Oracle® Solaris Studio 12.2 IDE クイックスタートチュートリアル

2010 年 9 月

プロジェクトの作成

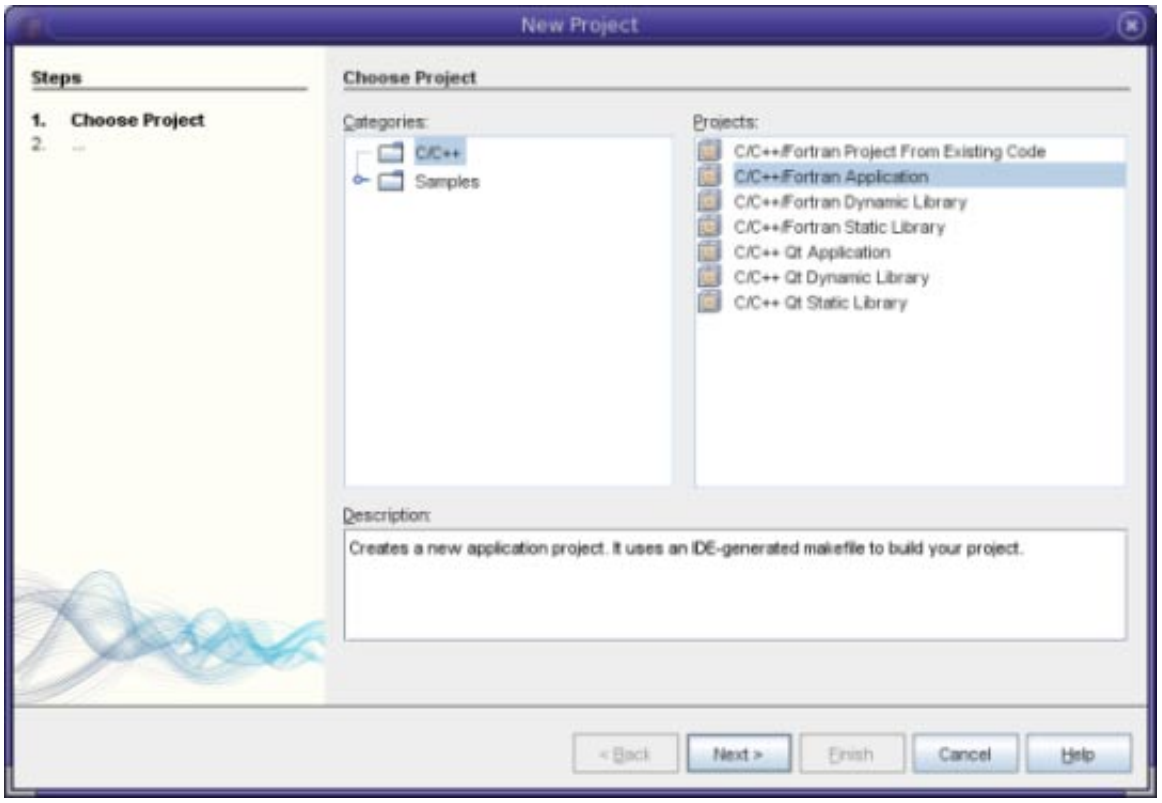
Oracle Solaris Studio では、生成済みメイクファイルを使用して C、C++、および Fortran アプリケーションやライブラリのプロジェクトを作成したり、既存のソースコードを持つプロジェクトを作成したりすることができます。

プロジェクトの構築、実行、およびデバッグは、IDE を起動したローカルホスト上、および Solaris オペレーティングシステムもしくは Linux オペレーティングシステムを実行しているリモートホストで行うことができます。

C/C++/Fortran アプリケーション、動的ライブラリ、もしくは静的ライブラリプロジェクトでは、IDE はアプリケーションの構築、実行、およびデバッグ方法のあらゆる面を制御します。プロジェクトを作成する際、または「プロジェクトのプロパティー (Project Properties)」ダイアログボックスで、プロジェクト設定を指定します。IDE はメイクファイルを生成し、ここにはすべての設定が保存されます。

アプリケーションプロジェクトの作成

1. 「ファイル (File)」>「新規プロジェクト (New Project)」を選択して、新規プロジェクトウィザードを開きます。
2. ウィザードで、C/C++ カテゴリを選択します。
3. ウィザードでは、新規プロジェクトのタイプを選択できます。「C/C++/Fortran アプリケーション (C/C++/Fortran Application)」を選択して、「次へ (Next)」をクリックします。



4. デフォルト値を使用して、新しい C/C++/Fortran アプリケーションプロジェクトを作成します。プロジェクトの名前とプロジェクトの場所を選択できます。
5. 「完了 (Finish)」をクリックしてウィザードを終了します。

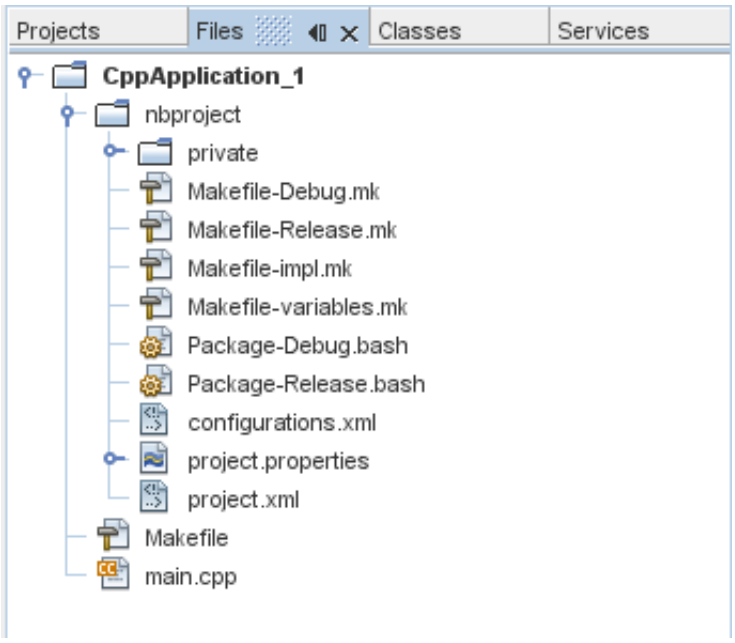
プロジェクトが作成され、いくつかの論理フォルダが作成されます。論理フォルダはディレクトリではありません。ファイルを整理する手段であり、ファイルがディスク上に物理的に保存される場所を示すものではありません。論理フォルダに追加されるファイルは自動的にプロジェクトの一部となり、プロジェクトを構築する際にコンパイルされます。

「重要なファイル」フォルダに追加されたファイルはプロジェクトの一部ではなく、プロジェクトの構築時にコンパイルされません。これらのファイルは参照用のみで、既存のメイクファイルがプロジェクトにある場合に便利です。

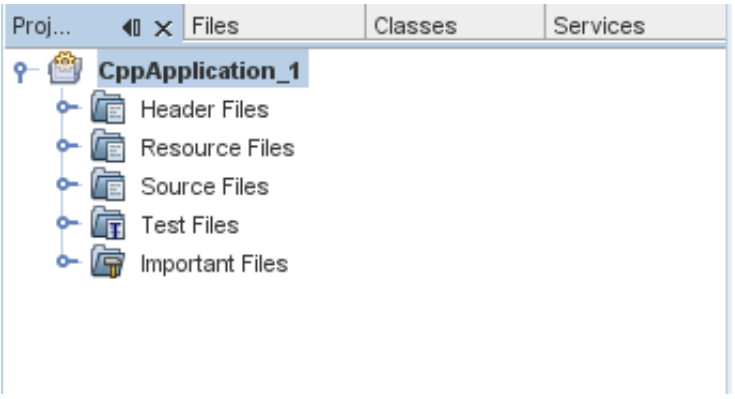
プロジェクトの論理ビューと物理ビューの切り換え

プロジェクトには、論理ビューと物理ビューがあります。プロジェクトの論理ビューと物理ビューを切り換えられます。

1. 「ファイル (Files)」タブを選択します。このウィンドウには、プロジェクトの物理ビューが表示されます。ディスクに保存されているファイルとフォルダが表示されます。



2. 「プロジェクト (Projects)」タブを選択します。このウィンドウには、プロジェクトの論理ビューが表示されます。



ファイルとフォルダのプロジェクトへの追加

論理フォルダをプロジェクトに追加できます。

1. `CppApplication_1` プロジェクトのプロジェクトノードを右クリックして、「新規論理フォルダ (New Logical Folder)」を選択します。新しい論理フォルダがプロジェクトに追加されます。
2. 新しい論理フォルダを右クリックして、「名前の変更 (Rename)」を選択します。新しいフォルダに付ける名前を入力します。

ファイルとフォルダの両方を既存のフォルダに追加できます。論理フォルダは入れ子にすることができます。

新規ファイルのプロジェクトへの追加

新しいファイルをプロジェクトに追加できます。

1. 「ソースファイル (Source Files)」フォルダを右クリックして、「新規 (New)」>「C main ファイル (Main C File)」を選択します。
2. 「名前と場所 (Name and Location)」ページで、`newfile` が「ファイル名 (File Name)」フィールドに表示されます。

3. 「完了 (Finish)」をクリックします。

`newfile.c` ファイルがプロジェクトディレクトリのディスクに作成され、「ソースファイル (Source Files)」フォルダに追加されます。このフォルダには、ソースファイルだけでなく、任意の種類のファイルを追加できます。

その他の新規ファイルのプロジェクトへの追加

1. 「ヘッダーファイル (Header Files)」フォルダを右クリックして、「新規 (New)」>「C ヘッダーファイル (C Header File)」を選択します。
2. 「名前と場所 (Name and Location)」ページで、`newfile` が「ファイル名 (File Name)」フィールドに表示されます。
3. 「完了 (Finish)」をクリックします。

`newfile.h` ファイルがプロジェクトディレクトリのディスクに作成され、「ヘッダーファイル (Header Files)」フォルダに追加されます。

既存ファイルのプロジェクトへの追加

2 つの方法で、既存のファイルをプロジェクトに追加できます。

- 「ソースファイル (Source Files)」フォルダを右クリックして、「既存の項目を追加 (Add Existing Item)」を選択します。「項目を選択 (Select Item)」ダイアログボックスを使用してディスク上の既存ファイルを選択して、ファイルをプロジェクトに追加できます。
- 「ソースファイル (Source Files)」フォルダを右クリックして、「フォルダから既存の項目を追加 (Add Existing Items from Folders)」を選択します。「フォルダの追加 (Add Folders)」ダイアログボックスを使用して、既存ファイルを含むフォルダを追加します。

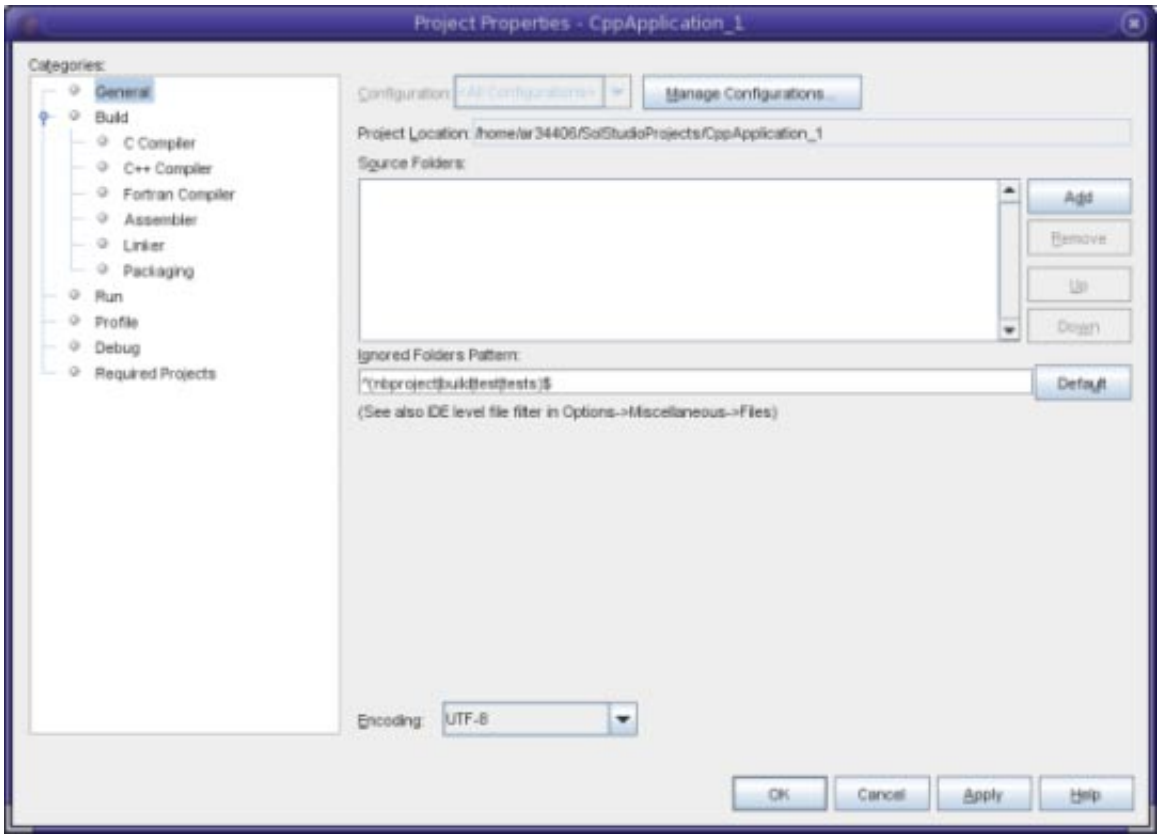
既存の項目の追加に「新規 (New)」メニュー項目を使用しないでください。「名前と場所 (Name and Location)」パネルから、ファイルがすでに存在していることが通知されます。

プロジェクトプロパティーの設定

プロジェクトを作成するとき、デバッグとリリースという 2 つの構成があります。構成はプロジェクトに使用される一連の設定で、多くのプロパティー設定を一度に簡単に切り換えられます。デバッグ構成では、デバッグ情報を含むバージョンのアプリケーションを構築します。リリース構成では、最適化バージョンを構築します。

「プロジェクトのプロパティー (Project Properties)」ダイアログボックスには、プロジェクトの構築情報と構成情報が含まれています。「プロジェクトのプロパティー (Project Properties)」ダイアログボックスを開くには、次の手順に従います。

- アプリケーションプロジェクトのプロジェクトノードを右クリックして、「プロパティー (Properties)」を選択します。



左側のパネルでノードを選択して右側のパネルでプロパティーを変更して、「プロジェクトのプロパティー (Project Properties)」ダイアログボックスでコンパイル設定およびその他の構成設定を変更できます。ノードとプロパティー値を選択して、設定できるプロパティーに注目します。一般プロパティーを設定すると、プロジェクトのすべての構成で設定が行われます。構築、実行、もしくはデバッグプロパティーを設定すると、現在設定されている構成のプロパティーが設定されます。

構成の管理

「プロジェクトのプロパティー (Project Properties)」ダイアログボックスで変更されたプロパティーは、現在の構成のメイクファイルに保存されます。デフォルトの構成を編集したり、新しい構成を作成したりできます。新しい構成を作成するには、次の手順に従います。

1. 「プロジェクトのプロパティー (Project Properties)」ダイアログボックスで「構成を管理 (Manage Configurations)」ボタンをクリックします。
2. 「構成 (Configurations)」ダイアログボックスで、目的の構成にもっとも近い構成を選択します。この場合、Release 構成を選択して、「コピー (Copy)」ボタンをクリックします。その後、「名前を変更 (Rename)」をクリックします。
3. 「名前を変更 (Rename)」ダイアログボックスで、構成の名前を「PerformanceRelease」に変更します。「OK」をクリックします。
4. 「構成 (Configurations)」ダイアログボックスで「OK」をクリックします。
5. 「プロジェクトのプロパティー (Project Properties)」ダイアログボックスで、左側のパネルの「C コンパイラ (C Compiler)」ノードを選択します。PerformanceRelease 構成は「構成 (Configuration)」ドロップダウンリストで選択されています。
6. 右側のパネルのプロパティーシートで、「開発モード (Development Mode)」を Release から PerformanceRelease に変更します。「OK」をクリックします。

別のオプションセットでアプリケーションをコンパイルする、新しい構成が作成されました。

ソースファイルのプロパティーの設定

プロジェクトにプロジェクトプロパティーを設定すると、関連するプロパティーがプロジェクト内のすべてのファイルに適用されます。特定のファイルにプロパティーを設定できます。

