Source Insight 3.5
ユーザー マニュアル

(参考訳)

発行 :

2008 年 7 月 7 日

エクセルソフト株式会社

〒 108-0014 東京都港区芝 5-1-9 ブゼンヤビル 4F

TRANSLATED BY ***XLSOFT CORPORATION***

第 1 章

はじめに

Source Insight をお買い上げ頂きありがとうございます。Source Insight は、10 年以上に渡り改良を重ね進化し続けている製品です。Source Insight は生産性を向上させ、プログラミングを楽しくします。

この章では、Source Insight の機能についてご紹介します。

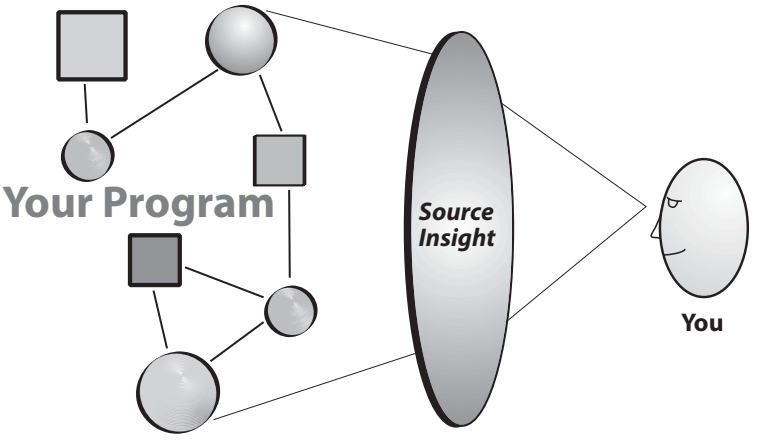
製品概要

問題点

開発者の周辺には、多数のソース ファイルが存在します。第三者が記述したプログラム ソースや関数を扱わなければならないこともあります。そのコードがどのように動作するかを理解し、関連するファイルについても調査や確認が必要です。自身が作成したコードではないコードも、また、かなり以前に記述したコードについても理解することが求められます。

世界で有数の開発者のひとりであっても、プログラム中の無数に存在するプログラム部分をすべて見つけることは不可能ですし、また、コードを理解して、それを記憶しなければ、作業の生産性を向上させることはできません。

Source Insight は、このような問題に対応する最適なソリューションです。



ソリューション

Source Insight はプログラムの理解と修正作業の効率を向上させるようデザインされています。 Source Insight の目標は、ソース コードを明確にして簡単な方法で情報を示し、プログラマが大規模で複雑なプロジェクトを修正できるようにすることで、プログラミング チームの生産性を高めることです。

プログラムのソース コードを自由形式の情報データベースとして考えてください。そこにはクラス、メンバ、機能情報だけではなく、多くの重要なコメントも含まれています。

ソース コードには履歴もあります。実際、多くの大規模なプログラムは何年にも渡って多くのプログラマが関わり、作業時間は長くなりま

す。以前のソース コードがわかりにくくても、無視するわけにはいきません。

Source Insight はプロジェクトのソース コードを取り巻く情報サーバーとして機能します。Source Insight があれば、プログラム内のシンボル情報やテキスト情報に簡単にアクセスできます。

プロジェクトに新規に参加した場合でも、Source Insight を使用することで、直ちに生産性の高い作業を開始できます。

製品の特長

Source Insight は、C/C++、C#、および Java プログラムを解析できる、プロジェクト指向のプログラム コード エディタ / ブラウザです。ソース コードを解析し、作業中に独自のシンボリック情報のデータベースをダイナミックに管理して、役立つコンテキスト情報を自動的に提供します。

Source Insight は、優れたプログラム エディタであるだけではなく、参照ツリー、クラス階層ダイアグラム、コール ツリーを表示することもできます。また、ソース コードと任意のプログラミング エディタのソース情報の迅速なナビゲーションも特長です。

Source Insight は、ソース コードとソース情報に対して、素早く、革新的なアクセスが可能です。多くの他のエディタ製品と異なり、Source Insight は、編集中にソース コードを解釈して、役立つ情報を素早く提供します。

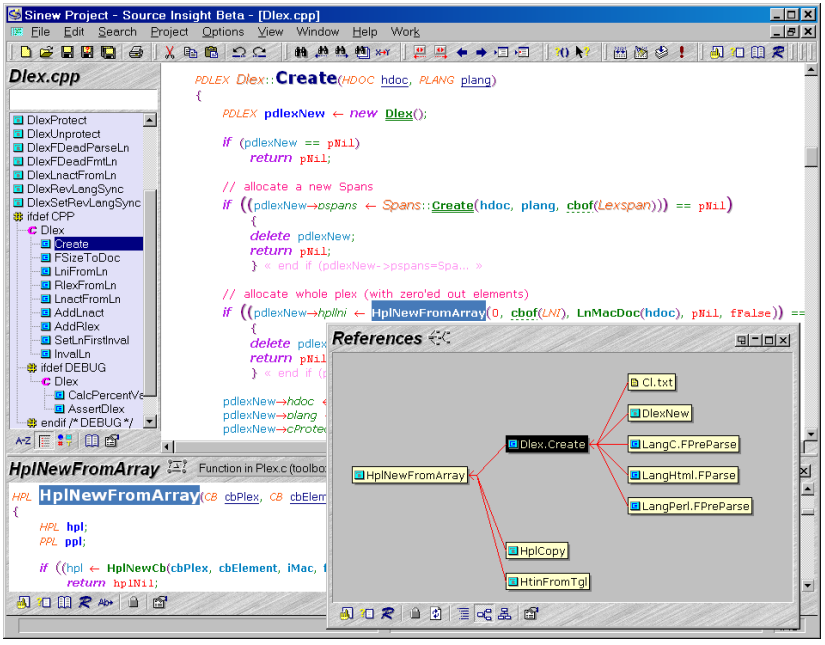


図 1.1 Source Insight は、革新的な構文フォーマットとダイナミックな関係表示を使用して、ソース コードをわかりやすく表示します。

常に最新の情報

プログラムの開発中はコードが頻繁に変更されるため、コンパイルされないコード中のシンボルを含めて細かい精度でコードを表示できることは重要です。Source Insight は、プロジェクトのソース ファイルで

定義されている関数、メソッド、グローバル変数、構造体、クラス、およびその他の種類のシンボルについて独自の高性能なシンボル データベースを作成して利用します。Source Insight は、このシンボル データベースを利用してコードを瞬時に表示します。プロジェクトをコンパイルしたり、コンパイラに依存するファイルを提供する必要はありません。コードの編集中でも、ファイルに影響を与えることなく、ファイルに関する情報を素早く更新できます。シンボル機能は、各 Source Insight プロジェクトに自動的に組み込まれます。余計なタグ ファイルをビルドする必要はありません。

状況に応じたダイナミックな型解決

Source Insight は、クラス継承を含む変数の型を、編集中にダイナミックにデコードします。状況に応じて、クラスの正確な情報が提供されます。

ファイルごとにシンボル ウィンドウを表示

シンボル ウィンドウは各ソース ウィンドウの横に表示され、ダイナミックに更新されます。各フィールド内を移動してファイルの概要を素早く表示できます。47 ページの「シンボル ウィンドウ」も参照してください。

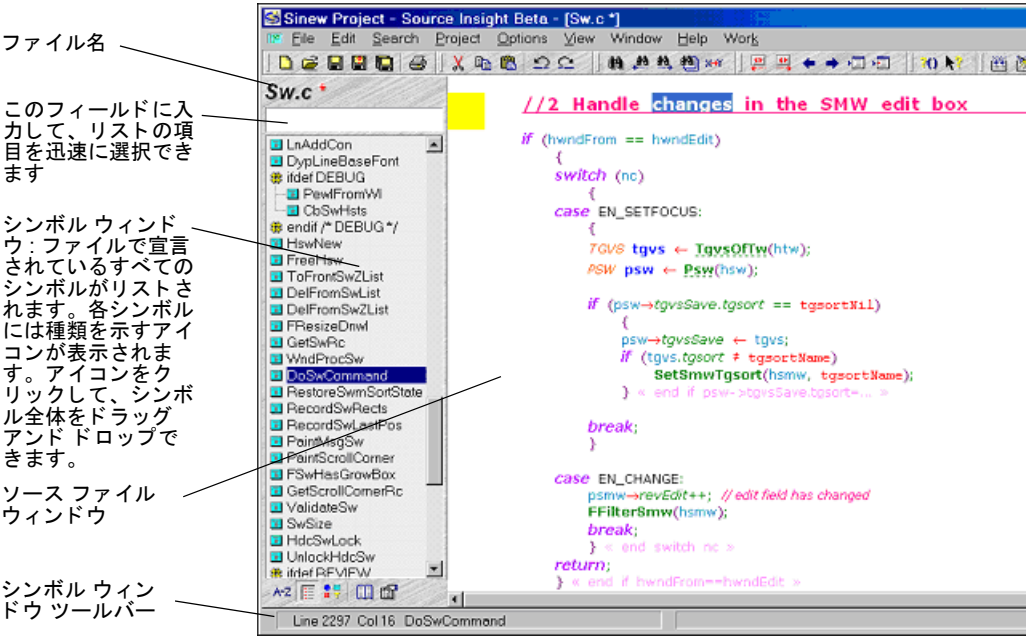


図 1.2 シンボル ウィンドウは各ソース ファイル ウィンドウの左側に表示されます。

シンボル ウィンドウのシンボルをクリックして、シンボルに素早くジャンプできます。また、シンボルをコードにドラッグ アンド ドロップして再配置することもできます。シンボル ウィンドウは、名前、行番号、または種類でソートできます。シンボル名の最初の数文字を入力するだけで、該当するシンボルに素早く移動します。シンボルを素早く識別できるように、#ifdef-#endif 入れ子レベルとシンボルの種類を示すアイコンも表示します。

コンテキスト ウィンドウでの宣言の自動表示

コンテキスト ウィンドウは、Source Insight 2.0 で追加された革新的な機能の 1 つです。カーソルの位置の識別子、または入力している内容に応じてシンボル定義を自動的に表示します。55 ページの「コンテキスト ウィンドウ」も参照してください。

コンテキスト ウィンドウはバック グラウンドで動作し、作業履歴を追跡します。識別子をクリックすると、シンボルの定義が自動的に表示されます。識別子が変数の場合、宣言をデコードして基本構造またはクラス タイプを表示します。

コンテキスト ウィンドウは、プロジェクト ウィンドウで選択されているファイル、リレーション ウィンドウのシンボル、クリップ ウィンドウで選択されているクリップも自動的に表示します。

コール グラフとコール ツリー ダイアグラム

リレーション ウィンドウは、シンボル間の関係を示す Source Insight の革新的な機能です。バックグラウンドで動作し、ユーザーが選択したシンボルを追跡します。このウィンドウで、クラス階層、コール ツリー、参照ツリー、その他を参照できます。59 ページの「リレーション ウィンドウ」も参照してください。

リレーション ウィンドウの利点は、何も特別なことを行う必要がないことです。作業中はバックグラウンドで動作していますが、必要なときに呼び出すことができます。

リレーション ウィンドウは、グラフィカルまたはアウトライン形式で表示できます。複数のリレーション ウィンドウを開いて、異なる種類の情報を表示することもできます。

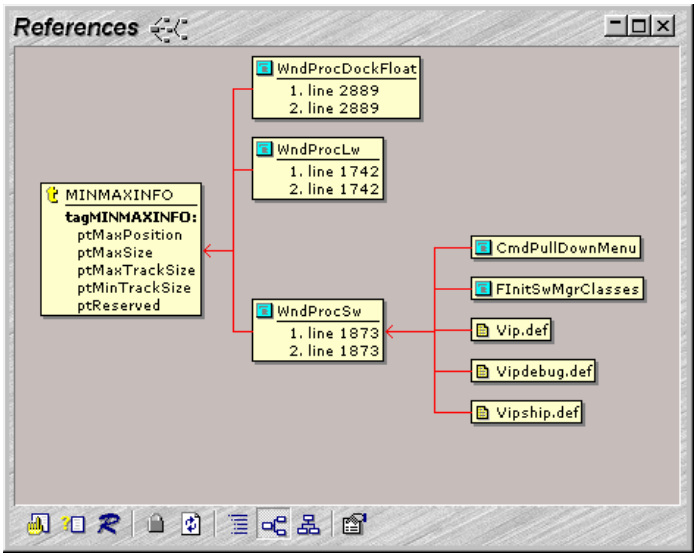


図 1.3 リレーション ウィンドウは、型への参照、および関数からの間接参照を表示します。

構文フォーマット

構文フォーマットは、詳細かつ簡単な方法で情報を提供する、Source Insight の革新的な機能の 1 つです。ユーザー定義スタイルのリッチ テキスト書式を含む、大幅に改良された表示機能を提供します。Source Insight は、プロジェクトの単語およびシンボリック情報に基づいてス

タイルを自動的に適用します。101 ページの「構文フォーマットとスタイル」も参照してください。

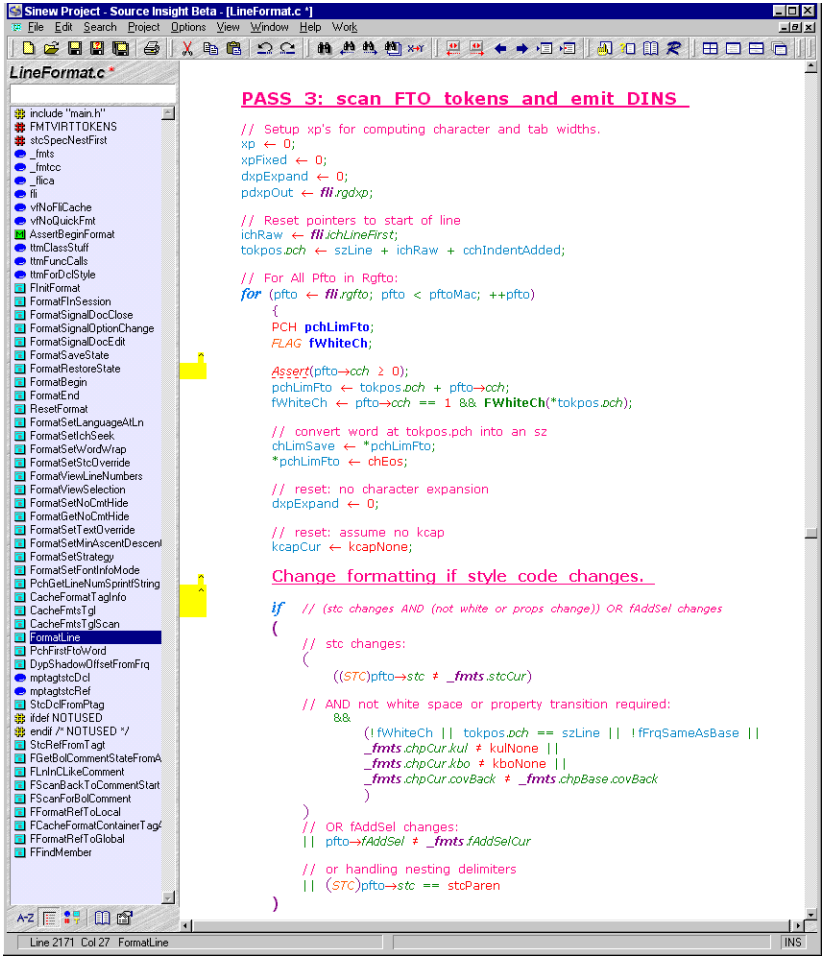


図 1.4 構文フォーマットは、コードを一目でわかるようにします。上記の例では、Source Insight の一意なコメント見出しスタイル、入れ子の括弧、およびさまざまなシンボル参照スタイルが表示されています。左余白の黄色いマークは編集された行を示しています。

構文フォーマットは、コードを読むときに役立つ情報を追加します。たとえば、ローカル変数への参照とグローバル変数への参照を区別して表示できます。また、関数への参照と C 関数形式のマクロへの参照を区別して表示することもできます。構文フォーマットを使用するこ

とで、識別子が何を参照しているか、またスペルミスがないか一目でわかるようになります。

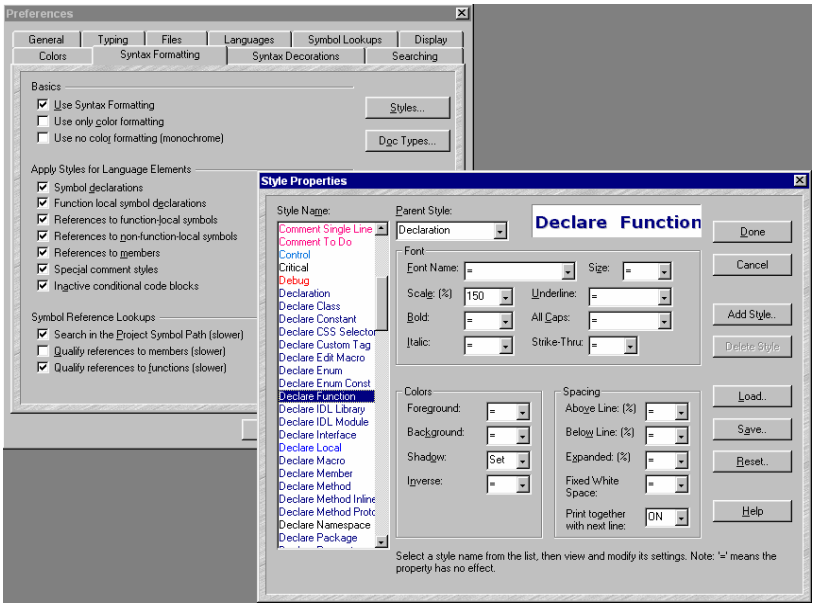


図 1.5 [Style Properties] ダイアログ ボックスで、ユーザーはフォーマット オプションを編集できます。[Preferences] ダイアログ ボックスで、ユーザーはどの言語要素にスタイルを適用させるか制御できます。

状況に応じて名前を変更

Source Insight のインデックスを使用すると、ワンステップで変数、関数、およびその他の識別子の名前を簡単に変更できます。Source Insight の状況依存マッチング機能は、ローカル スコープ変数、グローバル スコープ識別子またはクラス スコープ識別子の名前を変更します。307 ページの「Smart Rename」も参照してください。

混在言語の編集

Source Insight は、HTML およびアクティブ サーバー ページ ファイル (ASP および JSP) をサポートしています。スクリプトはシンボリックに閲覧可能で、適切な構文フォーマットを使用して表示されます。

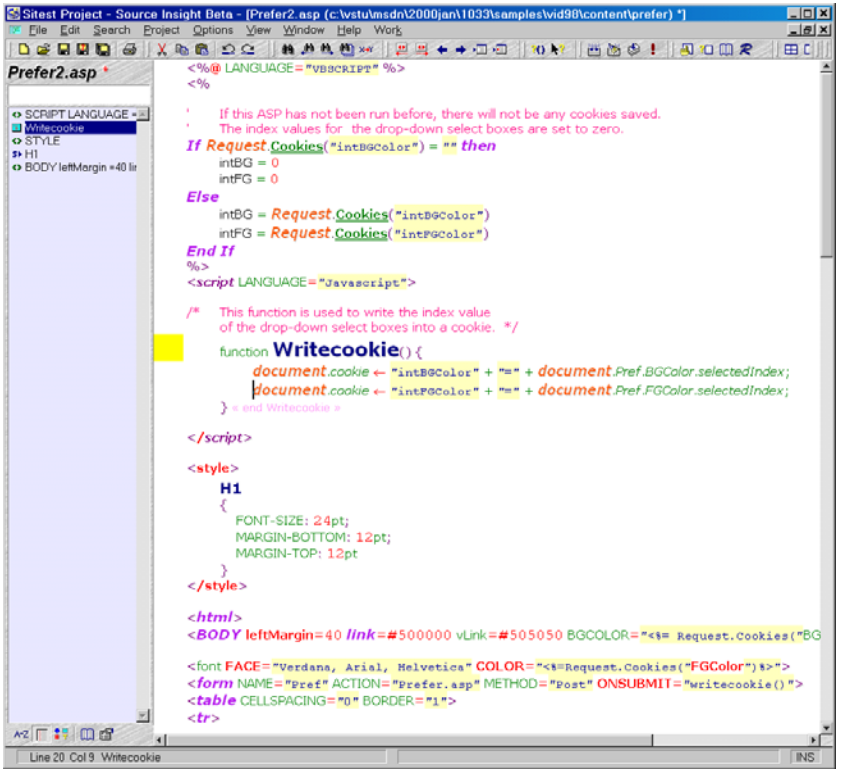


図 1.6 ASP および HTML に組み込まれているスクリプトも構文フォーマットを使用して表示されます。

コード ベースでインターネット検索のようなキーワード検索

[Search Project] コマンドを使用すると、インターネットの検索エンジンに似た、キーワード形式の検索を行うことができます。指定した行数内で、1 つ以上のトピックを参照するコードのセクションを検索します。たとえば、「save disk (copy または duplicate)」と入力すると、Source Insight は、「save」、「disk」、および「copy」と「duplicate」のいずれかの単語を含む検索を行います。この際、「saves」、「saved」、および「saving」のような変化形の単語も合わせて検索します。301 ページの「Search Project」も参照してください。

シンボルの自動補完

Source Insight は、識別子の名前の入力を開始すると、可能性のある識別子の名前の一覧をポップアップで表示します。関数名、変数名、深いレベルまでのクラスと構造体フィールドを表示できます。変数 (継承を含む) の型は瞬時にデコードされます。

すべてのシンボルとファイルに素早くアクセス

Source Insight は、他のエディタよりも多くの役立つプログラミング情報を提供します。シンボル定義をダイナミックに追跡するだけでなく、便利な形式で情報を提供します。

Source Insight を使用すると、ブラウザで Web サイトを参照するのと同じようにプロジェクトを参照できます。ローカルまたはグローバル シンボルをダブルクリックすると、Source Insight は定義を表示するか、または情報ウィンドウをポップアップします。シンボルをクリックすると、瞬時に、プロジェクト内にあるそのシンボルへのすべての参照が一覧で表示されます。シンボル参照ダイアログ ボックスでは、正規表現を使用してシンボルを検索できます。

プロジェクト指向

ネットワーク上の複数のディレクトリも含む、ソース全体のディレクトリ ツリーを Source Insight プロジェクトに追加できます。ファイルが含まれているディレクトリがわからない場合でも、ファイル名を素早く指定できます。Source Insight は、各プロジェクトのシンボルとインデックスの独自のデータベースを自動的に保守します。プロジェクト レポートおよびクロスリファレンスも生成されます。ソース管理プログラムでプロジェクトのファイルを更新すると、Source Insight はシンボル情報を自動的に更新します。

チーム プログラミングのサポート

Source Insight は、コード ベース全体をスキャンし、必要に応じて再同期を行うため、プログラミング チームのメンバが行った変更が自動的に反映されます。

シンボルの定義や使用方法に直ちにジャンプできるので、プログラマはプロジェクトやファイルの構成を考慮する必要がありません。ディレクトリ、マシン、ファイルなどの情報を知らなくても、モジュールとその他のシンボルにアクセスできます。

Source Insight を使用することで、各プログラマは、複数のグループで作成された大規模で複雑なプロジェクトを簡単に理解して編集できるようになります。

参照を素早く検索

Source Insight のシンボル インデックス機能を使用すると、プロジェクト全体にわたってシンボルへの参照を素早く検索して、アクティブな

ソースのリンクを含む一覧を作成できます。244 ページの「Lookup References」も参照してください。

コンパイラ エラーや検索結果へのハイパー ソース リンク

ソース リンクを使用すると、リンクされている場所を素早く移動できます。ソース リンクは、あるファイルの場所と別のファイルの場所を接続しているハイパーテキストのようなリンクです。ソース リンクは、検索結果と一致する場所へのリンク、およびコンパイラ エラーとエラーの場所へのリンクに使用されます。任意のファイルの仕様を解析して、それらのファイルへのリンクを作成することもできます。ファイルを変更すると、ソース リンクも更新されます。たとえば、ファイルにテキストを挿入すると、正しい行を指すようにリンクが更新されます。111 ページの「テキストの検索と置換」も参照してください。

プロジェクト全体の高速な検索と置換

Source Insight は、プロジェクト ファイル内を高速に検索および置換できます。各検索の結果は、検索結果ウィンドウに追加されます。このウィンドウには、検索結果へのアクティブなソースのリンクが含まれます。Source Insight は、検索インデックスを使用して、プロジェクト全体を素早く検索します。正規表現を使用した検索もサポートされています。

複数のビューで構成されたプロジェクト ウィンドウ

Source Insight のプロジェクト ウィンドウは、プロジェクトの内容を複数のモードで表示します。ファイル リスト モードでは、カレント プロジェクトのすべてのファイルを一覧で表示します。プロジェクト ウィンドウからプロジェクトのファイルを開いたり、プロジェクト ウィンドウに Windows エクスプローラからファイルをドラッグして、ファイルをプロジェクトに追加できます。50 ページの「プロジェクト ウィンドウ」も参照してください。

プロジェクト ウィンドウには、シンボルをカテゴリ別に表示、ファイルをカテゴリ別に表示、およびすべてのプロジェクト シンボルを一覧で表示するモードもあります。素早く項目を見つけるには、名前または名前の一部に部分一致機能を使用します。

Source Insight は、数百万行のコードと数十万のシンボル宣言を含むような、大規模なプロジェクトでも処理できます。

外部コンパイラおよびツールとの統合

Source Insight は、カスタム コマンドを使用して、コンパイラ、make プログラム、フィルタ、ソース管理プログラムなどの外部ツールを統合します。プロジェクトを Source Insight 内からコンパイルできます。コンパイラのエラーは編集時に自動的に追跡されます。外部ツールは、Source Insight 内からシェル コマンド ウィンドウで同時に起動できま

す。プログラムの出力は、ファイル バッファに転送できます。または、エラー メッセージ用に解析できます。外部ツールを起動する、独自のカスタム コマンドを追加できます。134 ページの「カスタム コマンド」も参照してください。

クリップボードと定型コード格納用のクリップ ウィンドウ

クリップ ウィンドウを使用して、コードの並べ替えおよび定型コードの挿入を簡単に行うことができます。クリップ ウィンドウには、必要なときにソース ファイルに追加するテキストのクリップが含まれています。クリップは、複数のセッションにわたって自動的に保存され管理されます。また、クリップは、元の関数またはシンボルを記憶しています。63 ページの「クリップ ウィンドウ」も参照してください。

2 段階の行改訂マークと選択した行の復元

Source Insight は、変更した行の余白 (行を削除した場合は次の行の余白) に行改訂マークを表示します。この機能により、ファイルで変更した場所を簡単に確認できます。変更した場所がわかるだけでなく、[Restore Line] コマンドを使用して元のテキストに戻すこともできます。[Restore Line] コマンドは元に戻すことができます。このコマンドは、パワフルな Undo (元に戻す) 機能を提供します。

各ファイルにおける作業の変更履歴は、ファイルを保存した後も保持されています。行改訂マークも、ファイルを保存すると色が変わります。ファイルを保存した後でも、Source Insight を終了するまで、それまでと同じように、変更した行の確認、オリジナルの行に戻す、コマンドの Undo を行うことができます。

ドキュメントの種類および言語の拡張

ドキュメントの種類を定義して、Source Insight に新しいファイルの種類を追加できます。これにより、異なる種類のファイルに対して、異なる編集、表示、言語解析オプションを設定できます。92 ページの「ドキュメント タイプ」も参照してください。

独自のカスタム言語を追加することもできます。カスタム言語には、構文規則、構文フォーマット キーワード、および単純な解析表現を指定します。82 ページの「カスタム言語」も参照してください。

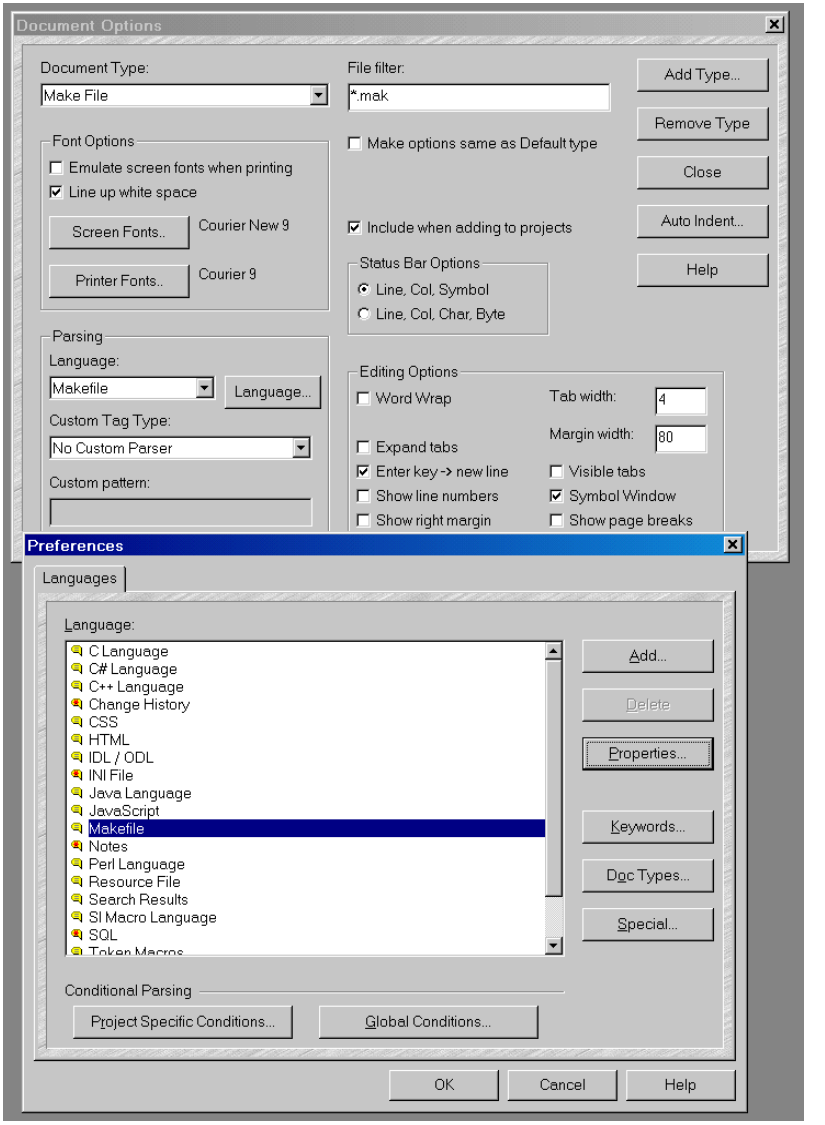


図 1.7 ファイルはドキュメント タイプにマップされ、ドキュメント タイプは言語にマップされます。新規ドキュメント タイプと言語を追加できます。

クラッシュ リカバリによるフルタイム保護の実現

Source Insight は、編集作業を透過的に、しかも頻繁にリカバリ ファイルに記録します。このため、コンピュータがクラッシュしたとき、変更を保存していない場合でも、それまで行った変更を回復できます。これは、作業を中断してファイルを保存する自動保存機能ではありません。アイドル状態のときに、それまで行った変更のみリカバリ ファイルに記録されます。リカバリ ファイルが変更を記録する頻度は指定できます。

継続されるワークスペース

ファイルのセットおよび他のセッション情報をワークスペースでグループ化できます。すべてのセッション状態をワークスペース ファイルに保存して、他のワークスペース ファイルからのセッションを簡単にリストアできます。Source Insight は、終了時に現在のワークスペースを自動的に保存します。Source Insight を再度実行するか、他のプロジェクトを開くと、ワークスペースがリストアされます。Source Insight を終了したとき、プロジェクトを閉じたとき、あるいはマシンがクラッシュしたときでも、以前の状態から再開できます。

カスタマイズ可能なメニューとキーボード

キーボードだけでなく、マウスのボタンやメニューもカスタマイズできます。カスタマイズした設定は、設定ファイルに保存でき、Source Insight 実行中に簡単にリストアできます。別の「セットアップ」プログラムを使用するためにカスタム マクロを記述する必要はありません。

Windows 2000/XP および Windows 9x/Me のサポート

Source Insight は、フローティング ツール ウィンドウ用の半透明ウィンドウやターミナル サーバー セッションなどの Windows 2000/XP の機能をサポートしています。

優れた Windows ユーザー インターフェイス

Source Insight は、32 ビットで実装され、複数のインスタンス、ロング ファイル名、UNC ファイル名、右クリックのショートカット メニュー、およびツールバーをサポートしています。

コマンドやオブジェクトのプロパティへの簡単なアクセスを提供するため、Source Insight は、ウィンドウの多くで右クリックのショートカット メニューを使用します。

ウィンドウは、柔軟に対応できるように、メイン アプリケーション ウィンドウから分離して表示したり、結合して表示できます。

フル機能のエディタ

Source Insight は、ファイルごとに複数回の [Undo] / [Redo] コマンド、自動インデント、構文の色分け、対応する括弧の表示、行番号の表示、キーストロークとコマンドの記録に加えて、ブロック、関数、段落、単語全体の選択用の特別選択モードなど、優れた編集機能も提供します。マウスは完全にサポートされています。同じファイルを複数のウィンドウで開くことができます。ワークスペースは、以前のセッションのファイルとウィンドウをリストアするために使用されます。

ドラッグ アンド ドロップ編集

Source Insight は、ソース ファイル間およびクリップ ウィンドウのクリップ間でテキストのドラッグ アンド ドロップ編集を行うことができます。シンボル ウィンドウからシンボル全体をドラッグ アンド ドロップして、関数の並べ替えを簡単に行うこともできます。プロジェクト ウィンドウにファイルをドラッグするとプロジェクトにファイルが追加され、クリップ ウィンドウにファイルをドラッグすると新しいクリップがロードされます。

