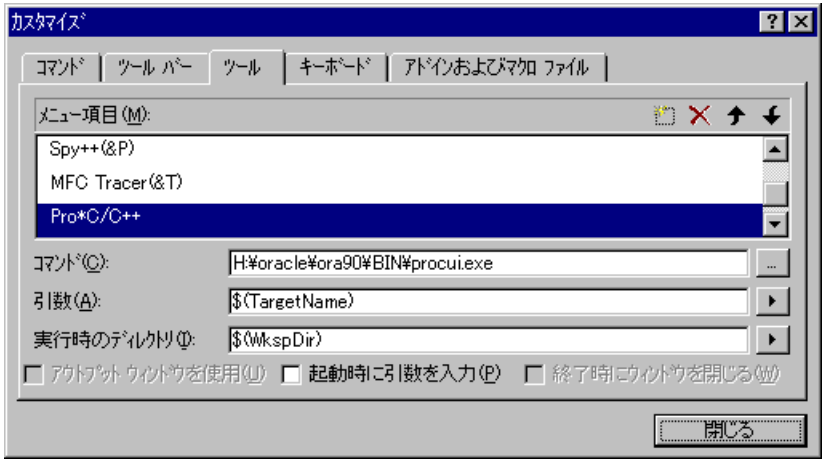
   ```
   $(TargetName)
   ```

   「ツール」メニューから「Pro*C/C++」を選択すると、Microsoft Visual C++ によって `$(TargetName)` 引数が使用され、現行の開発プロジェクトの名前が Pro*C/C++ に渡されます。これにより、Pro*C/C++ は開いているプロジェクトと同じ名前で拡張子が `.pre` の、プロジェクト・ディレクトリにあるプリコンパイル・プロジェクトを開きます。

7. 「実行時のディレクトリ」ボックスに、次のように入力します。

   ```
   $(WkspDir)
   ```

   この時点で「カスタマイズ」ダイアログ・ボックスの内容は次の図のようになります(ただし使用するコンピュータによっては Oracle ホーム・ディレクトリが異なる場合もあります)。



8. 「閉じる」をクリックします。Microsoft Visual C++ の「ツール」メニューに「Pro*C/C++」が追加されます。

索引

S

う

お

き

こ

さ

り