1. `newfile.c` ソースファイルを右クリックして、「プロパティー (Properties)」を選択します。
2. 「カテゴリ (Categories)」パネルの「一般 (General)」ノードをクリックして、このファイルの構築に別のコンパイラまたはその他のツールを選択できることを確認します。チェックボックスを選択して、現在選択されているプロジェクト構成の構築からファイルを除外することもできます。
3. 「C コンパイラ (C Compiler)」ノードをクリックして、プロジェクトコンパイラ設定およびこのファイルのその他のプロパティーを上書きできることを確認します。
4. 「プロジェクトのプロパティー (Project Properties)」ダイアログボックスをキャンセルします。

主プロジェクトの設定

「プロジェクト (Projects)」ウィンドウのプロジェクトノードを右クリックすると、選択したプロジェクトで実行できるアクションのポップアップメニューが表示されます。同時に複数のプロジェクトを開くと、プロジェクトノードのポップアップメニューが開き、そのプロジェクトで操作していることがわかります。

メニューバーおよびツールバー上のプロジェクト関連のアクションの多くは、主プロジェクトに対して動作します。主プロジェクトノードは「プロジェクト (Projects)」ウィンドウでボールドテキストで表示されます。

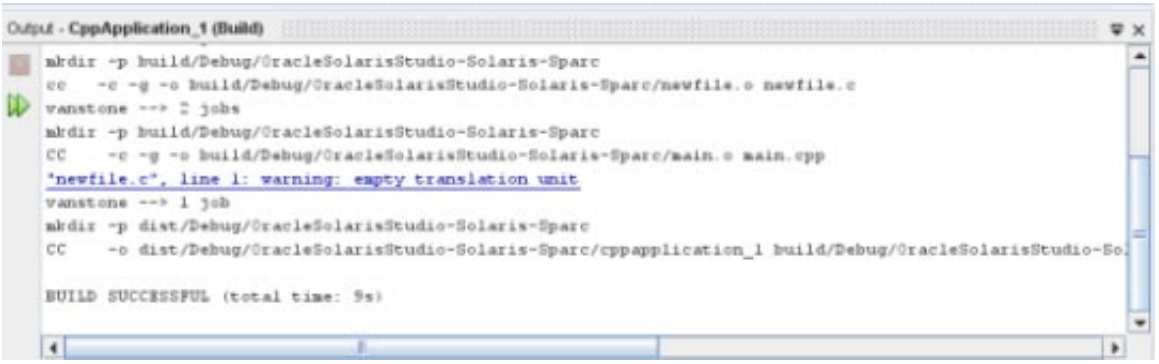
IDE で主プロジェクトを変更するには、次の手順に従います。

- 目的のプロジェクトノードを右クリックして、「主プロジェクトとして設定 (Set as Main Project)」を選択します。このプロジェクトは IDE で主プロジェクトとなり、メニューバーおよびツールバーのアクションはこのプロジェクトを参照するようになります。

プロジェクトの構築

プロジェクトを構築するには、次の手順に従います。

1. プロジェクトを右クリックして、「構築 (Build)」を選択します。プロジェクトが構築されます。構築生成物が「出力 (Output)」ウィンドウに表示されます。



2. メインツールバーの「構成 (configuration)」ドロップダウンリストで、構成を Debug から PerformanceRelease に変更します。プロジェクトは PerformanceRelease 構成を使用して構築されるようになります。
3. プロジェクトを右クリックして、「構築 (Build)」を選択します。プロジェクトが構築されます。構築生成物が「出力 (Output)」ウィンドウに表示されます。

プロジェクトの複数の構成を同時に構築するには、「実行 (Run)」>「主プロジェクトをバッチ構築 (Batch Build Main Project)」を選択して、「バッチ構築 (Batch Build)」ダイアログボックスで構築する構成を選択します。

プロジェクトを右クリックしてメニューからアクションを選択して、プロジェクトを構築、クリーン、およびクリーンと構築の両方を実行できます。プロジェクトにはオブジェクトファイルと実行可能ファイルが構成ごとに保管されるため、複数の構成でファイルが混在する心配はありません。

単体のファイルのコンパイル

単体のソースファイルをコンパイルするには、次の手順に従います。

- `newfile.c` ファイルを右クリックして、「ファイルのコンパイル (Compile File)」を選択します。このファイルのみがコンパイルされます。

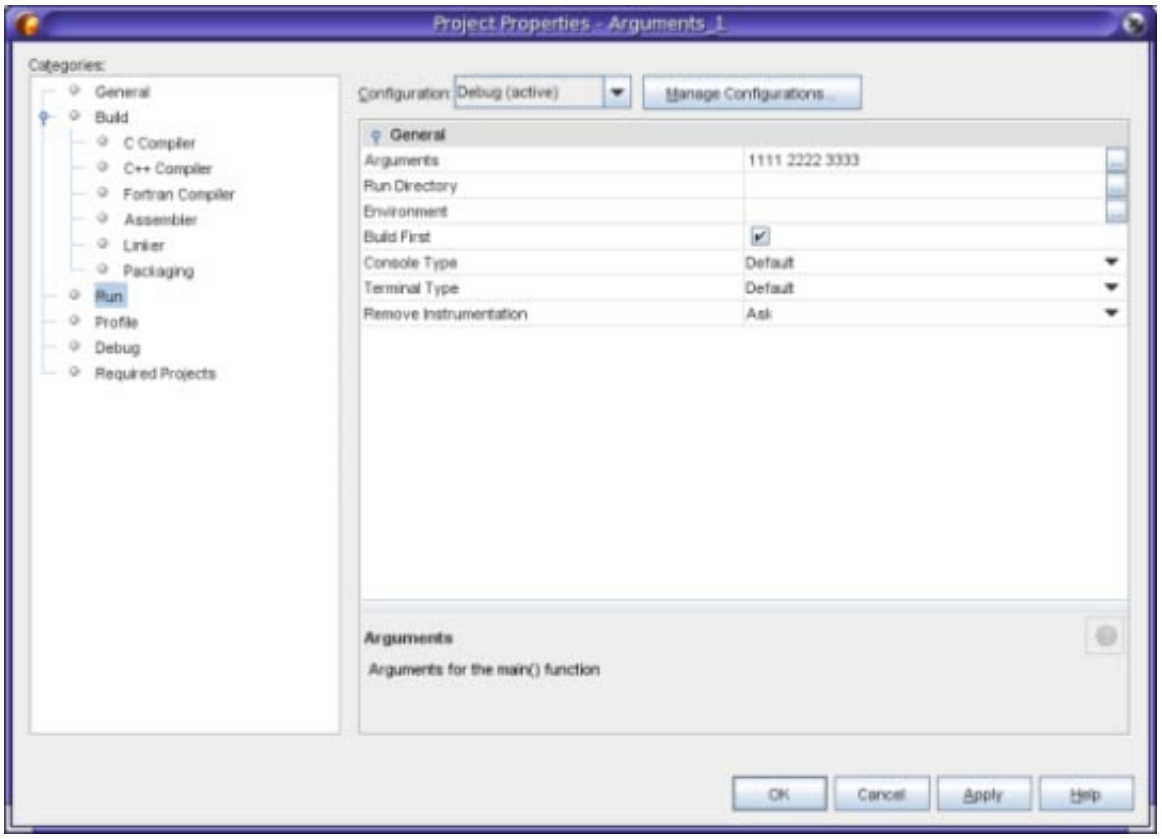
注 – 単体のファイルのコンパイルは、既存のコードからの C/C++/Fortran プロジェクトのプロジェクトタイプではサポートされていません。

プロジェクトの実行

`Arguments` サンプルプログラムは、コマンドライン引数を出力します。このプログラムを実行する前に、現在の構成で引数を設定します。その後、プログラムを実行します。

`Arguments_1` プロジェクトを作成するには、引数を設定してプロジェクトを実行します。

1. 「ファイル (File)」>「新規プロジェクト (New Project)」を選択します。
2. プロジェクトウィザードで、「サンプル (Samples)」カテゴリを展開します。
3. 「C/C++」サブカテゴリを選択して、`Arguments` プロジェクトを選択します。「次へ (Next)」をクリックして、「完了 (Finish)」をクリックします。
4. `Arguments_1` プロジェクトノードを右クリックして、「構築 (Build)」を選択します。プロジェクトが構築されます。
5. `Arguments_1` プロジェクトノードを右クリックして、「プロパティー (Properties)」を選択します。
6. 「プロジェクトのプロパティー (Project Properties)」ダイアログボックスで、「実行 (Run)」ノードを選択します。
7. 「引数 (Arguments)」テキストフィールドに、`1111 2222 3333` と入力します。「OK」をクリックします。



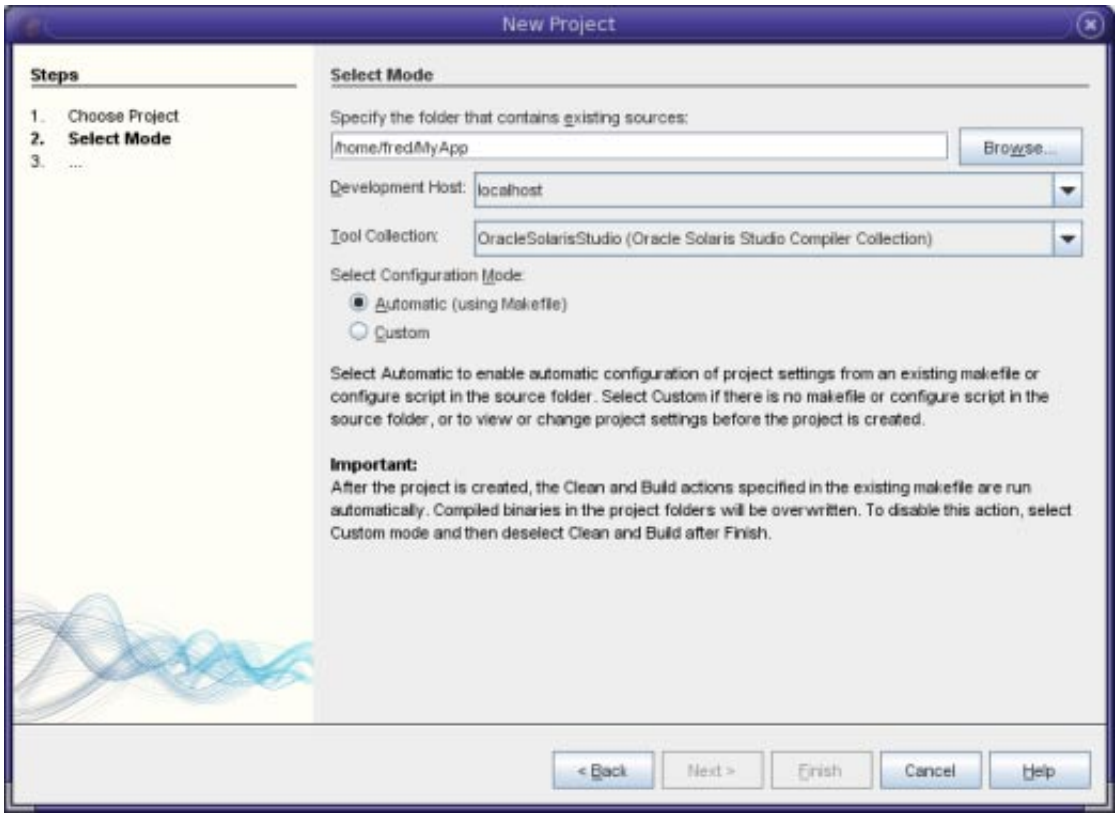
8. 「実行 (Run)」>「主プロジェクトを実行 (Run Main Project)」を選択します。アプリケーションが構築され、実行されます。引数は外部ウィンドウに表示されます。

ヒント – プロジェクトを実行すると「モニターの実行 (Run Monitor)」タブが開き、アプリケーションの動作を監視するプロファイルツールが表示されます。プロファイルツールは「プロジェクトのプロパティー (Project Properties)」ダイアログボックスでオフにできます。

既存のコードからのプロジェクトの作成

「既存のコードからの C/C++/Fortran プロジェクト (C/C++/Fortran Project From Existing Code)」で、IDE は既存のメイクファイルの命令を使用してアプリケーションをコンパイルおよび実行します。

1. 「ファイル (File)」>「新規プロジェクト (New Project)」を選択します。
2. C/C++ カテゴリを選択します。
3. 「既存のコードからの C/C++/Fortran プロジェクト (C/C++/Fortran Project From Existing Code)」を選択して「次へ (Next)」をクリックします。
4. 新規プロジェクトウィザードの「モードを選択 (Select Mode)」ページで、「参照 (Browse)」ボタンをクリックします。「プロジェクトフォルダを選択 (Select Project Folder)」ダイアログボックスで、ソースコードがあるディレクトリに移動します。「選択 (Select)」をクリックします。



5. デフォルトの構成モード「自動 (Automatic)」を使用します。「完了 (Finish)」をクリックします。
6. プロジェクトが作成され、「プロジェクト (Projects)」ウィンドウで開きます。また、既存のメイクファイルで指定された Clean セクションと Build セクションを IDE が自動的に実行します。プロジェクトにコード支援が自動的に設定されます。

プロジェクトが作成され、「プロジェクト (Projects)」ウィンドウで開きます。既存のコードの thin ラッパーとなるプロジェクトが作成されました。

プロジェクトの構築と再構築

プロジェクトを構築するには、次の手順に従います。

- プロジェクトのプロジェクトノードを右クリックして、「構築 (Build)」を選択します。

プロジェクトを再構築するには、次の手順に従います。

- プロジェクトのプロジェクトノードを右クリックして、「生成物を削除して構築 (Clean and Build)」を選択します。

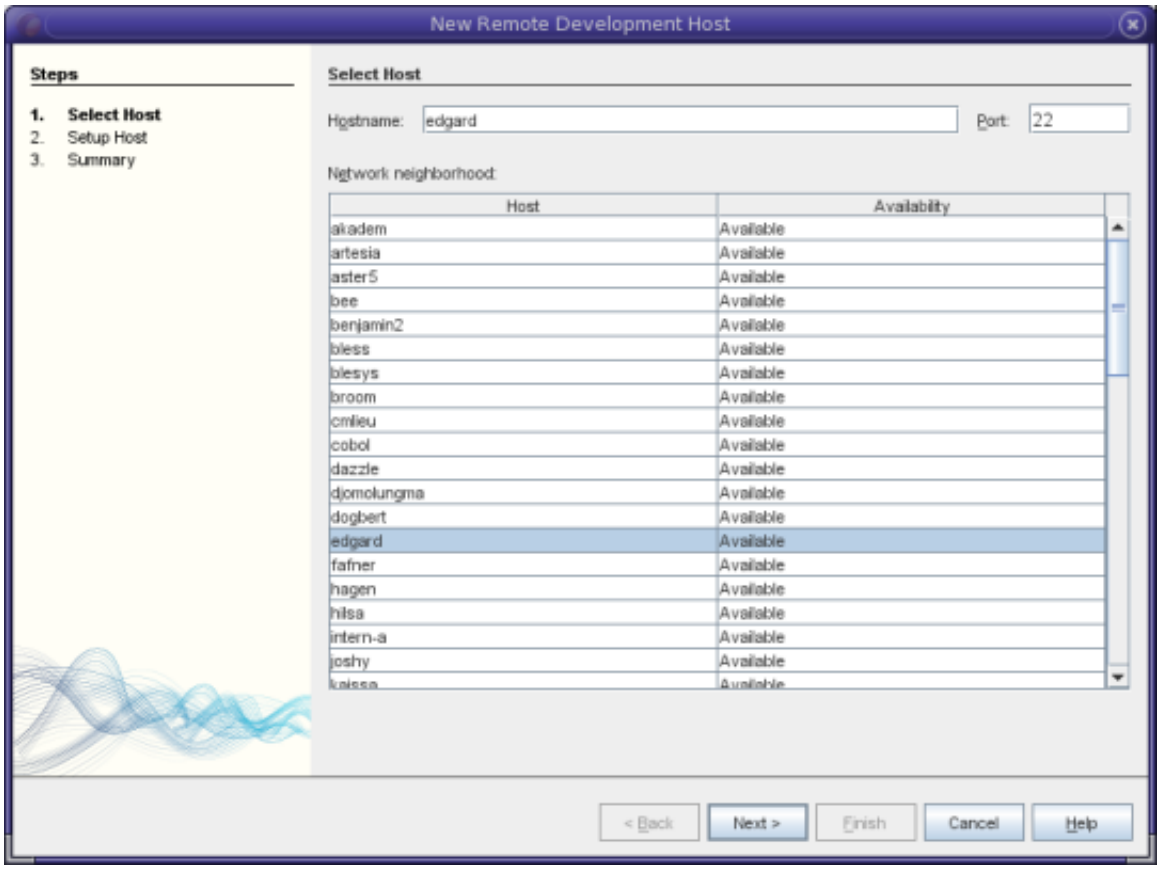
リモート開発の実行

ローカルホスト (IDE を起動したシステム)、または UNIX® オペレーティングシステムを実行しているリモートホスト上で、プロジェクトを構築、実行、デバッグできます。