主要な企業で実証済み

Source Insight は、主要なソフトウェア企業に携わるプログラマが使用している、業界でも有数のエディタです。これまで、数千のファイル、数百万行のコード、および数十万のシンボル宣言を含む、大規模で複雑な商用プロジェクトで使用されてきた実績があります。

スピーディで便利

Source Insight の目標は、ソース コードを明確にして簡単な方法で情報を示し、プログラマが大規模で複雑なプロジェクトを修正できるようにすることで、プログラミング チームの生産性を高めることです。プログラミング関連したあらゆる情報を素早く入手可能な、優れたプログラム編集環境を提供することにより、Source Insight は作業効率を向上させます。

システム要件

オペレーティング システム :

- Windows 2000/XP/Vista
- Windows NT 4.0 SP3 以上
- Windows 98/Me
- Windows 95 (Internet Explorer 4.0 以上必須)

マシン : Pentium またはそれ以上 (Pentium II またはそれ以上を推奨)

メモリ : 64 MB (128 MB またはそれ以上を推奨)

ディスク空き容量 : 4 MB (最小インストール) または 12 MB (フルインストール)

テクニカル サポート

Source Insight のテクニカル サポートおよびセールスに関する質問は、エクセルソフト株式会社までお問合せください。

Web サイト : http://www.xlsoft.com/jp

開発元の Web サイト : www.sourceinsight.com

第 2 章

セット アップと クイック スタート

この章では、Source Insight のインストールと実行手順について説明します。Source Insight セットアップ後にクイック ツアーが起動します。

Source Insight のインストール

Source Insight インストールはとても簡単です。ほとんどの場合、Source Insight は自動的に設定されます。

Windows NT/2000/XP へのインストール

マシンにソフトウェアをインストールする権限があることを確認してください。

Windows NT ベースのシステムに Source Insight をインストールするには、HKEY_LOCAL_MACHINE レジストリを更新する権限が必要です。しかし、Source Insight をインストールした後は、ほぼすべてのユーザーから利用可能です。Source Insight はユーザーごとに別々の設定情報を保存します。

バージョン 2 からのアップグレード

バージョン 2.0 または 2.1 からアップグレードする場合、403 ページの「バージョン 2 からのアップグレード」の情報を参照してください。

バージョン 3.0 および 3.1 からのアップグレード

バージョン 3.5 では、インストールで使用するプロジェクトとフォルダに関して重要な変更が行われました。バージョン 3.0 または 3.1 からアップグレードする場合、401 ページの「バージョン 3.1 またはバージョン 3.0 からのアップグレード」の情報を参照してください。

CD-ROM の挿入

Source Insight をインストールするには、Source Insight の CD-ROM を CD ドライブまたは DVD ROM ドライブに挿入してください。セットアップ プログラムが自動的に起動します。自動的に起動しない場合は、CD-ROM のルート フォルダにある Setup.exe を実行してください。

Source Insight は通常は CD-ROM で提供されます。CD-ROM 以外のメディアをご要望の方は、エクセルソフト株式会社までお問合せください。

インストール先の選択

セットアップ プログラムでは、Source Insight のインストール先のディレクトリ名を選択します。デフォルトでは、C:\Program Files\Source Insight が選択されます。

メモ： バージョン 2.x を続けて使用したい場合、バージョン 2.x. と違うディレクトリにバージョン 3.5 をインストールしてください。

Source Insight をインストールするには約 4 MB のディスク容量が必要です。.NET Framework の場合、さらに 6 MB のディスク容量が必要になります。

メモ： リモート ネットワーク ドライブに Source Insight をインストールしないでください。Source Insight のセットアップ プログラムは、マシンのレジストリを変更します。ネットワーク ドライブにインストールすると、他のユーザーから正しく利用できません。

バージョン 3 とバージョン 2 の併用

Source Insight の以前のバージョンを継続して使用できます。

たとえば、バージョン 3.5 と 2.x を同じマシンで使用できます。それぞれ別のレジストリ設定を使用するため、併用しても問題ありません。しかし、下記のガイドラインに従ってください。

- マシンにバージョン 2.x をインストールしている場合、バージョン 2.x. と違うディレクトリにバージョン 3.5 をインストールしてください。
- バージョン 2.x とバージョン 3.5 のインスタンスを同時に実行しないでください。
- プロジェクト ファイルは互換性がありますが、バージョン 3.5 用に別のプロジェクトを作成することをお勧めします。
- バージョン 2.x のプロジェクトを開くには、[Open Project] ダイアログ ボックスで [Browse] (参照) ボタンをクリックし、以前の .PR ファイルを指定する必要があります。新しいディレクトリにバージョン 3.5 をインストールした場合、バージョン 3.5 のディレクトリに以前のプロジェクトは含まれません。
- [Options] > [Load Configuration] コマンドで以前の設定ファイルを開くことができます。2.x のディレクトリで以前の *.CF ファイルを選択してください。新しい設定ファイルの拡張子は .CF3 に変更されていることに注意してください。

Source Insight の設定

Source Insight は、スタート メニューのプログラム メニューに追加されます。

セットアップ プログラムがファイルのコピーを完了すると、名前と組織名を入力する画面が表示されます。この画面は 1 回のみ表示されます。

シリアル番号の入力

継続して使用する場合、シリアル番号を購入する必要があります。

Source Insight を実行すると、シリアル番号を入力する画面が表示されます。Source Insight を評価版として使用する場合、シリアル番号を入力する必要はありません。Source Insight の評価期間は 30 日間です。シリアル番号を入力しないで Source Insight を評価版として使用するには、[Try it] ボタンをクリックしてください。

Source Insight のライセンスを購入した場合、シリアル番号を入力してください。いったんシリアル番号が入力されると、Source Insight を継続して使用できるようになります。

メモ： シリアル番号はユーザーではなくマシンに適用されます。同じマシン上のすべてのローカル ユーザーが同じシリアル番号を使用します。

コモン プロジェクトの作成

次に、コモン プロジェクトを作成するかどうか選択します。

コモン プロジェクトは一般的に使用される宣言を含む外部プロジェクトです。

Source Insight で C ランタイムや Java 標準パッケージのような標準ライブラリにシンボル展開などの機能を提供するには、それらのライブラリ用に別々のプロジェクトを設定する必要があります。カレント プロジェクトにシンボルが見つからない場合、Source Insight はプロジェクトを検索するためソートしなおします。

それぞれのコモン プロジェクトについて、対応するファイルを保存するディレクトリを指定します。ディスクにライブラリのソース コードをインストールした場合、Source Insight で、プロジェクトの基本としてそのソース コードを使用できます。たとえば、C ランタイム ライブラリの関数 strtok への呼び出しをクリックすると、Source Insight は strtok のソース コードを表示します。

この時点で作成されたプロジェクトは自動的に、シンボル パスに追加されます。後でパスを変更する場合は、[Options] メニューの [Preferences: Symbol Lookups] コマンドを実行してください。後からプロジェクトを作成してパスを追加することもできます。

304 ページの「Setup Common Projects」も参照してください。

プロジェクトの作成

プロジェクトとはソース ファイルの集まりです。

Source Insight はプロジェクト単位で構成されます。プロジェクトとはソース ファイルの集まりです。プロジェクトを作成する場合、プロジェクトにファイルを追加してください。Source Insight は、プロジェクトのファイル データベースにプロジェクトに含まれるファイルを記録します。

[Add and Remove Project Files] ダイアログ ボックスで、個々のファイルまたは全ディレクトリ ツリーをプロジェクトに追加できます。詳しく

は、153 ページの「Add and Remove Project Files」も参照してください。を参照してください。

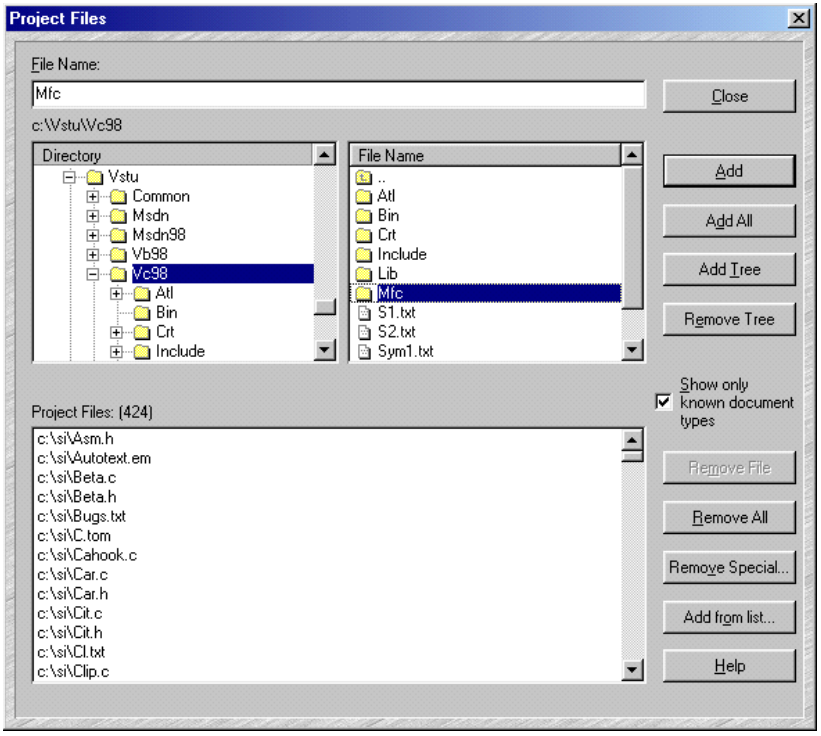


図 2.1 [Add and Remove Project Files] ダイアログ ボックスで、ソース ファイルをプロジェクトに追加できます。

新規ファイルを作成した場合、ファイルは保存時にプロジェクトに追加されます。プロジェクトのディレクトリまたはサブディレクトリに新規ファイルが表示された場合、[Synchronize Files] コマンドを実行して、新規ファイルをプロジェクトに自動的に追加することもできます。

第 3 章

ウィンドウの説明

この章では、Source Insight のユーザー インターフェイスとウィンドウについて説明します。

Source Insight アプリケーション ウィンドウ

Source Insight のユーザー インターフェイスは、主に下記の項目により構成されています。

- トップ画面のメイン メニューとメイン ツールバー。
- ファイルを編集するソース ファイル ウィンドウ。
- 結合または分離可能なツール ウィンドウ。

Source Insight は MDI (Multiple Document Interface) アプリケーションです。つまり、各ソース ファイルは Source Insight アプリケーション ウィンドウ内の子ウィンドウとして表示されます。

Source Insight アプリケーション ウィンドウの上部には、メイン ツールバーがあります。ユーザーは、このウィンドウで大半の作業を行います。

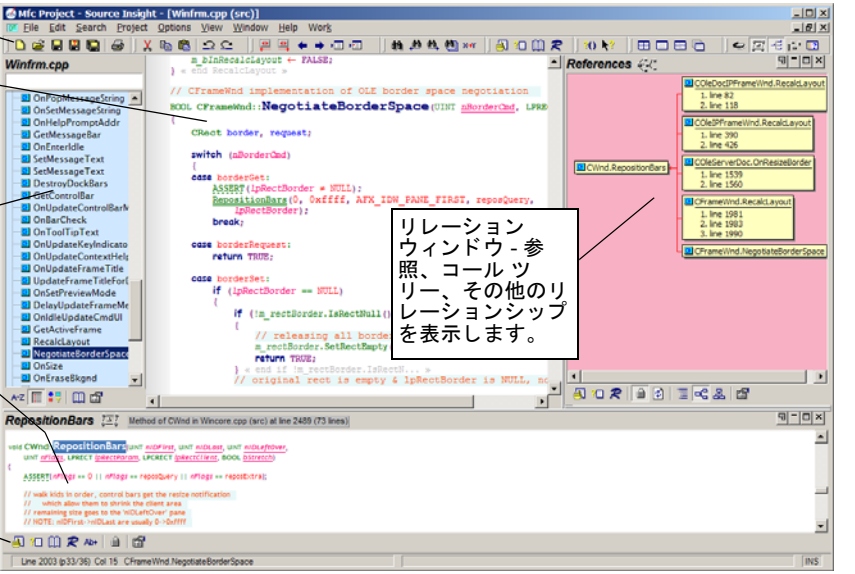


図 3.1 Source Insight のメイン プログラム ウィンドウに、ソース ウィンドウ (中央)、シンボル ウィンドウ (左側)、リレーション ウィンドウ (右側)、コンテキスト ウィンドウ (下部) が表示されています。コンテキスト ウィンドウは画面の下部に結合して表示されます。

ツールバー

メイン ツールバーは Source Insight プログラム ウィンドウの上部にあります。[View] > [Toolbars] > [Main Toolbar] コマンドで、メイン ツールバー全体のオン、オフを切り替えることができます。

[View] > [Toolbar] メニュー項目で、表示するツールバーを指定できます。

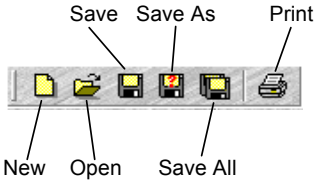
メイン ツールバーは、小さなサブ ツールバーにより構成されています。各サブ ツールバーは、[View] > [Toolbars] メニューを使用して、分離して表示できます。メイン ツールバー内で、サブ ツールバーをドラッグして移動することもできます。

各ツールバーの位置は、設定ファイルに自動的に保存されます。

ツールバーのアイコンはそれぞれ、Source Insight のコマンドと対応しています。各コマンドの詳細は、第 5 章を参照してください。

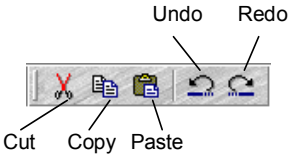
標準ツールバー

標準ツールバーには、基本的なファイル操作が含まれています。



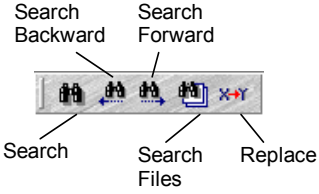
編集ツールバー

編集ツールバーには、切り取り、コピー、貼り付けのような基本的な編集操作が含まれています。



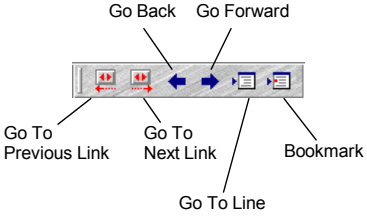
検索ツールバー

検索ツールバーには、検索コマンドが含まれています。



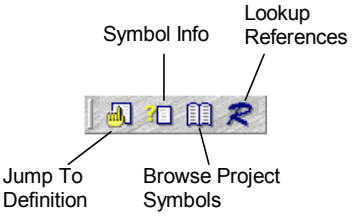
ナビゲーション ツールバー

ナビゲーション ツールバーには、ファイル内およびファイル間を移動するコマンドが含まれています。



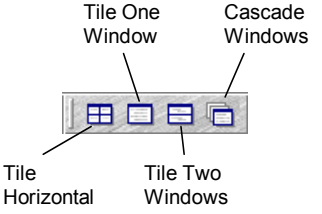
シンボル ツールバー

シンボル ツールバーには、シンボル情報へアクセスするコマンドが含まれています。



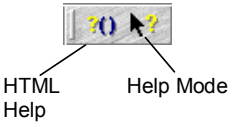
ウィンドウ ツールバー

ウィンドウ ツールバーには、ウィンドウを調整するコマンドが含まれています。



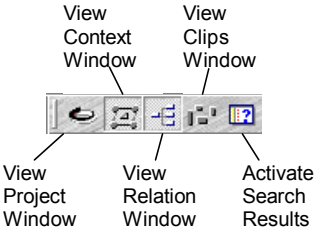
ヘルプ ツールバー

ヘルプ ツールバーには、オンライン ヘルプへアクセスするコマンドが含まれています。



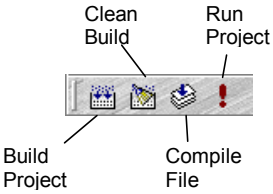
ビュー ツールバー

ビュー ツールバーには、コンテキスト ウィンドウやプロジェクト ウィンドウのような補助的なウィンドウを表示、非表示にするコマンドが含まれています。



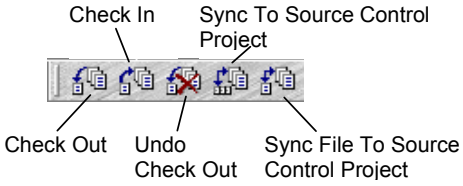
ビルド ツールバー

ビルト ツールバーには、プロジェクトの実行ファイルをビルドする際に使用するコマンドが含まれています。これらのコマンドは、カスタム コマンドとして定義されます。



ソース管理ツールバー

ソース管理ツールバーには、ソース管理システム (バージョン管理システム) へのアクセスに使用するコマンドが含まれています。これらのコマンドは、カスタム コマンドとして定義されます。



ヒント： これらのコマンドを素早く編集するには、[Ctrl] キーを押したままでツールバー ボタンをクリックし、[Custom Command] ダイアログ ボックスを表示します。

ソース ファイル ウィンドウ

開いたファイルはそれぞれ、別々のソース ファイル ウィンドウに表示されます。Source Insight は MDI (Multiple Document Interface) アプリ

ケーションです。各ソース ファイル ウィンドウの左側には、シンボル ウィンドウがあります。このウィンドウは非表示にできます。

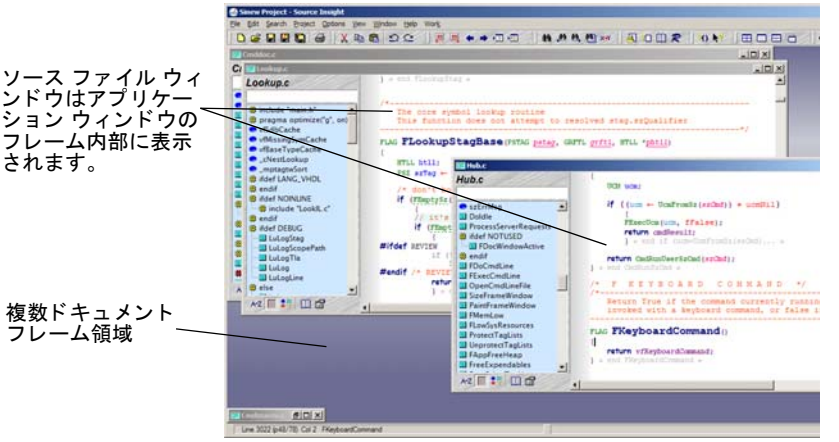


図 3.2 ソース ファイル ウィンドウ。

ソース ファイルを開くと、個々のソース ファイル ウィンドウが表示されます。通常の編集に関してはすべて、このウィンドウで行います。ソース ファイル ウィンドウは MDI ウィンドウです。

ソース ファイル ウィンドウはファイル バッファを表示します。

言語が添付されたファイルを開く場合、シンボル ウィンドウがソース ファイル ウィンドウの左側に表示されます。**ドキュメント オプション**を選択して**シンボル ウィンドウ** チェック ボックスを設定することで、シンボル ウィンドウを使用するかどうか指定できます。194 ページの「Document Options」も参照してください。

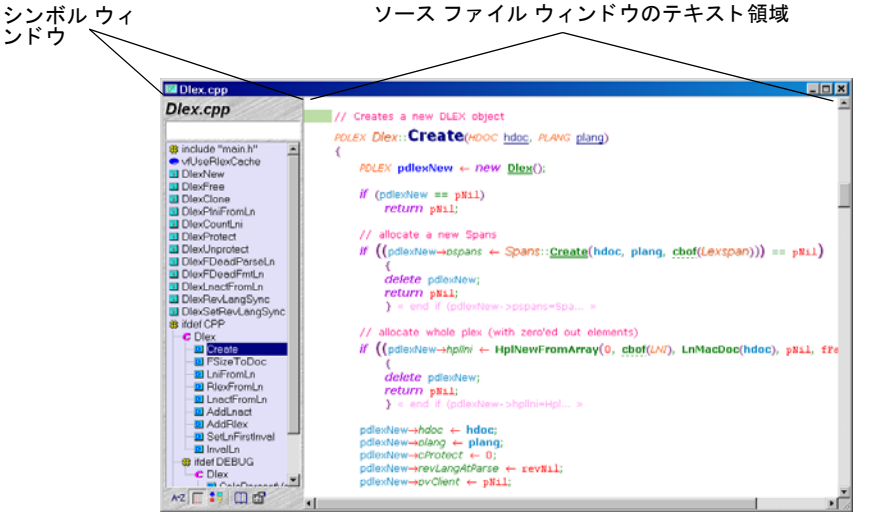


図 3.3 ソース ファイル ウィンドウの左側には、シンボル ウィンドウがあります。

シンボル ウィンドウ

シンボル ウィンドウは通常、ソース ファイル ウィンドウの左側に表示されます。

シンボル ウィンドウは言語が添付されたソース ファイル ウィンドウの左側に表示されます。シンボル ウィンドウには、ファイルで定義されているすべてのシンボルがリストされます。たとえば、すべての関数、構造体、クラス、メソッド、マクロ、定数、その他の情報がシンボル ウィンドウにリストされます。シンボル ウィンドウ リストの各項目の左には、シンボルの種類を示す小さなアイコンがあります。

Source Insight はバックグラウンドでファイルをスキャンし、シンボル ウィンドウを動的に更新します。新規宣言を入力するとすぐに、シンボル ウィンドウにシンボルが表示されます。

シンボル ウィンドウを使用すると、各ファイル内を簡単に移動できます。ファイルの概要を素早く表示することもできます。シンボル ウィンドウから別のウィンドウへシンボルをドラッグすることもできます。

シンボルを素早く識別できるように、#ifdef-#endif 入れ子レベルとシンボルの種類を示すアイコンも表示します。

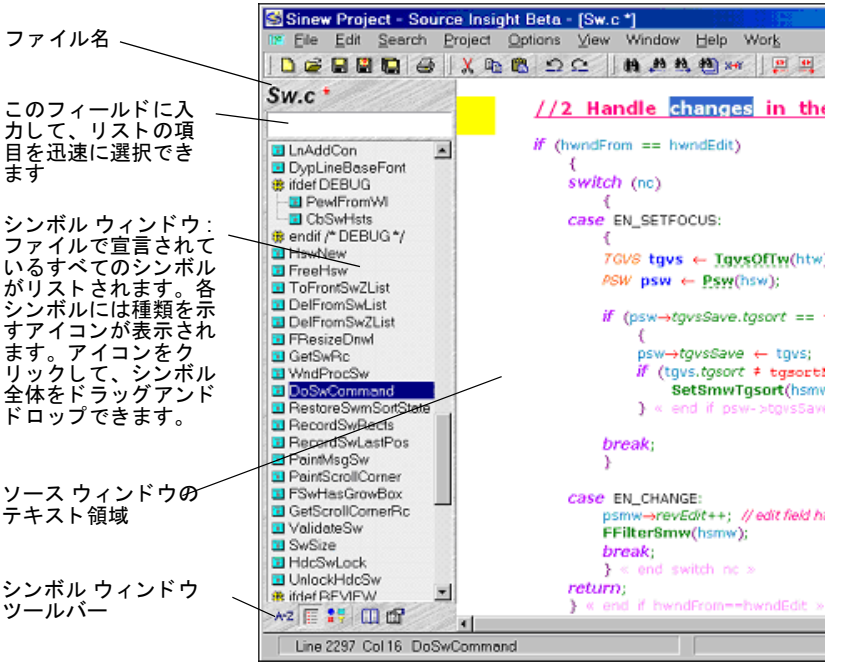


図 3.4 シンボル ウィンドウは各ソース ファイル ウィンドウの左側に表示されます。

シンボル ウィンドウの下部には小さなツールバーがあります。このツールバーには、リストをソートするコントロールと [Browse Local File Symbols] コマンドを実行するボタンがあります。シンボル ウィンドウを右クリックすると、ショートカット メニューが表示されます。

シンボル ウィンドウのカスタマイズ

シンボル ウィンドウの設定を変更するには、シンボル ウィンドウを右クリックして、[Symbol Window Properties] を選択します。318 ページの「Symbol Window Properties」も参照してください。

シンボル ウィンドウの幅の変更

シンボル ウィンドウの幅を変更するには、ウィンドウの右端をクリックしてドラッグします。

変更したシンボル ウィンドウの幅の固定

シンボル ウィンドウ (および、他のファイルのシンボル ウィンドウのすべて) の幅を変更して固定するには、ウィンドウの右端をドラッグ

してサイズを変更した後、シンボル ウィンドウを右クリックして、[Record New Default Properties] を選択します。ウィンドウの幅、シンボルのソート、シンボルの種類のフィルタリングが記録され、以降に作成される新規ウィンドウのデフォルト値として使用されます。

フローティング ウィンドウ

フローティング ウィンドウは、Source Insight のメイン ウィンドウに結合できます。

フローティング ウィンドウは、メイン アプリケーション ウィンドウの手前に表示されます。フローティング ウィンドウは、メイン ウィンドウの端にドラッグして、メイン ウィンドウに結合することもできます。ツール ウィンドウには次のウィンドウがあります。

- **プロジェクト ウィンドウ** - プロジェクト ファイルとシンボルを表示するマルチモードのウィンドウ。
- **コンテキスト ウィンドウ** - 状況依存の情報ウィンドウ。
- **クリップ ウィンドウ** - 簡単なコピー、貼り付けのためにクリップボード形式のクリップを表示するウィンドウ。
- **リレーション ウィンドウ** - コール ツリー、クラス ツリー、その他のリレーションシップを表示するウィンドウ。

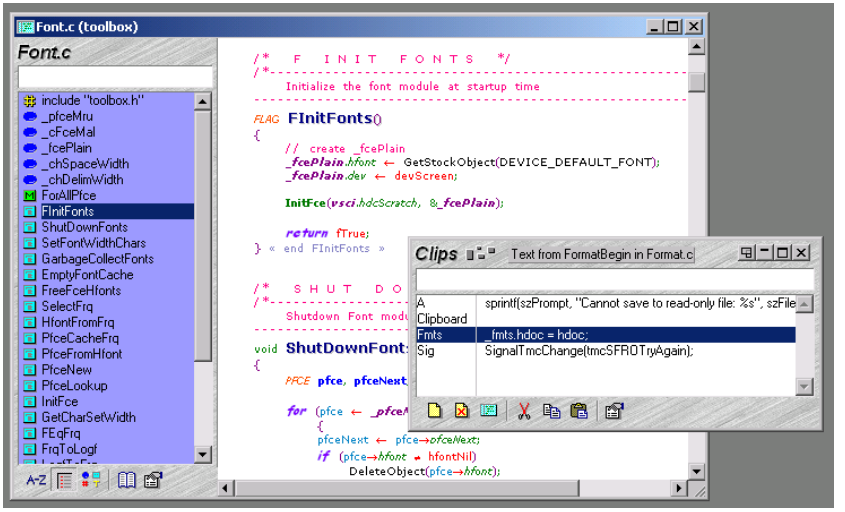


図 3.5 クリップ ウィンドウは、Source Insight のメイン ウィンドウの手前に表示されるフローティング ウィンドウの例です。プログラム ウィンドウの端に結合することもできます。

ヒント： フローティング ウィンドウを結合するには、マウス カーソルを結合するウィンドウの端までドラッグします。

透明なフローティング ウィンドウ

Source Insight は、半透明なフローティング ウィンドウをサポートしています。これは、ゲームのヘッドアップ ディスプレイ (オーバーレイ) のようなものです。

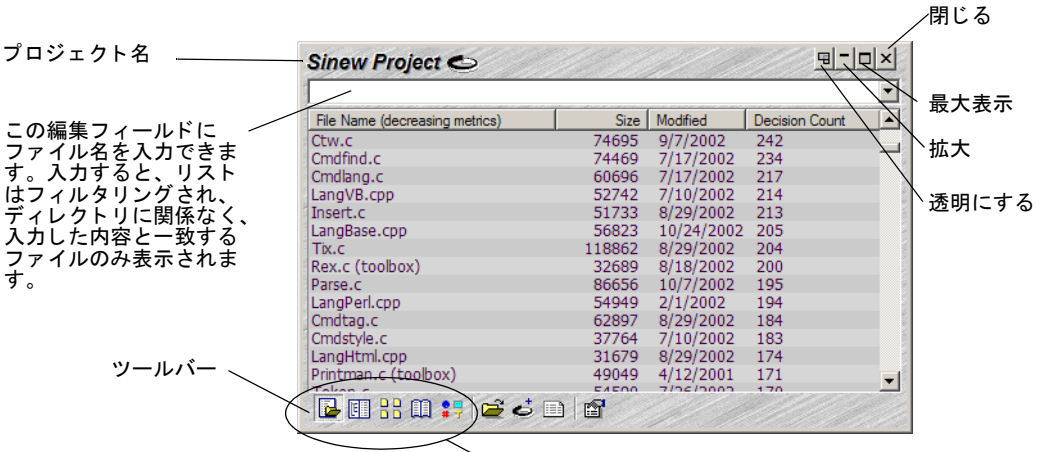
フローティング ウィンドウは、Windows 2000/XP では半透明にできます。

Windows 2000 または Windows XP (半透明モードをサポート) を使用している場合、フローティング ウィンドウにウィンドウを透明に切り替えるボタンが表示されます。

ウィンドウが透明の場合、フローティング ウィンドウ内のオブジェクトをクリックしていない限り、ウィンドウの下のテキストをクリックできます。

プロジェクト ウィンドウ

[Open] コマンドまたは [Project Window] コマンド ([View] メニュー) を実行すると、プロジェクト ウィンドウが表示されます。プロジェクト ウィンドウにはプロジェクト内のすべてのファイルとシンボルがリストされており、ファイルが含まれているディレクトリに関係なく、ファイルを素早く開くことができます。



これらのボタンで、プロジェクト ウィンドウのビューを切り替えることができます。5 つのビューがあります。

1. ファイル リスト ビュー - プロジェクトのすべてのファイルをリスト
2. ファイル ディレクトリ ビュー - ディレクトリごとにファイルをリスト
3. ファイル タイプ ビュー - ドキュメント タイプごとにファイルをリスト
4. シンボル リスト ビュー - プロジェクトのすべてのシンボルをリスト
5. シンボル クラス ビュー - クラスおよびタイプごとにシンボルをリスト

図 3.6 プロジェクト ウィンドウ

プロジェクト ウィンドウは Source Insight のメイン アプリケーション ウィンドウの横に結合するか、または手前に表示できます。

プロジェクト ウィンドウの下部には、プロジェクトを開く、プロジェクトへファイルを追加する、プロジェクトからファイルを削除するボタンを含む、小さなツールバーがあります。

ファイルを開く

ファイルを開くには、プロジェクト ウィンドウのファイル名をダブルクリックします。プロジェクト ウィンドウを右クリックすると、ショートカット メニューが表示されます。プロジェクト ウィンドウの上部のテキスト ボックスに入力すると、入力した内容と一致するファイルのみが表示されます。入力した内容とファイルを一致させる際には使用されないため、ファイルが存在するディレクトリはあまり重要ではありません。ファイルのディレクトリがわからない場合は、リーフ名を入力するだけでかまいません。プロジェクトのほとんどのファイルは、わずかなキーストロークで開くことができます。

プロジェクト ウィンドウ ビュー

プロジェクト ウィンドウ ビューには 5 つの異なるビューがあります。

- **ファイル リスト ビュー** - プロジェクトのすべてのファイルをリストします。
- **ファイル ディレクトリ ビュー** - ディレクトリごとにファイルをリストします。
- **ファイル タイプ ビュー** - ドキュメント タイプごとにファイルをリストします。
- **シンボル リスト ビュー** - プロジェクトのすべてのシンボルをリストします。
- **シンボル クラス ビュー** - クラスおよびタイプごとにシンボルをリストします。

プロジェクト ウィンドウの下部にあるツールバーのボタンをクリックして、これらのビューを切り替えることができます。

ファイル リスト ビュー

ファイル リスト ビューは、カレント プロジェクトのすべてのファイルをリストで表示します。ワイルドカードの入力やテキスト ボックスに入力することにより、作業ディレクトリの変更も可能です。

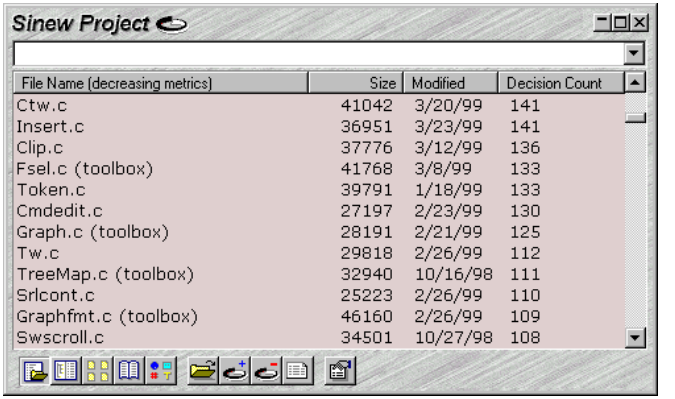


図 3.7 プロジェクト ウィンドウの「ファイル リスト」ビュー

シラブル (音節) マッチングを使用する場合、ディレクトリに関係なく、ファイル名の一部を入力し、ファイルを検索できます。

ワイルドカードでリストをフィルタリングするには、ワイルドカードを入力し、[Enter] を押します。

ワイルドカードを入力して [Enter] を押すと、指定と一致するファイル リストがフィルタリングされます。たとえば、*.c を入力して [Enter] を押した場合、ディレクトリに関係なく、プロジェクトのすべての *.c ファイルがリストされます。ワイルドカードを削除するには、* (アスタリスク) を押し、[Enter] を押します。

ノンプロジェクト ファイルの参照

ディスクを参照する際にプロジェクトの一部ではないファイルも表示するには、ドット (.) を入力し、[Enter] を押します。現在の作業ディレクトリの内容がリストで表示されます。プロジェクトの一部であるファイルのみを表示するビューに戻るには、** (アスタリスク 2 つ) を入力し、[Enter] を押します。または、プロジェクト ウィンドウのファイル ディレクトリ ビューに切り替えます。

ファイル ディレクトリ ビュー

ファイル ディレクトリ ビューはディスクのディレクトリ構造を表示します。

ファイル ディレクトリ ビューは、ディレクトリとファイルを表示します。このビューを使用して、基本ディレクトリとファイルのメンテナ

ンスを行い、ノンプロジェクト ファイルを簡単に開くことができます。

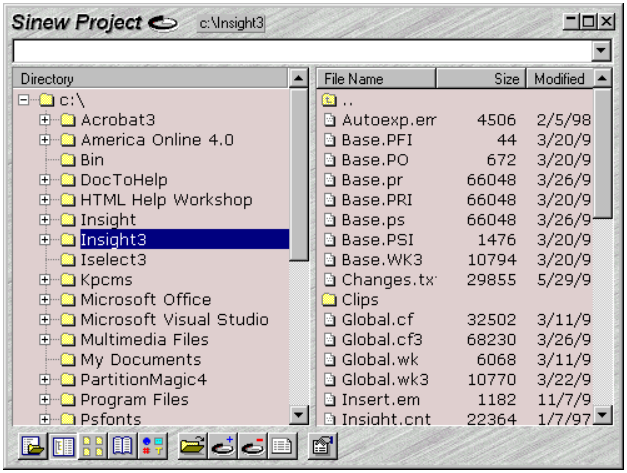


図 3.8 プロジェクト ウィンドウの「ファイル ディレクトリ」ビュー

シンボル リスト ビュー

シンボル リスト ビューは、プロジェクトのすべてのシンボルを表示します。

シンボル リスト ビューは、プロジェクト シンボル データベースのすべてのシンボルのリストを表示します。これは、[Browse All Symbols] ダイアログ ボックスに表示される内容と似ています。

シンボルを素早く検索するには、シンボル名の一部を入力します。入力した内容でリストがフィルタリングされます。

最初に疑問符 (?) を入力して正規表現を指定することで、シンボル名の正規表現検索を行うこともできます。たとえば、

```
?Insert.*Stack
```

は「Insert」、0 文字以上の文字、「Stack」を含むすべてのシンボルを検索します。

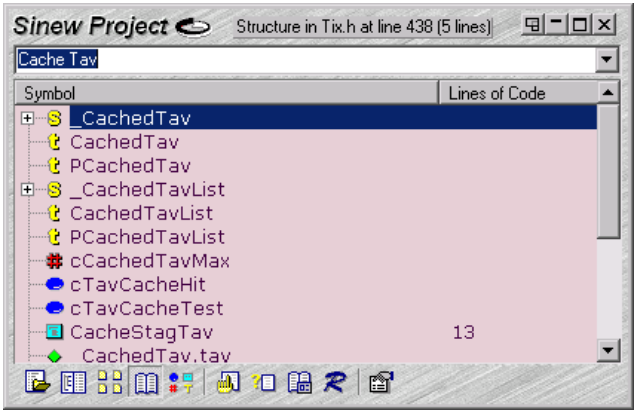


図 3.9 プロジェクト ウィンドウの「シンボル リスト」ビュー

ファイル タイプ ビュー

ファイル タイプ ビューは、ドキュメント タイプごとにファイルを表示します。

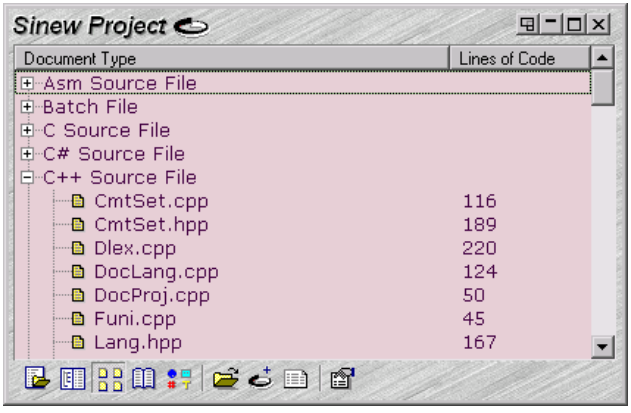


図 3.10 プロジェクト ウィンドウの「ファイル タイプ」ビュー は、ファイルのカテゴリ リストを表示します。

シンボル クラス ビュー

シンボル クラス ビューは、シンボルをクラスおよびタイプごとに表示します。

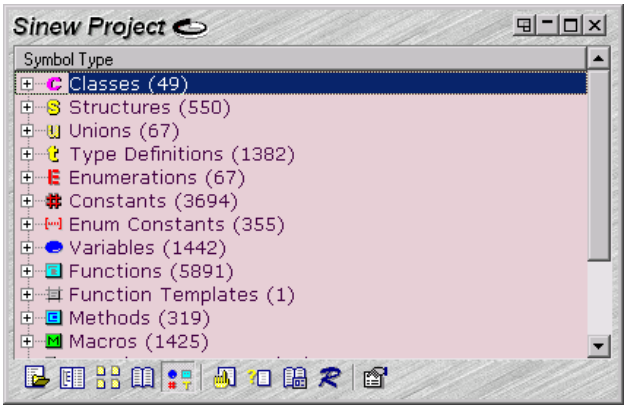


図 3.11 プロジェクト ウィンドウの「シンボル クラス」ビューは、プロジェクトのシンボルをカテゴリ別に表示します。

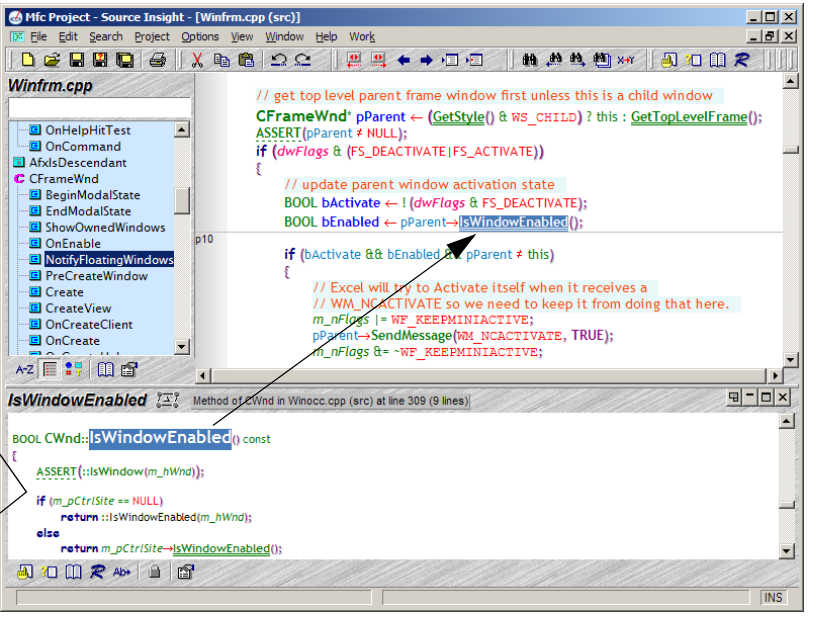
コンテキスト ウィンドウ

コンテキスト ウィンドウは、ソース コードを編集、表示する際に、関連する情報を自動的に表示する画期的な機能です、

コンテキスト ウィンドウはコンテキストに基づき、シンボル情報を表示します。

コンテキスト ウィンドウは、入力またはクリックした内容に関連する情報を表示する、結合および分離可能なウィンドウです。たとえば、関数呼び出しをクリックした場合、コンテキスト ウィンドウは関数の定義を表示します。変数をクリックした場合、コンテキスト ウィンドウは宣言を解読し、基本構造とクラス タイプを表示します。

コンテキスト ウィンドウは、プロジェクト ウィンドウで選択されているファイル、リレーション ウィンドウのシンボル、クリップ ウィンドウで選択されているクリップも自動的に表示します。



コンテキスト ウィンドウはソース ファイル ウィンドウで選択されている識別子に関する情報を表示します。ここでは、クラス メンバ関数の定義を表示しています。

図 3.12 ウィンドウの下部には選択されているシンボルの宣言が表示されます。

[Context Window] コマンドを実行し、コンテキスト ウィンドウのオン、オフを切り替えることができます。[Activate Global Symbol List] コマンドは、コンテキスト ウィンドウを表示してから、シンボルの名前を入力できるようにテキスト ボックスにフォーカスをセットします。[Browse Project Symbols] ダイアログ ボックスに似ています。

ファイルのプレビュー

コンテキスト ウィンドウはプロジェクト ウィンドウでファイルを選択すると、ファイル プレビューを表示します。

プロジェクト ウィンドウが手前に表示されている場合、コンテキスト ウィンドウにはプロジェクト ウィンドウで現在選択されているファイルが表示されます。クリップ ウィンドウが手前に表示されている場合、コンテキスト ウィンドウには選択されているクリップの内容が表示されます。

宣言と定義の表示

ソース ファイル ウィンドウの識別子名をクリックすると、コンテキスト ウィンドウには自動的にシンボルの宣言が表示されます。関数や他

のシンボルが、パラメータおよび他の定義情報と共に、コンテキスト ウィンドウに表示されます。

コンテキスト ウィンドウはクリックまたは入力するシンボルの種類を判断します。たとえば、変数をクリックした場合、変数の宣言が表示されます。変数がデータ構造インスタンスまたはポインタである場合、コンテキスト ウィンドウには構造やクラスの定義が表示されます。

コンテキスト ウィンドウは、プロジェクト ウィンドウ、リレーション ウィンドウ、クリップ ウィンドウのような他のウィンドウの選択内容を追跡します。

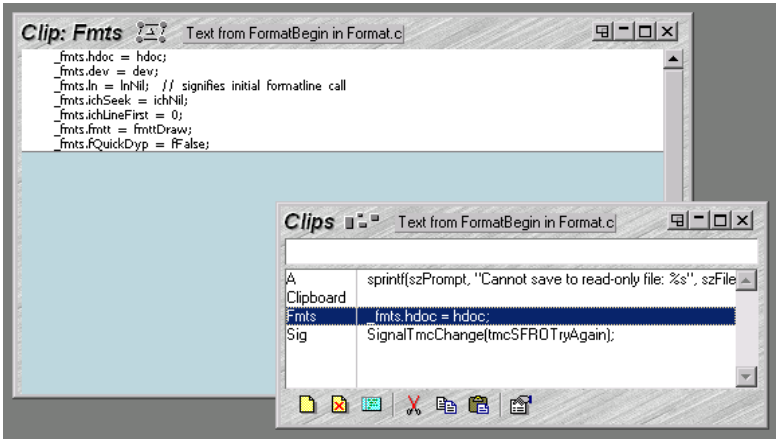


図 3.13 コンテキスト ウィンドウ (バックグラウンド) がクリップ ウィンドウで選択されているクリップの内容を表示している。

コンテキスト ウィンドウの動作を要約すると、下記の表のようになります。

表 3.1: コンテキスト ウィンドウの動作

アクション	コンテキスト ウィンドウの表示
ソース ファイル ウィンドウの識別子を選択 (クリック)	コンテキストに基づいてシンボル定義を表示します。
複数のシンボルが一致した場合、一致のリストを表示します。	プロジェクトで一致するシンボルが 1 つしかない場合、シンボルの宣言を表示します。

表 3.1: コンテキスト ウィンドウの動作 (続き)

アクション	コンテキスト ウィンドウの表示
ソース ファイル ウィンドウへ入力	自動完了機能が無効な場合、コンテキスト ウィンドウは接頭辞一致を表示します。一致する情報が 1 つしかない場合、シンボル定義を表示します たとえば、「Insert」と入力した場合、「Insert」で始まるすべてのシンボルを表示します。「Insert」という名前のシンボルが 1 つしかない場合、「Insert」の宣言を表示します。
プロジェクト ウィンドウでファイルを選択	ファイルの内容を表示します。
クリップ ウィンドウでクリップを選択	クリップの内容を表示します。
リレーション ウィンドウでアイテムを選択	選択されているシンボルの定義を表示します。リレーション ウィンドウのシンボルが参照の場合、参照の位置を表示します。

構造を表示するベース タイプの解読

コンテキスト ウィンドウはベース タイプを動的に解読します。

コンテキスト ウィンドウはタイプの階層を上ることで、ベースとなる構造タイプを決定します。つまり、typedefs はベース構造タイプに解読されます。たとえば、下記のようなコードの場合。

```
struct S { ... };
typedef S T;
typedef T *PT;
PT ptvar;
.
.
ptvar->foo....
```

`ptvar->foo` の `foo` を選択した場合、コンテキスト ウィンドウは `PT ptvar` の宣言を探し、`struct S` が見つかるまで、タイプの階層を解読します。そして、`struct S` 内のフィールド `foo` の宣言を表示します。

コンテキスト ウィンドウはクラス階層も動的に解読します。つまり、範囲にあるメンバを探しながら、クラス派生階層構造を探します。

コンテキスト ウィンドウのカスタマイズ

Context Window Properties コマンドを使用して、コンテキスト ウィンドウの設定を変更できます。上記のようにベース タイプの解読をオフに

して、コンテキスト ウィンドウがどのようにカーソルを追跡するか制御できます。171 ページの「Context Window Properties」も参照してください。

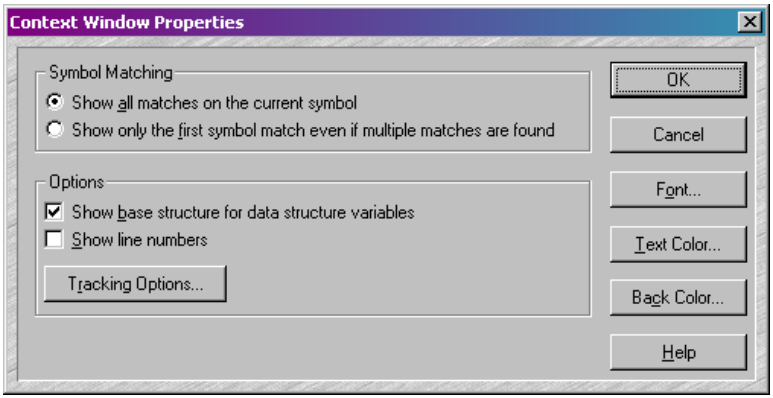


図 3.14 [Context Window Properties] ダイアログ ボックス

リレーション ウィンドウ

リレーション ウィンドウを使って、関数呼び出しツリーや参照ツリーを見ることができます。

リレーション ウィンドウは、現在選択しているシンボルと他のシンボルの関係を表示する Source Insight の革新的な機能です。現在の動作状況を追跡したり、リレーション情報を自動的に表示するなど、コンテ

キスト ウィンドウのように機能します。[View] > [Relation Window] コマンドは、リレーション ウィンドウのオンとオフを切り替えます。

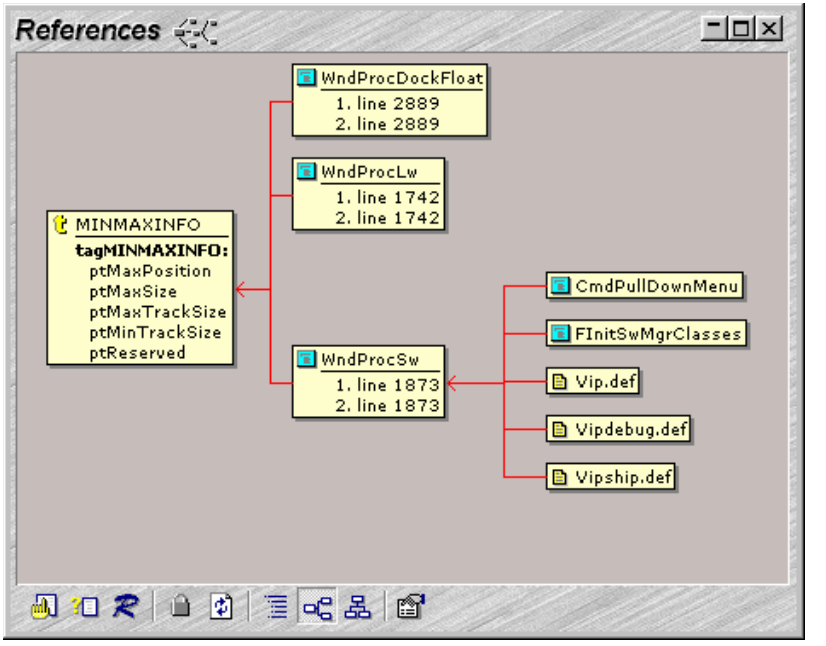


図 3.15 リレーション ウィンドウは、型への参照、および関数からの間接参照を表示します。

リレーション ウィンドウはバックグラウンドで動作し、ユーザーが選択したシンボルを追跡します。クラス階層、コール ツリー、参照ツリー、その他を参照できます。リレーション ウィンドウの利点は、何も特別なことを行う必要がないことです。作業中はバックグラウンドで動作していますが、必要なときに呼び出すことができます。複数のリレーション ウィンドウを開いて、異なる種類の情報を表示することもできます。

アウトラインとグラフ ビュー

リレーション ウィンドウでアウトラインとグラフ ビューを表示できます。

リレーション ウィンドウには、アウトライン ビューとグラフ ビューの 2 種類のビューがあります。グラフ ビューはシンボル同士を線で接続し、グラフ ノードとしてシンボルを表示します。[Relation Graph Properties] コマンド (リレーション ウィンドウで使用可能な右クリッ

ク ショートカット メニュー) により、グラフ ビューの表示方法をコントロールできます。

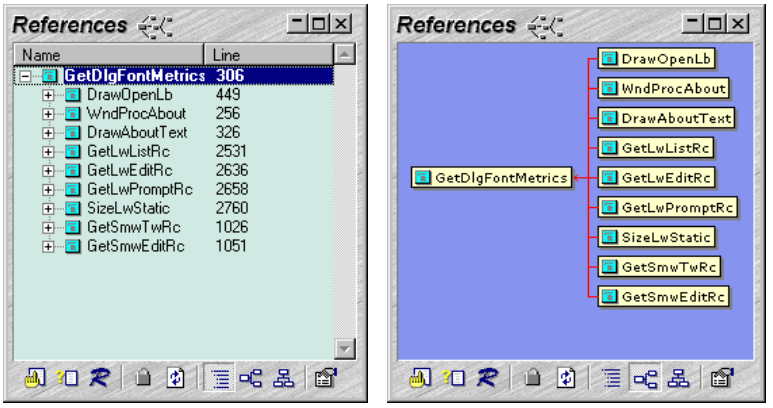


図 3.16 同じリレーション データのアウトライン ビューとグラフ ビュー

リレーションシップのタイプ

リレーション ウィンドウは異なるリレーションシップを表示できます。

リレーションシップは計算上最も速いものから最も遅いものまでリストされた 3 つのカテゴリに分類されます。

- **Contains** – 現在のシンボルの内容を表示します。たとえば、構造体のメンバ。
- **Calls** – 現在のシンボルによって参照されている他のシンボルを表示します。たとえば、現在の関数に呼び出されている関数。
- **References** – 現在のシンボルを参照している他のシンボルを表示します。たとえば、現在の関数を呼び出している関数。

リレーション ウィンドウの性能

リレーション ウィンドウは多くの処理を行います。一部のリレーションシップは遅くなります。非常に大規模なプロジェクトでは、「References」リレーションシップは処理が最も遅くなります。

リレーションシップのルール

リレーションシップには表示する情報を決めるルールがあります。

表示されるリレーションシップはシンボルのタイプによって決まります。[Relation Window Properties] ダイアログ ボックスで、異なるシンボル タイプに表示するリレーションシップを指定できます。たとえば、「Calls」では関数のリレーションシップ、「Inheritance」ではクラスのリレーションシップを表示するように指定できます。

リレーション ウィンドウがシンボルを拡張して新規レベルを表示する場合、拡張で表されるリレーションシップは拡張するシンボルのタイプに基づきます。これは、各リレーション ウィンドウが複数のリレーションシップを潜在的に表示できることを意味します。273 ページの「Relation Window Properties」も参照してください。

コール グラフ

関数呼び出しグラフ リレーションシップは、特定のパスのみを表示するようにフィルタリングできます。[Relation Window Properties] ダイアログ ボックスの [Call Graph Filtering] ボタンをクリックすると、除外する特定の関数を指定してフィルタリングを制御できるダイアログ ボックスが表示されます。コード メトリクスの規制による機能もフィルタリングできます。

Source Insight は C マクロを正当な関数形式のシンボルとみなすため、C マクロはコール グラフに表示されることに注意してください。C マクロは [Call Graph Filtering] ダイアログ ボックスでフィルタリングすることもできます。

複数のリレーション ウィンドウ

複数のリレーション ウィンドウを使用して、同時に異なるリレーションを表示できます。

リレーション ウィンドウで右クリック、または新規リレーション ウィンドウ コマンドを選択することにより、複数のリレーション ウィンドウを使用できます。2 つ以上のリレーション ウィンドウを使用することにより、リレーションシップの複数のタイプを表示したり、同時に異なるターゲットを追跡できます。たとえば、あるウィンドウでは選択した関数によって呼び出されている関数を表示し、別のウィンドウでは選択した関数を呼び出す関数を表示します。

リレーション ウィンドウのカスタマイズ

[Relation Window Properties] コマンドは、リレーション ウィンドウ ツールバーまたは、ショートカット メニューからアクセスできます。このコマンドから表示するリレーションシップ、およびウィンドウの表示方法を制御できます。273 ページの「Relation Window Properties」も参照してください。

[Relation Graph Properties] コマンドを使用して、リレーション ウィンドウのグラフ ビューの表示方法をカスタマイズすることもできます。272 ページの「Relation Graph Properties」も参照してください。

クリップ ウィンドウ

クリップ ウィンドウは、クリップを表示する分離および結合可能なウィンドウです。クリップ ウィンドウへテキストをドラッグ アンド ドロップしたり、クリップ ウィンドウからファイルへテキストをドラッグできます。

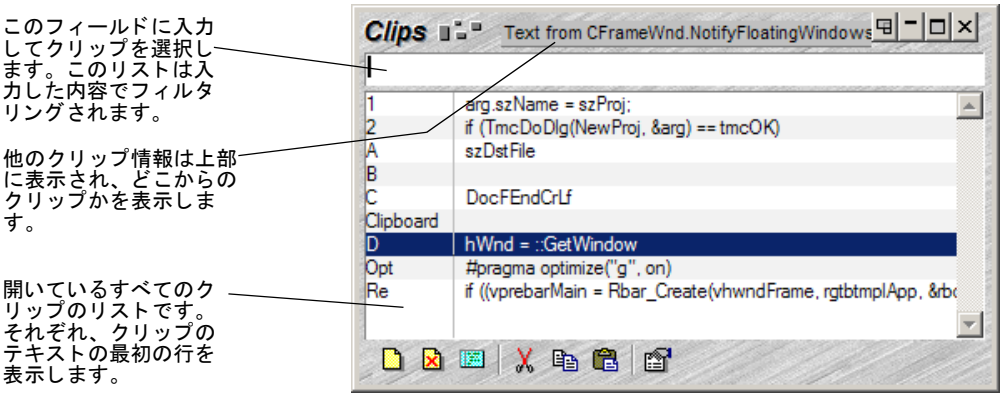


図 3.17 クリップ ウィンドウ

クリップとは

クリップは、コードやボイラープレート (定型) テキストの再配置に便利です。

クリップはクリップ ボード形式のドキュメントです。実際、クリップボードは Source Insight ではクリップとみなされます。Source Insight では、たくさんのクリップを使用できます。クリップは他のファイルと似ています。他のファイルと同じように編集できます。クリップと他のファイルの違いは、クリップはセッションの間に自動的に保存され、Paste From Clip コマンドで簡単に貼り付けることができる点です。

クリップは特に複数のファイル間でコードを再配置する場合に便利です。クリップはボイラープレート テキストを頻繁に挿入する場合にも便利です。

[Clip Window] コマンドまたは [Activate Clip Window] コマンドを実行して、クリップ ウィンドウのオン、オフを切り替えることができます。

新規クリップの作成

新規クリップを作成するには、[File] メニューまたはクリップ ウィンドウ ツールバーから [New Clip] コマンドを実行します。新規ソース ファイル ウィンドウが表示されます。他のファイルのように、編集や入力が可能です。ファイルを閉じた場合でも、クリップはファイルが削除されるまでクリップ ウィンドウに残ります。

新規クリップを作成するには、テキストをクリックしてクリップ ウィンドウへドラッグします。

[Copy to Clip] コマンドを使用してクリップを作成することもできます。エクスプローラからファイルをクリップ ウィンドウへドラッグして、ファイルをクリップとして開くこともできます。

クリップの保存

クリップは、Source Insight プログラム ディレクトリの Clips サブディレクトリに自動的に保存されます。Clips サブディレクトリに作成したテキスト ファイルは、Source Insight の起動時に自動的にロードされます。

検索結果ウィンドウ

マルチファイルの検索と置換結果は、検索結果ウィンドウに表示されます。

検索結果ウィンドウは、[Search Files] コマンドまたは [Lookup References] コマンドを実行すると作成されます。検索結果ウィンドウのそれぞれの行は、行番号のファイルに対応しています。検索結果ウィンドウにリストされた一致には、ソース リンク (一致が見つかった場所へのハイパーテキスト形式のリンク) が含まれます。

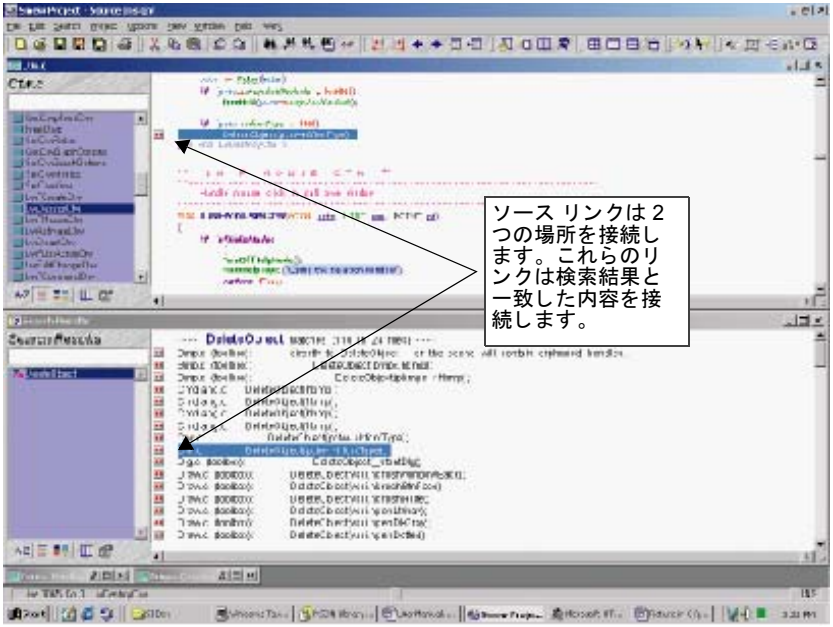


図 3.18 検索結果ウィンドウは、リストされている一致ごとにソース リンクを表示します。各ソース リンクはファイル中の位置に対応しています 上記の検索結果ウィンドウは、あるソース リンクのターゲット位置です。

検索結果ウィンドウの各セットの一致には、検索に使用されたパターンをリストする検索見出し、続いて一致回数が表示されます。検索見出しは Source Insight によって解析され、それぞれの検索パターンは検索結果ウィンドウの左側のシンボル ウィンドウに表示されます。

たとえば、下記の検索見出し、

```
---- GetStockObject (23) -----
```

は一致回数が 23 であることを意味します。

検索結果ウィンドウは、実際に編集可能なもう 1 つのファイル バッファです。ウィンドウから行を削除して結果を削除することもできます。

第 4 章

Source Insight の概念

この章では、Source Insight の概念と機能について説明します。少し時間をかけて読むことで、Source Insight に関する多くの情報が得られます。

この章を読むと、Source Insight の機能について詳しく知ることができます。Source Insight のコマンドの詳細は、第 5 章を参照してください。

プロジェクト

プロジェクトとはソース ファイルの集まりです。

Source Insight はプロジェクト単位で構成されます。プロジェクトとはソース ファイルの集まりです。Source Insight は、プロジェクトのファイル データベースにプロジェクトに含まれるファイルを記録します。

新規ファイルを作成した場合、ファイルは保存時にプロジェクトに追加されます。プロジェクトのディレクトリまたはサブディレクトリに新規ファイルが表示された場合、**[Synchronize Files]** コマンドを実行するか、Source Insight がバックグラウンドで自動的に同期するように設定することで、新規ファイルをプロジェクトに自動的に追加することもできます。

プロジェクトを開くと、Source Insight の一部の操作が変更または拡張されます。たとえば、プロジェクト ウィンドウでは、ディレクトリに関係なく、プロジェクトのファイルがすべてリストされます。

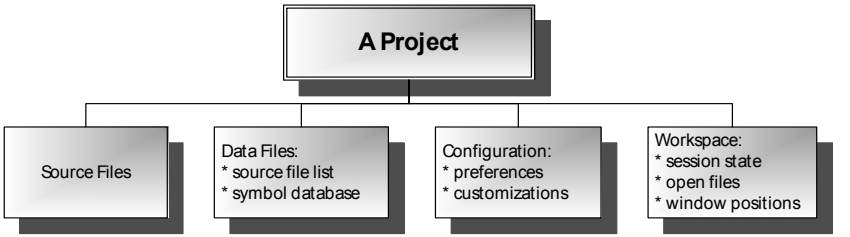


図 4.1 Source Insight プロジェクトのコンポーネント

プロジェクトにはシンボル データベースが含まれています。

プロジェクトには Source Insight が管理しているシンボル データベースが自動的に含まれます。プロジェクトにソース ファイルを追加する場合を除いて、別のタグ ファイルを生成する必要はありません。Source Insight が自動的にタグ ファイルを生成します。

各プロジェクトには、個々のセッション ワークスペースがあります。ワークスペースには、現在開いているファイルのリストやウィンドウの位置のようなセッション情報が含まれます。

プロジェクトには個々の設定情報があります。単一のグローバル設定を使用することもできます。設定には、[Options] メニューから設定した内容を含む、カスタマイズした情報が含まれます。

このマニュアルは、特に明記されていない限り、プロジェクト ファイルが開かれていることを前提として記述されています。プロジェクトを開いている場合と開いていない場合でコマンドの動作が異なる場合は、その点を説明します。

カレント プロジェクト

カレント プロジェクトとは Source Insight のインスタンスで開いているプロジェクトのことです。

開いているプロジェクトがあれば、それをカレント プロジェクトとして参照します。同時に開くことのできるプロジェクトは 1 つのみですが、Source Insight の異なるインスタンスを実行して、各インスタンスで異なるプロジェクトを開くことは可能です。

Source Insight は他のプロジェクトを開き、シンボル宣言を検索することがあります。しかし、Source Insight の各インスタンスで同時に開くことのできるプロジェクトは 1 つのみです。

プロジェクトの機能

Source Insight のプロジェクトにはいくつか重要な機能があります。

- プロジェクトは論理的にファイルを関連付けます。
- 開くファイルを指定する場合、ファイルのドライブやディレクトリを指定する必要はありません。Source Insight は、プロジェクト ファイルが異なるディレクトリやドライブにある場合でも、すべての場所を認識します。129 ページの「コマンド ラインの構文」も参照してください。
- Source Insight は、プロジェクトのすべてのシンボル宣言に関するデータを含むシンボル データベースを保持しています。Source Insight を使って、シンボルを素早く検索することもできます。ソース ファイルが保存されると、シンボル データベースは自動的にインクリメンタルに更新されます。この結果、Source Insight は常にシンボルの位置を把握しています。たとえば、ソース管理システムによってファイルが変更されると、Source Insight はそれらのファイルをプロジェクト シンボル データベースと自動的に同期させます。
- Source Insight はコール ツリー、参照ツリー、クラス階層のように、プロジェクトのシンボル リレーションシップを表示できます。
- Source Insight は、参照インデックスを使用して、シンボル参照のプロジェクト全体の検索を高速化します。参照インデックスは、ファイルを編集または保存すると、インクリメンタルに更新されます。
- 各プロジェクトには、個々のセッション ワークスペースがあります。プロジェクトを開くと、すべてのセッション状態はリストアされます。プロジェクトを閉じると、開いていたすべてのファイルは閉じられ、ワークスペースは保存されます。
- 各プロジェクトには、個々の設定ファイルがあります。つまり、各プロジェクトで個々のメニュー、キーボード割り当て、スクリーン カラーのセットを使用できます。

プロジェクトの作成

[Project] > [New Project] コマンドを使用して、新規プロジェクトを作成します。プロジェクト名および Source Insight がプロジェクト データを格納する場所を指定します。250 ページの「New Project」も参照してください。

プロジェクト ディレクトリ

プロジェクトを作成する場合、プロジェクトごとに 2 つのディレクトリを指定する必要があります。

- **プロジェクト データ ディレクトリ** - Source Insight がプロジェクト データ ファイルを格納する場所です。たとえば、.pr ファイルはここに格納されます。デフォルトでは、新規プロジェクトを作成する場合、Source Insight は「My Documents\Source Insight\Projects」フォルダにプロジェクト データ ディレクトリを作成します。
- **プロジェクト ソース ディレクトリ** - プロジェクト ソース ファイルの場所です。以前のバージョンの Source Insight では、*プロジェクト ルート ディレクトリ*と呼ばれていました。

これら 2 つの別のフォルダを使用することで、Source Insight のデータをソース ファイルと別の場所に格納できます。さらに、Source Insight のプロジェクト データ ファイルは個々のユーザー データ エリアに保存されるため、同じマシンの他のユーザーからアクセスできません。しかし、両方のフォルダで同じ場所を使うこともできます。