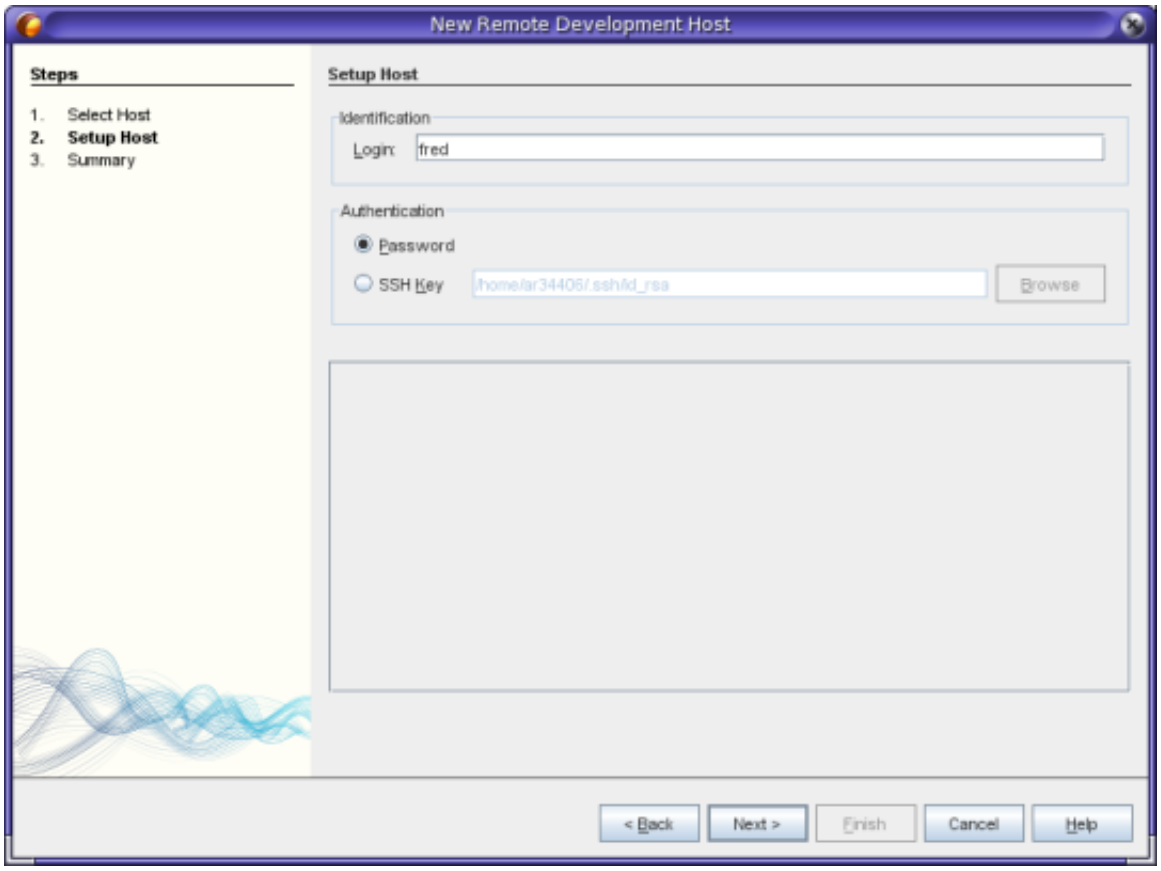
「オプション (Options)」ダイアログボックスの「構築ツール (Build Tools)」タブで、リモート開発ホストを定義できます。リモートホストを追加するには、次の手順に従います。

1. 「ツール (Tools)」>「オプション (Options)」を選択します。
2. 「オプション (Options)」ダイアログボックスの「構築ツール (Build Tools)」タブで、「編集 (Edit)」をクリックします。
3. 「開発ホストマネージャー (Development Host Manager)」ダイアログボックスで、「追加 (Add)」をクリックします。

4. 「新規リモート開発ホスト(New Remote Development Host)」ウィザードの「ホストを選択(Select Host)」ページで、「ホスト名 (Hostname)」フィールドにホストのシステム名を入力するか、または「隣接ネットワーク (Network neighborhood)」リスト内の使用できるホストをダブルクリックして選択します。「次へ」をクリックします。



5. 「ホストのセットアップ (Setup Host)」ページで、「ログイン (Login)」フィールドにログイン名を入力して「次へ (Next)」をクリックします。



6. ウィザードからパスワードが要求され、ホストに接続して「概要 (Summary)」ページが表示されます。「完了 (Finish)」をクリックします。
7. ホストが「開発ホストマネージャー (Development Hosts Manager)」ダイアログボックスの「開発ホスト (Development Hosts)」リストに追加されたら、「OK」をクリックします。

リモートホストにプロジェクトを開発するには、プロジェクトはローカルホストとリモートホストの両方で参照できる共有ファイルシステム上に存在する必要があります。通常このようなファイルシステムは、NFS または Samba を使用して共有されます。リモートホストを定義するときに、プロジェクトソースファイルへのローカルパスとリモートパスの間のマッピングを定義できます。

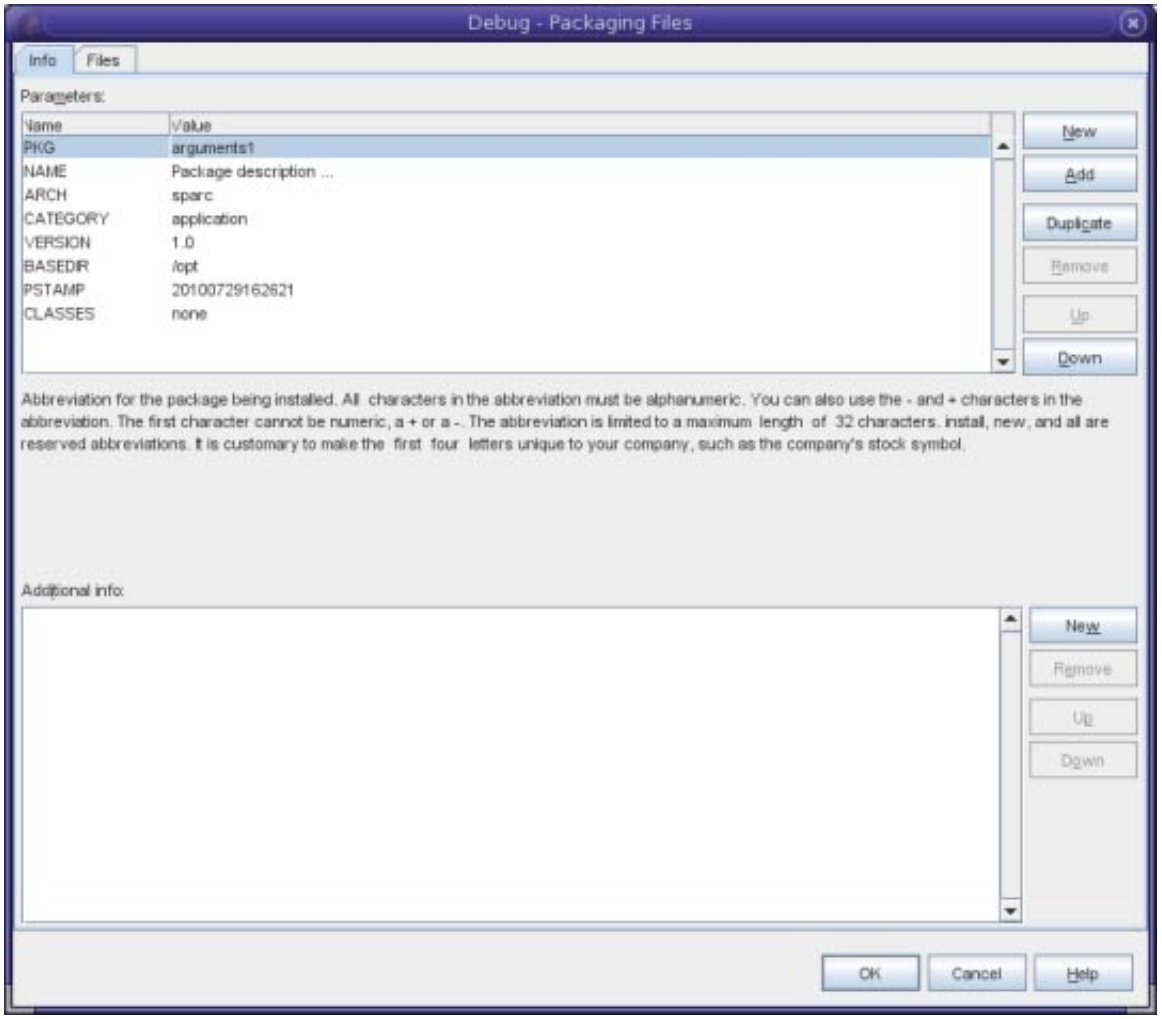
プロジェクトを作成するとき、現在の開発ホストがプロジェクトの開発ホストとして選択されます。「プロジェクトのプロパティー (Project Properties)」ダイアログボックスの「構築 (Build)」パネルで、プロジェクトの開発ホストを変更できます。実行可能ファイルまたはコアファイルをデバッグするときに、開発ホストを指定することもできます。

アプリケーションのパッケージ作成

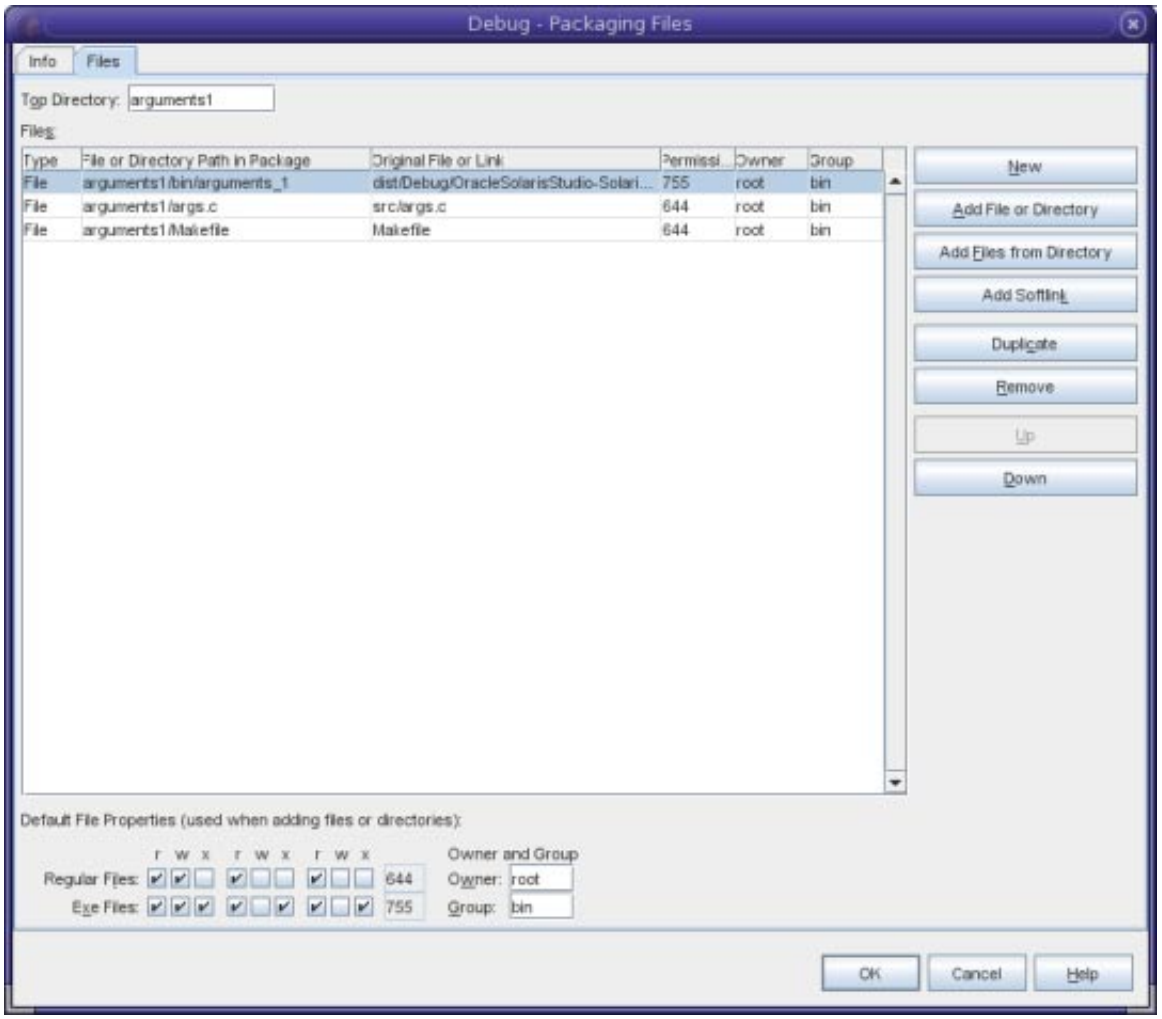
完成したアプリケーションを tar ファイル、zip ファイル、Solaris SVR4 パッケージ、RPM、もしくは Debian パッケージとしてパッケージできます。

1. `Arguments_1` プロジェクトを右クリックして、「プロパティー (Properties)」を選択します。
2. 「プロジェクトのプロパティー (Project Properties)」ダイアログボックスで、「パッケージング (Packaging)」ノードを選択します。
3. ドロップダウンリストから、Solaris SVR4 パッケージタイプを選択します。
4. パッケージ先に別のディレクトリまたはファイル名を使用する場合は、出力パスを変更します。

5. 「ファイルのパッケージング (Packaging Files)」参照ボタンをクリックします。「ファイルのパッケージング (Packaging Files)」ダイアログボックス (SVR4 パッケージの場合) で、必要に応じて「情報 (Info)」タブのパッケージパラメータを変更します。



6. すべてのパッケージタイプに対して、「ファイル (Files)」タブのボタンを使用してファイルをパッケージに追加します。各ファイルについて、「ファイル (Files)」リストの「パッケージ内のファイルまたはディレクトリパス (File or Directory Path in Package)」列に、パッケージ内でのパスを指定できます。「ファイル (Files)」リストが完成したら、「OK」をクリックします。



7. 必要に応じて、チェックボックスをクリックして冗長モードをオフにします。
8. 「OK」をクリックします。
9. パッケージを構築するには、プロジェクトを右クリックして「その他の構築コマンド (More Build Commands)」>「パッケージの構築 (Build Package)」を選択します。

ソースファイルの編集

Oracle Solaris Studio IDE には高度な編集機能およびコード支援機能があり、ソースコードの表示と変更に役立ちます。これらの機能を確認するため、`Quote` プロジェクトを使用します。

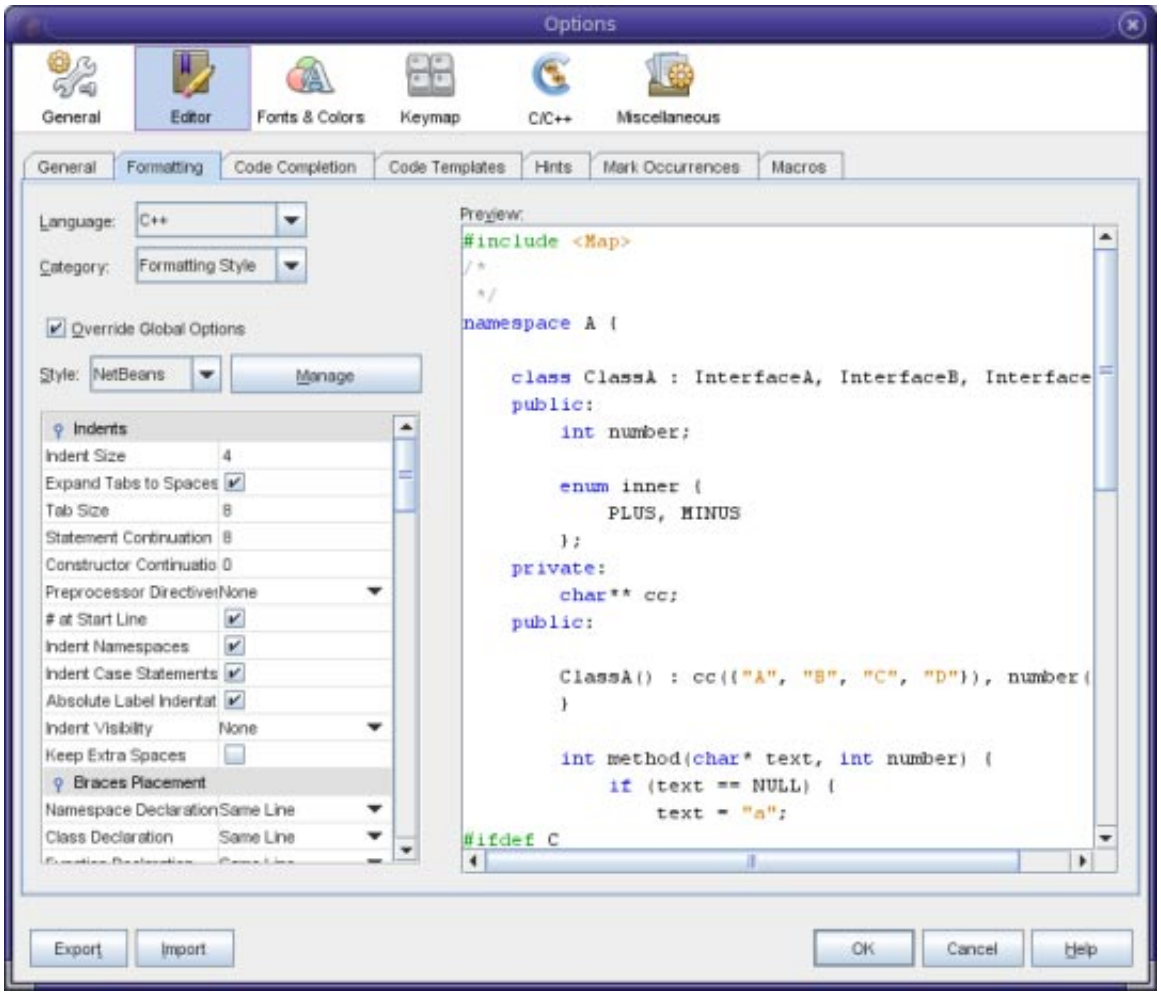
1. 「ファイル (File)」>「新規プロジェクト (New Project)」を選択します。
2. プロジェクトウィザードで、「サンプル (Samples)」カテゴリと「C/C++」サブカテゴリを展開して、`Quote` プロジェクトを選択します。「次へ (Next)」をクリックして、「完了 (Finish)」をクリックします。