プロジェクト ソース ディレクトリの場所を編集するには、**[Project Settings]** コマンドを使用します。263 ページの「Project Settings」も参照してください。

プロジェクト ソース ディレクトリ

プロジェクト ソース ディレクトリは、ソース ファイルの「ホーム」ディレクトリと見なす場所のことです。

プロジェクト ソース ディレクトリは、ソース ファイルのメインのディレクトリと見なす場所のことです。プロジェクト ソース ディレクトリは、ほとんどのソース ファイルを含む最上位のディレクトリまたは「ルート」ディレクトリです。プロジェクトの「ホーム」ディレクトリと見なされます。Source Insight は、このディレクトリに関連するプロジェクト ファイル名を正規化します。(71 ページの図 4.2 を参照してください)。デフォルトでは、Source Insight は、プロジェクト ソース ディレクトリとプロジェクト データ ディレクトリを同じにします。プロジェクトを作成した後、**[Project Settings]** コマンドを使用して、ソース ディレクトリをディスク上の任意の場所に設定できます。

プロジェクトの場所の選択についての詳細は、250 ページの「プロジェクトを保存する場所」を参照してください。

いったんプロジェクトが指定したディレクトリに作成されると、ネットワーク ドライブや UNC パスを含む任意のディレクトリとドライブからファイルを追加できます。

たとえば、ゲーム プログラムのプロジェクトを作成する場№・ 痰 … あげます。ソース ファイルをカテゴリ別に分類し、それぞれのカテゴリ

別にディレクトリを作成するとします。必要なディレクトリを作成し、ルート ディレクトリが C:\Game であるプロジェクトを作成します。

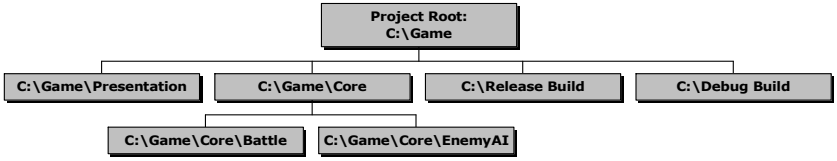


図 4.2 プロジェクト ソース ツリーのサンプル。

ゲーム プログラムのプロジェクト ソース ディレクトリは C:\Game です。「Presentation」、「Core」ディレクトリおよび「Core」のサブディレクトリにソース コードがあります。Source Insight プロジェクトには、これらのすべてのディレクトリのファイルが含まれます。

ファイル名の正規化

Source Insight がファイル名を表示し、そのファイルがプロジェクトの一部である場合、名前とパスを分かりやすいものに変更し、他のすべてのディレクトリ パスと重複しないベース ファイル名をつけます。この処理はファイル名の正規化と呼ばれます。多くのプロジェクトはファイルを複数のサブディレクトリに分けて保存しているため、この処理はとても重要です。また、ディレクトリ ツリーをフラットにすると、ファイル名の最も重要な部分の入力や選択がとても簡単になります。

正規化されたファイル名は、常にリーフ ファイル名で始まり、括弧付きのディレクトリ パスが続きます。

正規化されたファイル名は、常にリーフ ファイル名で始まり、括弧付きのディレクトリ パスが続きます。ファイルが同じドライブにある場合、ディレクトリ パスはプロジェクトのソース ディレクトリに対する相対パスです。ファイルが同じドライブにない場合、またはプロジェクト ソース ディレクトリ ツリーの一部ではない場合、フル パスが括弧付きで表示されます。

プロジェクト ウィンドウで正規化されたファイル名を参照しない場合、**[Project Window Properties]** コマンドを使用し、**[File Directory]** ボックスをチェックしてディレクトリ名のリストに別の列を追加することで機能をオフにできます。

前述のゲーム プロジェクトを使用した例をいくつか示します。

ファイル パス :	表示されるファイル名 :
C:\Game\File.c	File.c
C:\Game\Core\File.c	File.c (Core)
C:\Game\Core\EnemyAI\File.c	File.c (Core\EnemyAI)
C:\SomeDir\File.c	File.c (C:\SomeDir)
D:\OtherDir\File.c	File.c (D:\OtherDir)

プロジェクト リスト

プロジェクト リストには、最近作成したプロジェクトのリストが含まれます。

プロジェクトを作成すると、Source Insight はプロジェクト リストでプロジェクトを追跡します。プロジェクト リストは 1 つのみで、Source Insight を最初に実行するときに作成されます。ファイルの名前は Projects.db3 で、「My Documents\Source Insight\Projects」ディレクトリに作成されます。プロジェクト リストにはプロジェクトが作成されたディレクトリを含む、作成または開かれたすべてのプロジェクトの名前が格納されます。

プロジェクトにファイルを追加する

いったんプロジェクトを作成して開いたら、プロジェクトにソース ファイルを追加する必要があります。2 つの方法があります。

Source Insight で新規ファイルを作成して保存すると、カレント プロジェクトにファイルを追加するかどうか確認するメッセージが表示されます。新規コードを記述して新規ソース ファイルを作成している場合、この方法が最も一般的です。

[Add and Remove Project Files] コマンドを使って、プロジェクトにファイルを追加します。

既存のソース ファイルがあり、カレント プロジェクトにこれらのファイルを追加する場合は、**[Add and Remove Project Files]** コマンドを使用します。このコマンドを使用すると、ディレクトリ ツリー全体を含む既存のファイルを、ディスクの任意の場所からカレント プロジェクトに追加できます。詳細は、153 ページの「Add and Remove Project Files」および 157 ページの「Add File List」を参照してください。

プロジェクトにファイルを追加すると、以下のような影響があります。

- ファイル名がプロジェクトのファイル名データベースに格納されます。Source Insight でファイルのリストを表示するときに、ファイル名がリストに含まれます。たとえば、[Open] コマンドを使用すると、ファイル名がリスト ボックスに表示されます。
- ファイルは言語形式に基づいて解析されます。シンボル定義はプロジェクトのシンボル データベースに追加されます。各ファイルで使われる言語パーサーはドキュメント タイプにより決定されます。92 ページの「ドキュメント タイプ」も参照してください。
- ファイルが Source Insight 以外で (たとえば、ソース管理システムによって) 変更された場合、Source Insight がプロジェクト シンボル データベースを同期できるように、ファイルの変更日付がファイル名データベースに記録されます。
- ファイル名の表示方法が変更されます。ファイル名はプロジェクトのソース ディレクトリに正規化されます。
- ファイルは、コール ツリーのようにシンボル リレーションを表示するとき検索されるプロジェクト コード ベースの一部になります。

プロジェクトからファイルを削除する

カレント プロジェクトのファイル リストからファイルを削除するには、**[Project] > [Add and Remove Project Files]** コマンドを使用します。ファイルがプロジェクトから削除されると、ファイル内のすべてのシンボルはプロジェクトのシンボル データベースから削除されます。Source Insight はディスクからファイルを実際に削除しません。153 ページの「Add and Remove Project Files」も参照してください。

プロジェクトを閉じる

カレント プロジェクトを閉じるには、**[Close Project]** コマンドを使用します。プロジェクトを閉じるとプロジェクトは中断されます。このため、Source Insight は編集したファイルを保存するかどうか確認メッセージを表示した後、すべてのファイルを閉じます。実際に中断する代わりに、Source Insight は何もプロジェクトを開かないで続行します。

プロジェクトを開く

プロジェクトを開くには、[Open Project] コマンドを使用します。

異なるプロジェクトを開くには、**[Project] > [Open Project]** コマンド (252 ページの「Open Project」も参照してください。) を使用します。Source Insight では同時に開くことができるのは 1 プロジェクトのみなので、既に開いているプロジェクトがある場合、カレント プロジェク

トを閉じるかどうか確認するメッセージが表示されます。カレント プロジェクトを閉じると、選択可能な既存プロジェクトのリスト ボックスが表示されます。

プロジェクトが開かれている場合、プロジェクトの設定ファイルとワークスペース ファイルがロードされます。この結果、表示、メニュー、キーストロークの設定が変更され、以前のセッションで開かれていたファイルが再び使用されます。

プロジェクトが開かれている場合、現在の作業ディレクトリはプロジェクトのソース ディレクトリに変更されます。

プロジェクトの削除

プロジェクトを削除するには、**[Remove Project]** コマンドを使用します。このコマンドは、Source Insight が作成してプロジェクトと関連付けたプロジェクト データ ファイルをすべて削除します。ソース ファイルは削除 *されません*。281 ページの「Remove Project」も参照してください。

プロジェクト設定の変更

[Project Settings] コマンドを使用してプロジェクトのインデックス オプションを設定します。

[Project Settings] コマンドを使用して、カレント プロジェクトを管理するさまざまなオプションを設定できます。開いているプロジェクトがない場合、[Project Settings] コマンドはデフォルトのオプションを設定します。次回以降作成されるプロジェクトでは、このオプションが使用されます。

新規プロジェクトを作成すると、[Project Settings] ダイアログ ボックスが表示されます。

プロジェクトで個々の設定を使用するか、グローバル設定ファイルを使用するかを指定できます。プログラムのソース ディレクトリの場所やインデックスするシンボル情報の種類を指定することもできます。

263 ページの「Project Settings」も参照してください。

チームでの作業

Source Insight はチームでの作業で効率的に動作するように設計されています。チームのプログラマがコード ベースを編集すると、Source Insight は自動的に編集を認識し、シンボル情報を更新します。また、多くの新規コードをプロジェクトに追加したり、モジュール間で移動した場合も、Source Insight は行われた処理を追跡します。

ネットワークの使用

ローカル ファイルをプロジェクトに追加することをお勧めします。

Source Insight は、各チーム メンバが個々のローカル マシンでプロジェクト ソース ファイルのコピーを使用した場合に最も効果的に動作します。通常、ソース コードの「マスタ」コピーは中央サーバーで管理されます。

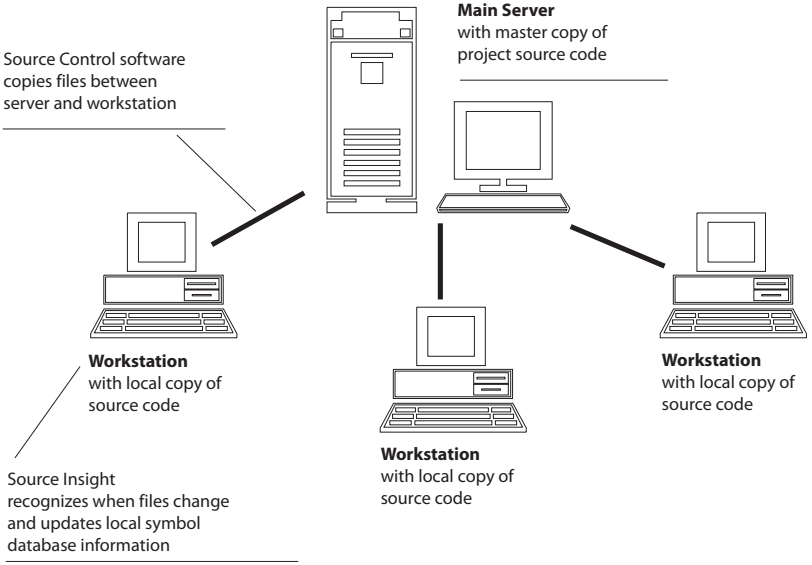


図 4.3 典型的なネットワークにおけるソースの管理。

プロジェクトにリモート ファイルを追加する

ファイルのローカル コピーではなく、プロジェクト サーバーのソース ファイルに直接アクセスする必要がある場合もあります。もちろん、ネットワーク上のすべてのファイルを直接、開くことが可能です。しかし、別のユーザーがファイルを開いている場合、ファイルがロックされるか、ファイルの競合が生じる可能性があることを常に注意してください。また、リモート ファイルをプロジェクトに追加しない場合、Source Insight のプロジェクト機能は十分に活用できません。

必要な場合、リモート ドライブからプロジェクトにソース ファイルを追加できます。

プロジェクトに直接サーバーのファイルを参照させる 1 つの方法は、ワークステーション上にプロジェクトをローカルに作成し、[Add and Remove Project Files] コマンドを使用して、リモート サーバーからプロジェクトにファイルを追加する方法です。この方法では、Source Insight のシンボル データベース ファイルはマシン上にローカルに格納されますが、ソース ファイルはサーバーにあります。153 ページの「Add and Remove Project Files」も参照してください。

[Project Settings] コマンドを使用して、サーバー上のリモート ソース コード ディレクトリをプロジェクト ソース ディレクトリとして指定

する必要があります。この方法では、ファイルはローカル プロジェクトのデータ ファイル ディレクトリではなく、メイン ソース ディレクトリに大して相対的に表示されます。263 ページの「Project Settings」も参照してください。

ソース管理の使用

典型的なネットワーク環境では、各開発者はワークステーション上に、個々のローカル ソース ファイルを持っています。開発者がファイルを変更する場合、最初にソース管理ソフトウェアを使用してファイルを「チェックアウト」します。ファイルがチェックアウトされたら、ローカル ファイルを編集します。変更を反映する場合は、ソース管理ソフトウェアを使用して変更したファイルを「チェックイン」します。開発者のローカル マシンからメイン プロジェクト サーバーにファイルがコピーされます。

別の開発者がソース ファイルの「最新」バージョンを取得するように設定している場合、ソース制御ソフトウェアを使用して、その開発者のローカル ディレクトリとメイン プロジェクト サーバーを「同期」させます。プロジェクト サーバーから開発者のローカル ワークステーションに最新ファイルがコピーされます。

Source Insight は、各ファイルの「更新日時」タイムスタンプとファイルのサイズを確認して、ファイルの変更を認識します。ファイルの変更が検出されると、ファイルを再解析して、開発者のローカル マシン上の Source Insight シンボル データベースを更新します。

ソース管理コマンド

ソース管理を行う、標準のカスタム コマンドです。

Source Insight には、ソース管理を行ういくつかの標準的なカスタム コマンドが用意されています。[Options] > [Custom Commands] ダイアログ ボックスで、コマンドを編集できます。デフォルトでは、コマンドは Microsoft Visual Source Safe に合わせて設定されています。しかし、他のソース管理システムやバージョン管理システムをサポートするように簡単に変更できます。

ソース管理コマンドは、下記の表のとおりです。各コマンドの正確な意味は、[Custom Commands] ダイアログ ボックスでのセットアップ方法に基づきます。176 ページの「Custom Commands」も参照してくだ

さい。Source Insight では、以下のコマンド セマンティクスを定義しています。

表 4.1: ソース管理コマンド

コマンド名	動作
Check Out	ファイルを編集できるように、ソース管理プロジェクトのカレント ファイルをチェックアウトします。
Check In	カレント ファイルをソース管理プロジェクトにチェックインします。ファイルを編集した後、他のチーム メンバがアクセスするメイン ソース管理プロジェクトに戻すには、[Check In] コマンドを使用してください。
Undo Check Out	チェックアウトの動作を元に戻します。ファイルのチェックインは行いません。
Sync to Source Control Project	ソース管理プロジェクトに対して最新になるように、ローカル プロジェクトのすべてのファイルを更新します。
Sync File to Source Control Project	ソース管理プロジェクトに対して最新になるように、カレント ファイルを更新します。

ソース管理ツールバー

ソース管理ツールバーには、ソース管理コマンドごとに別のボタンが存在します。

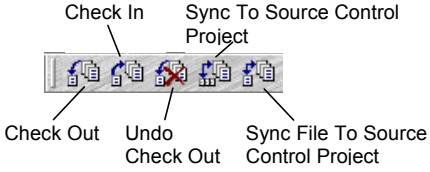


図 4.4 ソース管理ツールバー

シンボルおよびプロジェクトについて

Source Insight は、ソース ファイルの編集中に、シンボル定義を動的に解析します。シンボル情報は、プロジェクトに必要不可欠な、インデックスされたシンボル データベースのディスク上に格納されます。

シンボル データベースは、ファイルが変更されると自動的に更新されます。

シンボル データベースは、ファイルを開いて保存するたびにインクリメンタルに更新されます。他のプロジェクト チーム メンバが変更したファイルは、バックグラウンドでシンボル データベースと自動的に同期されます。

ソース ファイルの解析に使用する言語

各ファイルで使用される言語パーサーは、ドキュメント タイプにより決定されます。(92 ページの「ドキュメント タイプ」も参照してください。) Source Insight の言語パーサーは、宣言を幅広く認識します。

Source Insight は、洗練された、エラーに寛容なパターン マッチング パーサーを使用して、ソース ファイルのシンボル宣言を検索します。これは、「ctags」のような、他のバッチ ツールよりはるかに優れています。[Preferences: Languages] コマンドを使用して [Properties] ボタンをクリックすることで、ファイルからシンボルを解析する際に使用するカスタム正規表現パターンを指定することもできます。

シンボルの命名

Source Insight では、シンボル名は「ドット付きパス」として格納されます。ドット付きパスには、シンボルのコンテナ名、ドット (.)、シンボルの名前が含まれます。たとえば、クラスのメンバは下記のようになります。

```
MyClass.member
```

別のシンボルの内部で宣言を入れ子にするすべてのシンボルは、ドット付きパスを使用します。プロジェクト ウィンドウにリストされたシンボルを見ると、ドット付きパスが見つかります。メンバの宣言にスコープ解決演算子 (::) を使用する C++ のような言語でも、シンボル名はドット付きパスとして内部的に格納されます。

シンボルのフル ネームを入力するときにコンテナも指定する場合は、シンボル名にドットを使用する必要があります。

メモ： シンボルの名前には、組み込みドット (.) 文字を使用できます。Source Insight は、組み込みドット文字とドット付きパスのドット文字を混同しないようにシンボル名を保管します。

シンボル データベースの更新

プロジェクトにファイルが追加されるか、ファイルが保存されると、Source Insight はファイルで定義されているシンボルを判断して、ディ

スク上に格納されているプロジェクト シンボル データベースをインクリメンタルに更新します。

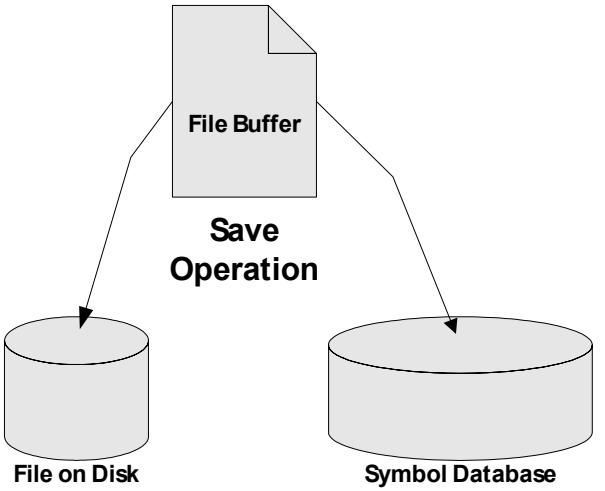


図 4.5 保存操作は、ディスク上のソース ファイルとシンボル データベースを更新します。

シンボル形式のファイル名

Source Insight ではファイル名はシンボルとして扱われます。したがって、シンボルが指定できるすべての場所でファイル名を指定できます。たとえば、任意のファイル参照ダイアログ ボックスでファイル名 (および拡張子) を入力してファイルを開くことができます。#include ステートメントのファイル名をクリックし、[Jump To Definition] コマンドを使用してファイルを開くこともできます。

プロジェクト ファイルの同期

同期とは、シンボル データベースをすべての変更したファイルに合わせて更新することを意味します。

Source Insight を使用しないで、ファイルを編集することがあります。たとえば、ソース管理システムを使用してファイルを更新した場合や、ビルド プロセスによりファイルが生成された場合などです。この場合、Source Insight はプロジェクトを最新の状態にするため、日付が古いファイルを再解析します。この処理はプロジェクトの同期と呼ばれます。通常、この処理はバックグラウンドで自動的に行われます。

この同期処理により、プロジェクト全体が最新であることが保証されます。プロジェクトの各ファイルをスキャンし、Source Insight が最後にファイルを開いた後に変更されたファイルのシンボル データベースを更新します。存在しなくなったプロジェクトの一部であるファイルはプロジェクトから削除されます。オプションで、同期処理により新規ファイルをプロジェクトに自動的に追加することもできます。

以下の 2 つのうち 1 つの方法でプロジェクトを同期できます。

- **[Preferences: General]** ダイアログ ボックスで [background synchronization] オプションをオンにします。Source Insight で引き続き編集を行うと、プロジェクトの同期処理がバックグラウンドで行われます。このオプションは、デフォルトでオンです。209 ページの「General Options」も参照してください。
- **[Synchronize Files]** コマンドを実行して、プロジェクトをオンデマンドに同期します。319 ページの「Synchronize Files」も参照してください。

通常、Source Insight が最後にファイルを開いた後に変更されたファイルを開くと、シンボル データベースが自動的に更新されます。自動同期では、更新作業はユーザーには見えません。しかし、シンボルがあるファイルから別のファイルに移動された場合、両方のファイルがプロジェクトと同期していない限り、Source Insight はシンボルの場所の追跡に失敗します。また、新規シンボルが定義された場合、ファイルが同期されるまで Source Insight は新規シンボルを把握できません。

コモン プロジェクトの使用 : プロジェクト シンボル パス

Source Insight は、カレント プロジェクトで宣言を見つけられなかった場合、プロジェクト シンボル パスにあるすべてのプロジェクトを検索します。

複数のプロジェクトで利用するライブラリのヘッダー ファイルのセットを使用することがあります。これらのファイルを各プロジェクトに追加することは可能ですが、冗長になります。より良い解決法は、コモン ヘッダー ファイルのセットごとに、1 つのプロジェクトを作成することです。これらのプロジェクトはパスに保存できます。Source Insight はプロジェクトでシンボルが検索できない場合は常に、パスを検索します。このパスはプロジェクト シンボル パスと呼ばれます。

Source Insight は、シンボルを検索するとき、現在開いているファイル、プロジェクト シンボル データベース、プロジェクト シンボル パスのプロジェクトを検索します。

[Preferences: Symbol Lookups] コマンドを使用して、プロジェクト シンボル パスを編集できます。315 ページの「Symbol Lookup Options」も参照してください。

プロジェクト シンボル パスは、Source Insight がシンボルの検索に使用するプロジェクトのリストです。プロジェクト シンボル パスは [Preferences: Symbol Lookups] ダイアログ ボックスで指定します。プロジェクト シンボル パスを使用すると、小さな自己完結型のプロジェクトが作成できますが、他のプロジェクトのシンボルを検索することもできます。ベース プロジェクト (81 ページの「ベース プロジェクト」も参照してください。) は Source Insight がシンボルを検索する最後の場所です。プロジェクト シンボル パスの最後に暗黙的に指定されます。

プロジェクト シンボル パスは、シンボル定義を検索する ¾ めにのみ使用されます。プロジェクト シンボル パスは、シンボル参照の検索、

[Search Files] および [Smart Rename] コマンドでは使用されません。これらの操作は、カレント プロジェクトのファイルに対してのみ有効です。

プロジェクト シンボル パスの検索

プロジェクト シンボル パスは、シンボル定義の検索に使用されます。

Source Insight は、開いているすべてのファイルとカレント プロジェクトのシンボル データベースを検索します。シンボルが見つからない場合、プロジェクト シンボル パスにあるすべてのプロジェクトを検索します。それでもシンボルが見つからない場合、ベース プロジェクトを検索します。

指定した名前で複数のシンボルが見つかった場合、リストからシンボルを選択します。

[Preferences: Symbol Lookups] ダイアログ ボックスの **[Always search symbol path]** オプションを指定できます。このオプションがオンの場合、シンボルが開いているファイルやカレント プロジェクトで既に見つかっている場合でも、Source Insight がシンボルを参照するたびに、シンボル パスのすべてのプロジェクトが検索されます。冗長な (または似た名前の) シンボルがプロジェクトまたはプロジェクト シンボル パスのプロジェクトにあるかどうか確認する場合は、このオプションをオンにしてください。

プロジェクトを開かないで作業する

プロジェクトが開かれていない場合、Source Insight は現在開いているファイルを検索します。シンボルが見つからない場合、プロジェクト シンボル パスにあるすべてのプロジェクトを検索します。それでもシンボルが見つからない場合、ベース プロジェクトを検索します。

ベース プロジェクト

ベース プロジェクトは最後の手段として検索されます。

Source Insight を初めて実行すると、Base という名前のプロジェクトが自動的に作成されます。ベース プロジェクトは Source Insight がシンボルを検索する最後の場所です。ベース プロジェクトは、プロジェクト シンボル パスの最後に暗黙的に指定されます。つまり、Source Insight は、開かれているファイル、カレント プロジェクト、プロジェクト シンボル パスのすべてのプロジェクトでシンボルを見つけられない場合、ベース プロジェクトを検索します。

メモ： プロジェクト シンボル パスにベース プロジェクトを追加する必要はありません。Source Insight はベース プロジェクトがリストに追加されているものとして自動的に検索します。

ベース プロジェクトは、コモン シンボルを保存する場所として適しています。ベース プロジェクトに格納されているすべてのシンボルは、他のプロジェクトから参照可能です。たとえば、標準の C/C++ インクルード ファイルをベース プロジェクトに追加可能です。ベース プロジェクトは Source Insight のエディタ マクロ ファイルを追加する場所としても適しています。

プログラミング言語

Source Insight は、言語の抽象化を使用して各種プログラム言語の特徴をカプセル化します。

ファイルのドキュメント タイプは言語を決定します。

指定されたファイル バッファについて、ファイルの名前がドキュメント タイプを決定します。ドキュメント タイプはその言語を決定します。ドキュメント タイプとファイル名および言語の関連付けは、[Options] > [Document Options] ダイアログ ボックスを使用して行います。

Source Insight は、構文フォーマットでソース コードを表示し、コードからシンボル定義を解析し、プロジェクト シンボル データベースにシンボル情報を保存することにより、言語をサポートしています。

解析はファイルのドキュメント タイプと関連付けられた言語タイプによって制御されます。

シンボル宣言の解析はファイルのドキュメント タイプと関連付けられた言語タイプによって制御されます。

現在サポートしている言語のリストを参照するには、**[Options] > [Preferences: Languages]** ダイアログ ボックスを使用します。 新規言語は、プログラム更新時にリストに追加されます。

Source Insight の言語は、ビルトイン言語とカスタム言語の 2 つのカテゴリに分類されます。

ビルトイン言語

Source Insight には、C/C++、Java、IDL/ODL、Perl、C#、ASP、Visual Basic、その他を含むさまざまな言語向けにビルトインの最適化サポートが含まれています。

ほとんどのビルトイン言語は、リファレンスの検索やコール ツリーの生成のような追加機能をサポートしています。

カスタム言語

[Preferences: Languages] ダイアログ ボックスを使用してカスタム言語のサポートを Source Insight に追加することもできます。カスタム言語は、構文規則、構文フォーマット キーワード、および解析表現を指定

するシンプルなジェネリック言語です。228 ページの「Language Options」も参照してください。

カスタム言語はエクスポートおよびインポート可能です。

カスタム言語を追加するには

新規言語のサポートの追加は、2 ステップのプロセスです。

1. **[Options] > [Preferences: Languages]** を使用して言語を追加します。228 ページの「Language Options」を参照してください。
2. **[Options] > [Document Options]** を使用して言語を参照するドキュメント タイプを追加します。92 ページの「ドキュメント タイプ」を参照してください。

メモ： [Document Options] ダイアログでは、新規言語を追加する代わりに、カスタム解析パターンをドキュメント タイプのプロパティに直接追加することもできます。しかし、新規カスタム言語を追加し、新規言語を示すドキュメント タイプを使用することにより、より優れたコントロールが可能になります。

コマンド リファレンスの言語オプション トピックには、カスタム言語プロパティの詳細情報が含まれています。

.NET Framework のサポート

C# ファイルを編集する場合、Source Insight は .NET Framework クラス ライブラリ シンボルのシンボル補完を実行できます。シンボル補完は、マシン上の .NET Framework プロジェクトを維持することにより達成されます。

.NET Framework クラス ライブラリ シンボルは NetFramework プロジェクトに格納されます。

.NET Framework クラス ライブラリ シンボルは Source Insight が作成した **NetFramework** プロジェクトに格納されます。Source Insight は、ユーザーのプロジェクト フォルダの NetFramework フォルダにプロジェクトを保存します。

Source Insight は、.NET Framework クラス ライブラリのシンボルを宣言するマスタ「ソース ファイル」のセットもインストールします。これらのソースは、Source Insight プログラム フォルダの NetFramework フォルダに保存されます。マシンごとに 1 つのコピーがあります。これらの「ソース ファイル」は C# 構文のマシン生成ファイルです。しかし、これらのソース ファイルは厳密な C# 互換ではありません。ファイルの内容は、Source Insight の新しいバージョンでは変更されることがあります。

Source Insight で NetFramework プロジェクトを作成するには、**[Setup Common Projects]** コマンドを使用するか、**[Preferences: Symbol Lookups]** ダイアログ ボックスで **[Create Common Projects]** ボタンをクリックします。

HTML の使用

Source Insight には HTML 言語を制御する特別な機能があります。HTML 言語パーサーはファイルをスキャンして構造タグを検索します。これらのタグは、各ソース ウィンドウの左側のシンボル ウィンドウに表示されます。プロジェクト シンボル データベースに追加されるシンボル タイプは「TITLE」タグのみです。HTML ファイルでは下記のように記述されます。

```
<TITLE>Programming Wisely</TITLE>
```

この場合、「TITLE: Programming Wisely」のようにシンボル データベースに追加されます。シンボルの最大長は 79 文字です。

HTML と ASP の複合言語の使用

HTML と ASP は、組み込みスクリプトを含む複合言語です。

HTML 言語は組み込みスクリプトもサポートしています。サーバーおよびクライアントサイド スクリプト ブロックは、すべてのスクリプト言語で適切な構文形式を使用して表示されます。スクリプトで定義された要素もシンボル ウィンドウに表示され、シンボル データベースに保存されます。

[Options] > [Preferences: Languages] ダイアログ ボックスで、HTML または ASP で使用するデフォルトのスクリプト言語を指定できます。**[Special]** ボタンをクリックし、デフォルトの言語を選択します。

Java 言語の編集

標準 Java パッケージでシンボルの自動補完を使用するには、マシン上にパッケージのソース コードがインストールされている (またはネットワーク ドライブ経由で利用できる) 必要があります。ソースを参照する JavaStandard コモン プロジェクトも必要です。

JavaStandard コモン プロジェクトを作成するには

1. **[Options] > [Preferences: Symbol Lookups]** ダイアログ ボックスで、**[Create Common Projects]** ボタンをクリックします。
2. **[Common Projects]** ダイアログ ボックスで、**[Standard Java Libraries]** チェック ボックスをクリックし、その隣にある **[Browse]** ボタンをクリックします。ソース コードを含むディレクトリ (通常は JDK Java/Src サブディレクトリの 1 つ) に移動します。ディレクトリを選択し、**[OK]** をクリックします。
3. **[Common Projects]** ダイアログ ボックスで **[Continue]** をクリックします。[Save As] ダイアログ ボックスが表示されます。JavaStandard プロジェクトの名前と保存先を指定します。JDK またはそのサブディレクトリの 1 つと同じディレクトリに保存してください。**[Save]** をクリックします。
4. **[Project Settings]** ダイアログ ボックスが表示されたら、**[OK]** をクリックします。
5. **[Add and Remove Project Files]** ダイアログ ボックスが表示されます。**[Add Tree]** をクリックし、関連するファイルをすべて JavaStandard プロジェクトに追加します。**[Close]** をクリックします。以前のプロジェクトに戻ります。

ファイルにインポートされたパッケージで、自動補完が有効になりました。

C/C++ 言語の機能

Source Insight には、プログラミングに C/C++ を使用している場合に考慮すべきいくつかの特別な機能が用意されています。

非アクティブ コードの処理 - ifdef のサポート

ifdef は [Edit Conditions] ダイアログ ボックスで条件値が指定された場合に有効になります。

Source Insight の C/C++ およびリソース ファイル パーサーは、#ifdef、#if、および #elif ディレクティブを使用してコンパイル時に無効にされる非アクティブ コードのブロックを認識します。**[Options > Preferences: Language]** ダイアログ ボックスには、既知の条件付き定数のリストを編集できる [Conditions] ボタンがあります。ソース ウィンドウの右クリック メニューの **[Edit Condition]** コマンドを使用して条件を編集することもできます。

デフォルトでは、Source Insight は条件付きディレクティブをすべて無視して、条件付きコンパイル構造のすべての分岐を処理します。通常、条件付き分岐の宣言は互いに干渉しないので、処理は正常に行われます。

しかし、宣言がわかりにくい場合、#ifdef が正しく解釈されないことがあります。この場合、Source Insight で処理が正常に行われません。例 :

```
void DoThing(
    int param1,
#ifdef ABC
    int param2)
#else
    int param2, param3)
#endif
```

さらに、非アクティブ コードは通常は参照しません。これらの理由から、Source Insight は条件値を指定します。

非アクティブ コードのブロックは、「Inactive Code」スタイルで表示されます。例 :

```
#ifdef NEVER
        TluParentRefFromSrl(hsrl, ist, &tlu);
        tlu.hdoc = hdocNil;
#endif /* NEVER */
        return CmdGotoTlu(&tlu) == cmdOK;
```

図 4.6 「Inactive Code」スタイルが表示されます。

非アクティブ コードは Inactive Code スタイルで表示されます。

条件付き解析

条件付き解析は、Source Insight で条件付きコンパイルをサポートする言語 (C/C++ および Windows リソース ファイル) にのみ適用されます。