書式設定スタイルの設定

「オプション (Options)」ダイアログボックスを使用して、プロジェクトのデフォルトの書式設定スタイルを設定できます。

1. 「ツール (Tools)」>「オプション (Options)」を選択します。
2. ダイアログボックスの上部ペインの「エディタ (Editor)」をクリックします。
3. 「書式設定 (Formatting)」タブをクリックします。

4. 「言語 (Language)」ドロップダウンリストから、書式設定スタイルを設定する言語を選択します。
5. 「スタイル (Style)」ドロップダウンリストから、設定するスタイルを選択します。



6. 必要に応じてスタイルプロパティーを変更します。

C および C++ ファイルでのコードのブロックの折り畳み

一部のタイプのファイルでは、コード折り畳み機能を使用して、コードのブロックを折りたたんでブロックの最初の行のみをソースエディタに表示できます。

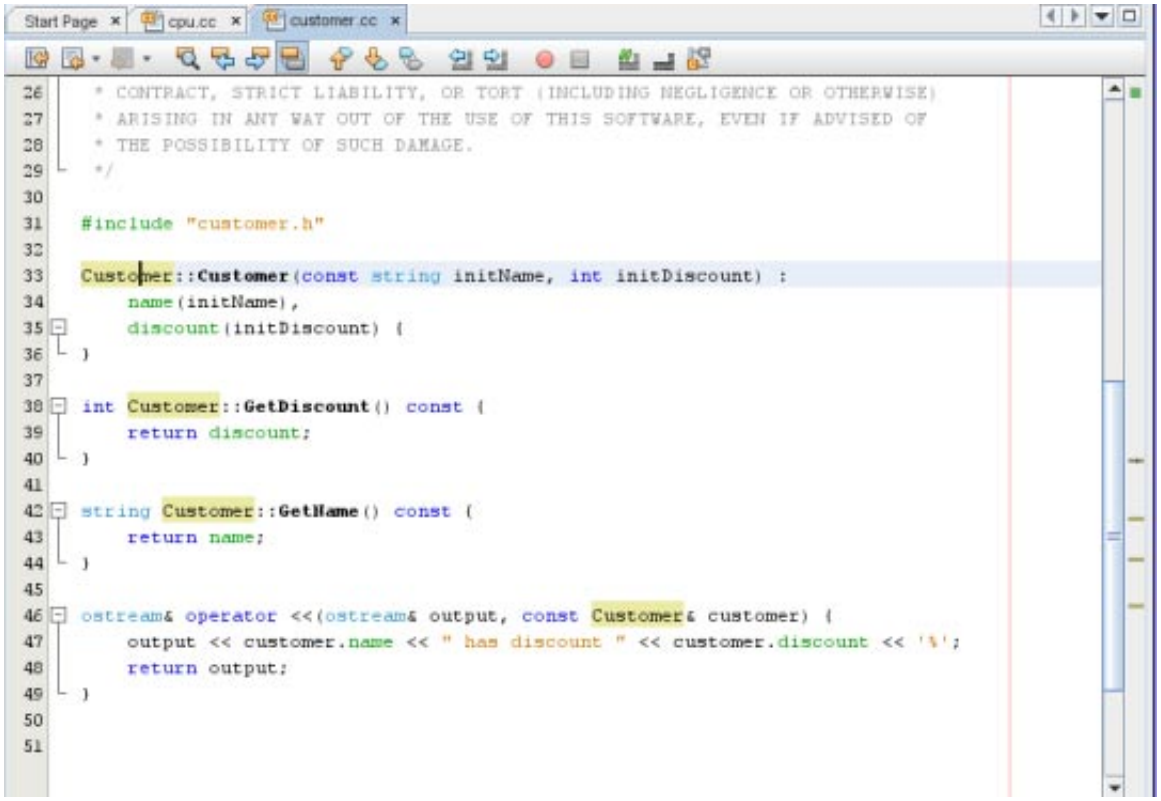
1. `Quote_1` アプリケーションプロジェクトで、「ソースファイル (Source Files)」フォルダを開き、`cpu.cc` ファイルをダブルクリックしてソースエディタで開きます。
2. 左端の折り畳みアイコン (マイナス記号付きの小さなボックス) をクリックして、メソッドの 1 つのコードを折り畳みます。
3. 折り畳んだブロックの右側の {...} 記号にマウスオーバーして、ブロック内のコードを表示します。

意味上の強調表示の使用

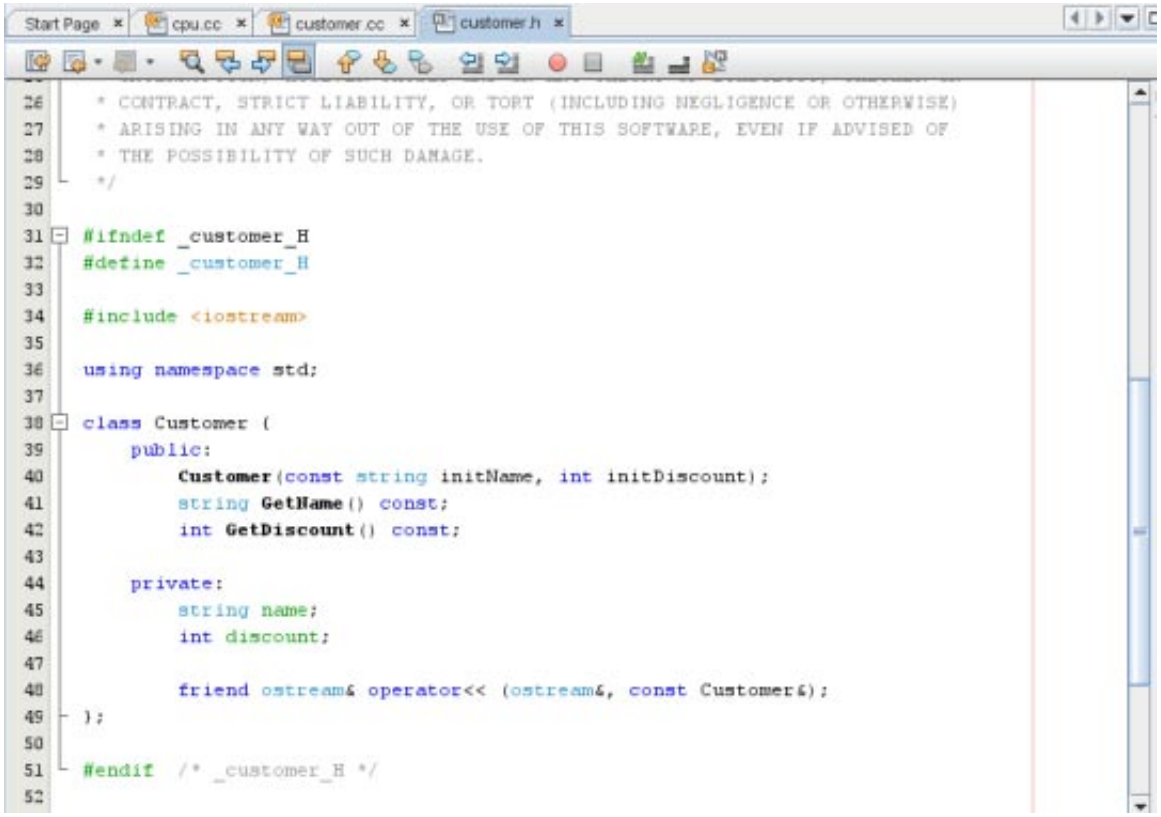
オプションを設定して、クラス、関数、変数、もしくはマクロをクリックしたときに、現在のファイル内のこのクラス、関数、変数、もしくはマクロのすべての出現箇所が強調されるようにできます。

1. 「ツール (Tools)」>「オプション (Options)」を選択します。

2. ダイアログボックスの上部ペインの「C/C++」をクリックします。
3. 「強調表示 (Highlighting)」タブをクリックします。
4. すべてのチェックボックスがチェックされていることを確認します。
5. 「OK」をクリックします。
6. `Quote_1` プロジェクトの `customer.cc` ファイルで、関数名がボールドで強調表示されていることを確認します。
7. `Customer` クラスの出現箇所をクリックします。
8. ファイル内の `Customer` クラスのすべての出現箇所が黄色の背景で強調表示されます。



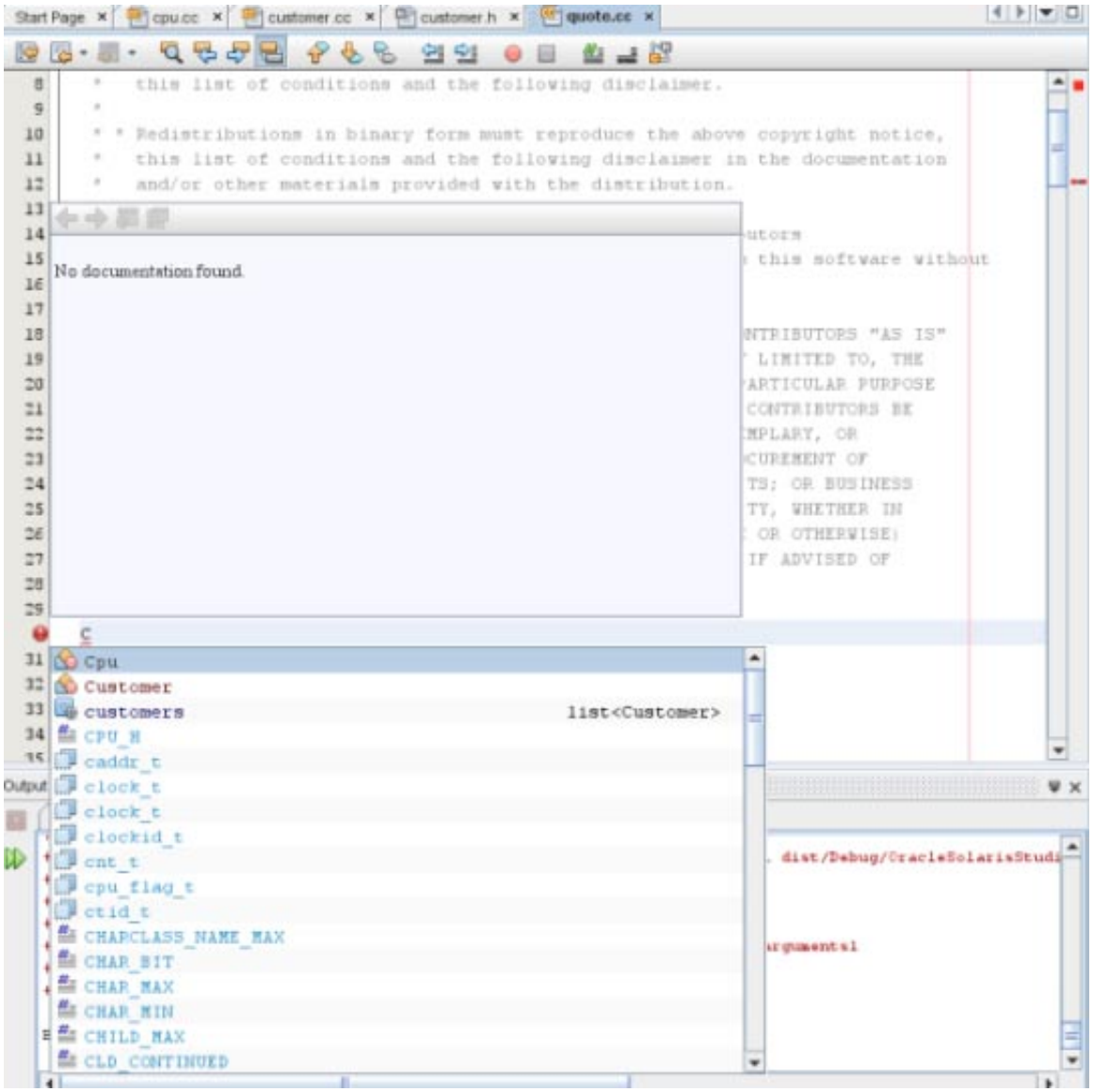
9. `customer.h` ファイルで、クラスフィールドがボールドで強調表示されていることを確認します。



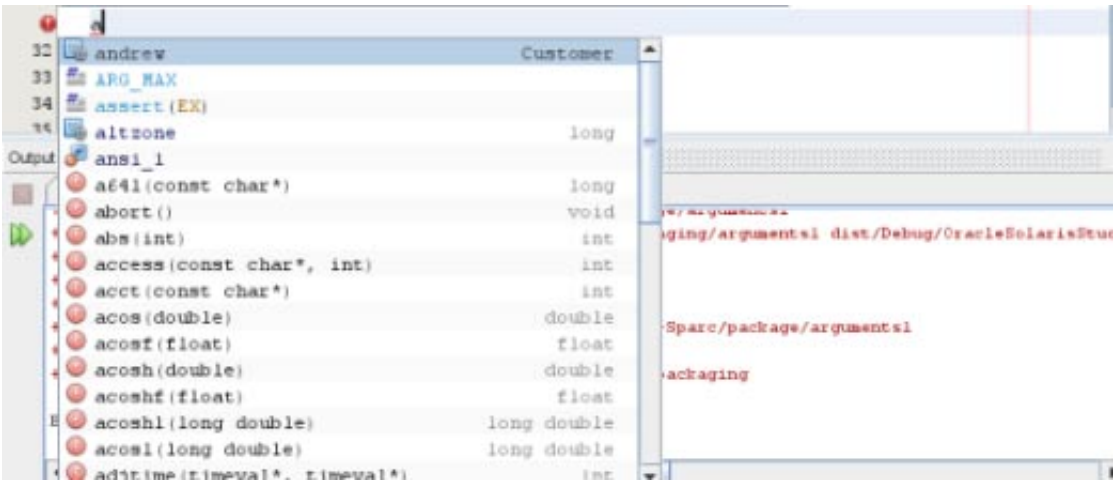
コード補完の使用

IDE にはC および C++ の動的コード補完機能があり、1 文字以上を入力すると該当するクラス、メソッド、変数などのリストが表示され、これを使用して式を補完できます。

1. `Quote_1` プロジェクトの `quote.cc` ファイルを開きます。
2. `quote.cc` ファイルの最初の空行で、大文字の C を」入力して Ctrl-Space を押します。コード補完ボックスに、`Cpu` および `Customre` クラスを含む短いリストが表示されます。ドキュメントウィンドウも開き、プロジェクトソースコードにドキュメントがないため、「ドキュメントがありません (No documentation found)」というメッセージが表示されます。
3. Ctrl-Space をもう一度押して、コード補完リストを展開します。