Source Insight は 2 種類の条件値リストを管理します。

- **条件リストに基づく設定ファイル**。リストは、カスタマイズ設定を含む、現在の設定ファイルに保存されます。通常は、1 つのグローバル設定ファイルをすべてのプロジェクトで使用します。
- **プロジェクト固有の条件リスト**。プロジェクトごとに保存されます。このため、個々のプロジェクトで異なる条件値を使用できます。たとえば、あるプロジェクトで「RETAIL」定義を使用し、別のプロジェクトで「DEBUG」定義を使用できます。

2 つの条件リストは Source Insight がファイルを解析するときに組み合わされます。プロジェクト固有の条件は、「グローバル」設定の条件よりも優先されます。

条件値

条件値は、#if、#ifdef、#ifndef、および #elif ステートメントの式で使用できます。
例 :

```
#if VER < 3 && DEF_OPEN != 0
    ....
```

この例では、VER と DEF_OPEN の 2 つの条件値が使用されています。条件値は、[Edit Condition] コマンドを使用して指定できます。

条件値にはテキスト値を指定できます。C および C++ では、ゼロの値は「False」、非ゼロの値は「True」と見なされます。

条件値を無視する

変数の値を指定しない場合、その変数を含むすべてのステートメントはスキップされ無視されます。下記は #if-type ステートメントのデフォルトの動作です。

例 :

```
#if VER < 3 && WINVER >= 5
    int a = 1;
#else
    int a = 2;
#endif
```

VER と WINVER の両方が [Edit Condition] コマンドで定義されている場合、#if ステートメントの式が評価され、1 つの分岐のみがアクティブになります。しかし、これらの変数のいずれかが Source Insight で定義されていない場合、*両方の*分岐がアクティブになります。

条件値の編集

条件付き変数の値を編集するには、変数を右クリックして [Edit Condition] を選択します。条件リストを編集する際、Source Insight はプロジェクト全体を再解析するかどうか確認します。最初に条件リストの変更をすべて行ってから、プロジェクト全体を再解析してください。プロジェクトが再解析されるまで、行った変更は Source Insight のシンボル データベースに格納されているシンボル情報に反映されません。

プリプロセッサ トークン マクロ

トークン マクロを使用してマクロの拡張方法を定義できます。

Source Insight には、ファイルを解析するときに使用する、独自のプリプロセッサが用意されています。プリプロセッサ マクロはトークン マクロと呼ばれます。このマクロは、Source Insight がファイルを解析するときに生じるトークン置換です。この機能により、Source Insight は、Source Insight のパーサーが認識しない特殊言語キーワードを処理したり、特殊な C/C++ プリプロセッサ置換を処理できます。

C マクロは通常、Source Insight によって拡張されません。

Source Insight は、ファイルを解析するときに、C/C++ プリプロセッサ マクロを拡張しません。この結果、特定のプリプロセッサ マクロと定数は Source Insight で正しく処理されません。トークン マクロを使用することで、Source Insight は一部のプリプロセッサ置換を選択して拡張できるようになります。

1 つの例は、シンボルの宣言に使用される標準 COM (または ActiveX) マクロのセットです。たとえば、STDMETHOD(メソッド名) マクロは COM メソッド関数の宣言に使用されます。Source Insight には、このマクロおよびその他のエントリを含むデフォルトの C/C++ トークン マクロ ファイル (c.tom) が含まれています。

トークン マクロ ファイル

トークン マクロ ファイルの拡張子は .tom です。

トークン マクロは拡張子 .tom のファイルにリストされます。グローバル トークン マクロ ファイルは、Source Insight のプログラム ディレクトリに格納されます。プロジェクト固有のトークン マクロは、プロジェクトのデータ ディレクトリに格納されます。プロジェクト トークン マクロ ファイルは、グローバル ファイル、およびプロジェクト マクロ (グローバル マクロよりも優先) と組み合わされます。

トークン マクロの構文

トークン マクロは 1 行に 1 つのトークン マクロで構成されています。トークン マクロのフォーマットは下記のとおりです。

```
macroname<no text here means macro is a no-op>
macronamesubstituted text here
macroname(parameter list)substituted text with parameter names
macroname(parameter)text##parameter // concatenates text
; comments begin with a semicolon
```

トークン マクロの例は下記のとおりです。

```
MyStructure(sname)struct sname
NoOperation
BuildName(name1, name2)name1##name2
```

ビルトイン言語パーサーにはそれぞれ、対応するトークン マクロ ファイルがあります。各言語のトークン マクロ ファイルの名前は下記の表のようになります。

表 4.2: 異なる言語用のトークン マクロ ファイル

言語	ファイル名
C および C++	C.tom – Source Insight に含まれるデフォルトのコピーです。
HTML	Html.tom
Java	Java.tom

表 4.2: 異なる言語用のトークン マクロ ファイル

言語	ファイル名
リソース ファイル	Rc.tom
x86 アセンブリ言語	X86.tom
Perl	Perl.tom

トークン マクロの編集

トークン マクロを変更するには、トークン マクロ ファイルを開いて、変更を行った後で保存します。Source Insight は適切な言語用のトークン マクロが変更されたことを認識します。開いているファイルは自動的に再解析されます。

トークン マクロ ファイルを保存すると Source Insight はマクロを認識します。

トークン マクロ ファイルを編集するとき、Source Insight が開いているファイルを再解析する前に、ディスクに保存する必要があります。Source Insight はプロジェクト全体を自動的に再解析しません。最初にトークン マクロ ファイルの変更をすべて行ってから、[Rebuild Project] コマンドを使用してプロジェクト全体を再解析してください。プロジェクトが再解析されるまで、トークン マクロへ行った変更は Source Insight のシンボル データベースに格納されているシンボル情報に反映されません。

プロジェクト固有のトークン マクロ

プロジェクトは、それぞれトークン マクロ ファイルを持ち、グローバル ファイルとマージします。

プロジェクトは、それぞれ独自のトークン マクロ ファイルを持っています。Source Insight は、トークン マクロ ファイルを自動生成しません。ユーザーが生成します。プロジェクト トークン マクロ ファイルはプロジェクト データ ディレクトリに保存されます。Source Insight がソース ファイルを解析すると、プロジェクト トークン マクロと Source Insight プログラム ディレクトリに保存されているグローバル ファイルをマージします。プロジェクト トークン マクロはグローバルなものを優先します。プロジェクト固有のトークン マクロを追加することにより、個々のプロジェクトごとにトークン マクロの拡張を調整できます。

解析の考慮事項

厳格な言語解析ではなくエラーに寛容なパターン マッチングによってシンボルが見つかることは、重要な意味があります。

- ソース ファイルに構文エラーがあります。構文エラーがあるファイルはコンパイルできません。つまり、コンパイルしていない、または中間状態のソース コードでシンボルが見つかることを意味します。
- プリプロセッサ マクロは Source Insight がコードを解析する前に拡張されません。これは良い意味に聞こえないかもしれませんが、実際はとても良く機能します。Source Insight は、最も抽象的なレベル (記述されたレベルと同じレベル) でプログラムを解析します。このため、コール ツリーに関数形式のマクロを含むことができます。
- シンボル宣言は、コンパイルの時点でアクティブだったものだけではなく、すべてのソース コードから解析されます。たとえば、コンパイル時に #ifdef 分岐がアクティブでない場合でも、#ifdef-#endif 内部の C/C++ コードがシンボル データベースに追加されます。これは、マルチステート プログラムで作業している場合に非常に役に立ちます。コンパイラが参照しているものだけではなく、すべてのケースに注意する必要があります。非アクティブ コードのブロックを省略するには、[Edit Condition] コマンドまたは [Preferences: Languages] ダイアログ ボックスを使用して条件値を定義します。
- コメントと定数のテキストも検索用にインデックスされます。Source Insight は、コンパイラがオブジェクト コードに変換するコードだけでなく、ソース レベルのコードに焦点を当てています。
- すべてのヘッダー ファイルは、すべてのソース ファイルからアクセスできると仮定されます。Source Insight は、ソース ファイルにインクルードされているヘッダー ファイルに注目しません。そのため、すべてのシンボルがすべてのポイントで認識されます。コンパイラがプログラムをどのように参照するかという点については技術的に不正確ですが、ほとんどの場合は問題なく動作します。
- プログラミング スタイルが原因で、シンボルが見つからないことがあります。Source Insight が使用するデフォルトの解析は、ほとんどのプログラミング スタイルで問題なく動作します。Source Insight が認識できない宣言タイプを含む場合は、プリプロセス フェーズを拡張するトークン マクロを追加するか、カスタム解析正規表現パターンを追加します。

解析の結果を良くするコーディングのコツ

コーディング スタイルが解析の正確さに影響することがあります。

通常の C/C++ プリプロセスは Source Insight のパーサーでは実行されないため、新規コードを記述する際に、以下の点に注意する必要があります。

#ifdef-#endif ブロックを個々の宣言に分解しないようにしてください。どうしても宣言が分解される場合は、Source Insight が正しく解析できない非アクティブ コードのブロックが無効になるように、[Edit Condition] を使用して条件値を定義する必要があります。86 ページの「条件付き解析」も参照してください。

例 :

```
void MyFunc
#ifdef XYZ
    (int param1, int param2)
#else
    (long param1, long param2)
#endif
{
    …
}
```

標準言語キーワードや組み合わせを #define で置換しないでください。置換を回避できない場合、置換をサポートするようにトークン マクロを定義する必要があります。87 ページの「プリプロセッサ トークン マクロ」も参照してください。

例 :

```
#define ourpublic public
class D : ourpublic B { … }
```

Source Insight はキーワード `ourpublic` が `public` を意味することを知らないため、問題が発生します。

ドキュメント タイプ

ドキュメント タイプは [Document Options] コマンドで定義されるファイルの分類です。Source Insight はファイルの名前を使用してドキュメント タイプを決定します。

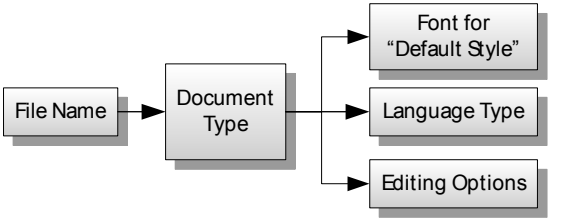


図 4.7 ファイル名はドキュメント タイプを決定します。ドキュメント タイプは、フォント、ファイルの解析と構文フォーマットの表示に使用される言語タイプ、その他の編集オプションを決定します。

新規ドキュメント タイプを定義したりビルトイン タイプを変更するには、[Document Options] コマンドを使用します。194 ページの「Document Options」も参照してください。

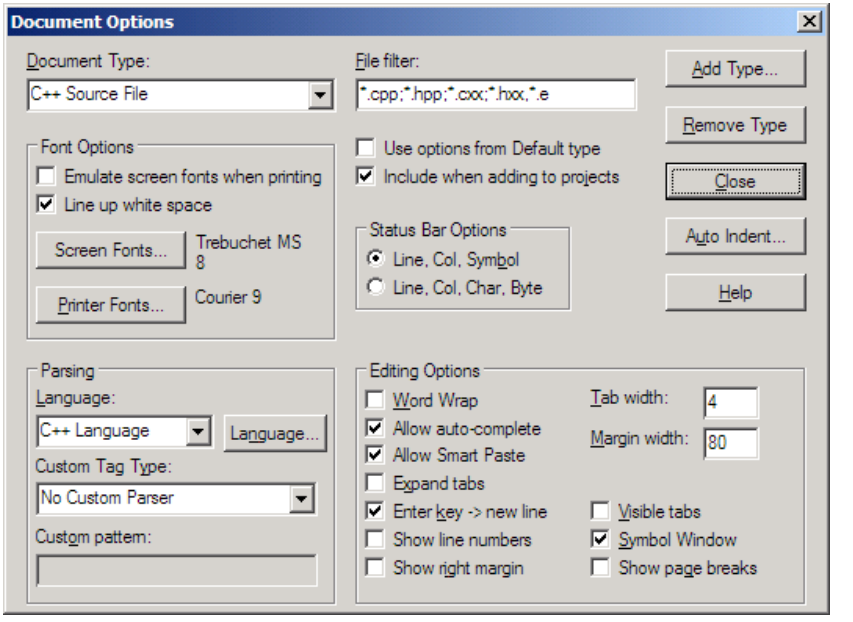


図 4.8 [Document Options] ダイアログ ボックス

ドキュメント固有のオプション

ドキュメント タイプは、ファイル タイプの言語と編集オプションを決定します。

ドキュメント タイプは、Source Insight がファイルの処理方法を決定する重要な要素です。最も重要なのは、プログラム言語と各ファイルの関連付けを制御する点です。

ドキュメント タイプは、タブ幅、ワードラップ、自動インデント、表示フォントのような編集と表示オプションも指定します。

ヒント： カレント ファイルに関連付けられているドキュメント タイプを参照するには、ソース ファイル ウィンドウを右クリックし、**[Document Options]** を選択してください。ファイルのドキュメント タイプがダイアログ ボックスで自動的に選択されます。

ファイルとドキュメント タイプの関連付け

ファイルは名前と拡張子でドキュメント タイプと関連付けられています。

[Document Options] コマンドでドキュメント タイプとファイル名ワイルドカードを関連付けることができます。指定されたすべてのファイルについて、Source Insight は、ファイル名を (ドキュメント タイプの定義で指定された) ファイル名ワイルドカードと一致させてドキュメント タイプを決定します。たとえば、*.c ファイルは「C Source File」ドキュメント タイプに属し、*.asm ファイルは「Asm Source File」ドキュメント タイプに属します。

特殊なファイル名の関連付け

filealias.txt ファイルには、ドキュメント タイプの決定に使用するファイル名の別名が含まれています。

プロジェクトに拡張子がないファイルが含まれる場合があります。たとえば、「complex」のような標準 C++ ヘッダー ファイルには拡張子はありません。この場合、ワイルドカードを使用してファイルとドキュメント タイプを関連付けることは困難です。このため、Source Insight では、ファイルのドキュメント タイプを決定する目的のためにファイル名の別名を作成する filealias.txt という名前の特別なファイルを使用しています。

filealias.txt ファイルは、Source Insight のプログラム ディレクトリに格納されます。このファイルは編集できます。デフォルトでは、標準 C++ ヘッダー ファイルの名前が含まれます。

filealias.txt のフォーマットは下記のとおりです。

```
oldfilename=newfilename
```

ファイルのドキュメント タイプを決定する前に oldfilename を newfilename にマップします。
例 :

```
algorithm=algorithm.h
```

Source Insight がファイル algorithm を参照する場合、ファイルのドキュメント タイプを確定するときに別名 algorithm.h を使用します。

新規ファイル タイプの追加

新規ドキュメント タイプを追加すると、Source Insight が処理するファイル タイプが拡張されます。

ドキュメント タイプは、Source Insight が処理するファイルを決定します。Source Insight は、ディレクトリのファイルを表示する場合、現在定義されているドキュメント タイプに属しているファイルのみを表示します。また、Source Insight がファイルをプロジェクトに自動的に追加する場合、既知のドキュメント タイプに属しているファイルのみを追加します。つまり、新規ドキュメント タイプを追加することで、Source Insight が処理するファイル タイプが拡張されます。

ドキュメント オプションの編集

ファイルのドキュメント オプションを編集する (またはファイルが属しているドキュメント タイプを確認する) 最も簡単な方法は、ファイル ウィンドウでファイルを右クリックしてショートカット メニューから [Document Options] を選択する方法です。194 ページの「Document Options」も参照してください。

シラブル インデックスを使用したシンボル名の入力

シラブル マッチングは、シンボルとファイル名の部分一致を見つけます。

シラブル インデックスとシラブル マッチングは、シンボルの名前が分からない場合に、シンボルを見つけるための機能です。一貫した命名規則に従っている API では、シラブル マッチングを使用して、特定のトピックに関連するシンボルを見つけることができます。一部の名前を入力することで、関連する項目の検索範囲を狭めることができます。たとえば、「Win」と入力すると、「Win」に関連する関数がすべて検索されます。

シラブル マッチングは、シンボル リストだけでなく、リストに関連付けられているほとんどの入力ボックスでも使用できます。

シンボル名のシラブルを見つけるには、シンボル データベースをインデックス付けする必要があります。

Source Insight は、データベースのシンボルの名前だけではなく、シンボル名に含まれるシラブルもインデックス付けします。Source Insight では、シンボルまたはファイル名のシラブルは大文字で始まる 2 文字以上の文字であると見なされます。たとえば、シンボル名「CreateWindow」には、「Create」と「Window」の 2 つのシラブルが含まれます。Source Insight は両方のシラブルをインデックス付けするので、「Cre」、「Win」などの文字を任意に組み合わせ、任意の順序で入力して、シンボルを参照できます。各シラブルは入力した文字と一致する接頭辞です。

[Browse Project Symbols] ダイアログ ボックスを使用して、プロジェクト ウィンドウのシンボルやファイル リストのシンボルを参照できます。ソース ウィンドウの左側にある、シンボル ウィンドウの上部のテキスト ボックスでも同様です。

さらに、テキスト ボックスと関連するすべてのリストで、同様の機能が利用できるようになりました。

シンボル シラブルとは

シラブルは大文字で始まる 2 文字以上の文字です。

シンボルまたはファイル名のシラブルは大文字で始まる 2 文字以上の文字です。すべての文字が大文字の場合もあります。いくつかの例を示します。

シンボル名	シラブル
CreateWindow	Create Çý Window
OpenHTML	Open Çý HTML
HTMLOpen	HTML Çý Open
FOpenDoc	Open Çý Doc
Vip32Test	Vip32 Çý Test

プロジェクトのシンボル インデックス

Source Insight は、シンボル データベースのシンボルをすべてインデックス付けします。各シンボルはインデックスに 2 回以上表示されることがあります。これは、Source Insight が以下の 3 つのインデックスを管理しているためです。

フル ネーム インデックス。. シンボルのフル ネームをインデックス付けします。フル ネームにはシンボルの親も含まれます。たとえば、「Class1.Member1」。このタイプのインデックスは常に管理され、シンボル定義にナビゲートするために使用されます。

メンバ名インデックス。. シンボルの構造体とクラス メンバ名のみをインデックス付けします。たとえば、クラス名「Document」内部のメンバ関数「Help」のフル ネーム インデックスは「Document.Help」ですが、メンバ名インデックスは「Help」です。このインデックスを使用すると、属するクラスや構造体を指定しなくても、メンバを素早く取得できます。

シラブル インデックス。. シンボル名のシラブルをインデックス付けします。このインデックスを使用すると、名前の一部しか知らない場合でもシンボルを検索できます

プロジェクト用インデックス オプションの設定

[Project Settings] ダイアログ ボックスでシラブルとメンバ インデックス オプションを指定できます。

シラブルとメンバ インデックスはディスクとメモリのインデックス スペースを使用します。データベースの処理は少し遅くなります。**[Project Settings]** ダイアログ ボックスでインデックス付けを制御できます。

インデックスを制御する 2 つのチェック ボックス :

Quick browsing for member names メンバ名インデックスを有効または無効にします。このオプションがチェックされている場合、クラスまたは構造体の名前、ドット (.)、メンバ名を入力する代わりに、クラスまたは構造体のメンバ名のみを入力できます。このオプションがチェックされていない場合、ディスク スペースおよびメモリ容量が節約されます。

Quick browsing for symbol syllables シラブル インデックスを有効または無効にします。このオプションがチェックされている場合、一部のシラブルを入力して、シラブルを含むシンボルを検索できます。このオプションがチェックされていない場合、ディスク スペースおよびメモリ容量が節約されます。大規模なプロジェクトでは、シンボル データベースの参照やファイルの同期は遅くなります。

両方のチェック ボックスをオフにすると、[Browse All Symbols] ダイアログ ボックスおよびプロジェクト ウィンドウのシンボル リスト フィルタは単純な接頭辞マッチングに戻されます。しかし、各ソース ウィンドウの左側のシンボル ウィンドウおよびプロジェクト ウィンドウのファイル リストではシラブル参照を使用できます。

シラブル マッチングの制御

シラブル マッチングは [Preferences: Typing] ダイアログ ボックスで有効または無効にします。

[Preferences: Typing] ダイアログ ボックスのシラブル マッチングで影響を受けるリストを制御できます。チェック ボックス **[Match syllables while typing]** は、プロジェクト設定でシラブルが (シンボル データベースで) インデックス付けされることを示しているかどうかに関係なく、シラブル マッチングを有効または無効にします。

シラブルの前にスペースを入力して個々のケースのシラブル マッチングを切り替えることができます。

シラブルの前にスペースを追加して入力することで、シラブル マッチングの使用を切り替えることができます。たとえば、[Preferences: Typing] ダイアログ ボックスでシラブル マッチングをオフにした場合、シラブルの前にスペースを追加して入力することでシラブルの使用を選択できます。

シラブル マッチングの使用

リスト ボックスと関連するテキスト ボックスにシラブルを入力します。

シンボルまたはファイル名を入力できるテキスト ボックスの多くで、シラブルを入力できます。テキスト ボックスと関連するリスト ボックスは、入力した内容に基づいてフィルタされます。

複数のシラブルを入力することもできます。

複数のシラブルを入力する場合は、「CreWin」のように、シラブルの 2 文字以上を入力して、別のシラブルを入力します。「Cre Win」のように、スペースでシラブルを区切ることも可能です。ほとんどの場合、任意の順序でシラブルを入力できます。

シラブル フィルタリングは大文字と小文字を区別しません。しかし、フィルタされたリストが表示されると、Source Insight は大文字と小文字を含めて、入力した内容に最も近いリストの項目を選択します。

たとえば、以下の指定はすべて等価です。

```
CreWin
Cre Win
cre win
WinCre
Win Cre
win cre
win_cre
Win.cre
```

一致した名前は一致率でソートされます。

Source Insight はリストの内容をフィルタして、結果をソートし、最も一致率の高いものを一番上に表示します。一致率が同じ場合、シンボルの最初からの一致率が高いほうが上に表示されます。

たとえば、「TopBottom」と「TopAndBottom」という 2 つのシンボルがあり、「TopBot」と入力した場合、「TopBottom」シンボルが上に表示されます。

シラブル ショートカットの使用

厳密な接頭辞マッチングやメンバ名マッチングを指定するには、シンボル名フィールドに入力するときにシラブル ショートカットを使用します。

接頭辞のみを厳格に一致させるには、単一スペースまたはキャレット (^) 文字を先頭に指定します。たとえば、「^Format」および「<space>Format」は「format」で始まるシンボルとのみ一致します。(大文字と小文字は区別されません。) 「LineFormat」とは一致しません。接頭辞はシラブル マッチングのオン / オフを切り替えます。

リーフ メンバ名の接頭辞を厳格に一致させるには、先頭にドット (.) を指定します。たとえば、「.fdirty」は「fdirty」で始まる構造体またはクラス名とのみ一致します。入れ子のクラス メンバにも適用されます。たとえば、「.draw」はメソッド「Class1::Class2::Draw」と一致します。

下線は (プログラムでは一般的ですが) 無視されます。たとえば、「_insert」および「insert」は等価です。

解析機能

Source Insight は、シンボルの検索およびシンボル情報のレビューを行う多くの方法を提供しています。これは Source Insight の重要な機能です。

このセクションでは、シンボル情報のアクセスを助ける Source Insight の機能について簡単に説明します。

構文解析

シンボル データベースは、自動的に最新状態に更新されます。

ファイルの編集中に、Source Insight はプロジェクト ファイルを解析します。この結果、ファイルをコンパイルすることなく、クラス、メソッド、関数、その他の情報が表示されます。

プロジェクトにファイルが追加されると、ファイル内のシンボル定義の名前と位置が、プロジェクトのシンボル データベースに保存されます。シンボル定義は以下のテクニックを使用して素早く見つけることができます。Source Insight の多くのオブジェクトは、右クリックするとオブジェクトのショートカット メニューが表示されます。以下のコマンドのいくつかは、ショートカット メニューにあります。

シンボル ナビゲーション コマンド

これらのコマンドは、シンボル定義や関数の呼び出し元に移動するために最も一般的に使用されるコマンドです。

Jump to Definition コマンド

定義にジャンプするには、[Ctrl] キーを押したままで、名前をダブルクリックします。

[Jump to Definition] コマンドは、カーソル位置のシンボルの定義にジャンプします。221 ページの「Jump To Definition」も参照してください。コマンドを呼び出すには、[Alt]+[=] キーを押すか、[Ctrl] キーを押したままでダブルクリックします。シンボルを右クリックし、ショートカット メニューを使用してこのコマンドを実行することもできます。

シンボルが見つかると、シンボルのファイルが開かれ、ウィンドウに表示されて、シンボル名が選択されます。プロジェクト内に同じ名前のシンボルが複数ある場合、Source Insight は選択するシンボルを確認します。

Jump to Caller コマンド

Jump to Caller コマンドは、カーソル位置の関数の呼び出し元にジャンプします。221 ページの「Jump To Caller」も参照してください。シンボルを右クリックし、ショートカット メニューを使用してこのコマンドを実行することもできます。このコマンドは、カーソル位置のオブジェクトが関数名である場合のみ動作します。

Refresh Relation Window コマンド

現在選択されているシンボルが反映されるように、リレーション ウィンドウを更新します。たとえば、リレーション ウィンドウが関数参照を表示するように設定されていてカーソルが関数名の上にある場合、関数へのすべての参照をリレーション ウィンドウに表示します。このコマンドは、自動更新を行わないようにリレーション ウィンドウをロックしている場合に便利です。59 ページの「リレーション ウィンドウ」も参照してください。

Browse Project Symbols コマンド

[Browse Project Symbols] ダイアログ ボックスはプロジェクトのすべてのシンボルをリストします。

[Browse Project Symbols] コマンドは、プロジェクト内のすべてのシンボルのリストを含むダイアログ ボックスを開きます。リストからシンボルを選択するか、シンボル名を入力できます。最初にダイアログ ボックスが表示されたとき、カーソル位置の単語が [Symbol] テキスト ボックスに自動的にロードされます。

[Browse] ダイアログ ボックスで入力する場合、リストは入力した内容と一致するように自動的にフィルタされます。通常、数文字入力するだけで目的のシンボルが表示されます。

プロジェクト ウィンドウのシンボル リスト

プロジェクト ウィンドウにも、プロジェクト内のすべてのシンボルがモードレスでリストされます。

プロジェクト ウィンドウは、すべてのプロジェクト シンボルのリストを表示します。[Browse Project Symbols] ダイアログ ボックスのように、テキスト ボックスに入力する際に、接頭辞とシラブル名マッチングが行われます。しかし、プロジェクト ウィンドウはモードレスなウィンドウです。そのため、プロジェクト ウィンドウでシンボルを選択すると、コンテキスト ウィンドウおよびリレーション ウィンドウで選択したシンボルに関する情報が更新されます。50 ページの「プロジェクト ウィンドウ」も参照してください。

プロジェクト ウィンドウにはクラス ビューもあります。クラス ビューは、シンボルをカテゴリ別に分類し、表示する階層ビューです。たとえば、すべてのグローバル変数、すべての構造体、のようにグループ化されます。クラス、構造体、ユニオンのような、構造体の分類はプロジェクト ウィンドウで表示できます。

コール ツリーと参照ツリー

リレーション ウィンドウではコール ツリーと参照ツリーを表示できます。

リレーション ウィンドウは現在選択しているシンボルと他のシンボルの関係を表示する Source Insight のもう 1 つの新機能です。現在の動作状況を追跡したり、リレーション情報を自動的に表示するなど、コンテキスト ウィンドウのように機能します。59 ページの「リレーション ウィンドウ」も参照してください。

リレーション ウィンドウには、関数呼び出しツリー、クラス階層、構造体メンバ、参照ツリー、その他の情報が表示されます。

コンテキスト ウィンドウ

コンテキスト ウィンドウには、選択したシンボルの宣言が表示されます。

コンテキスト ウィンドウは、Source Insight バージョン 2.0 で導入された新機能です。シンボル情報を自動的に表示します。選択および入力した情報を追跡し、関連するシンボル宣言を自動的に表示します。55 ページの「コンテキスト ウィンドウ」も参照してください。

コマンド ライン シンボル アクセス

Source Insight の起動時に、コマンド ラインで -f <symbol> オプションを使用することにより、シンボルにジャンプできます。

例 :

```
insight3 -f myfunc
```

「myfunc」という名前のシンボルを指します。

特別なコマンド オプションなしで、ファイル名のように、コマンド ラインでシンボルの名前を指定することもできます。この場合、Source Insight は、指定されたシンボルがファイルなのか解析したシンボルなのか判断します。ファイル名のように、コマンド ラインで複数のシンボルをリストできます。

例 :

```
insight3 myfunc fcb init
```

シンボル「myfunc」、「fcb」、および「init」を含むすべてのファイルを開きます。129 ページの「コマンド ラインの構文」も参照してください。

シンボルへの参照の検索

シンボルへの参照を素早く検索するには、[Lookup References] コマンドを使用します。

[Lookup References] コマンドは、現在選択している単語へのすべてのテキスト形式の参照を素早く検索します。たとえば、TextOut へのすべての呼び出しを検索するには、挿入ポイントが単語 TextOut を指すようにして、[Lookup References] コマンドを実行します。検索結果ウィンドウに、見つかったすべての参照情報がリストされます。

[Browse Project Symbols]、[Browse Local File Symbols]、[Symbol Info] ダイアログ ボックスにも、同様の処理を行う [References] ボタンがあります。244 ページの「Lookup References」も参照してください。

プロジェクト レポートの作成

シンボルのクロスリファレンス、およびプロジェクトのファイルに関する統計を含むレポート ファイルを作成するには、[Project Report] コ

マンドを使用します。268 ページの「Project Report」も参照してください。

名前の変更

複数のプロジェクト ファイルで状況に依存して名前の変更を行うには、[Smart Rename] コマンドを使用します。

[Smart Rename] コマンドは、状況依存形式のグローバル検索と置換です。スマート状況依存メソッドを使用して、すべてのプロジェクト ファイルの識別子の名前を変更します。Source Insight の検索インデックスで、検索が非常に速くなります。これは単一の単語識別子を新しい文字列に置換する最も簡単な方法です。検索結果ウィンドウに置換情報のログを出力することもできます。置換された行と、変更された行へのソース リンクがリストされます。307 ページの「Smart Rename」も参照してください。

構文フォーマットとスタイル

構文フォーマットは、詳細かつ簡単な方法で情報を提供する、Source Insight の重要な機能の 1 つです。

構文フォーマットは、プログラム情報に基づくリッチ テキスト フォーマットを使用します。

Source Insight は、パーサーが収集した情報を使用してソース コードの書式を設定します。識別子は、太字や斜体のような効果に加えて、異なるフォントやフォント サイズで表示できます。

フォーマットは「スタイル」を使用して適用されます。スタイルはフォーマット プロパティのセットです。たとえば、スタイルで太字 + 斜体を指定します。各スタイルのフォーマット プロパティを編集するには、[Style Properties] コマンドを使用します。311 ページの「Style Properties」も参照してください。

フォーマット スタイルはソース コードの要素に適用されます。

事前に定義されているスタイルもいくつかあります。独自のスタイルを追加することもできます。スタイルは設定ファイルに格納されます。

スタイルの動作

スタイルには、他のスタイルと組み合わせることができるフォーマット命令が含まれます。

スタイルがテキストに適用されるとき、テキスト プロパティと組み合わされます。基本テキスト プロパティは、各ドキュメント タイプのフォント設定で設定されます。[Options] > [Document Options] コマンドで制御できます。スタイル プロパティは基本テキスト プロパティと組み合わされます。スタイルは、他の「親」スタイルからプロパティを継承することもできます。102 ページの「親スタイル」も参照してください。

すべての適切なスタイルのフォーマット プロパティは、指定したファイルのドキュメント タイプで選択されているフォントと組み合わされます。

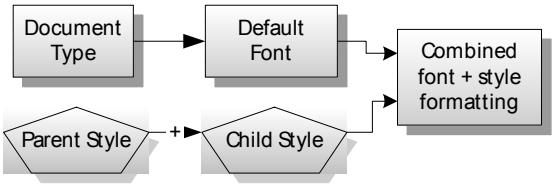


図 4.9 スタイル プロパティは、各プロパティ、および [Document Options] ダイアログ ボックスのデフォルトのフォント設定と組み合わされます。

フォーマット プロパティ

スタイルは、親スタイルとのフォーマットの違いをリストしたものです。たとえば、スタイルに「太字」プロパティが含まれている場合、親スタイルにも「太字」プロパティが含まれています。それぞれのフォーマット項目には、以下の状態があります。

- **On** フォーマット プロパティが追加されます。例 : Bold-On
- **Off** フォーマット プロパティが削除されます。例 : Bold-Off
- **A Number** スケーリング、フォント サイズのような項目に適用されます。例 : scaling = 120%
- **Font Name** フォント名項目に適用されます。
- **No Change** フォーマット プロパティは変更されません。親スタイルのプロパティを継承します。

親スタイル

スタイルは階層構造です。

スタイルには、「親スタイル」プロパティがあります。スタイルは、親スタイルからフォーマット プロパティを継承します。その結果、スタ

イルの階層が作成されます。たとえば、「Declarations」の階層構造には、図のようなスタイルが含まれます。

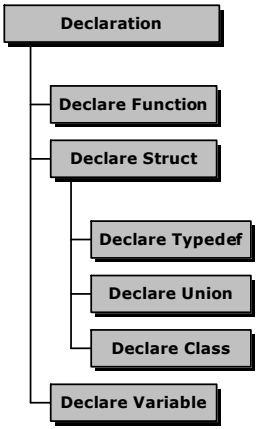


図 4.10 スタイル階層の例。

スタイルのフォーマット プロパティは親スタイルと組み合わされます。

スタイルのフォーマット プロパティは親スタイルと組み合わされます。このため、「Declare Struct」スタイルは、「Declaration」スタイルのフォーマット プロパティを継承します。「Declaration」スタイルのプロパティを変更すると、その階層のすべてのスタイルは変更の影響を受けます。