4. `calloc()` などの標準ライブラリ関数をリストから選択すると、ドキュメントウィンドウにその関数のマニュアルページが表示されます (IDE でマニュアルページにアクセスできる場合)。
5. `Customer` クラスを選択して、Enter を押します。
6. `andrew;` と入力して、`Customer` クラスの新しいインスタンスを完成させます。次の行で、文字 a を入力して Ctrl-Space を押します。コード補完ボックスに、メソッド引数、クラスフィールド、およびグローバル名など、現在のコンテキストでアクセスできる、文字 a で始まる選択対象のリストが表示されます。



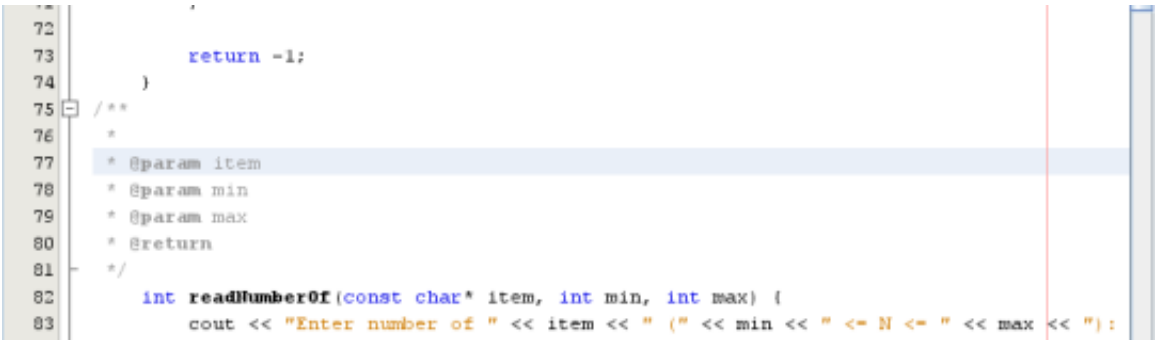
7. andrew オプションをダブルクリックして結果を受け入れ、その後にピリオドを入力します。Customer クラスのパブリックメソッドとフィールドのリストが自動的に指定されます。

8. 追加したコードを削除します。

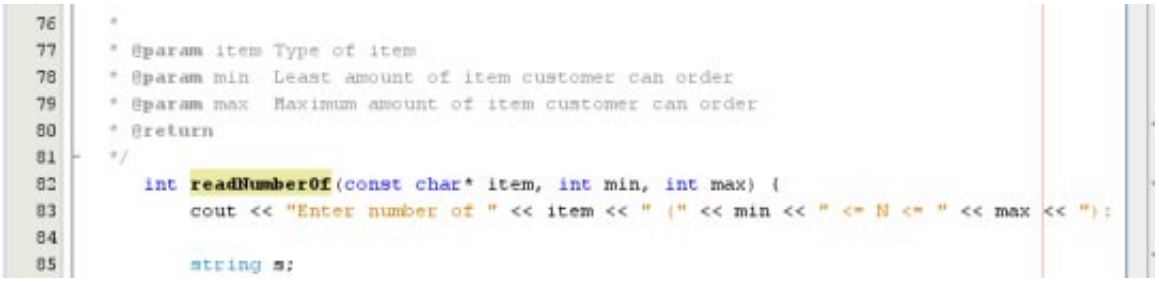
ソースコードドキュメントの追加

コードにコメントを追加して、関数、クラス、およびメソッドのドキュメントを生成できます。IDE は Doxygen 構文を使用するコメントを認識し、ドキュメントを自動的に生成します。また、コメントブロックを自動的に生成して、コメントの下の関数のドキュメントを作成します。

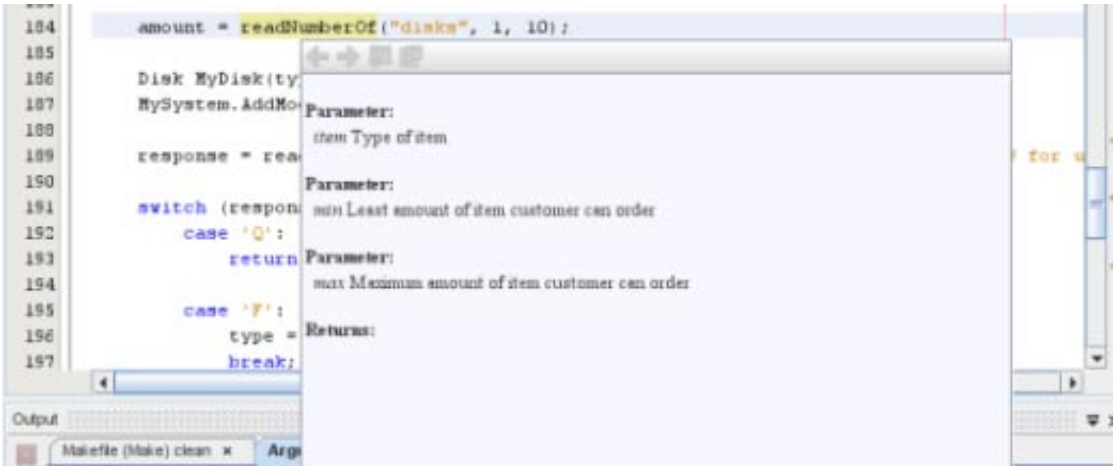
1. quote.cc ファイルで、行 `int readNumberOf(const char* item, int min, int max) {` の上の行にカーソルを置きます。
2. スラッシュ 1 つとアスタリスク 2 つを入力して Enter を押します。エディタから、Doxygen で書式設定されたコメントが `readNumberOf` クラスに挿入されます。



3. `@param` の各行に説明のテキストを追加して、ファイルを保存します。
4. `readNumberOf` クラスをクリックして黄色で強調表示し、右側の出現箇所マークの 1 つをクリックしてクラスが使用されている場所にジャンプします。



5. ジャンプ先の `readNumberOf` クラスをクリックして、Ctrl-Shift-Space を押してパラメータに追加したドキュメントを表示します。



6. ファイル内の任意の場所をクリックしてドキュメントウィンドウを閉じて、`readNumberOF` クラスを再度クリックします。
7. 「ソース (Source)」>「ドキュメントの表示 (Show documentation)」を選択して、クラスのドキュメントウィンドウを再度開きます。

コードテンプレートの使用

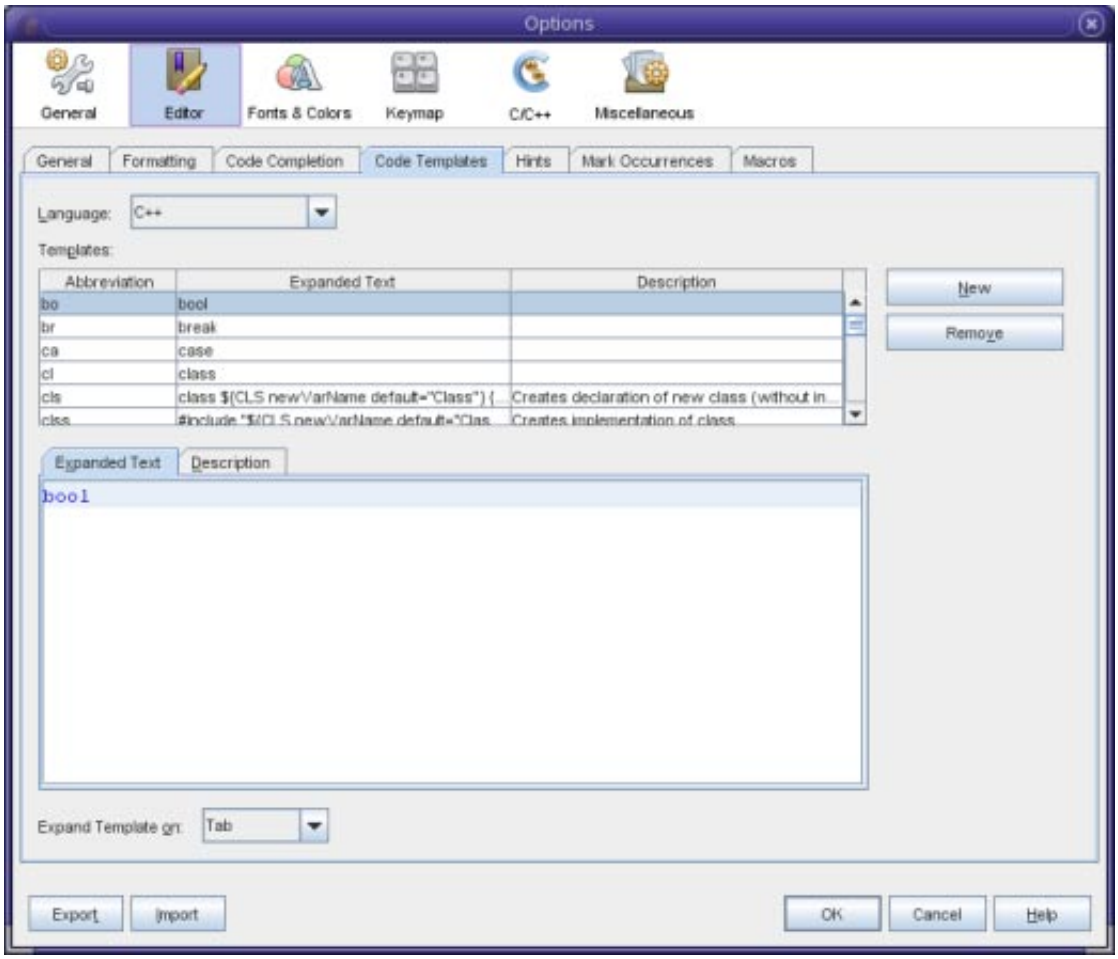
ソースエディタには、C、C++、および Fortran コードの共通スニペット用の、一連のカスタマイズ可能なコードテンプレートがあります。略語を入力して Tab キーを押すと、コードスニペット全体を生成できます。たとえば、`Quote_1` プロジェクトの `quote.cc` ファイルで、次のようにします。

- `uns` と入力した後に Tab キーを押して、`uns` を `unsigned` に展開します。
- `iff` と入力した後に Tab キーを押して、`iff` を `if (exp) {}` に展開します。
- `ifs` と入力した後に Tab キーを押して、`ifs` を `if (exp) {} else {}` に展開します。
- `fori` と入力した後に Tab キーを押して、`fori` expands を `for (int i=0; i< size; i++) { Object size = array[i]; }` に展開します。

使用できるコードテンプレートすべてを表示するには、テンプレートを変更してユーザー固有のコードテンプレートを作成するか、または別のキーを選択してテンプレートを展開します。

1. 「ツール (Tools)」>「オプション (Options)」を選択します。
2. 「オプション (Options)」ダイアログボックスで、「C/C++」を選択して、「コードテンプレート (Code Templates)」タブをクリックします。

3. 「言語 (Language)」ドロップダウンリストから言語を選択します。



ペア補完の使用

C および C++ ソースファイルを編集すると、ソースエディタは角括弧、丸括弧、および引用符など、ペアで使用される文字の「スマート」照合を実行します。これらの文字の片方を入力すると、ソースエディタはもう片方の文字を自動的に挿入します。

1. `Quote_1` プロジェクトで、`module.cc` ファイルの行 116 の `{` の後にカーソルを置き、Return を押して新しい行を開きます。
2. `enum state {` と入力して Return を押します。閉じる大括弧とセミコロンが自動的に追加され、カーソルが括弧の間に置かれます。
3. `invalid=0, success=1` と入力して、列挙法を補完します。
4. 列挙の閉じる `};` の後の行で、`if` と入力します。閉じ括弧が自動的に追加され、カーソルが括弧の間に置かれます。
5. `v==null` と入力します。右側の括弧の後に、`i` と改行を入力します。閉じ括弧が自動的に追加されます。
6. 追加したコードを削除します。

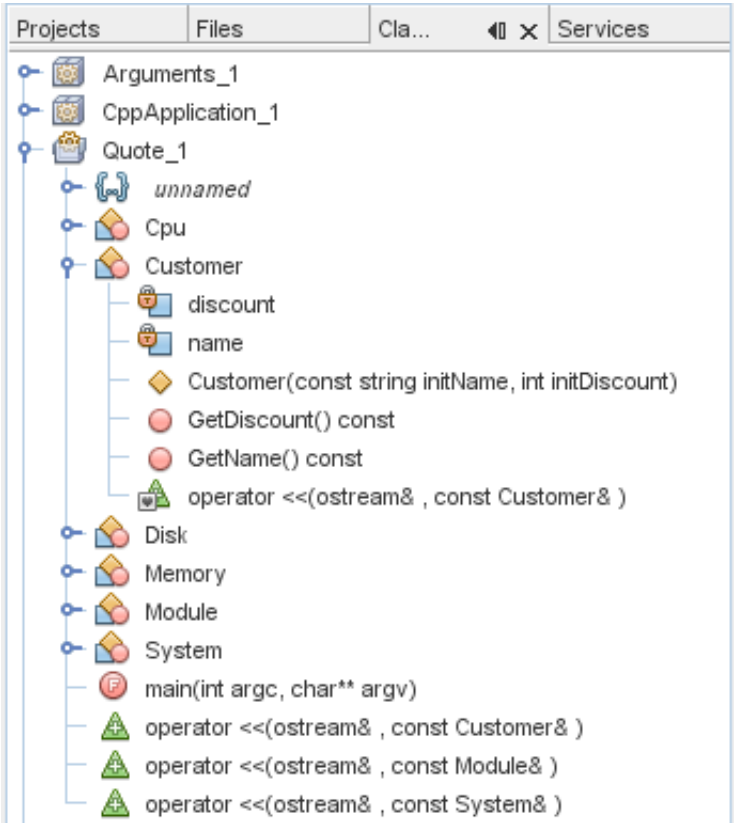
ソースファイルのナビゲーション

IDE には、ソースコードを表示する高度なナビゲーション機能があります。これらの機能を確認するため、`Quote_1` プロジェクトを使用します。

「クラス (Classes)」ウィンドウの使用

「クラス (Classes)」ウィンドウでは、プロジェクトのすべてのクラスと、各クラスのメンバーとフィールドを表示できます。

1. 「クラス (Classes)」タブをクリックして、「クラス (Classes)」ウィンドウを表示します。
2. `Quote_1` ノードを展開します。プロジェクト内のすべてのクラスが一覧表示されます。
3. `Customer` クラスを展開します。

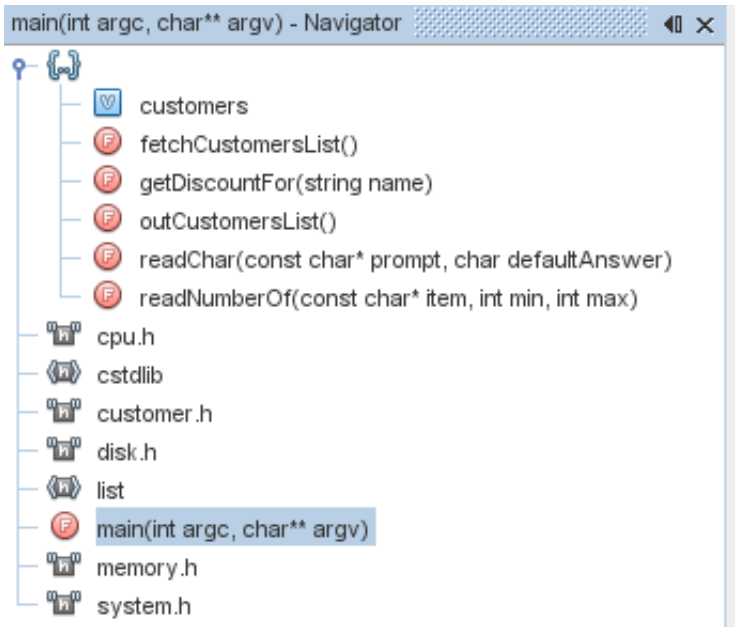


4. `name` 変数をダブルクリックして `customer.h` ヘッダーファイルを開きます。

「ナビゲータ (Navigator)」ウィンドウの使用

「ナビゲータ (Navigator)」ウィンドウには、現在選択されているファイルの簡易ビューが表示され、ファイルの異なる部分へのアクセスが簡単になります。「ナビゲータ (Navigator)」ウィンドウが開いていない場合、「ウィンドウ (Window)」>「ナビゲート (Navigating)」>「ナビゲータ (Navigator)」の順に選択して開きます。

1. エディタウィンドウ内で、`quote.cc` ファイルの任意の場所をクリックします。
2. ファイルの簡易ビューが「ナビゲータ (Navigator)」ウィンドウに表示されます。ウィンドウの上部のノードをクリックして、ビューを展開します。