最上位の親スタイルは「Default Text」スタイルです。このスタイルのフォーマット プロパティは、ドキュメント タイプのフォント設定で決定されます。

[Style Properties] コマンドを使用して、任意のスタイルの親スタイルを変更できます。

下記の図は、ドキュメント タイプのフォント設定とスタイルを組み合わせた例です。

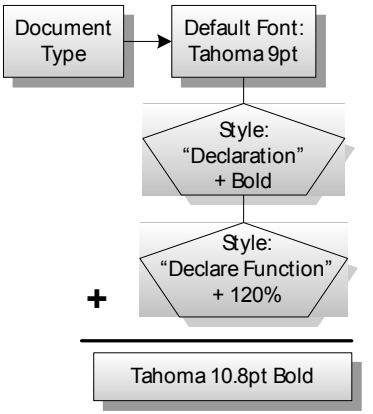


図 4.11 スタイル プロパティは、ドキュメント タイプのデフォルトのフォント設定に「追加」されます。

スタイルのソース コードへの適用

スタイルはソース コード テキストに自動的に適用されます。

スタイルはソース コード テキストに自動的に適用されます。コメント スタイルを除き、スタイルをワード プロセッサのように明示的に適用することはできません。単語情報に基づくスタイルが言語パーサーによって適用されます。下記のトピックで、適用されるスタイルについて説明します。

言語キーワード スタイル

独自のキーワードと任意のスタイルを関連付けることができます。

最も単純なスタイルのアプリケーションは、言語キーワードのフォーマットです。言語にはそれぞれ、キーワード リストが含まれます。キーワード リストはそれぞれ、キーワードとスタイルを関連付けます。キーワード リストは編集可能で、設定ファイルに格納されます。Source Insight はキーワードを認識し、関連付けられているスタイルをキーワードに適用します。言語キーワード リストは、[Preferences: Languages] ダイアログ ボックスから、または [Keyword List] コマンドを直接起動して編集できます。224 ページの「Keyword List」も参照してください。

指定された単語のフォーマットを決定するため、Source Insight は適切な言語タイプのキーワード リストで単語を検索します。キーワード リストには、スタイルと関連付けられているフォーマットを示すスタイル名が含まれます。

つまり、ファイル名およびファイル内の単語で始まります。Source Insight は、この情報を使用して単語のスタイルを決定します。

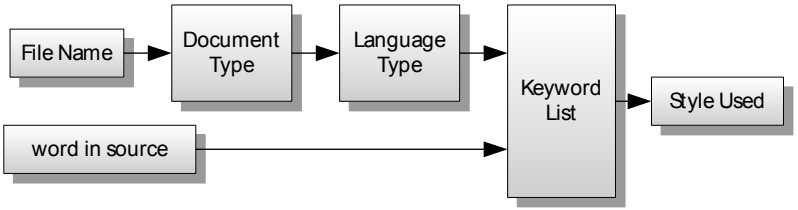


図 4.12 ソース テキストの単語に使用されるスタイルは、ファイルのドキュメント タイプで指定されている言語のキーワード リストによ闌・閧 S れます。

宣言スタイル

宣言は、「Declare...」スタイルでフォーマットされます。

シンボルが宣言または定義されると、その名前は適切な「Declaration」スタイルでフォーマットされます。構造体、クラス、ユニオン、関数、その他のように、異なるタイプの宣言スタイルがあります。

たとえば、構造体は下記のようになります。

```
typedef struct CLIP
    {
    HSZ         hszSource;      // where it came from
    HSZ         hszWhat;        // what it is
    } CLIP;
```

構造体名「CLIP」は「Declare Struct」スタイルで表示されます。構造体内のメンバ フィールドは「Declare Member」スタイルで表示されます。

参照スタイル

シンボルの参照は、「Ref to...」スタイルでフォーマットされます。

実際の宣言ではないシンボルの参照は、適切な「Reference」スタイルでフォーマットされます。異なったタイプのシンボル向けの参照スタイルがあります。

たとえば、コード フラグメントは下記のようになります。

```
// seek the class name token
SeekTokenLnIch(hpar, HdocOfTgl(htglLocal), kswaChangeMark);
```

「SeekTokenLnIch」の呼び出しは、「Ref to Function」スタイルでフォーマットされます。C マクロ関数「HdocOfTgl」は、マクロを参照していることを示す「Ref to Macro」スタイルでフォーマットされます。パラメータ「hpar」は、現在の関数のローカルなパラメータであることを示す「Ref to Parameter」スタイルでフォーマットされます。識別子「htglLocal」は、現在の関数のローカル変数であることを示す「Ref to Local Variable」スタイルでフォーマットされます。定数

「kswaChangeMark」は、「Ref to Function」スタイルでフォーマットされます。

参照スタイルは非常に便利ですが、システムのリソースを消費します。

参照フォーマットは、要求しなくても多くの情報を提供します。関数名のスペルを間違えた場合や、定数、ローカル変数、グローバル変数を使用しているかどうかが、直ちに判明します。

参照フォーマットを有効にすると、場合によっては表示が遅くなることに注意してください。Source Insight では、潜在的なシンボル参照が行われるたびに、シンボルのルックアップ操作を実行する必要があります。

Inactive Code スタイル

Source Insight の C/C++ およびリソース ファイル パーサーは、#ifdef ディレクティブを使用してコンパイル時に無効にされる非アクティブ コードのブロックを認識します。[Options > Preferences: Language] ダイアログ ボックスには、既知の条件付き定数のリストを編集できる [Conditions] ボタンがあります。

```
#ifdef NEVER
        TluParentRefFromSrl(hsrl, ist, &tlu);
        tlu.hdoc = hdocNil;
#endif /* NEVER */
        return CmdGotoTlu(&tlu) == cmdOK;
```

非アクティブ コードは Inactive Code スタイルで表示されます。

非アクティブ コードのブロックは、「Inactive Code」スタイルで表示されます。詳細は、86 ページの「条件付き解析」を参照してください。

コメント スタイル

Source Insight では、コメントにいくつかのフォーマットを追加できます。コメント スタイル階層は下記のように表示されます。

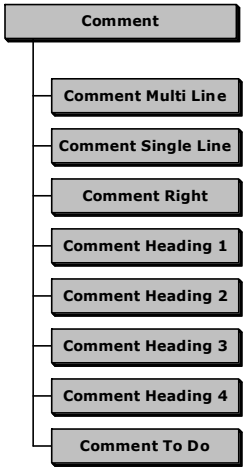


図 4.13 コメント スタイル階層

コメント見出しスタイル

コメント見出しスタイルは、コメントを分かりやすくします。

コメント見出しスタイルは、コードの一部を分割する場合に非常に便利です。

コメント見出しスタイルは //n (n は 1 から 4) のように指定します。

例 :

```
//1 This is a Heading 1 comment
//2 This is a Heading 2 comment
```

コメントを表示するとき、カーソルがその行の上になければ、コメントの先頭にある //n は表示されません。

カーソルが行の上に移動すると、//n が表示され、編集できるようになります。

Comment Right スタイル

コードの右に表示されるコメントには、「Comment Right」スタイルを使用します。
例 :

```
// this comment is formatted with the Comment Line style
a = 0;  // this is formatted with the Comment Right style
```

Comment Line と Comment Multi Line スタイル

複数行のコメントは「Comment Multi Line」スタイルで表示されます。C/C++ や Java の /* と */ デリミタ を使用するコメントが含まれます。

単一行のコメントは、「Comment Line」スタイルで表示されます。C/C++ や Java の // デリミタ を使用するコメントが含まれます。

コメント スタイルとカスタム言語

カスタム言語を定義する場合、[Preferences: Languages] ダイアログ ボックス を使用して、コメントやテキスト範囲のタイプを指定して、スタイルに関連付けることができます。たとえば、(* で開始して *) で終了するコメント範囲を作成して、テキストを「Comment Multi Line」スタイルでフォーマットできます。任意の区切られた範囲のテキストを実際に定義し、(非コメント スタイルを含む) 任意のスタイルに関連付けることができます。詳細は、82 ページの「プログラミング言語」および 229 ページの「[Special Language Options]」を参照してください。

構文修飾

構文修飾は、コード表示に情報を追加します。

Source Insight では、一般的な演算子とシンボル文字を置換できます。[Preferences: Syntax Decorations] コマンドを使用して、使用する装飾を制御できます。

シンボリック文字は、「Symbol Characters」スタイルでフォーマットされます。[Style Properties] コマンドを使用して Symbol Characters スタイルを参照すると、このスタイルは Symbol フォントを使用していることがわかります。色やフォント サイズのようなスタイルのプロパティは変更できますが、フォント名を Symbol 以外に変更しても、構文装飾シンボルはこの時点では表示されません。

メモ : シンボル置換を行ってもソース ファイルのテキストは変更されません。画面の表示が特別なシンボルを使用するように変更されるだけです。このため、コードを編集または検索する場合は、演算子を入力する必要があります。

演算子の置換

特別なシンボルを使用して演算子を表示できます。

ポインタ逆参照右矢印 (->) や、代入演算子 (=) のような一般的な演算子は、矢印などのシンボリック演算子に置換できます。たとえば、下記の内容の代わりに、

```
DIM dimLine = DimOfRgch(hdc, psnoMem->tav.sz, 1);

DrawNodeLine(psnoMem->tav.sz, hdc, xp, yp, prc);
```

装飾がオンの場合、= と -> は矢印に置換されます。

```
DIM dimLine ← DimOfRgch(hdc, psnoMem→tav.sz, 1);

DrawNodeLine(psnoMem→tav.sz, hdc, xp, yp, prc);
```

論理型、数学、その他の演算子も装飾シンボルに置換できます。

入れ子の括弧

Source Insight は、一致するセットが簡単に識別できるように、入れ子の括弧を異なるフォントで表示できます。これは、複数行にわたって、機能します。

```
i ← (ITIR) (((UINT)iMin + (UINT)iLim) >> 1);
```

Goto 矢印

「Goto 矢印」は、もう 1 つの便利な装飾です。上矢印と下矢印は、ターゲット レベルの向きを示す goto ステートメントに表示されます。

```
krel ← krelCalls;
goto ↑LSet;
```

閉じ括弧の注釈

Source Insight では、C/C++ および Java コードの閉じ括弧 (}) に「閉じ括弧」注釈を自動的に追加できます。入れ子の if、while、switch、その他のブロックを見るときに便利です。

```
        } « end switch *pch »
    } « end if stcNewWord!=stcString... »
} « end if !FWhiteCh(*pch) »  // !fWhiteCh
```

構文フォーマットの制御

コードのフォーマットを多くの方法で制御できます。

スタイル プロパティの変更

スタイルのフォーマットを変更するには、[Style Properties] コマンドを使用します。

[Style Properties] コマンドを使用すると、各スタイルのフォーマット プロパティを制御 Ý きます。独自のスタイルを追加して、インポートおよびエクスポートすることもできます。

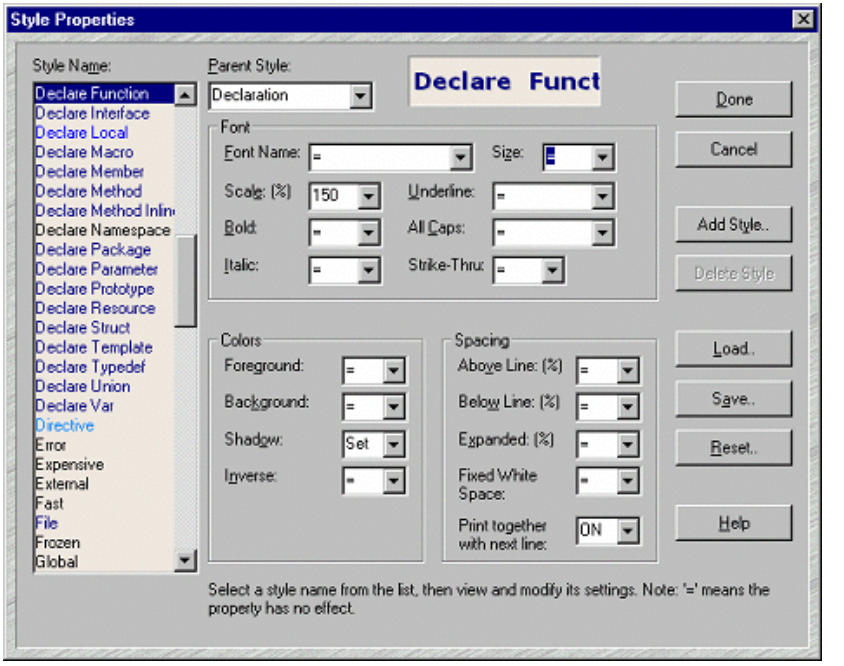


図 4.14 [Style Properties] ダイアログ ボックスで、ユーザーはスタイルのフォーマット オプションを編集できます。

[Syntax Formatting] コマンド

[Syntax Formatting] ダイアログ ボックスで、スタイルの使用方法を制御します。

[Preferences: Syntax Formatting] コマンドを使用して、適用する表示フォーマットを制御できます。いくつかのオプションを無効にすると、表示が速くなります。詳細は、322 ページの「Syntax Formatting」を参照してください。

[Syntax Decorations] コマンド

[Preferences: Syntax Decorations] コマンドを使用して、許可する構文装飾を制御できます。

構文フォーマットをオフにする

Source Insight の構文フォーマット機能は強力ですが、単一フォントを使用しているときにテキストが別のエディタや簡易表示モードでどのように表示されるか確認しておく必要があります。

構文フォーマットを一時的にオフにする

ファイルを一時的に構文フォーマットなしで確認するには、[Draft View] コマンドを使用します。

ファイルを一時的に構文フォーマットなしで確認する場合、[View] > [Draft View] コマンドを使用します。ドラフト モードはディスプレイを基本的な等幅フォントに素早く切り替える際、役に立ちます。これは、カラムを並べるのに、タブの代わりにスペースを使う場合は特に、役に立ちます。ドラフト モードが有効な場合、[Preferences: Syntax Formatting] および [Preferences: Syntax Decorations] ダイアログ ボックスの設定よりも優先されます。

フォントを変更しない

フォントを変更しないで (または、太字や斜体のようなフォント装飾を行わないで) 色のみを変更するには、[Options] > [Preferences: Syntax Formatting] ダイアログ ボックスで **[Use only color formatting]** ボックスをチェックします。

すべての文字を同じ幅にする

すべての文字を同じ幅にしてファイルを表示するには、ファイルを右クリックして、[Document Options] を選択します。[Screen Fonts] ボタンをクリックして、Courier New などの等幅フォントを選択します。もちろん、そのドキュメント タイプで使用されているデフォルト フォントを他のフォントに変更することもできます。

テキストの検索と置換

Source Insight には、カレント ファイルや複数ファイルのテキストを検索および複数する多くのコマンドが用意されています。複数ファイルを検索するとき、ソース リンクを使用して検索結果に素早くリンクできます。

検索はプログラマにとって重要な処理です。Source Insight は、豊富な機能により、大量のソース コードに取り組むユーザーを助けます。

シンボル参照の検索

シンボルへの状況依存参照を検索するには、[Lookup References] を使用します。

現在のプロジェクトで現在選択されているシンボルへの参照を検索するには、[Lookup References] コマンド ([Ctrl]+[/]) を使用します。244 ページの「Lookup References」も参照してください。たとえば、「BeginPaint」をクリックし、[Lookup References] コマンドを実行すると、Source Insight は検索結果ウィンドウを開いて、プロジェクトで BeginPaint が使用されている場所をすべてリストします。検索結果ウィンドウにリストされる一致行には、検索した単語を含む行の場所へのソース リンクもあります。

[Lookup References] コマンドは状況依存なので、指定されたコンテキスト範囲で正確に一致する参照のみを検索します。

識別子名の変更

シンボル名を状況 (グローバルまたはローカル) に依存して変更するには、[Smart Rename] を使用します。

スマート状況依存メソッドを使用してすべてのプロジェクト ファイルの識別子を変更するには、[Smart Rename] コマンド ([Ctrl]+['']) を使用します。[Smart Rename] コマンドは、状況依存形式のグローバル検索と置換です。307 ページの「Smart Rename」も参照してください。Source Insight の検索インデックスで、検索が非常に速くなります。これは単一の単語識別子を新しい文字列に置換する最も簡単な方法です。検索結果ウィンドウに置換情報のログを出力することもできます。置換された行と、変更された行へのソース リンクがリストされます。

Smart Rename は、ローカル スコープ変数の名前変更にも便利です。

カレント ファイルの検索

[Search] コマンドは、カレント ファイル バッファを検索します。

[Search] コマンド ([Alt]+[F]) は、検索パターンでカレント ファイルを検索します。下方向または上方向に検索できます。さらに、検索はバッチ スタイルにできます。検索結果は検索結果ウィンドウに表示されます。295 ページの「Search」も参照してください。

[Search Forward] コマンド (F4) は、指定されたパターンで、ファイルの次の候補を検索します。

[Search Forward for Selection] コマンドは、現在の選択範囲にある最初の単語の次の候補を検索します。

[Incremental Search] コマンド (F12) は、入力した文字の一致を検索します。216 ページの「Incremental Search」も参照してください。

カレント ファイルの置換

[Replace] コマンドを使用して、検索パターンでカレント ファイルを検索し、既存パターンと新規パターンを置換します。置換の範囲は、ファイル全体または現在の選択範囲のみです。284 ページの「Replace」も参照してください。

複数ファイルの検索

[Search Files] コマンドは、複数ファイルで検索を行います。

すべてのプロジェクト ファイル、非プロジェクト ファイルでパターンを検索するには、[Search Files] コマンドを使用します。297 ページの「Search Files」も参照してください。検索するファイルを指定できる点を除けば、[Search Files] コマンドは [Search] コマンドに似ています。検索結果は検索結果ウィンドウに表示されます。検索結果には、ソース リンクと呼ばれる非表示の情報も含まれます。

プロジェクト全体を単一の単語で参照する場合は、[Lookup References] コマンドを使用するほうが高速です。244 ページの「Lookup References」も参照してください。

複数ファイルの置換

[Replace Files] コマンドは、複数ファイルで置換を行います。

286 ページの「Replace Files」も参照してください。置換するファイルを指定できる点を除けば、[Replace Files] コマンドは [Replace] コマンドに似ています。

複数ファイルのシンボル名を変更する場合は、状況依存の [Smart Rename] コマンドを使用してください。307 ページの「Smart Rename」も参照してください。

キーワードの検索

[Search Project] コマンドは、プロジェクト内をインターネット スタイルで検索します。

インターネットの Web サイトを検索するようにプロジェクト内を検索するには、[Search Project] コマンドを使用します。キーワード検索を使用すると、コンテキストの指定した行の範囲内にある用語の組み合わせを検索できます。301 ページの「Search Project」も参照してください。

正規表現

正規表現は複雑なパターンのマッチングに役立つ特殊な検索文字列です。正規表現では、多くの文字は特殊な意味を持ちます。たとえば、「行頭」という意味の特殊文字があります。このセクションでは、Source Insight が認識できる特殊文字について説明します。

ワイルドカード マッチング

`.(dot)`

ドット (.) は、すべての文字とマッチします。

例：`b.g` は、big、beg、bag とマッチしますが、bp、baag とはマッチしません。

行頭または行末のマッチング

`^ および $`

キャレット ^ が検索パターンの最初の文字として現れた場合、行頭とマッチします。

例：`^Hello` は、行頭に Hello が現れたときのみマッチします。

`$` は行末とマッチします。

例 : TRUE$ は、行末に TRUE が現れたときのみマッチします。

タグまたはスペースのマッチング

```
\t
\s
\w
```

\t は、単一タブ文字とマッチします。

例 : \tint abc; は、int abc; が続くタブ文字とマッチします。

\s は、単一スペース文字とマッチします。

例 : \sif は、if が続くスペースとマッチします。

\w は、単一空白文字とマッチします。つまり、\w は、タブ文字またはスペース文字のいずれかとマッチします。

例 : \wwhile は、while が続くタブまたはスペースとマッチします。

0 または 1 回以上の繰り返しマッチング

```
* および +
```

* は、先行する文字の 0 回以上の繰り返しとマッチします。先行する文字の繰り返し回数が 0 でもマッチします。

例 : a*b は、b、ab、aab、aaab、aaaab、... とマッチします。

+ は、先行する文字の 1 回以上の繰り返しとマッチします。

例 : a+b は、ab、aab、aaab、aaaab、... とマッチしますが、b とはマッチしません。

文字セットのマッチング

```
[ ..]
```

文字セットが [..] で囲まれている場合、セットに含まれているすべての文字がマッチします。

例 : [abc] は、a、b、c とマッチしますが、d とはマッチしません。

セットの先頭にキャレット (^) が追加されると、セットに含まれていない文字のみマッチします。

例 : [^abc] は、d、e、f とマッチしますが、a、b、c とはマッチしません。

セットは範囲の指定に便利です。範囲は、[a-z] のように 2 つの文字をダッシュ (-) で区切って指定します。最初の文字は、最後の文字よりも ASCII 値で小さくなければなりません。

例 : [a-z] は、a から z のすべての文字とマッチしますが、A、1、2 とはマッチしません。

セットに複数の範囲を含めることができます。

例 1: `[a-zA-Z]` は、すべての英字とマッチします。

例 2: `[^a-zA-Z0-9]` は、すべての非英数字とマッチします。

正規表現グループ

`\(` および `\)`

正規表現の一部を `\(` および `\)` で囲んで、グループにできます。グループは、置換パターンで使用したマッチの一部を利用する際に便利です。パターンの各グループは、左から右に、1 から始まる数が割り当てられます。

例: `abc\(xyz\)` は、abcxyz とマッチします。xyz はグループ #1 と見なされます。

グループは、[Replace] コマンドを使用するときに威力を発揮します。置換文字列に、\< 数字 > 形式でグループ文字を指定できます。グループ文字が置換パターンにマッチすることは、マッチしたパターンからグループ値を置換することを意味します。

例 1: `\(abc\)\(xyz\)` を `\2\1` で置換。マッチした文字列 abcxyz を、グループ #2 の内容 xyz、グループ #1 の内容 abc の順に置換します。このため、abcxyz は xyzabc に置換されます。驚くのはこれからです。次のサンプルを参照してください。

例 2: `\(\w+\)\(.*\)ing` を `\1\2ed` で置換。ing で終わる文字列を ed で終わる文字列に変更します。英語の先生には迷惑かもしれませんね。

正規表現文字の無効化

`\` (バックスラッシュ)

メタ文字の前にあるバックスラッシュ (\) はその特殊な意味を無効にします。バックスラッシュ (\) は文字列から無視されます。

例: `a\*b` は、a*b とマッチします。この場合、* 文字に「先行する文字の 0 回以上の繰り返しとマッチする」という意味はありません。

正規表現のまとめ

以下の特殊文字は正規表現で解釈されます。

表 4.3: 正規表現文字

文字	マッチする対象
^ (先頭にある場合のみ)	行頭
.	すべての単一文字
`[abc]`	セット abc に属するすべての単一文字

表 4.3: 正規表現文字

文字	マッチする対象
[^abc]	セット abc に属さないすべての単一文字
*	先行する文字の 0 回以上の繰り返し
+	先行する文字の 1 回以上の繰り返し
\t	タブ
\s	スペース
\w	空白 (タブまたはスペース)
$	行末

[abc] のようなセットには、以下の意味があります。

表 4.4: 正規表現文字列

セットの種類	意味
[< 文字リスト >] 例 : [abcde]	セット内のすべての文字とマッチします。セットの文字数に制限はありません。
[x-y] 例 : [a-z]	x から y の範囲のすべての文字とマッチします。x の ASCII 値は y の ASCII 値よりも小さくなければなりません。
組み合わせ 例 : [WXYa-z0-9]	文字リストと範囲が組み合わされます。

ブックマーク

ブックマークは、ファイルの特定の場所をマークします。ブックマークを設定および削除、またはブックマークへジャンプするには、**[Bookmark]** コマンドを使用します。各マークには名前があり、ファイルと行番号を指定します。[Bookmark] コマンドは、マークがある関数およびシンボルも表示します。

ブックマークは、ファイルの特定の場所を記録する際に便利です。設定したマークはカレント ワークスペースに保存され、セッションからセッションへ維持されます。

ブックマークは、ファイルの編集に応じて更新されます。たとえば、ある行にマークを設定し、その行の前に行を挿入した場合、行番号が変更されてもマークはオリジナル行に設定されたままです。マークを

設定した行が削除されるか、ファイルが閉じられると、マークも削除されます。

選択履歴のナビゲーション

選択履歴は、現在開いているファイルで最近選択した範囲 (最大 100) の循環リストです。

選択履歴は、最近アクセスした位置の循環リストです。

選択履歴の位置を表示、または選択履歴の位置にジャンプするには、[Selection History] コマンドを使用します。[Selection History] コマンドは、各履歴項目がある関数またはシンボル、ファイルおよび行番号も表示します。

[Go Back] と [Go Forward] コマンド

[Go Back] と [Go Forward] コマンドは、インターネット ブラウザのコマンドに似ています。

インターネット ブラウザの [進む] と [戻る] ボタンの使い方を知っているなら、[Go Back] と [Go Forwards] コマンドも簡単に使用できます。

[Go Back] と [Jump To Definition] コマンドを使用すると、関数呼び出しチェーンの移動や、異なる関数呼び出しを含む別のパスへの移動が簡単になります。

[Go Back] コマンドは、選択履歴の前の場所にジャンプします。[Go Forward] コマンドは、選択履歴の次の場所にジャンプします。選択履歴は循環リストです。そのため、リストの最後で [Go Forward] コマンドを使用するとリストの先頭に、リストの先頭で [Go Back] コマンドを使用するとリストの最後にジャンプします。

ソース リンクを使用したナビゲーション

ソース リンクは、2 つの異なるテキスト ファイルの 2 つの場所を接続します。ソース リンクは、「リンク ソース」ファイルのテキストの行を「リンク ターゲット」ファイルの場所に接続します。リンクは個々の行と関連付けられます。ソース リンクは、カレント ワークスペースの一部です。

ソース リンクは、2 つのファイルの 2 つの行を接続します。

リンクを移動するには、[Jump To Link] コマンドを使用します。ソース リンクのリンク先が表示されます。リンクは、ソース ファイルが開い

ている限り移動できます。ターゲット ファイルが開いていない場合、[Jump To Link] コマンドを使用すると自動的にファイルが開きます。

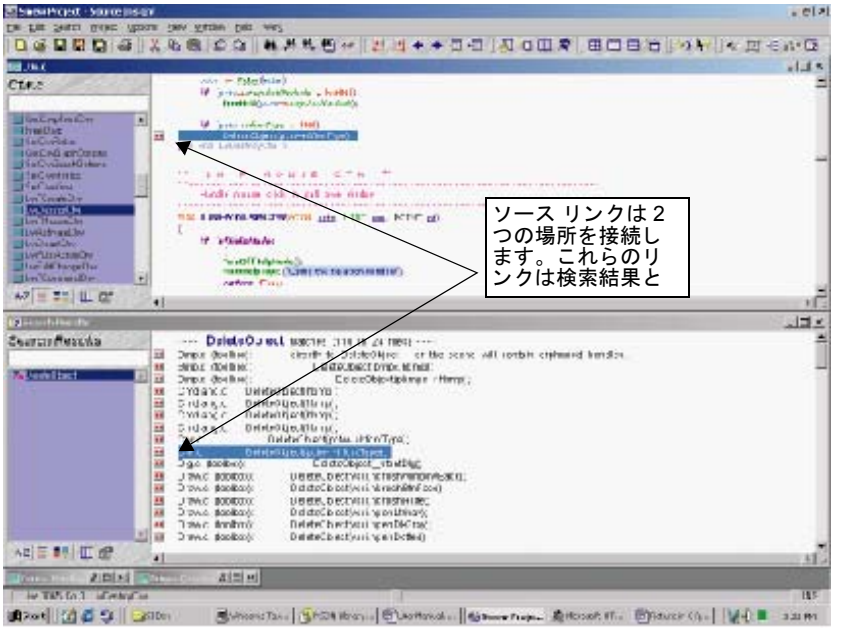


図 4.15 検索結果ウィンドウとソース リンクのリンク先を示すソース ファイル ウィンドウ。

ソース リンクは双方向のリンクです。そのため、**[Jump To Link]** コマンドを使用して、ソースからターゲットに、またはターゲットからソースに移動できます。

また、リンク情報は、ブックマークのようにファイルの編集に応じて更新されます。リンク ソース行が削除されるか、リンク ソース ファイルが閉じられると、リンクも削除されます。

検索とソース リンク

[Search]、[Search Files]、および [Lookup References] コマンドを使用して検索を実行すると、検索結果ウィンドウの検索結果にソース リンクが含まれます。リンク ソース ファイルは検索結果のテンポラリ ファイルで、各リンクのリンク ターゲットは検索したファイルでパターンが一致した場所です。

ソース リンクはカレント ワークスペースの一部です。つまり、セッション状態の一部です。検索結果を新しいファイル名で保存した場合、ソース リンクはカレント ワークスペースに残されます。たとえば、あるパターンを検索して、検索結果を S1.OUT に保存し、次の検索を行

うと、S1.OUT と検索結果に別のソース リンクが含まれます。検索結果ウィンドウを閉じると、ソース リンクは削除されます。

ソース リンクの作成

[Parse Source Links] コマンドは、カレント ファイルにソース リンクを作成します。

カレント ファイルにソース リンクを作成するには、[Parse Source Links] コマンドを使用します。[Parse Source Links] コマンドを使用するには、ファイル名と行番号の解析に使用する検索パターンを指定する必要があります。パターンがマッチするたびに、新規ソース リンクがカレント ファイルに挿入されます。

[Parse Source Links] コマンドは、コンパイラ出力およびエラー メッセージを含む「ログ」ファイルがある場合に便利です。ログ ファイルを開いて、[Parse Source Links] コマンドを実行するだけです。リンクがエラー メッセージを含むログ ファイルの各行に設定されます。

カスタム コマンド出力からのソース リンク

カスタム コマンド出力に、エラーのソース リンクを含めることができます。

カスタム コマンドの出力からソース リンクを作成できます。コマンドを定義するとき、「Parse Source Links」オプションはオンです。ファイル名と行番号の解析に使用する検索パターンを指定する必要があります。コマンドを終了すると、Source Insight は [Go To First Link] コマンドを自動的に呼び出します。

Source Insight から C コンパイラを呼び出す場合は、この方法を推奨します。この方法を使用すると、コンパイラを実行してエラーが発生した場合でも、Source Insight は C ソース行を表示できます。

ソース リンクの移動

Source Insight では、リンク間の移動用に、以下のコマンドが用意されています。

表 4.5: ソース リンクのコマンド

コマンド	キー	説明
Jump To Link	Ctrl+L	ソース リンクのリンク先へ移動します。
Go To First Link		リンク ソース ファイルの最初のリンク行を選択し、リンク ターゲット ファイルの関連するリンク行を選択して、両方のファイルをウィンドウに表示します。

表 4.5: ソース リンクのコマンド

コマンド	キー	説明
Go To Next Link	Shift+F9	リンク ソース ファイルの次のリンクを選択することを除き、上記と同じです。
Go To Previous Link		リンク ソース ファイルの前のリンクにジャンプします。

テキストのスクロールと選択

選択範囲は、ソース ファイル ウィンドウでハイライトされているゼロまたはそれ以上の文字です。選択範囲のテキストは、「Selection」スタイルでハイライトされます。

ソース ファイル ウィンドウごとに、1 つの選択範囲があります。ウィンドウを切り替えても、各ウィンドウの選択範囲には影響しません。

選択範囲はキーボードとマウスで指定します。テキストの選択に使用するコマンドは、この後の表を参照してください。マウスを使用してテキストを選択する場合は、テキストをポイントしてドラッグします。マウスをダブルクリックすると、単語全体が選択されます。

[Preferences: General] ダイアログ ボックスの [Keep cursor in window when paging up and down] オプションがオンの場合のみ、ウィンドウをスクロールすると、選択範囲が変更されます。このオプションがオフの場合は、カーソル移動キーを使用した場合、マウスを使用してポイントした場合、ファイルを編集した場合のみ、選択範囲が変更されます。

カレント ウィンドウとカレント ファイルの選択範囲は、現在の選択範囲として参照されます。

ファイル内の移動

ファイル内を移動するコマンドはたくさんあります。移動には、選択とスクロールの 2 種類の移動があることに注意してください。

選択とは、ファイルの現在の選択範囲 (通常は挿入ポイント) を動かすことです。

スクロールとは、ファイル ウィンドウにファイルの (ウィンドウに表示されていなかった) 新しい部分が表示されることです。ウィンドウは、上下左右にスクロールできます。スクロールは、選択範囲の場所に影響するとは限りません。

選択している場所がウィンドウに表示されていない場合、選択コマンドを使用すると該当する場所にスクロールします。たとえば、カーソルがウィンドウの一番下の行にあるとき、[Cursor Down] コマンドを使用すると、ウィンドウはスクロールして下の行を表示します。

スクロール コマンド

スクロール コマンドはアクティブなウィンドウをスクロールします。現在の選択内容は影響を受けません。ウィンドウのみ影響を受けます。

表 4.6: スクロール コマンド

コマンド	キー	説明
Scroll Line Up	Alt+Up	1 行上にスクロールします。
Scroll Line Down	Alt+Down	1 行下にスクロールします。
Page Up	PgUp	1 ウィンドウ上にスクロールします。
Page Down	PgDn	1 ウィンドウ下にスクロールします。
Scroll Half Page Up	Ctrl+PgUp	ウィンドウの半分上にスクロールします。
Scroll Half Page Down	Ctrl+PgDn	ウィンドウの半分下にスクロールします。
Scroll Left	Alt+Left	約 1 タブ左にスクロールします。
Scroll Right	Alt+Right	約 1 タブ右にスクロールします。