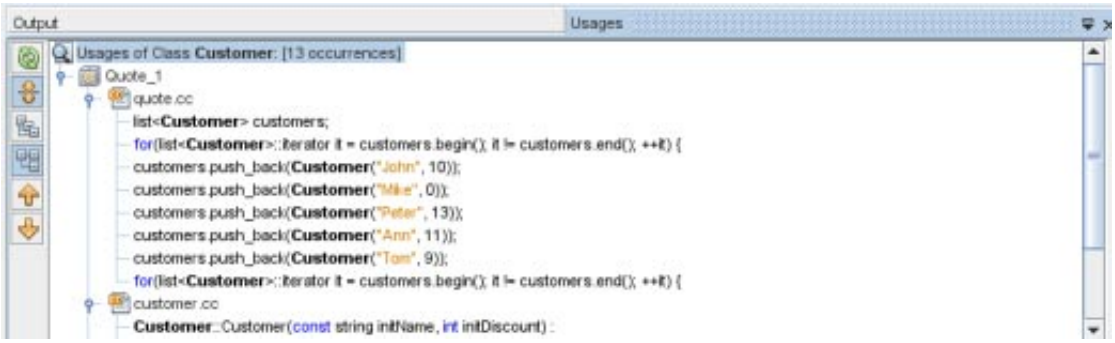


3. ファイルの要素にナビゲートするには、「ナビゲータ (Navigator)」ウィンドウで要素をダブルクリックして、エディタウィンドウのカーソルをその要素に移動させます。
4. 「ナビゲータ (Navigator)」ウィンドウ内を右クリックして、ウィンドウ内の要素のソート、項目のグループ化、フィルタのオプションを表示させます。
5. 「ナビゲータ (Navigator)」ウィンドウにあるアイコンを確認するには、「ヘルプ (Help)」>「ヘルプの目次 (Help Contents)」を選択して、IDE オンラインヘルプを開きます。ヘルプブラウザで、「検索 (Search)」タブをクリックして、「検索 (Find)」フィールドにナビゲータアイコンを入力します。

クラス、メソッド、およびフィールドの使用状況の検出

「使用状況 (Usages)」ウィンドウを使用して、プロジェクトのソースコード内で使用されている、あらゆる場所にあるクラス (構造)、関数、変数、マクロ、もしくはファイルを表示できます。

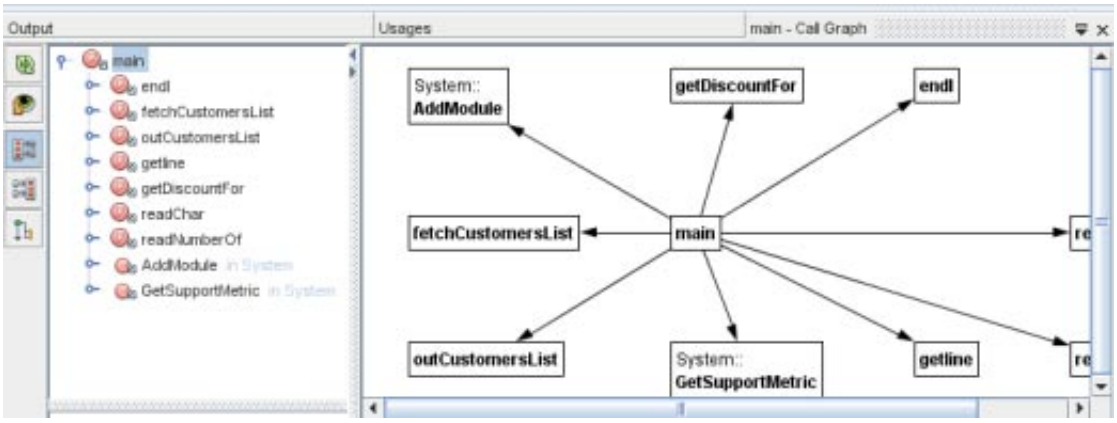
1. `customer.cc` ファイルで、42 行目の `Customer` クラスを右クリックして「使用状況を検索 (Find Usages)」を選択します。
2. 「使用状況を検索 (Find Usages)」ダイアログボックスで、「検索 (Find)」をクリックします。
3. 「使用状況 (Usages)」ウィンドウが開き、プロジェクトのソースファイルでの `Customer` クラスのすべての使用状況が表示されます。



コールグラフの使用

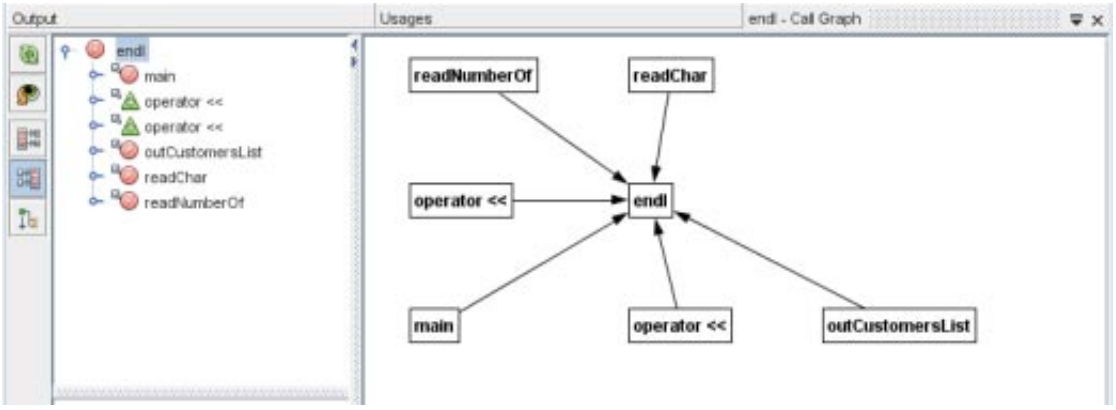
「コールグラフ (Call Graph)」ウィンドウには、クラス内の関数の予呼び出し関係の 2 つのビューが表示されます。ツリービューに、選択した関数から呼び出された関数、もしくはこの関数を呼び出す関数が表示されます。グラフィカル表示には、呼び出し元および呼び出し先の関数の間に矢印を使用して、呼び出し関係が示されます。

1. `quote.cc` ファイルで、メイン関数を右クリックして「コールグラフの表示 (Show Call Graph)」を選択します。
2. 「コールグラフ (Call Graph)」ウィンドウが開き、ツリービューと、`main` 関数から呼び出されるすべての関数のグラフ表示が表示されます。

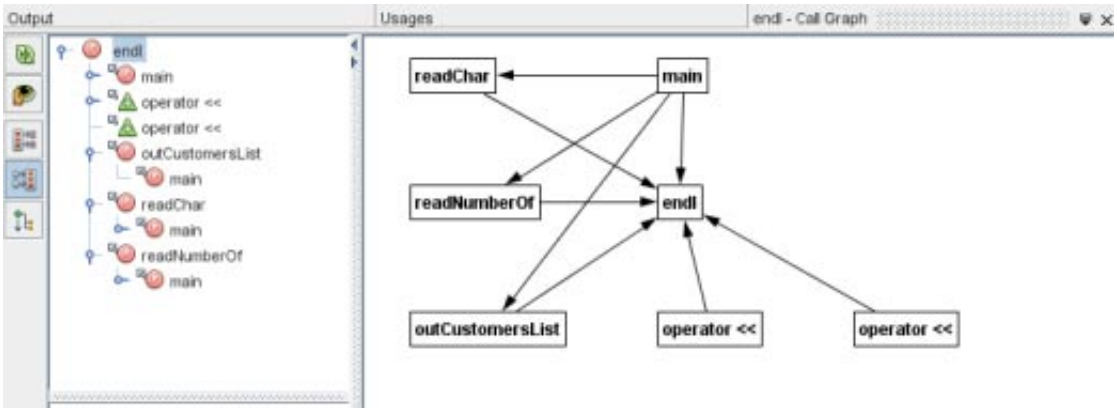


スクリーンショットに示す関数の一部が表示されない場合は、「コールグラフ (Call Graph)」ウィンドウの左側の 3 番目のボタンをクリックして、この関数から呼び出される関数を表示します。

3. `end1` ノードを展開して、この関数によって呼び出される関数を表示します。グラフが更新されて、`end1` によって呼び出される関数が追加されます。
4. `end1` ノードを選択してウィンドウの左側の 2 番目のボタンをクリックして、`endl` 関数にフォーカスし、4 番目のボタンをクリックして `end1` 関数を呼び出すすべての関数を表示します。



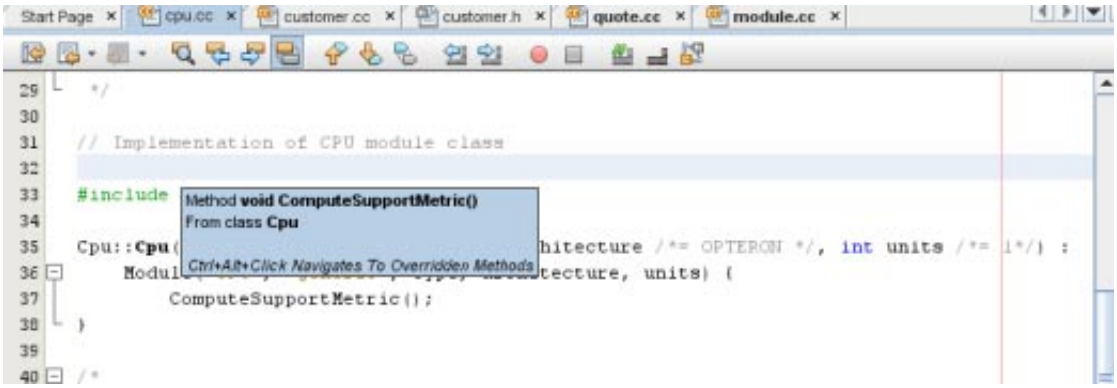
5. ツリー内のいくつかのノードを展開して、その他の関数を表示します。



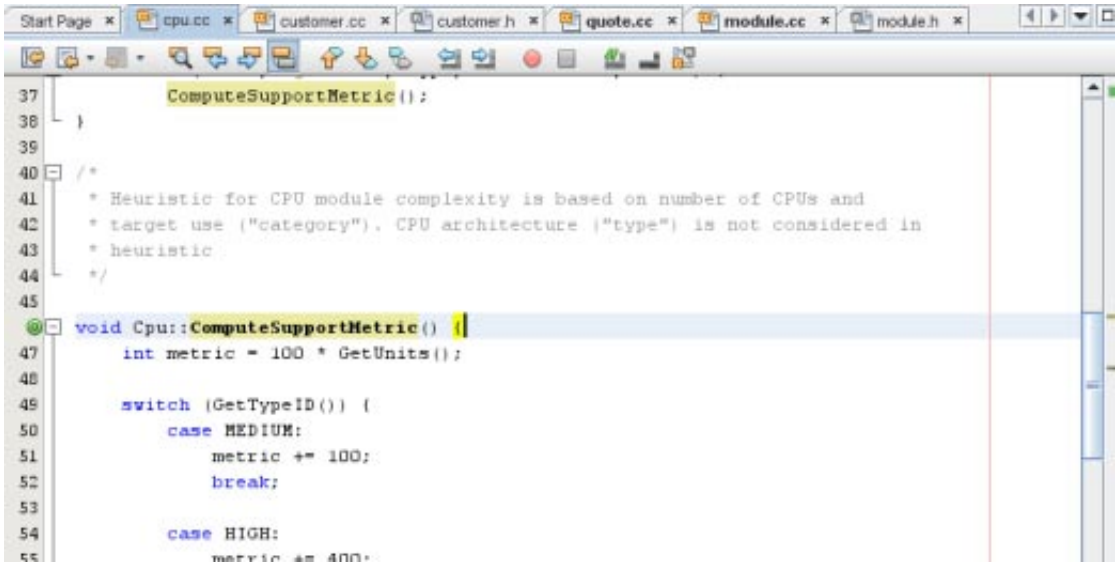
ハイパーリンクの使用

ハイパーリンクナビゲーションによって、クラス、メソッド、変数、もしくは定数の呼び出しから宣言へジャンプしたり、宣言から定義にジャンプしたりすることができます。ハイパーリンクでは、オーバーライドされるメソッドからオーバーライドするメソッドへ、またはこの逆方向にジャンプすることもできます。

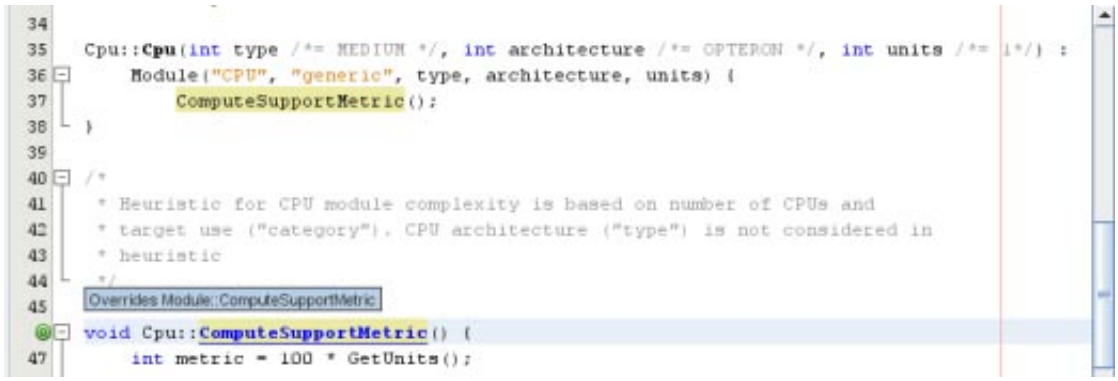
1. `Quote_1` プロジェクトの `cpu.cc` ファイルで、Ctrl を押しながら 37 行目にマウスオーバーします。`ComputeSupportMetric` 関数が強調表示され、関数についての情報を示す注釈が表示されます。



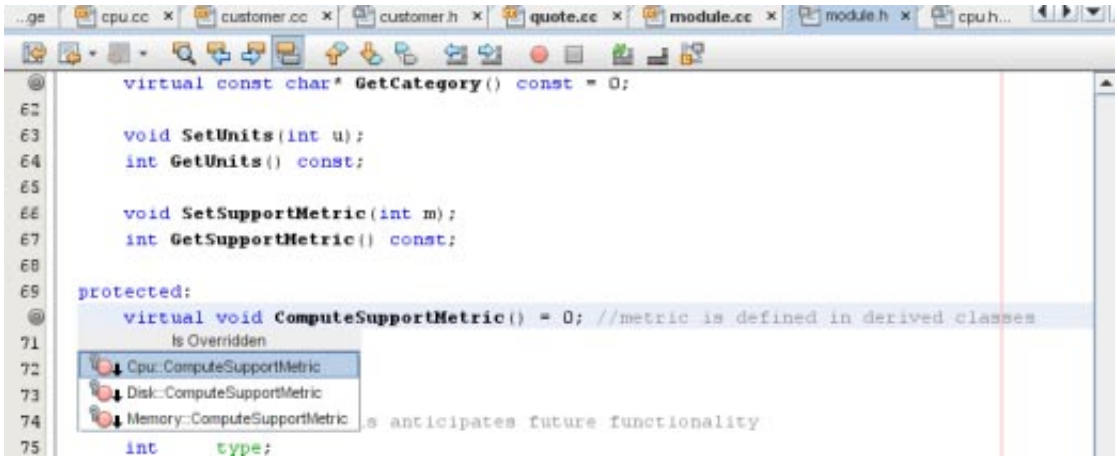
2. ハイパーリンクをクリックすると、エディタで関数の定義にジャンプします。



3. Ctrl を押しながら定義にマウスオーバーし、ハイパーリンクをクリックします。エディタで、cpu.h ヘッダーファイルの関数の定義にジャンプします。
4. エディタツールバーの左向き矢印をクリックすると、エディタは cpu.cc 内の定義に戻ります。
5. マウスカーソルを左端の緑の円の上に置くと、このメソッドが別のメソッドをオーバーライドすることを示す注釈が表示されます。



6. 緑の円をクリックしてオーバーライドされたメソッドに移動すると、エディタは module.h ヘッダーファイルにジャンプします。マージンにグレーの円が表示され、メソッドがオーバーライドされていることを示します。
7. グレーの円をクリックすると、エディタにはこのメソッドがオーバーライドするメソッドのリストが表示されます。