選択コマンド

選択コマンドは現在の選択範囲を変更します。通常、変更後の選択範囲がウィンドウに表示されていない場合、選択範囲を表示するようにウィンドウがスクロールされます。これらのコマンドは、挿入ポイントに選択範囲を変更して、ファイルの新しい位置に移動します。

表 4.7: カーソル移動コマンド

コマンド	キー	説明
Cursor Down	↓	1 行下に移動します。
Cursor Up	↑	1 行上に移動します。
Cursor Left	←	1 文字左に移動します。
Cursor Right	→	1 文字右に移動します。

表 4.7: カーソル移動コマンド

コマンド	キー	説明
Beginning of Line	Home	行頭に移動します。
End of Line	End	行末に移動します。
Top of File	Ctrl+Home	ファイルの先頭に移動します。
Bottom of File	Ctrl+End	ファイルの最後に移動します。
Beginning of Selection	Ctrl+Alt+[	選択範囲の先頭に移動します。
End of Selection	Ctrl+Alt+]	選択範囲の最後に移動します。
Top of Window		ウィンドウの先頭に移動します。
Bottom of Window		ウィンドウの最後に移動します。
Word Left	Ctrl+Left	1 単語左に移動します。
Word Right	Ctrl+Right	1 単語右に移動します。
Function Down	Keypad +	次の関数定義に移動します。
Function Up	Keypad -	前の関数定義に移動します。
Blank Line Down		次の空白行に移動します。
Blank Line Up		前の空白行に移動します。
Paren Left	Ctrl+9	前の括弧に移動します。
Paren Right	Ctrl+0	次の括弧に移動します。
Blank Line Up		前の空白行に移動します。
Blank Line Up		前の空白行に移動します。
Block Up	Ctrl+Shift+[	前の { ブロック レベルに移動します。
Block Down	Ctrl+Shift+]	次の { ブロック レベルに移動します。
Go To Line	F5、Ctrl+G	指定した行番号に移動します。
Search	Ctrl+F	パターンを検索します。
Search Forward	F4	次の候補を検索します。
Search Backward	F3	前の候補を検索します。

表 4.7: カーソル移動コマンド

コマンド	キー	説明
Search Forward for Selection	Shfit+F4	カーソルがある単語の次の候補を検索します。
Selection History	Ctrl+Shift+M	過去の選択範囲のリストを表示します。

選択範囲の拡張

以下のコマンドは、範囲の選択 (選択範囲の作成) または選択範囲の拡張に使用します。キーストロークはデフォルトです。

表 4.8: 選択コマンド

コマンド	キー	説明
Select All	Ctrl を押したまま左の選択バーをマウスでクリック	ファイル全体を選択します。
Select Block	Ctrl+-	中括弧または括弧で閉じた次のブロックを選択します。
Select Function Or Symbol	ウィンドウの左の選択 oーをダブルクリック	シンボル定義全体を選択します。
Select Char Left	Shift+Left	選択範囲を 1 文字左に拡張します。
Select Char Right	Shift+Right	選択範囲を 1 文字右に拡張します。
Select Line	Shift+F6	行全体を選択します。
Select Line Down	Shift+Down	選択範囲を 1 行下に拡張します。
Select Line Up	Shift+Up	選択範囲を 1 行上に拡張します。
Select Match	Alt+=	対応する中括弧、括弧または引用符まで選択します。
Select Sentence	Shift+F7、Ctrl+.	文全体 (次のピリオドまで) を選択します。
Select Word	Shift+F5、ダブルクリック	単語全体を選択します。

表 4.8: 選択コマンド

コマンド	キー	説明
Select Word Left	Ctrl+Shift+Left	左に単語単位で選択範囲を拡張します。
Select Word Right	Ctrl+Shift+Right	右に単語単位で選択範囲を拡張します。
Select To	Shift を押したままクリック	カーソルの位置まで選択範囲を拡張します。
Select To Top Of File	Shift+Home	ファイルの先頭まで選択します。
Select To End Of File	Shift+End	ファイルの最後まで選択します。
Select To End Of Line		カレント行の行末まで選択します。
Toggle Extend Mode		オンの場合、移動キーで選択範囲を拡張します。

[Toggle Extend Mode] コマンドは、拡張モードのオン、オフを切り替えます。拡張モードがオンの場合に、[Cursor Left] のような移動コマンドを使用すると、現在の選択範囲がその方向に拡張されます。

98 ページの「解析機能」も参照してください。

選択ショートカット

Source Insight には、ソース コード内のオブジェクトを選択する多くのショートカットが用意されています。

Source Insight のソース ファイル ウィンドウの左には、「選択バー」があります。ショートカットの多くは、選択バーをクリックして実行できます。

単語全体の選択

単語全体を選択するには、単語をダブルクリックします。

単語全体を選択するには、単語をダブルクリックします。選択範囲をドラッグして単語全体を選択することもできます。

関数全体またはシンボルの選択

関数全体を選択するには、左の選択バーをダブルクリックします。

関数全体、構造体、その他のシンボルを選択するには、ソース ファイル ウィンドウの選択バーをダブルクリックします。Source Insight は、ドラッグした関数全体またはシンボルを選択します。

[Select Symbol] コマンドを使用して、現在の選択範囲を囲むシンボルを選択することもできます。

(ソース ファイル ウィンドウの左にある) シンボル ウィンドウもシンボル全体の選択に使用できます。リストでエントリをダブルクリックすると、シンボルが選択されます。

対応する括弧とブロックの選択

対応する括弧または引用符まで選択するには、選択範囲をダブルクリックするか、[Alt]+[=] キーを押します。

対応する括弧または引用符まで選択するには、括弧または引用符をダブルクリックします。Source Insight が該当する範囲を選択します。[Select Match] コマンド ([Alt]+[=]) を使用して対応する括弧まで選択することもできます。

囲みブロックの選択

[Select Block] コマンド ([Ctrl]+[-]) は、囲みブロックを選択します。囲みブロックとは括弧、中括弧、または大括弧のいずれかで囲まれているブロックです。このコマンドを繰り返すと、次の外側の囲みブロックが選択されます。

行全体の選択

テキストの行全体を選択するには、選択バーをクリックします。テキストの行全体が選択されます。マウスのボタンを押したままドラッグすると、行全体が選択されます。[Select Line] コマンド ([Shift]+[F6]) も行全体を選択します。

ファイル全体の選択

ファイル全体を選択するには、[Ctrl] キーを押したままで左の余白をマウスでクリックします。または、[Select All] コマンド ([Ctrl]+[A]) を使用します。

テキストの段落の選択

言語パーサーがないファイル (194 ページの「Document Options」を参照) では、選択バーをダブルクリックしてテキストの段落全体を選択できます。

段落は、黒線で囲まれたテキスト行のグループとして定義されます。

行間の選択

Source Insight では、カーソルが行の先頭の近く、またはその上下にあるときに表示される、くさび型のカーソルを使用して行間を選択できます。マウスでくさび型のカーソルをクリックすると、Source Insight は行間を選択します。入力を開始すると、Source Insight は最初に改行を自動的に挿入します。これは新しい行を挿入する便利な方法です。

ファイル バッファについて

[Open] コマンドはファイルを作業「バッファ」にロードします。

[File] > [Open] コマンドは、ディスク上のファイルを開いて、ファイル バッファにロードします。ファイル バッファは編集可能なファイルのテンポラリ イメージです。[Save] コマンドを使用してファイルを保存するまで、オリジナル ファイルに影響を及ぼすことなく、ファイル バッファを編集できます。

[Save] コマンドは、オリジナル ファイルを上書きします。

[File] > [Save] コマンドは、ファイル バッファの内容をオリジナル ファイルに上書きします。オリジナル ファイルが変更されるのは、この場合だけです。

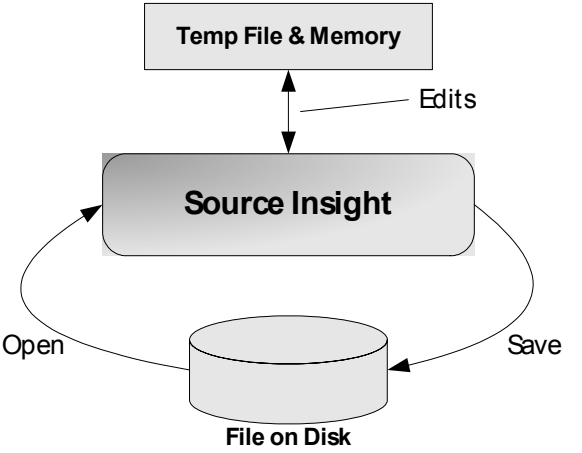


図 4.16 開くと保存

ファイルを閉じると、ファイル バッファの内容は破棄されます。ディスク上のオリジナル ファイルは [Save] コマンドを使用しない限り変更されません。

オリジナル ファイルはファイルを保存するまで変更されません。

ファイルが保存されるまで、すべての変更はテンポラリ ファイル バッファに記録されます。ファイルが保存されると、新規ファイルが作成され、オリジナル ファイルは削除されるか、バックアップ ディレクトリに移動されます。いったん新規ファイルがディスクに書き込まれると、ファイルの名前がオリジナル ファイルの名前に戻されます。保存されるまでオリジナル ファイルが変更されないため、Source Insight では安全にファイルを変更することができます。

ファイル編集時にファイル バッファの使用を考慮する必要はありません。このマニュアルでは、「開いているファイル」という記述はファイル バッファを意味しています。Source Insight は、開いているファイルとオリジナルのベース ファイル間の関係を維持します。このため、[Open] コマンドは、ファイルと無関係なバッファにオリジナル ファイ

ルをコピーするコマンドではなく、オリジナル ファイルを開いて編集するコマンドと見なされます。[Save] コマンドは、オリジナル ファイルと編集したファイルの同期と見なすことができます。

Source Insight は、テキスト ファイルの編集に使用します。

Source Insight は、ASCII ファイルの編集に使用します。非 ASCII ファイルを編集するようには設計されていません。しかし、非 ASCII ファイルを開くことはできます。非 ASCII 文字が含まれている場合、Source Insight はファイルを保存する前に、本当に保存してもいいかどうか確認します。Source Insight は、CR/LF シーケンスを行末マーカとして扱い、CR/LF シーケンスを含むファイルを保存する場合は常に確認します。

Source Insight のアプリケーション ウィンドウにファイルをドラッグ アンド ドロップして、さまざまな種類のファイルを仮想的に開くこともできます。通常のテキスト ファイルに加えて、プロジェクト ファイル (.PR)、設定ファイル (.CF3)、ワークス ペース ファイル (.WK3)、およびクリップ ファイル (.CLI) を開くことができます。

タイムスタンプ

ファイルのタイムスタンプとサイズは、「署名」として使用されます。

Source Insight でファイルを保存すると、Source Insight はプロジェクトにファイルの更新日時を記録します。プロジェクト ファイルを開くとき、ファイルの変更日時がプロジェクトに記録されている日時より新しい場合、Source Insight はファイルのプロジェクト シンボル データベースを再同期します。自動的に同期することで、ユーザーが別のテキスト エディタやソース管理システムを使用した場合でも、Source Insight のプロジェクト データベースが最新の状態に保たれます。319 ページの「Synchronize Files」も参照してください。

Source Insight 起動時の状態

Source Insight は、起動後にまずレジストリを調べて、開くプロジェクトを確認します。次に、プロジェクトを開き、プロジェクトの設定とワーク スペースをロードします。ワークスペースをロードすると、以前開かれていたファイルがすべて、再度開かれます。

ファイル名をコマンド ラインで指定した場合、ワークスペースにリストされているファイルは再度開かれません。

プロジェクトが開かれると、現在の作業ディレクトリがプロジェクトのソース ディレクトリに変更されます。

クラッシュからのリカバリ

リカバリ ファイルは編集中にバックグラウンドで保存されます。

一定間隔ごとに、保存していない編集項目はテンポラリ リカバリ ファイルに保存されます。この保存処理は非常に高速に行われ、通常は気づくことはありません。作業を中断される心配はありません。

リカバリ ファイルには、他のファイルへのポインタのみ保存されます。このため、書き込み作業は非常に高速です。編集内容を徐々にリカバリ ファイルに保存することで、Source Insight はシステム障害、停電、またはクラッシュに対処します。Source Insight は、次回の起動時にリカバリが必要かどうか認識します。

リカバリ ファイルの保存間隔は、[Preferences: General] で設定します。

[Preferences: General] コマンドでリカバリ ファイルの保存間隔を設定できます。リカバリ時間は秒単位で指定できます。デフォルト値は 15 秒です。この値よりも短く設定すると、リカバリ ファイルの更新頻度は高くなります。この値よりも長く設定すると、リカバリ ファイルの更新頻度は低くなります。設定時間を長くしすぎると、リカバリ ファイルに変更を記録しないで、ファイルに変更を加えることができます。なるべく短い間隔を設定してください。しかし、リカバリ ファイルの更新頻度が高ければ高いほど、Source Insight のパフォーマンスは低下します。高速なマシンでは数秒で更新が完了するため、それほど気になりません。

リカバリ手順

オーファン リカバリ ファイルはクラッシュが発生したことを示します。

ファイルを保存する前にクラッシュが発生した場合、Source Insight を再起動してください。Source Insight は、オーファン リカバリ ファイルを使用して、ファイルに対して行われた変更を認識します。ダイアログ ボックスが表示され、前回の Source Insight のセッションが正常に終了しなかったことを知らせます。3 つのオプションがあります。

- **[Recover] ボタンをクリックする**。Source Insight は、過去に開かれたファイルをすべてリカバリします。リカバリ後、セッションはリカバリ ファイルが最後に同期されたときと同じにように見えます。保存されなかった編集はすべてそのままです。

  (または)

- **[Continue] ボタンをクリックする**。Source Insight は、リカバリしないで処理を続行します。続行後にリカバリすることはできません。

  (または)

- **[Quit] ボタンをクリックする**。Source Insight は、リカバリしないで直ちに処理を終了します。再度 Source Insight を実行すれば、リカバリすることは可能です。

リカバリが行われると、Source Insight は編集セッションの途中から再開します。以前開かれていたすべてのファイルが開かれ、以前編集していた内容がそのまま表示されます。リカバリ後、通常通り編集を続行できます。終了すると、それまでに変更したファイルが保存されます。

警告

保存する前にリカバリの結果を確認してください。

Source Insight のリカバリ システムは非常に優れていますが、リカバリ システムは複雑なシステムです。脆弱性が存在する可能性は常にあります。リカバリが必要になるのは、ハードウェア障害、停電、Source Insight が原因で発生したプログラムの不具合、Source Insight そのものの不具合などの、異常な状況です。理由が何であれ、ファイルを保存する前にリカバリしたファイルを確認して問題がないことを確かめることを推奨します。Source Insight のリカバリ ファイルの一部または全部が壊れていたり、リカバリ ファイルの更新中に問題が発生した場合、Source Insight はファイルを正しくリカバリできません。リカバリが正しく行われなかった場合、ファイルを保存しないでください。ファイルを保存しないで閉じ、オリジナルの保存していないファイルを使用してください。リカバリできなかった変更は保存されていませんが、ファイルは正常です。

メモ： リカバリを行うには、以前開いたオリジナル ファイルがまだ存在していて、Source Insight で開いた後、または保存した後に変更されていない必要があります。リカバリを行う必要がある場合は、オリジナル ファイルを変更する前に行ってください。

コマンド ラインの構文

Source Insight のコマンド ラインの構文は以下のとおりです。

```
insight3 [- オプション ] [ [+ 行番号 ファイル ] [+ ファイル ] [ ファイル ]
    [ シンボル名 ] ]
```

大括弧 [...] で囲まれているのはオプション パラメータです。任意の数のオプション、ファイル、およびシンボル名をコマンド ラインで指定できます。

コマンド ラインで指定するオプションの前には、ダッシュ (-) または右上がりスラッシュ (/) を記述する必要があります。

ファイルの引数の指定

ファイルの引数は、以下の種類の 1 つになります。

- ファイル名。Source Insight では、ファイルの拡張子を省略できます。この場合、Source Insight はカレント プロジェクト内で入力したファイル名 (< ファイル名 >.*) と一致するファイルを検索します。同じ名前のファイルが複数存在する場合、ファイルを選択するリストが表示されます。
- シンボル名。Source Insight はファイル名をシンボルとして扱うため、各ファイル引数はシンボルとして見なすことができます。Source Insight は、シンボルが含まれるファイルを開き、その場所にジャンプします。任意の数のシンボルを指定できます。シンボル名が (拡張子を除く) ファイル名、または他のシンボルと重複している場合、シンボルを選択するリストが表示されます。
- ワークスペース ファイルの名前 (拡張子 .WK3)。Source Insight は、ワークスペースに含まれるファイルをすべて開きます。
- 設定ファイルの名前 (拡張子 .CF3)。Source Insight は、指定された設定ファイルをロードします。

ファイルを開く

Source Insight のコマンド ラインでは、ファイルを開くときに以下のオプションを指定できます。ファイル名の前にオプションでプラス記号 (+) と行番号を指定すると、ファイルを開いたときにその行が表示されます。

例 :

```
insight3 +100 file.c
```

FILE.C を開き、100 行目を表示します。

ファイル名の前にオプションでプラス記号 (+) のみを指定すると、カレント ワークスペースで開いているファイルに加えて、指定したファイルを開きます。

例 :

```
insight3 +file.c
```

カレント ワークスペースで開いているすべてのファイルに加えて、FILE.C を開きます。

プラス記号なしでファイルを指定した場合、カレント ワークスペースで開かれていたファイルは閉じられ、コマンド ラインで指定したファイルのみ開かれます。

例 :

```
insight3 file.c other.c
```

FILE.C と OTHER.C のみを開きます。カレント ワークスペースでそれまで開かれていたファイルは閉じられます。

ファイルの検索方法

コマンド ラインで相対ファイル パスを指定した場合、Source Insight は最初に Source Insight を開始したディレクトリに対する相対ファイルを検索します。ファイルが見つからない場合、プロジェクトを開いて、プロジェクトのファイル リストのファイルを検索します。

つまり、任意のディレクトリで Source Insight を開始できます。指定したファイルがカレント プロジェクトに含まれていれば、Source Insight はそのファイルを検索します。スタートアップ ディレクトリ (または相対ディレクトリ) に同じ名前のファイルがある場合、Source Insight はそのファイルを開きます。

ワークスペースを開く

通常のテキスト ファイルに加えて、Source Insight のワークスペース ファイルの名前も指定できます。

例 :

```
insight3 myset.wk3
```

ワークスペース ファイル myset.wk3 のすべてのファイルを開きます。

通常のファイル名とワークスペース ファイルを指定することもできます。

例 :

```
insight3 +100 file.c myset.vw print.c myset2.wk3
```

ファイルとワークスペース ファイルの両方を開きます。

コマンド ライン オプション

Source Insight のコマンド ラインでは、以下のオプションを指定できます。

新規プログラム インスタンスの抑制

```
-i <コマンド ラインの残り >
```

既に実行している Source Insight のインスタンスに、コマンド ラインの残りを指示します。実行しているインスタンスがない場合、新規インスタンスが開始されます。

例 :

```
insight3 -i myfile.cpp
```

既に実行している Source Insight のインスタンスに、myfile.cpp を開くように指示します。

Source Insight コマンドの実行

```
-c <コマンド名 >
```

Source Insight を開始して、指定されたコマンドを実行します。ビルトイン コマンド、定義されたカスタム コマンド、またはマクロ コマンドです。

開くプロジェクトの指定

```
-p <プロジェクト名 >
```

カレント プロジェクトを閉じて、指定されたプロジェクトを開きます。プロジェクトが存在しない場合、Source Insight はエラー メッセージを表示します。

例 :

```
insight3 -p myproj
```

カレント プロジェクトを閉じる

```
-pc
```

カレント プロジェクトを閉じます。他のプロジェクトは開かれません。

例 :

```
insight3 -pc
```

テンポラリ プロジェクトの使用

```
-pt <プロジェクト名 >
```

カレント プロジェクトを閉じて、指定されたプロジェクトを開きます。-p オプションとは異なり、次回 Source Insight を実行するとき、旧カレント プロジェクトが開かれます。カレント プロジェクトを変更しないでバッチ ファイルに Source Insight のコマンドを追加する場合に便利です。

例 :

```
insight3 -pt mail
```

シンボルの検索

```
-f <symbol_name>
```

symbol_name で指定されたシンボル名を検索し、ファイルを開いて、シンボルの挿入ポイントに移動します。 シンボルが見つからない場合、Source Insight はエラー メッセージを表示します。このコマンドは、ファイルではなく解析したシンボルを検索していることを Source Insight に明示的に知らせるため、ファイル名の代わりにシンボルを指定することとは違います。

例 :

```
insight3 -f DoIdle
```

プロジェクト ファイルの同期

```
-u
```

Source Insight の起動時にすべてのプロジェクト ファイルを更新します。[Project] メニューで [Synchronize Files] コマンドを使用することと同じです。

例 :

```
insight3 -u
```

バッチ モードでのファイルの同期

```
-ub
```

Source Insight がファイルを更新した後に終了することを除いて、-u と同じです。このコマンドを使用すると、バッチ ファイルに Source Insight プロジェクトを同期するコマンドを追加できます。

例 :

```
insight3 -ub
```

スプラッシュ スクリーンの抑制

```
-s
```

Source Insight の起動時に表示されるスプラッシュ スクリーンをオフにします。

ユーザーレベル コマンド

*コマンド*とは、ユーザーがメニュー項目を選択するか、またはキーストロークを行うことにより Source Insight が実行する、ユーザーレベル

の操作のことを言います。たとえば、[Open] コマンドはファイルを開き、[Save] コマンドはファイルを保存します。各コマンドには名前と、動作があります。

コマンドはメニュー、キーストローク、およびマウス クリックに割り当て可能なリソースです。そして、それらの割り当ては設定の一部です。

コマンドにキーを割り当てるには、[Options] > [Key Assignments] を使用します。

キーストロークとマウス クリックがコマンドに割り当てられます。たとえば、Ctrl+O キーストロークは [Open] コマンドに割り当てられています。1 つ以上のキーストロークをコマンドに割り当て可能です。キー割り当てをカスタマイズするには、[Key Assignments] コマンドを使用します。

メニューにコマンドを割り当てるには、[Options] > [Menu Assignments] を使用します。

コマンドがメニューに割り当てられます。たとえば、[Open] コマンドは [File] メニューに割り当てられています。メニューの内容をカスタマイズするには、[Menu Assignments] コマンドを使用します。

Source Insight では、コンパイラや他の外部ツールを Source Insight から起動する*カスタム コマンド*を定義することもできます。

カスタム コマンド

Source Insight に用意されている標準コマンドに加えて、カスタム コマンドを定義できます。カスタム コマンドは、シェル バッチ ファイルに似ています。外部のコマンド ライン プログラムと Windows GUI プログラムを実行します。Source Insight は、バックグラウンドでカスタム コマンドを実行できます。カスタム コマンドの出力は、ファイルに保存するか、貼り付けることができます。

カスタム コマンドは、Source Insight の内部から起動するシェル コマンドです。

[Options] > [Custom Command] ダイアログ ボックスを使用して、カスタム コマンドを定義、実行できます。一度定義したら、カスタム コマンドは、他のコマンドと同様に処理できます。メニューに割り当てたり、キーストロークを割り当て可能です。カスタム コマンドは現在の設定の一部です。

カスタム コマンドはコンパイルの実行に便利です。コンパイラや make プログラムを実行するカスタム コマンドを定義することで、コンパイラ エラー メッセージをキャプチャしてエラーを解析し、間違っているソース コードを自動的に指すソース リンクを生成できます。

カスタム コマンドを使用するさまざまなテキスト フィルタを実装することも可能です。たとえば、ソート フィルタを実行して現在の選択範囲に出力を貼り付ける [Sort] カスタム コマンドを定義できます。

詳細は、176 ページの「Custom Commands」、222 ページの「Key Assignments」、および 248 ページの「Menu Assignments」を参照してください。

Source Insight のカスタマイズ

Source Insight では多くの項目をカスタマイズできます。オプションのセット全体は設定と呼ばれ、設定ファイルを保存およびロードできます。

カスタマイズ設定は設定ファイルに格納されます。

通常、カスタマイズ設定はすべて Source Insight プログラム ディレクトリ、またはユーザー データ ディレクトリにある global.cf3 というファイルに保存されます。これを「グローバル」設定ファイルと言います。[Project Settings] ダイアログ ボックスのオプションを使用して、プロジェクトで独自の設定ファイルを使用できます。

選択 . [Preferences] コマンドで、ファイル制御、表示オプション、言語サポートなどのさまざまなユーザー オプションを設定できます。

ドキュメント オプション . [Document Options] コマンドで、ドキュメント タイプを定義および変更できます。ドキュメント タイプとは、ファイルの名前や拡張子によって Source Insight の動作を指定できるファイル タイプのことです。

キー割り当て . [Key Assignments] コマンドで、Source Insight のキーボードを再配置できます。Source Insight の各コマンドがこのダイアログ ボックスにリストされます。各コマンドにキーストロークやマウス ボタン ショートカットを指定可能です。

メニュー割り当て . [Menu Assignments] コマンドで、Source Insight のメニュー バーをカスタマイズできます。Source Insight の各コマンドがこのダイアログ ボックスにリストされます。任意のメニューに各コマンドを割り当て可能です。

設定のロードと保存

[Save Configuration] コマンドを使用して設定を保存できます。[Load Configuration] コマンドを使用して設定オプションをロードできます。設定をロードすると、ディスクに格納されている現在の設定が置換されます。

[Save Configuration] コマンドを使用して、キー割り当てのような個々の設定を保存することも可能です。

プロジェクト設定

プロジェクト設定はプロジェクトに格納され、設定ファイルには格納されません。

[Project Settings] コマンドを使用して、各プロジェクトに特別なオプションを設定できます。前述した他のカスタマイズとは異なり、プロジェクト設定はプロジェクトに格納され、設定ファイルには格納されません。

プロジェクト固有の設定

プロジェクト固有の設定を行うには、[Project Settings] コマンドを使用して設定を指定します。プロジェクトに独自の設定ファイルがある場合、プロジェクトを開くときにすべてのユーザーの選択が変更されます。

設定の保存

カスタマイズ設定は設定ファイルに格納されます。

Source Insight はあらゆる面でカスタマイズ可能です。たとえば、画面の色や、コマンドへのキーストロークの割り当てなど。この一連のオプションは設定と呼ばれます。設定には、通常はあまり変更されない情報が含まれます。通常は、Source Insight のインストール時にのみ設定を行い、必要に応じて変更します。

設定には、[Options] メニューや [Preferences] ダイアログ ボックスで指定するオプションのほとんどが含まれます。

- 表示オプション
- 言語オプション
- 構文フォーマット スタイル
- 構文修飾
- キー割り当て
- メニュー割り当て
- ツールバー
- カスタム コマンド
- ドキュメント タイプとオプション
- ページ設定
- ファイル オプション
- 全般オプション
- シンボル参照オプション
- 入力オプション

設定ファイルは、オプション変更時に更新されます。

実行中の Source Insight に影響する設定はカレント設定と呼ばれます。カスタマイズ設定を変更するか、新規設定をロードすると、ディスク上の設定ファイルも更新されます。

設定ファイル

Source Insight を終了すると、カレント設定が設定ファイルに保存されます。Source Insight を開始すると、前のセッションの設定情報がリストアされます。

プロジェクトが開かれていない場合、カレント設定ファイルの名前は global.cf3 で、Source Insight プログラム ディレクトリ (セットアップ プログラムで Source Insight をインストールした場所) に格納されます。

プロジェクトが開かれている場合、カレント設定ファイルの名前は [Project Settings] ダイアログの設定に応じて < プロジェクト名 >.cf3 または Global.cf3 のいずれかになります。

設定ファイルの格納場所

設定ファイルは、通常、「My Documents\Source Insight」フォルダ内の「Settings」フォルダに保存されます。

Source Insight にログインして実行するユーザーごとに、「My Documents\Source Insight」フォルダ内のユーザー データ ディレクトリが作成されます。そのため、各ユーザーは別々に独自の設定を格納できます。

ヒント： すべてのカスタマイズ設定を含むグローバル設定ファイルのバックアップをとることを推奨します。[Load Configuration] コマンドを使用するか、Source Insight 内のカスタマイズ設定を変更すると、設定ファイルは自動的に変更されます。

ヒント： Source Insight を更新する場合、あらかじめバックアップをとることを推奨します。Source Insight の新しいビルドは以前の設定ファイルと互換性がありますが、以前のビルドで新しい設定ファイルを開けない場合があります。以前のビルドに戻す場合は、古い設定ファイルを使用してください。

設定のロード

[Load Configuration] コマンドを使用して、ロードする設定ファイルを選択します。設定項目全体、またはメニュー項目のような設定の一部をロードできます。

独自の設定ファイルがあるプロジェクトを開くと、プロジェクトの設定もロードされます。

設定の保存

[Save Configuration] コマンドを使用して、現在の設定を他の設定ファイルに保存できます。設定ファイルを保存することで、メニュー、

キーストローク割り当て、画面の色、その他が異なる Source Insight のカスタマイズ バージョンを作成できます。[Save Configuration] コマンドを使用して現在の設定の一部を保存することもできます。

Source Insight は変更されたオプションを自動的に保存するので、通常は [Save Configuration] コマンドを使用する必要はありません。

ワークス ペースの保存とリストア

ワークスペースにはセッション間で変更されるセッション状態が含まれます。編集中に複数のファイルを開く場合があります。各ファイルでテキストを選択できます。各ファイルでブックマークを設定できます。この情報はワークスペースの一部分です。ワークスペースには、以下の項目が含まれます。

- 開いたファイルの名前
- 各ファイルの選択範囲
- 各ファイルのブックマーク
- 選択履歴 (各ファイルで選択していた場所)
- 各ソース ファイル ウィンドウのサイズと位置
- 文字列の検索と置換
- ソース リンク
- ダイアログ ボックスの入力履歴
- コマンド記録 (311 ページの「Start Recording」を参照)

Source Insight を終了すると、カレント ワークスペースがワークスペース ファイルに保存されます。Source Insight を開始すると、前のセッションのワークスペースがリストアされます。

プロジェクトが開かれていない場合、カレント ワークスペース ファイルの名前は Global.wk3 です。プロジェクトが開かれている場合、カレント ワークスペース ファイルの名前は [Project Settings] ダイアログの設定に応じて <プロジェクト名 >.WK3 または Global.wk3 のいずれかになります。

ワークス スペースのロードと保存

ワークスペースを開くには、[File] > [Open Workspace] コマンドを使用します。

[File] > [Save Workspace] コマンドを使用して、カレント ワークスペースの内容を他のワークスペース ファイルに保存できます。

ワーク スペースでタスクを管理する

複数のワークスペースを使用することで、タスクをグループにして個別に保存できます。

たとえば、2 つの異なるタスクで作業しているとします。タスク 1 で、タスクのソース ファイルをすべて開き、作業を開始します。タスク 1 の作業を停止する場合、[Save Workspace] コマンドを使用してファイル「Session1」を保存します。すべてのファイルを閉じ、タスク 2 のソース ファイルを開いて作業を開始します。タスク 2 の作業を停止してタスク 1 の作業に戻す場合、[Save Workspace] コマンドを使用してファイル「Session2」を保存し、次に [Open Workspace] コマンドを使用してファイル「Session1」を開きます。

パフォーマンスの調整

プロジェクトのサイズとマシンの種類に応じて、最適なパフォーマンスが得られるように Source Insight を調整します。[Preferences] ダイアログ ボックスと [Project Settings] ダイアログ ボックスにパフォーマンスに影響するオプションが含まれています。

パフォーマンスに影響する要因

Source Insight は、プログラミング エディタの機能の制限を拡張するように設計されています。このため、非常に遅いシステムや最適化されていないシステムではパフォーマンスに大きく影響する特徴があります。このセクションでは、Source Insight のパフォーマンスに影響する要因と制御方法について説明します。

マシンの速度

Pentium II 以上を推奨します。

Source Insight のすべての機能を活用するには、Pentium II クラスのマシン (またはそれ以上) を推奨します。マシンが要件を満たしていない場合、多くのオプションをオフにすることで、機能は制限されますがパフォーマンスを向上できます。

ほとんどの場合、正しいオプションを選択することで、バージョン 2.x の機能と速度を再現できます。

Source Insight には、速度の低下を最小限に抑えてより多くの機能を提供する、多くの新しいパフォーマンス最適化機能が用意されています。

プロジェクトのサイズ

プロジェクトのサイズは、パフォーマンスとメモリ使用量に影響します。

プロジェクトのサイズは、Source Insight の特定の機能のパフォーマンスに大きく影響します。プロジェクトのサイズは、ファイル数と宣言されているシンボルの数で測定できます。

プロジェクトのサイズを確認するには、[Project Report] コマンドを使用します。

プロジェクトのサイズを確認するには、[Project] > [Project Report] コマンドを使用してプロジェクト レポートを出力します。Source Insight は、プロジェクト レポートの先頭に、プロジェクトに含まれるファイルの数、シンボルの数、およびシンボル インデックス エントリの数を出力します。[Project] > [Rebuild Project] ダイアログ ボックスの下部にも同じ情報が表示されます。(情報を確認するためにプロジェクトをリビルドする必要はありません。)

Source Insight 3.5 は、バージョン 2.1 よりも多くのシンボル情報を処理します。この結果、シンボル インデックスのサイズも大きくなりました。非常に大規模なプロジェクトではパフォーマンスに影響します。

プロジェクト インデックスの設定

[Project Settings] コマンドはプロジェクトのインデックス オプションを管理します。

[Project Settings] ダイアログ ボックスには、プロジェクトのインデックス オプションが含まれています。すべてのインデックス オプションを有効にした場合、プロジェクトが大きくなると、パフォーマンスに影響します。