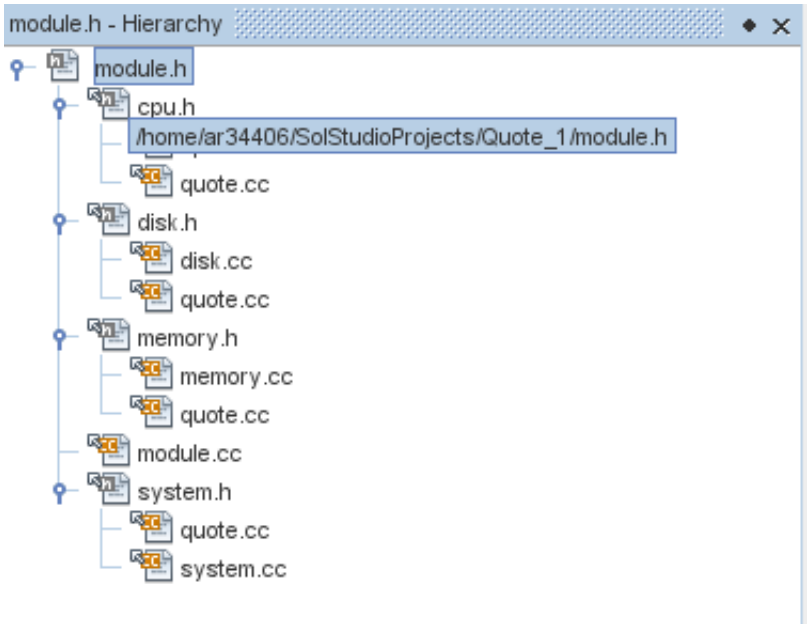


8. 「Cpu::ComputerSupportMetric」項目をクリックすると、エディタは cpu.h ヘッダーファイルのメソッドの宣言に戻ります。

インクルード階層の使用

「インクルードの階層 (Include Hierarchy)」ウィンドウでは、直接的または間接的にソースファイルにインクルードされたすべてのヘッダーファイルとソースファイル、または直接的または間接的にヘッダーファイルにインクルードされたすべてのソースファイルおよびヘッダーファイルを検査できます。

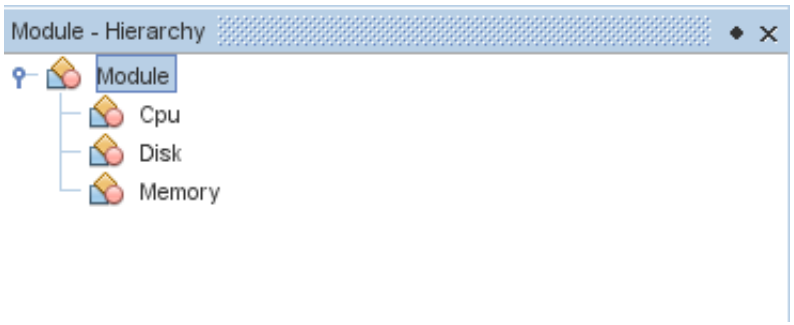
1. Quote_1 プロジェクトで、ソースファイルに module.cc ファイルを開きます。
2. ファイルの #include "module.h" 行を右クリックして、「ナビゲート (Navigate)」>「インクルードの階層を表示 (View Includes Hierarchy)」を選択します。
3. デフォルトで、「階層 (Hierarchy)」ウィンドウにはヘッダーファイルに直接インクルードされるファイルのプレーンリストが表示されます。ウィンドウ下部の右端のボタンをクリックして、表示をツリービューに変更します。右から 2 番目のボタンをクリックして、インクルードまたはインクルードされるすべてのファイルに表示を変更します。ツリービューのノードを展開して、ヘッダーファイルをインクルードするすべてのソースファイルを表示します。



タイプの階層の使用

「タイプの階層 (Type Hierarchy)」ウィンドウでは、クラスのすべてのサブタイプまたはスーパータイプを検査できます。

1. `Quote_1` プロジェクトで、`module.h` ファイルを開きます。
2. `Module` クラスの宣言を右クリックして、「ナビゲート (Navigate)」>「タイプの階層を表示 (View Type Hierarchy)」を選択します。
3. 「階層 (Hierarchy)」ウィンドウに、`Module` クラスのすべてのサブタイプが表示されます。

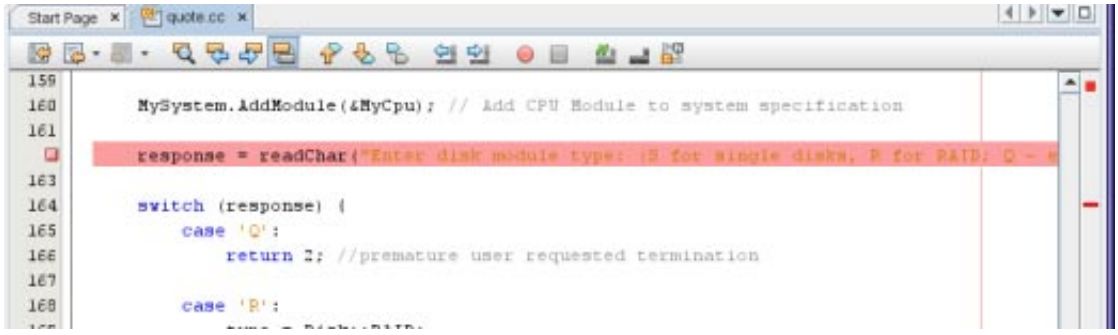


ブレークポイントの作成

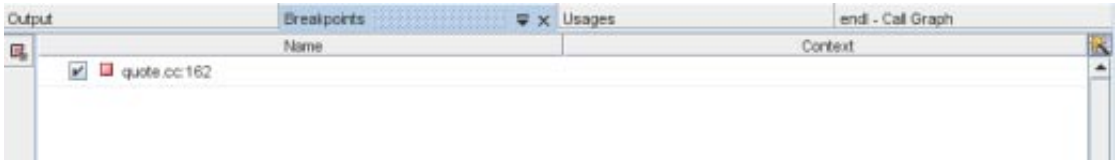
コード内でいつでもブレークポイントを作成し、操作できます。

行ブレークポイントの作成と削除

1. `Quote_1` プロジェクトで、`quote.cc` ファイルを開きます。
2. エディタウィンドウの 173 行目 (`response = readChar("Enter disk module type: (S for single disks, R for RAID; Q - exit)", 'S');`) の横の左マージンをクリックして行ブレークポイントを設定します。行が赤で強調表示され、このブレークポイントが設定されたことを示します。

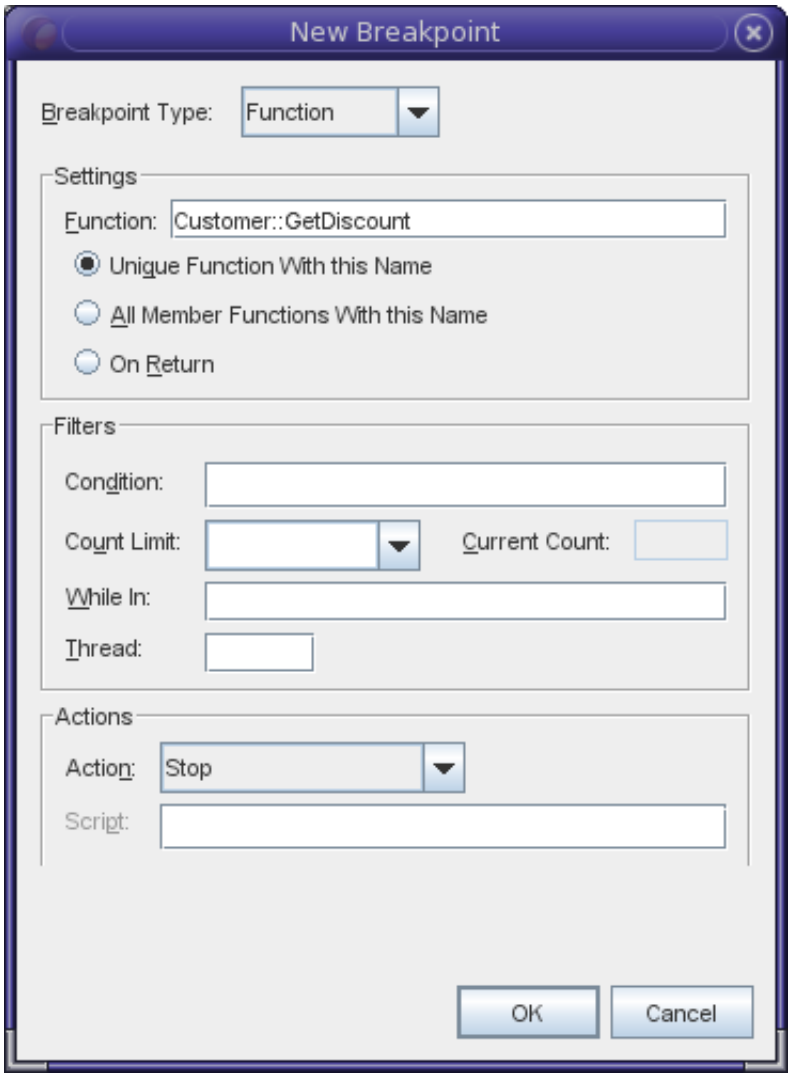


3. 左マージンのアイコンをクリックして、ブレークポイントを削除できます。
4. 「ウィンドウ (Window)」>「デバッグ (Debugging)」>「ブレークポイント (Breakpoints)」を選択して、「ブレークポイント (Breakpoints)」ウィンドウを開きます。行ブレークポイントがウィンドウに一覧表示されます。

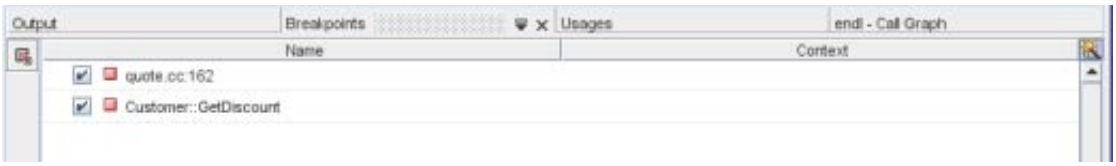


関数ブレークポイントの作成

1. 「デバッグ (Debug)」>「新規ブレークポイント (New Breakpoint)」(Ctrl+Shift+f8) を選択して、「新規ブレークポイント (New Breakpoint)」ダイアログボックスを開きます。
2. 「ブレークポイントの種類 (Breakpoint Type)」ドロップダウンリストで、タイプを「関数 (Function)」に設定します。
3. 関数名 `Customer::GetDiscount i`を「関数 (Function)」テキストフィールドに入力します。「OK」をクリックします。



4. 関数ブレークポイントが設定され、「ブレークポイント (Breakpoints)」ウィンドウのリストに追加されます。



プロジェクトのデバッグ

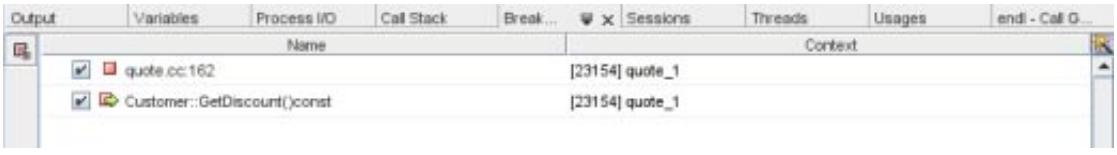
デバッグセッションを開始すると、IDE は dbx デバッガを起動し、デバッガ内部でアプリケーションを実行します。IDE はデバッガウィンドウを自動的に開き、デバッガ出力を「出力 (Output)」ウィンドウに出力します。

デバッグセッションを開始する

1. プロジェクトノードを右クリックして「デバッグ (Debug)」を選択して、`Quote_1` プロジェクトのデバッグセッションを開始します。デバッガが起動してアプリケーションが実行され、「デバッグ (Debugging)」ウィンドウに「変数 (Variables)」、「呼び出しスタック (Call Stack)」、「スレッド (Threads)」、「プロセス I/O (Process I/O)」、および「Dbx コンソール (Dbx Console)」ウィンドウが開きます。
2. 「ウィンドウ (Window)」>「デバッグ (Debugging)」>「セッション (Sessions)」を選択して、「セッション (Sessions)」ウィンドウを開きます。このウィンドウにデバッグセッションが表示されます。

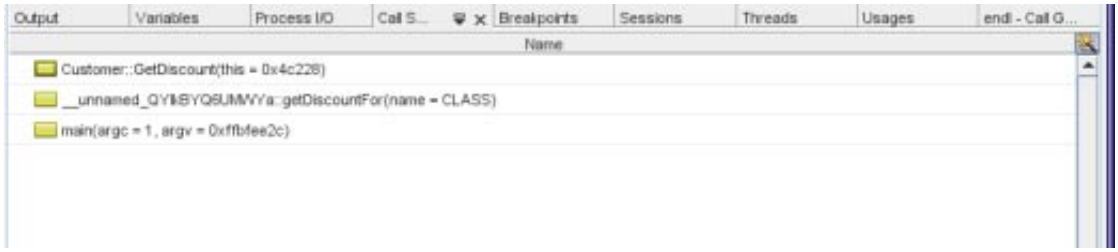
アプリケーションの状態の検査

1. `Quote_1` アプリケーションから、「プロセス I/O (Process I/O)」ウィンドウに入力するよう通知されます。
2. 「顧客名を入力してください `(Enter customer name:)`」というメッセージの後に、顧客名を入力します。
3. 以前設定した関数ブレークポイントで、アプリケーションが停止します。「ブレークポイント (Breakpoints)」ウィンドウに、以前設定した 2 つのブレークポイントが表示されます。関数ブレークポイントのブレークポイントアイコンの上に、緑のプログラムカウンタ矢印が表示されます。

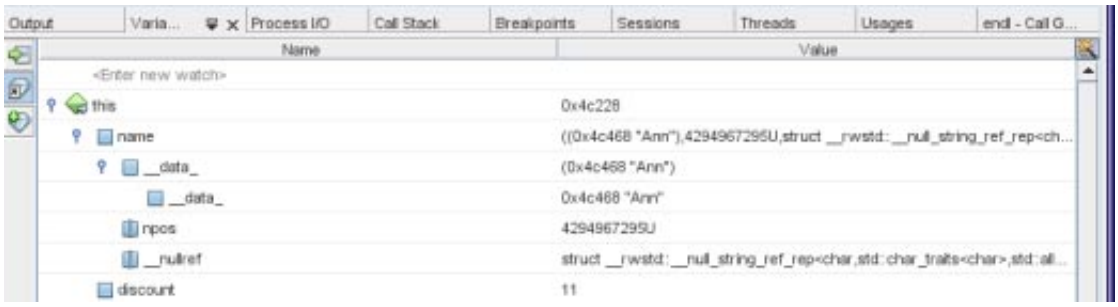


4. `customer.cc` ファイルで、 `GetDiscount` 関数の最初の行にあるブレークポイントアイコンの上に、緑のプログラムカウンタ矢印が表示されます。

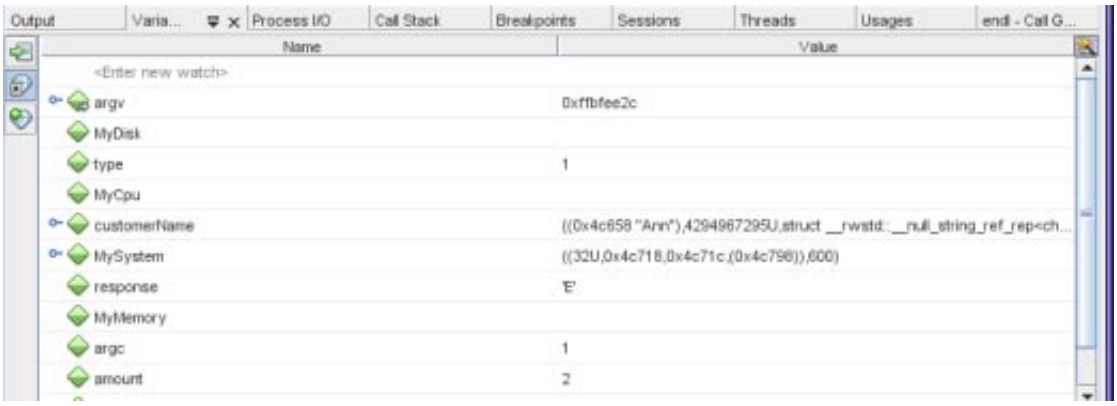
5. 「呼び出しスタック (Call Stack)」ウィンドウをクリックします。呼び出しスタックには 3 つのフレームが表示されます。



6. 「変数 (Variables)」ウィンドウをクリックし、1 つの変数が表示されていることに注意します。ノードをクリックして構造を展開します。



7. 「続行 (Continue)」ボタンをクリックします。`GetDiscount` 関数が実行され、「プロセス I/O (Process I/O)」ウィンドウに顧客割引が出力されます。次に、入力を求められます。
8. プロンプトに従って入力します。プログラムが次のブレークポイント (以前設定した行ブレークポイント) で停止します。「変数 (Variables)」ウィンドウをクリックして、ローカル変数の長いリストを確認します。

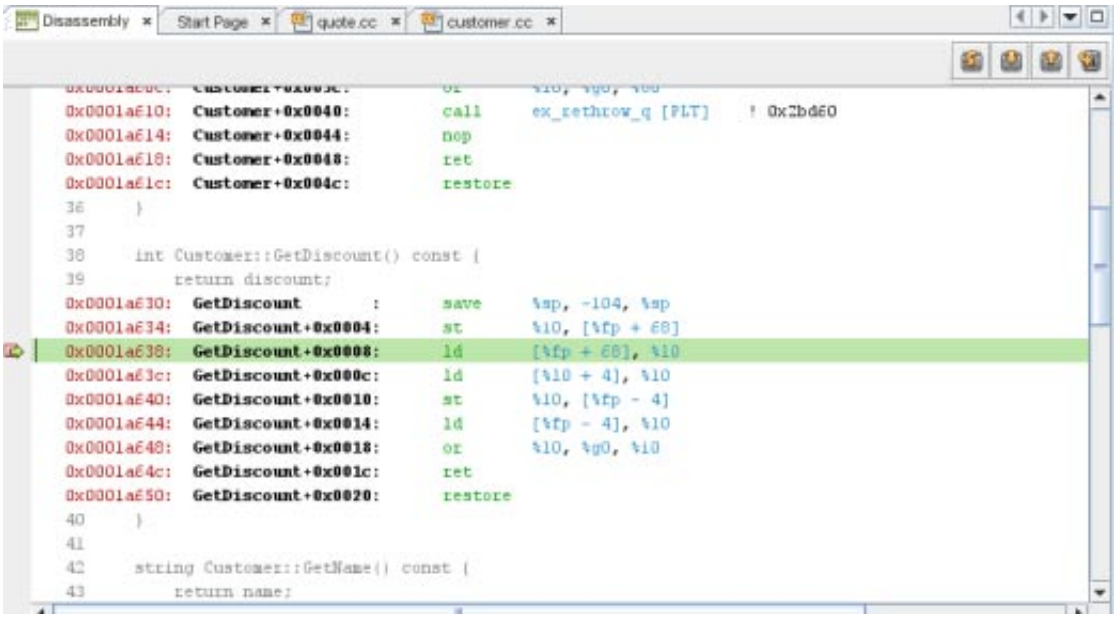


9. 「呼び出しスタック (Call Stack)」ウィンドウをクリックして、スタックにフレームが 1 つしかないことを確認します。
10. 「続行 (Continue)」ボタンをクリックして、プログラムが完了するまで、「プロセス I/O (Process I/O)」ウィンドウのプロンプトに従って入力を続行します。プログラムに最後の入力を行うと、デバッグセッションは終了します。プログラムが完了する前にデバッグセッションを終了するには、「セッション (Sessions)」ウィンドウでセッションを右クリックして「完了 (Finish)」を選択します。

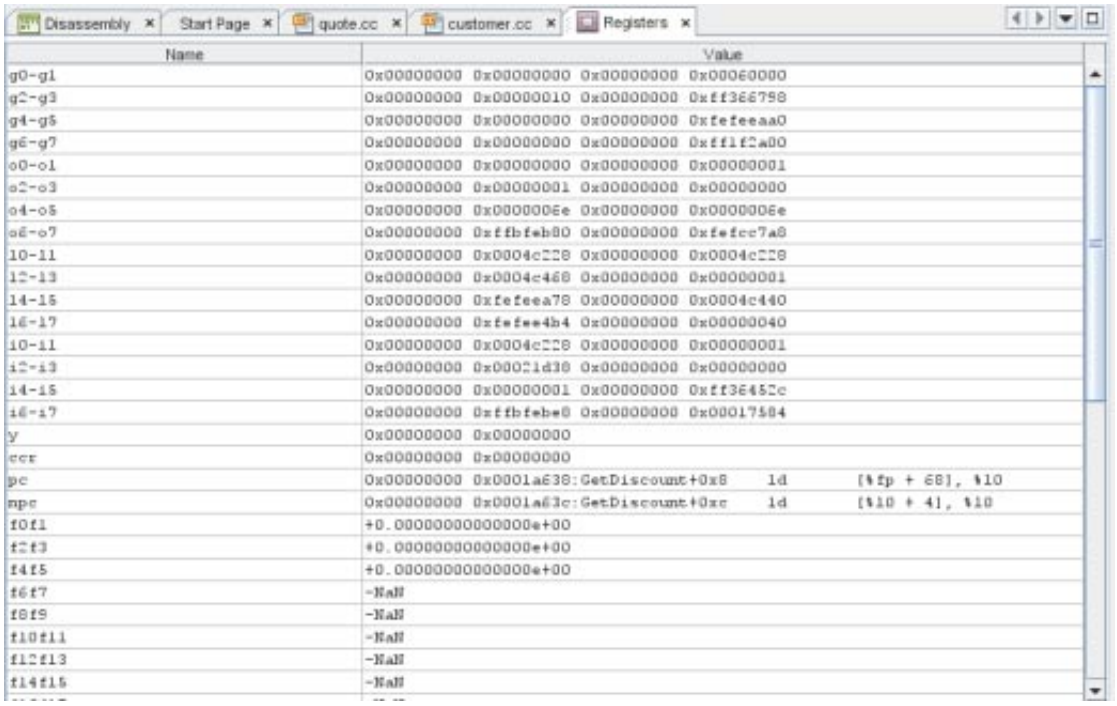
機械命令レベルでのデバッグ

デバッガには、プロジェクトを機械命令レベルでデバッグできるウィンドウがあります。

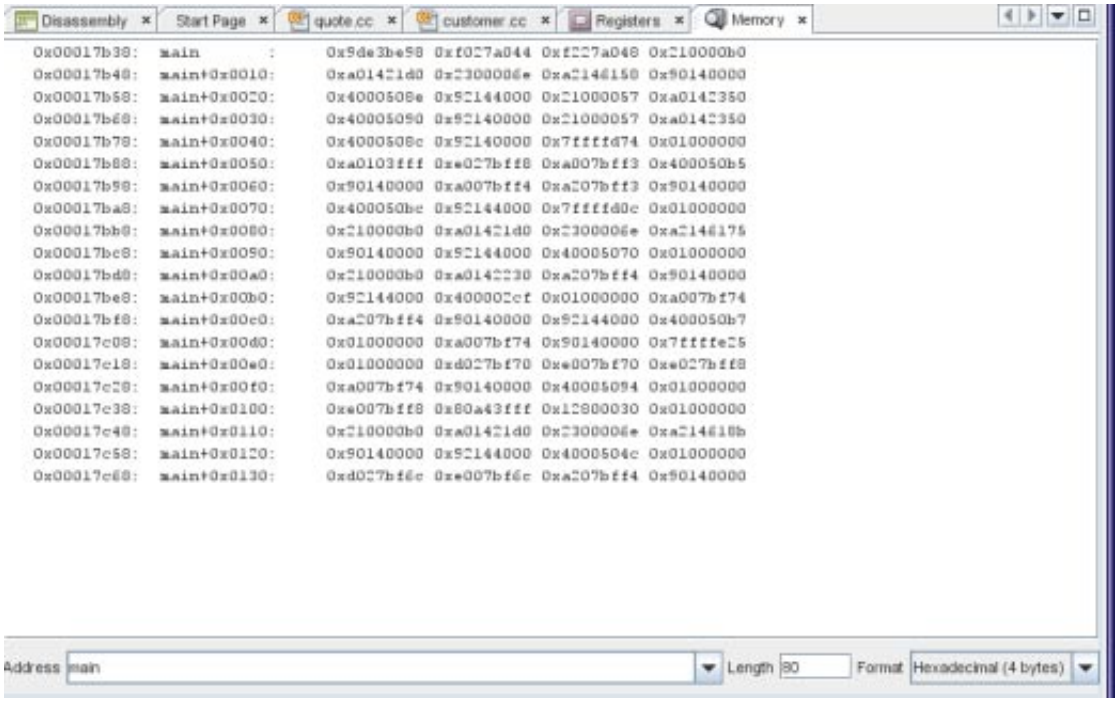
1. `Quote_1` プロジェクトを右クリックして、「デバッグ (Debug)」を選択します。
2. 「プロセス I/O (Process I/O)」ウィンドウで、プロンプトに従って顧客名を入力します。
3. プログラムが `GetDiscount` 関数のブレークポイントで一時停止したら、エディタウィンドウと同様に、「ウィンドウ (Window)」>「デバッグ (Debugging)」>「逆アセンブリ (Disassembly)」ウィンドウを選択します。プログラムが一時停止した命令のブレークポイントアイコンの上に、緑のプログラムカウンタ矢印が表示されます。



4. 「ウィンドウ (Window)」>「デバッグ (Debugging)」>「レジスタ (Registers)」を選択して「レジスタ (Registers)」ウィンドウを開きます。ここにはレジスタの内容が表示されます。



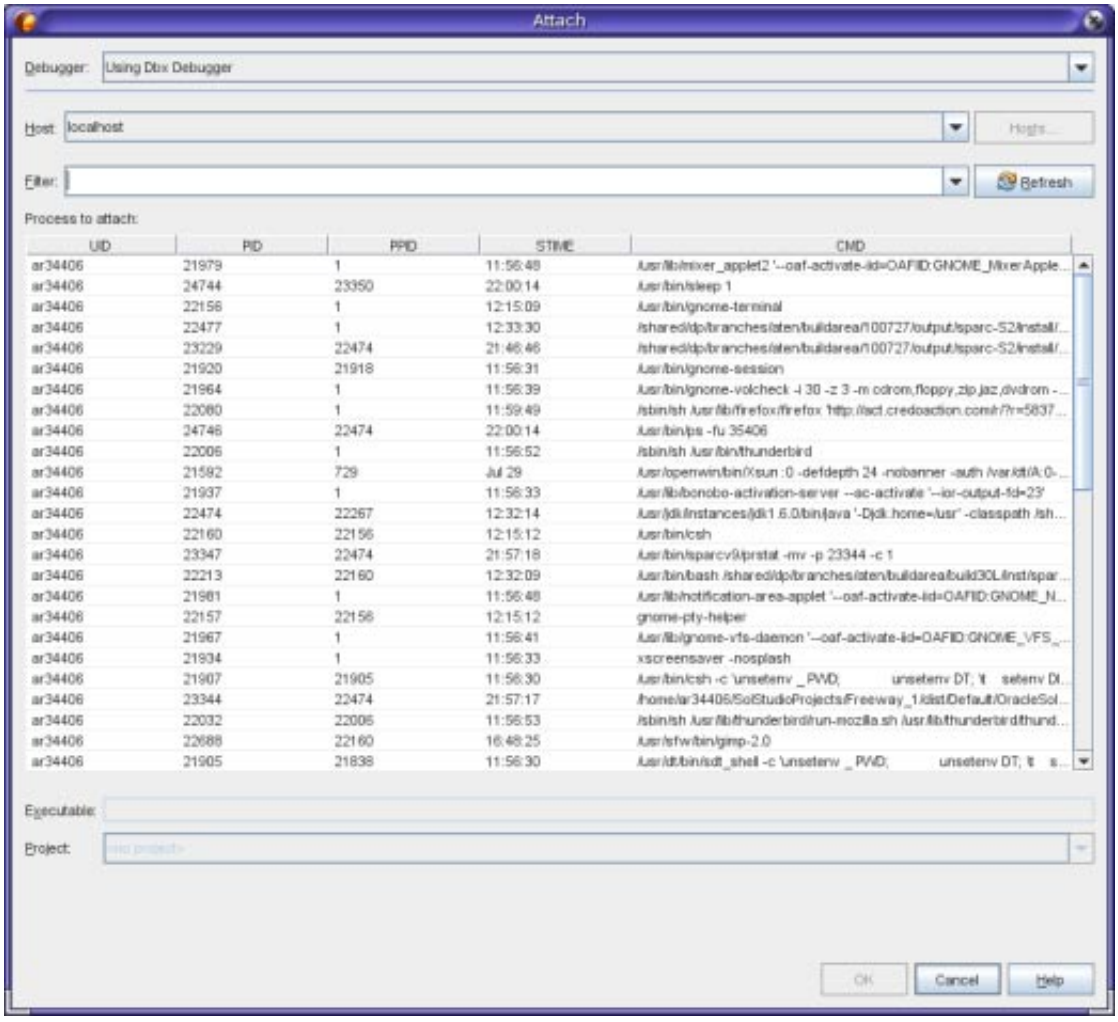
5. 「ウィンドウ (Window)」>「デバッグ (Debugging)」>「メモリー (Memory)」を選択して「メモリー (Memory)」ウィンドウを開きます。ここには、現在プロジェクトで使用されているメモリーの内容が表示されます。ウィンドウの下部で、参照するメモリーアドレスの指定、メモリー参照の長さの変更、メモリー情報の形式の変更を行えます。



実行中のプログラムを接続してデバッグ

すでに実行されているプログラムをデバッグするため、デバッガを該当するプロセスに接続できます。

1. 「ファイル (File)」>「新規プロジェクト (New Project)」を選択します。
2. 新規プロジェクトウィザードで、「サンプル (Samples)」ノードを展開して、「C/C++」カテゴリを選択します。
3. Freeway Simulator プロジェクトを選択します。「次へ (Next)」をクリックして、「完了 (Finish)」をクリックします。
4. 作成した `Freeway_1` プロジェクトを右クリックして、「実行する (Run)」を選択します。プロジェクトが構築され、Freeway アプリケーションが開始されます。
5. 「デバッグ (Debug)」>「デバッガを接続 (Attach Debugger)」を選択します。

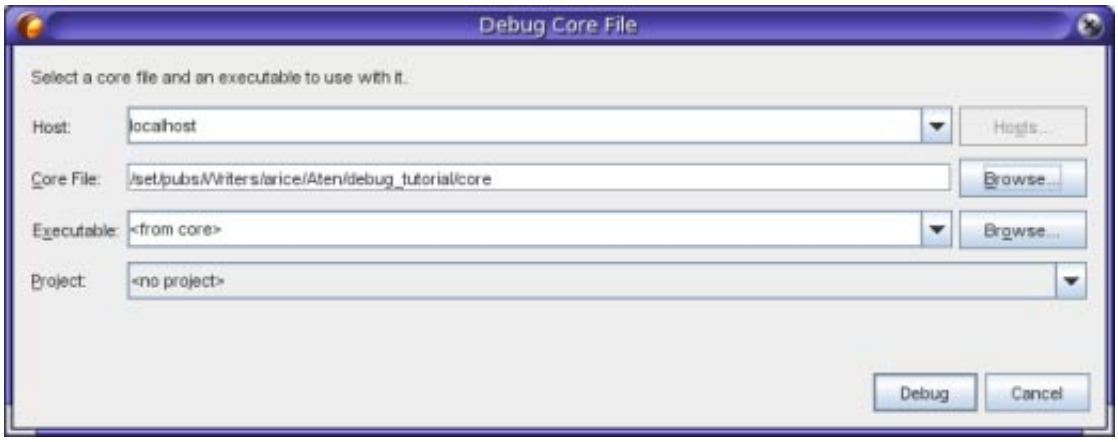


6. 「接続 (Attach)」ダイアログボックスで、「フィルタ (Filter)」フィールドに `Freeway` と入力して、プロセスのリストをフィルタします。
7. フィルタしたリストから `Freeway` プロセスを選択します。
8. 「OK」をクリックします。
9. デバッグセッションが開始し、`Freeway` プロセスの実行がデバッガが接続されたポイントで一時停止します。

既存のコアファイルのデバッグ

プログラムがクラッシュする場合、コアファイル (クラッシュしたときのプログラムのメモリーイメージ) をデバッグできます。コアファイルをデバッガにロードするには、次の手順に従います。

1. 「デバッグ (Debug)」>「コアファイルのデバッグ (Debug core file)」を選択します。
2. 「コアファイル (Core File)」フィールドにコアファイルのフルパスを入力するか、または「コアファイルを選択 (Select Core File)」ダイアログボックスで「参照 (Browse)」をクリックしてコアファイルにナビゲートします。



3. 実行可能ファイルで指定したコアファイルにデバッガを接続できない場合、「実行可能ファイルを選択 (Select Executable)」ダイアログボックスが開き、実行可能ファイルを指定できます。この場合、「実行可能ファイル (Executable)」テキストボックスに実行可能ファイルのパス名を入力するか、または「参照 (Browse)」ボタンをクリックして「実行可能ファイル (Executable)」ダイアログボックスを使用して実行可能ファイルを選択します。
4. デフォルトで、「プロジェクト (Project)」テキストフィールドには、<プロジェクトなし> (`no project`)、または実行可能ファイルの名前と完全に一致する既存のプロジェクトの名前が表示されます。実行可能ファイルに新規プロジェクトを作成するには、<新規プロジェクトの作成> (`create new project`) を選択します。
5. 「デバッグ (Debug)」をクリックします。

デバッグの詳細なチュートリアルは、Oracle Solaris Studio 12.2 dbxtool チュートリアルを参照してください。

821-2501

Oracle Corporation 500 Oracle Parkway, Redwood City, CA 94065 U.S.A.