インデックスが大きすぎると、以下のような現象が起こります。

- 大規模なプロジェクトのビルドまたはリビルド中にディスク スラッシングが発生します。非常に大規模なプロジェクトでは、リビルドの最終段階でディスクにアクセスする回数が非常に多くなるのは正常です。
- プロジェクトを開くまたは閉じるのに時間がかかります。
- 個々のファイルの同期処理が遅くなります。
- [Browse Project Symbols] ダイアログ ボックス (F7) の表示に時間がかかります。最初に使用するときに数秒停止するのは正常です。
- 128 MB のシステムで数百万のインデックス エントリが生成されます。

大規模なプロジェクトでシラブル インデックスを実行すると、システムが遅くなり、メモリの使用量が増加します。

大規模なプロジェクト (200,000 シンボル以上) の場合、シンボルのシラブル インデックスをオフにしてください。[Project: Rebuild Project] を選択してダイアログ ボックスの下部に表示される情報を確認することでデータベースのサイズが予測できます。情報を確認したら、[Cancel] ボタンをクリックしてダイアログ ボックスを閉じてください。インデックス エントリの数が百万を超えると、システムは遅くなります。マシンにメモリを追加すると、パフォーマンスが向上します。

シラブル インデックスを削除するには、[Project] > [Project Settings] コマンドを実行して、[Quick browsing for symbol syllables] をオフにします。次に、[Rebuild Project] コマンドを使用してプロジェクトをリビルドします。

シンボルのメモリ使用量

Source Insight は、プロジェクトのシンボル インデックス エントリごとに 16 バイトの RAM を使用します。すべてのシンボル インデックス オプションを有効にしたプロジェクト (263 ページの「Project Settings」を参照) では、シンボル定義の約 5 倍のインデックス エントリを使用します。このため、たとえば、プロジェクトに 100,000 個のシンボル定義がある場合、インデックス エントリの数は約 500,000 個になります。つまり、16 x 約 500,000 = 約 8,000,000 バイト必要です。

仮想メモリ容量

大規模プロジェクトは多くの仮想メモリを使用します。

Source Insight は、プロジェクトのサイズに比例するメモリを使用します。プロジェクトが大規模になると、Source Insight が必要なメモリも増加します。

Win32 プログラミング インターフェイスでは、Source Insight のようなプログラムがマシンに物理的に搭載されている RAM よりも多くのメモリを使用できます。この機能は、オペレーティング システムが「ページング」と呼ばれるプロセスによりハード ディスクを使用して RAM をエミュレートするため、仮想メモリと呼ばれます。

Source Insight は、2 つの方法でメモリ (仮想メモリ) を使用します。最初の方法は、Source Insight がヒープと仮想メモリを割り当てる方法です。割り当てに使用されるサイズは最も小さくなります。この場合でも、割り当て可能かどうかシステムのページング ファイルのサイズを確認する必要があります。プロジェクトで 100,000 個のシンボルが宣言されるごとに約 2-5 MB 必要です。

大規模なプロジェクト データベース ファイルはメモリにマップされ、報告されるメモリ使用量が多くなります。

Source Insight は、メモリマップ ファイルの数を増やすこともできます。これはファイルを仮想アドレス空間へマップする Win32 の機能で、プログラムからはファイルがメモリのブロックのように見えます。Source Insight は、メモリマップ ファイルを使用して、ソース ファイルやデータベース ファイルにできるだけ速くアクセスできるようにします。

Source Insight がプロジェクト データベース ファイルをメモリにマップする場合、ファイルが開かれたときに大量の仮想アドレス空間が使用されます。Source Insight がデータベースの異なるレコードにアクセスするので、オペレーティング システムはデータベース ファイルのコンテンツを保持するために、より多くの物理メモリをコミットします。オペレーティング システムは、他のプログラムやシステムで使用するために常に一部のメモリを予約します。

プロジェクト データベースのサイズはプロジェクトで宣言されたシンボルの数に比例します。このため、プロジェクトが非常に大きい場合、Source Insight によって使用されるメモリの量が 100 MB を超えることもあります。しかし、タスク マネージャや他のパフォーマンス モニタリング ツールで報告される使用中のメモリの大部分は、メモリにマップされたシンボル データベースの一部を表していることを理解する必要があります。Source Insight でこの量の物理メモリが必要な訳ではありません。

物理メモリ容量

プロジェクトが大規模になると、Source Insight が必要なメモリも増加します。システムに多くの RAM があれば、パフォーマンスが向上します。Source Insight の最小限の機能を使用する場合でも、少なくとも 64 MB の RAM が必要です。200,000 以上のシンボルを使用する大規模なプロジェクトでは、128 MB 以上の RAM を推奨します。

オペレーティング システム

Windows 2000/XP を推奨します。

仮想メモリに関して最良のパフォーマンスを得るには、Windows 2000/XP またはそれ以降で Source Insight を実行することを推奨します。これらのオペレーティング システムは、Windows 9.x/Me よりもメモリを有効に活用します。

カスタム解析表現

Source Insight では、独自のカスタム解析正規表現を使用して、言語を増やすことが可能です。ソース ファイルを右クリックし、[Language Properties] を選択することで、これらの表現を編集できます。

カスタム解析表現の使用には注意してください。

表現が多すぎる場合、または一致が本質的に遅い正規表現を使用している場合、ファイルの解析に時間がかかります。

正規表現は、簡単に一致するパターンで開始してあまり一致しないパターンで終了する場合、一致が遅くなります。
例 :

```
[a-z]*(\([a-z]+\))
```

この表現は、パターン [a-z]* (任意のゼロ以上の英文字) で開始するため、一致が遅くなります。ファイルのほとんどの文字が一致します。

ネットワーク上のファイルの場所

リモート ネットワーク ドライブからプロジェクトにソース ファイルを追加することは可能ですが、ネットワーク ドライブへのアクセスは一般的に遅いため、Source Insight の速度が低下します。

さらに、ネットワーク ドライブにプロジェクト データ フォルダを設定することもパフォーマンスの低下につながります。

「My Documents」フォルダの場所

Windows では、「My Documents」フォルダはローカル、またはリモート ネットワーク ドライブのいずれかに存在する仮想フォルダです。Source Insight は「My Documents」フォルダ内にユーザーごとのデータを格納するため、「My Documents」フォルダがネットワーク ドライブ上に存在している場合、一部の処理は遅くなります。特に、C# コードを編集している場合、.NET Framework シンボルは「My Documents\Source Insight\Projects\NetFramework」フォルダに格納されるため、.NET Framework のシンボル補完が遅くなります。

できれば、ローカル ドライブの「My Documents」フォルダを指定してください。ローカル ドライブに指定できない場合、Source Insight がユーザーのデータ フォルダにローカル ドライブを使用するようにレジストリ エントリを設定します。

regedit で次のキーを検索します。

```
HKEY_CURRENT_USER\Software\Source Dynamics\Source
  Insight\3.0\Paths
```

「UserDataDir」という名前で新規文字列値を追加します。ユーザーのデータ フォルダごとに、文字列値をフォルダのフル パスに設定します。

プログラムのスピードアップ

このセクションでは、Source Insight のスピードアップについて異なる点から説明します。

構文フォーマットのスピードアップ

Source Insight には、ソース コードのフォーマット用に多くの便利な表示機能が用意されています。一部の機能では、複雑な処理が必要です。

フォーマット オプションは、表示速度に影響します。

表示コードは、許容されるパフォーマンスでより多くの機能を提供できるように大幅に高速化されました。しかし、バージョン 3.5 ではより多くの情報が表示されるようになったため、表示速度がバージョン 2.1 よりも多少遅くなりました。表示する情報をバージョン 2.1 と同じになるように減らすことで、スピードアップできます。

[Preferences: Syntax Formatting] ダイアログ ボックスで、表示方法を細かく指定できます。各オプションには、パフォーマンス コストがあります。重要なオプションを以下にリストします。コストが最も低いオ

プションから、最も遅くコストが最も高いオプションの順にリストしています。

1. Apply Styles for Language Elements: References to members
2. Apply Styles for Language Elements: Symbol declarations
3. Apply Styles for Language Elements: Function-local symbol declarations
4. Apply Styles for Language Elements: References to function-local symbols
5. Apply Styles for Language Elements: References to non-function-local symbols
6. Symbol Reference Lookups: Search in the Project Symbol Path
7. Symbol Reference Lookups: Qualify references to members

「参照」のフォーマットは「宣言」より遅くなります。

一般に、「参照」の識別は「宣言」より遅くなります。プロジェクトが非常に大きい場合、各参照をシンボル ルックアップで解決する必要があるため、シンボルの参照の識別に時間がかかります。

シンボル参照エンジンは、「表示速度」でシンボルの参照をフォーマットできるように大幅に最適化されました。多くの情報はプロセス中にキャッシュされるため、最初にファイルを開いてスクロールすると、速度が低下していることがわかります。その後、ファイルの表示は速くなります。

表示する情報を増やすと速度は遅くなります。

速度を最優先する場合もありますが、ほとんどの場合は、速度が多少低下しても、生産性を高める有益な情報を表示することを優先します。

[Browse] ダイアログ ボックスでの入力のスピードアップ

プロジェクトが大規模で、**[Project Settings: Quick browsing for symbol syllables]** がオンになっている場合、シンボル リストの入力とフィルタは遅くなります。Source Insight は大規模なシラブル インデックスでシラブル マッチングを実行します。

大規模なプロジェクトではシラブル マッチングは遅くなります。

2 つの異なる方法でこの問題に対処できます。[Project Settings] ダイアログ ボックスでシラブル インデックスをオフにする方法と、[Preferences: Typing] ダイアログ ボックスでリスト フィルタの動作を変更する方法です。

[Project Settings] オプションは、シンボル データベースに格納およびインデックス付けする内容を制御します。

[Preferences: Typing] オプションは、情報がリストでどのように使用されるかを制御します。リストのシンボル シラブル マッチング機能を使用するには、データベースにシラブル情報が必要です。(ファイル名リスト用ではありません。)

メモリに余裕がある場合は、**[Project Settings: Quick browsing for symbol syllables]** をオンにして、**[Preferences: Typing: Match syllables while typing]** をオフにします。入力中のシラブル マッチングが無効になっていても、前にスペースを入力することで、シラブル マッチング

を使用できます。つまり、シラブルの前にスペースを追加して入力することで、シラブル マッチングのオン、オフを切り替えることができます。

リストのシラブルを一致するときに大量のディスク スラッシングが発生した場合、利用可能なメモリに対してプロジェクトが大きすぎます。

[Preferences: Typing] で、[Match syllables while typing] と [Match members while typing] オプションを両方ともオフにすると、リスト フィルタがバージョン 2.1 の機能に戻され、速度が速くなります。.

プロジェクトのビルドと同期処理のスピードアップ

非常に大規模なプロジェクトは、ビルドやリビルドに時間がかかります。スピードアップするには、以下の操作を実行してください。

- [Project Settings] で、**[Quick browsing for symbol syllables]** をオフにします。このインデックス オプションは、多くのメモリを使用し、インデックス処理が遅くなります。
- [Project Settings] で、**[Quick browsing for member names]** をオフにします。このインデックス オプションも、多くのメモリを使用します。
- プロジェクトに追加するドキュメント タイプ (たとえば、ファイル タイプ) の数を減らします。デフォルトでは、Visual Basic ファイルや ASP ファイルのような多くのドキュメント タイプが定義されています。ソース ツリーに使用しないドキュメント タイプが含まれている場合、プロジェクトから削除してください。[Document Options] を実行して、プロジェクトから削除するドキュメント タイプごとに **[Include when adding to projects]** チェック ボックスをオフにすることで、これらのドキュメント タイプをプロジェクトから削除できます。
- プロジェクトをより小さなプロジェクトに分割します。プロジェクト間で [Jump To Definition] コマンドを使用する場合、[Preferences: Symbol Lookups] ダイアログ ボックスで定義するプロジェクト シンボル パスに「サブプロジェクト」を追加できます。

リレーション ウィンドウのスピードアップ

リレーション ウィンドウは Source Insight 3.0 の新機能です。シンボル間のリレーションを表示する機能です。選択されている項目に関する情報をバックグラウンドで自動的に表示します。たとえば、関数呼び出しを選択した場合、該当する関数を開始するコール ツリーを表示します。

リレーション ウィンドウは多くの処理を行います。

リレーション ウィンドウは多くの処理を行います。一部のリレーションシップは遅くなります。リレーションシップは最も速いものから最も遅いものまでリストされた 3 つのカテゴリに分類されます。

- **Contains** – 現在のシンボルの内容を表示します。たとえば、構造体のメンバを表示します。
- **Calls** – 現在のシンボルによって参照されている他のシンボルを表示します。たとえば、現在の関数に呼び出されている関数を表示します。
- **References** – 現在のシンボルを参照している他のシンボルを表示します。たとえば、現在の関数を呼び出している関数を表示します。

「参照」を表示する場合はさらに処理が多くなります。

非常に大規模なプロジェクトでは、「References」リレーションシップは処理が最も遅くなります。中規模サイズのプロジェクト (約 200,000 コード行) であれば、Pentium II マシンでも問題なく処理できます。

リレーションを非参照型のリレーションに制限すると、リレーション ウィンドウの表示が速くなります。

ロックとリフレッシュを手動で行います。

リレーション ウィンドウをロックしたままにします。リレーション ウィンドウをロックするには、リレーション ウィンドウ ツールバーの [Lock Relation Window] ボタンをクリックします。リレーション ウィンドウは、[Refresh Relation Window] コマンドを使用するか、リレーション ウィンドウ ツールバーの [Refresh Relation Window] ボタンをクリックして、いつでもリフレッシュできます。

自動補完のスピードアップ

識別子を入力するとき、一致する識別子名を示す自動補完ウィンドウがポップ アップします。文字を入力するたび、Source Insight はそのファイルのシンボル データが「古く」なると見なします。最も正確に自動補完が行われるようにするには、文字が入力されるたびにファイルを解析し直す必要があります。このオプションは、[Preferences: Symbol Lookups] ダイアログ ボックスにあります。

[Parse locally before lookup] オプションをオンにすると、Source Insight は自動補完ウィンドウを表示する前にファイルを解析し直します。高速なマシンや小さなファイルでは、あまり速度には影響しません。しかし、このオプションをオフにする方が動作は速くなります。

.NET Framework 自動補完のスピードアップ

「My Documents」フォルダがネットワーク ドライブにある場合、C# コードを編集するときに、.NET Framework のシンボル補完は遅くなります。原因は、.NET Framework シンボルが「My Documents\Source Insight\Projects\NetFramework」に格納されるためです。 143 ページの「「My Documents」フォルダの場所」も参照してください。

ファイル検索のスピードアップ

Source Insight で複数のファイルを検索する方法はいくつかあります。[Search Files]、[Lookup References]、および [Search Project] コマンドはすべて、複数ファイルの検索を行います。

[Lookup References] コマンドを使用した検索が最も高速です。

Source Insight で最も高速な検索は、[Lookup References] コマンドを使用した検索です。この方法で検索する場合、Source Insight は単一の単語全体への参照を検索します。単一の単語全体を検索する場合、Source Insight は検索を高速に行うために特別なインデックス ファイルを使用します。可能であれば、[Search Files] の代わりに [Lookup References] コマンドを使用してください。

[Lookup References] コマンドのスピードアップ

[Lookup References] コマンドには、速度に影響するいくつかのオプションがあります。

状況依存オプションを使用すると、[Lookup References] は遅くなります。

[Smart Reference Matching] オプションは、検索が状況依存であることを意味します。検索結果には、周りのコンテキストを使用して、指定したシンボルへの参照と一致するもののみが含まれます。同じ文字列の検索を瞬時の解析が必要なシンボル ルックアップで行う必要があるため、このオプションを使用すると処理は遅くなります。このオプションをオフにすると、Source Insight のバージョン 2.x のように動作します。

[Skip Comments] と [Search Only Comments] オプションでも検索は多少遅くなりますが、[Smart Reference Matching] オプションほどではありません。

[Smart Rename] コマンドも、同じメカニズムを使用しています。

Source Insight により作成されるファイル

Source Insight は、ハード ディスクに以下のファイルとファイル タイプを作成します。

プログラム ディレクトリのファイル

以下のファイルが、Source Insight のインストール時にインストール ディレクトリに作成されます。インストール ディレクトリは、セットアップ プログラムの実行時に指定したインストール先です。

ファイル	説明
Insight3.exe	Source Insight のプログラム。
Insight.hlp	Source Insight のヘルプ ファイル。

ファイル	説明
ReadMe.txt	最新の情報を含むテキスト ファイル。
Sihook.exe	Source Insight がカスタム コマンドを実行する際に利用するユーティリティ プログラム。
FileAlias.txt	指定したファイル名に関連付けられているドキュメント タイプよりも優先される、ファイル名の別名ファイル。
*.CLF	[Preferences: Languages] ダイアログ ボックスからカスタム言語をエクスポートして作成するカスタム言語ファイル。

ユーザーごとのデータ フォルダ

Windows では、「My Documents」フォルダはユーザーごとに別々に存在する仮想フォルダです。Source Insight は、「My Documents\Source Insight」フォルダ内にユーザーごとのデータを格納します。

Source Insight のプロジェクト データ ファイルは個々のユーザー データ エリアに保存されるため、同じマシンの他のユーザーからアクセスできません。

できれば、ローカル ドライブの「My Documents」フォルダを指定してください。ローカル ドライブに指定できない場合、Source Insight がユーザーのデータ フォルダにローカル ドライブを使用するようにレジストリ エントリを設定します。

regedit で次のキーを検索します。

```
HKEY_CURRENT_USER\Software\Source Dynamics\Source Insight\3.0\Paths
```

「UserDataDir」という名前で新規文字列値を追加します。ユーザーのデータ フォルダごとに、文字列値をフォルダのフル パスに設定します。その後、Source Insight を再起動してプロジェクトを作成し直してください。

ユーザーごとに作成されるファイル

Windows では、ユーザーは、「My Documents」として知られる個人のデータ フォルダを使用します。Source Insight は、「My Documents」フォルダに「Source Insight」フォルダを作成し、ユーザー固有のデータを格納します。

ユーザー データ ディレクトリには、ユーザー固有のグローバル設定ファイル (global.cf3) 、「プロジェクトを開かない」モード用のワークスペース、およびプロジェクト データが含まれています。

以下のフォルダが、カレント ユーザーの Source Insight フォルダに作成されます。

ファイルまたはフォルダ	説明
Backup	Source Insight で保存されるバックアップ ファイルを含むフォルダ。
Clips	クリップ ウィンドウにリストされるクリップ ファイルを含むフォルダ。独自のテキスト ファイルをコピーすると、クリップ ウィンドウに自動的に表示されます。
Projects	ユーザーのマシンで作成された Source Insight プロジェクトを含むフォルダ。プロジェクトごとにサブフォルダが作成されます。
Projects\Projects.db3	作成されたすべてのプロジェクトのリストを含むファイル。
Projects\Base	「Base」プロジェクト用のプロジェクト ファイルを含むフォルダ。
Project\NetFramework	「NetFramework」プロジェクト用のプロジェクト ファイルを含むフォルダ。C# で使用される .NET Framework クラスのシンボル定義を含みます。
Settings	設定ファイルを含むフォルダ。
Settings\Global.CF3	プロジェクトを開いていない場合、およびプロジェクトが開かれていて [Project Settings] コマンドで「グローバル設定」を指定した場合に使用されるグローバル設定ファイル。
c.tom	C/C++ トークン マクロ。
*.RCO	異常終了後にデータを修復するために必要な情報を含むクラッシュ リカバリ ファイル。セッションがクラッシュした場合にのみ存在します。
Global.WK3	プロジェクトを開いていない場合に使用されるセッション状態を含むグローバル ワークスペース ファイル。

すべてのユーザー用のテンプレート設定

Source Insight プログラム ディレクトリに global.cf3 という名前で設定ファイルを保存すると、Source Insight を実行するすべての新規ユーザーに global.cf3 ファイルがコピーされます。したがって、Source Insight プログラム ディレクトリにある global.cf3 ファイルを、新規ユーザー向けのテンプレート設定として使用できます。

プロジェクトごとに作成されるファイル

Source Insight プロジェクトを作成すると、プロジェクトごとに以下のデータ ファイルが作成されます。「Name」は指定したプロジェクトの名前を意味します。

ファイル	説明
Name.pr	プロジェクトのファイルのリストを含むメイン プロジェクト ファイル。
Name.wk3	プロジェクト ワークスペース ファイル。
Name.cf3	プロジェクト固有の設定ファイル。
Name.po	プロジェクト オプション。
Name.ps	シンボル定義データベース。
Name.pri	シンボル参照インデックス。
Name.pfi	プロジェクト ファイル インデックス。
Name.imd, Name.imb	メイン シンボル インデックス。
Name.iad, Name.iab	メンバとシラブル用の補助シンボル インデックス。

第 5 章

コマンド リファレンス

この章は、ユーザーレベルのすべての Source Insight コマンドをアルファベット順にリストしたコマンド リファレンスです。各コマンドについて、詳細に説明します。

重要な概念の概要は、67 ページの第 4 章「Source Insight の概念」を参照してください。

コマンドの概要

コマンドとは、ユーザーがメニュー項目を選択するか、またはキーストロークを行うことにより Source Insight が実行する、ユーザーレベルの操作のことを言います。たとえば、[Open] コマンドはファイルを開き、[Save] コマンドはファイルを保存します。各コマンドには名前と、動作があります。

コマンドはメニュー、キーストローク、およびマウス クリックに割り当て可能なリソースです。そして、それらの割り当ては設定の一部です。

コマンドにキーを割り当てるには、[Options] > [Key Assignments] を使用します。

キーストロークとマウス クリックがコマンドに割り当てられます。たとえば、Ctrl+O キーストロークは [Open] コマンドに割り当てられています。1 つ以上のキーストロークをコマンドに割り当て可能です。キー割り当てをカスタマイズするには、[Key Assignments] コマンドを使用します。

メニューにコマンドを割り当てるには、[Options] > [Menu Assignments] を使用します。

コマンドがメニューに割り当てられます。たとえば、[Open] コマンドは [File] メニューに割り当てられています。メニューの内容をカスタマイズするには、[Menu Assignments] コマンドを使用します。

Source Insight では、コンパイラや他の外部ツールを Source Insight から起動する*カスタム コマンド*を定義することもできます。134 ページの「カスタム コマンド」も参照してください。

About Source Insight

[About Source Insight] コマンドを実行すると、Source Insight の著作権情報とバージョン番号を含むウィンドウが表示されます。Source Insight のバージョン番号とビルドの日付を参照するには、このコマンドを実行します。

Activate Menu コマンド

- Activate Edit Menu
- Activate File Menu
- Activate Help Menu
- Activate Option Menu
- Activate Project Menu
- Activate Search Menu
- Activate View Menu
- Activate Window Menu
- Activate System Menu
- Activate System Doc Menu

[Activate Menu] コマンドは、指定したメニューをメニュー バーからドロップダウンします。

Alt キーを押して Source Insight のメニュー バーを選択した後、文字キーを押すことで、対応するメニューを表示することもできます。たとえば、Alt を押した後に F を押すと、[File] メニューが選択されます。

Source Insight では、メニューの選択に Windows の標準的なキー操作である Alt+< メニュー文字 > を必ず使用する必要はありません。Alt キーをメニューの選択以外にも使用できます。Source Insight では、**[Key Assignments]** コマンドを使用して任意のキーを任意のコマンドに割り当てることができます。たとえば、[File] メニューにはデフォルトでは Alt+F キーが割り当てられていますが、F1 キーに割り当てを変更すると、F1 キーを押すことにより [File] メニューが表示されます。

Activate Global Symbol List

コンテキスト ウィンドウを有効にして、リストのすべてのプロジェクト シンボルを表示し、カーソルを上部のテキスト ボックスに移動します。テキスト ボックスに入力すると、入力した内容に基づいてグローバル シンボル リストがフィルタされます。Enter または Esc キーを押すと、ソース ファイル ウィンドウに戻ります。

Activate Relation Window

リレーション ウィンドウを開いて選択します。入力フォーカスはリレーション ウィンドウに移動します。

Activate Search Results

検索結果ウィンドウを有効にします。ウィンドウが開かれている場合、前面に表示します。検索結果に素早く戻ることができます。

Activate Symbol Window

シンボル ウィンドウを有効にして、カーソルを上部のテキスト ボックスに移動します。テキスト ボックスに入力すると、入力した内容に基づいてシンボル リストがフィルタされます。Enter または Esc キーを押すと、ソース ファイル ウィンドウに戻ります。

Add and Remove Project Files

ソース ファイルをカレント プロジェクトに追加します。または、ソース ファイルをカレント プロジェクトから削除します。

多くのファイルをプロジェクトに追加する主要な方法です。単一ファイル、ファイルのグループ、またはソース ディレクトリ ツリー全体を追加または削除できます。

プロジェクトに追加できるファイル

プロジェクトにはテキスト ファイルのみ追加できます。

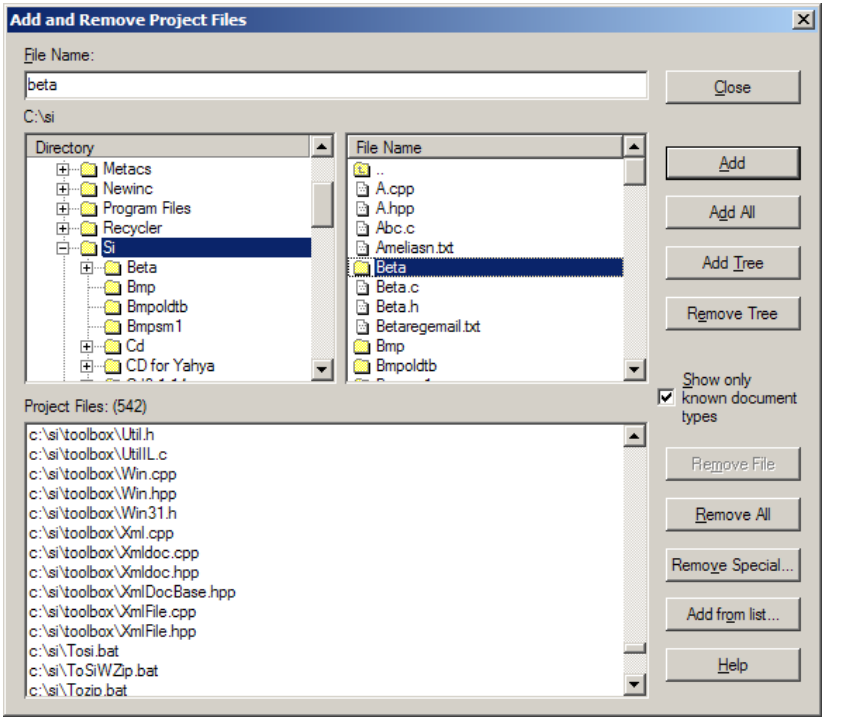
Source Insight プロジェクトは、プロジェクト ソース ファイルとテキスト ファイルのみで構成されます。バイナリ フォーマットのファイルを Source Insight プロジェクトに追加しても意味はありません。たとえば、EXE ファイルをプロジェクトに追加しても意味はありません。

デフォルトで定義されているドキュメント タイプは Source Insight で使用するソース ファイルの種類に対応しています。通常、これらの種類のファイルのみプロジェクトに追加します。

プロジェクトに追加するファイルの種類を制御するには、[Document Options] を使用します。

[Document Options] ダイアログ ボックスには、**[Include when adding to projects]** チェック ボックスがあります。このチェック ボックスを使用して、Source Insight がプロジェクトに自動的に追加するファイルの種類、または **[Add and Remove Project Files]** ダイアログ ボックスのリスト ボックスに表示されるファイルの種類を制御できます。

[Add and Remove Project Files] ダイアログ ボックス



[File Name] このテキスト ボックスに追加または削除するファイルの名前を入力します。入力した内容と一致するファイルがリストで自動的に選択されます。ワイルドカードを入力して Enter を押すと、ワイルドカードと一致するファイルのみを表示するようにファイル リストがフィルタされます。フル パスを入力するか、ドライブ名にコロン (:) を入力して他のディレクトリを指定することもできます。

[Directory] リスト カレント ドライブのディレクトリ ツリーを表示します。このリスト ボックスでディレクトリ名を選択すると、[File Name] リストに、そのディレクトリのファイルが表示されます。現在の作業ディレクトリとワイルドカード フィルタがリスト ボックスの上に表示されます。

[File Name] リスト 現在選択しているディレクトリにあるすべてのファイルのリストを表示します。このリスト ボックスからファイルを選択すると、そのファイル名が [File Name] テキスト ボックスにロードされます。既にプロジェクトの一部であるファイルはこのリスト ボックスに表示されません。

[Project Files] リスト プロジェクトに追加されているファイルをすべて表示します。このリストからファイルを選択して [Remove…] ボタンをクリックすると、プロジェクトからファイルが削除されます。

[Close] ダイアログ ボックスを閉じます。行った変更は保存されます。

[Add] 選択しているファイルをプロジェクトに追加します。ディレクトリが選択されている場合、カレント ディレクトリがそのディレクトリに変更されます。

[Add All] [File Name] リストのすべての項目を選択してプロジェクトに追加します。ディレクトリが含まれている場合、そのディレクトリの内容も追加されます。その場合、Source Insight はディレクトリの追加を確認します。

[Add Tree] ソース ツリー全体をプロジェクトに追加します。

ディレクトリが選択されている場合、ディレクトリ ツリー全体をプロジェクトに追加します。つまり、サブ ツリーのディレクトリはすべて、既知のドキュメント タイプと一致するファイル用にスキャンされ、プロジェクトに追加されます。

[Remove Tree] ディレクトリが選択されている場合、ディレクトリ ツリーにあるすべてのファイルを削除します。

[Show only known document types] 既知のドキュメント タイプに属するファイルのみをファイル リストに表示します。[Include when adding to project] オプションが選択されているドキュメント タイプのみが含まれます。既知のドキュメント タイプは **[Document Options]** コマンドを使用して変更できます。

チェックされていない場合、すべてのファイルが [File Name] リストに表示されます。

[Remove File] [Project Files] リストで選択しているファイルを削除します。

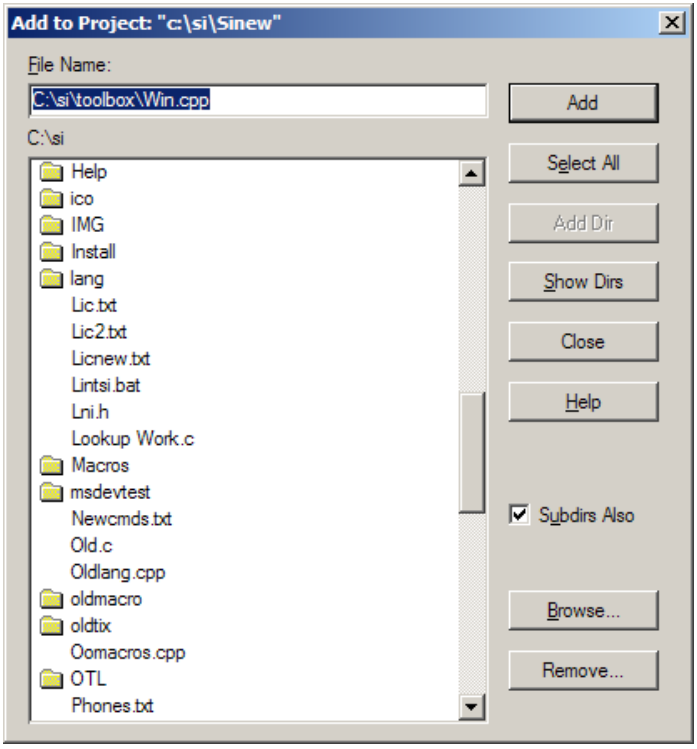
[Remove All] プロジェクトからすべてのファイルを削除します。プロジェクトは空になります。

[Remove Special] **[Remove File]** ダイアログ ボックスを表示します。このダイアログ ボックスで、*.h ファイルをすべて削除する、のような特殊な削除処理が実行できます。

[Add from list] **[Add File List]** ダイアログ ボックスを表示します。このダイアログ ボックスで、プロジェクトに追加するファイルとディレクトリのリストを含む入力テキスト ファイルを指定します。

Add File

ソース ファイルをカレント プロジェクトに追加します。このコマンドは、Source Insight の以前のバージョンに含まれていた古いコマンドです。ファイルをプロジェクトに追加またはファイルをプロジェクトから削除するには、新しく追加された **[Add and Remove Project Files]** コマンドを使用してください。



[File Name] プロジェクトに追加するファイルの名前です。ファイル名またはワイルドカードを入力できます。ワイルドカードを入力した場合、一致するファイルがリストに表示されます。

[File] リスト ボックス カレント プロジェクトに含まれていないカレント ドライブの作業ディレクトリにあるすべてのファイルのリストが表示されます。このリスト ボックスからファイルを選択すると、そのファイル名が [File Name] テキスト ボックスにロードされます。現在の作業ディレクトリがリスト ボックスの上に表示されます。既にプロジェクトの一部であるファイルはこのリスト ボックスに表示されませ

ん。[Include when adding to project] オプションが選択されているドキュメント タイプに属するファイルのみが表示されます。194 ページの「Document Options」も参照してください。

[Add] [File Name] テキスト ボックスの名前のファイルをプロジェクトに追加してダイアログ ボックスを閉じます。[File Name] テキスト ボックスにワイルドカード文字が含まれている場合、該当するファイルが [File] リスト ボックスに表示されます。ダイアログ ボックスは開いたままです。[File] リスト ボックスで複数のファイルが選択されている場合、選択されているすべてのファイルがプロジェクトに追加されます。

[Select All] [File] リスト ボックスに含まれているすべてのファイルを選択します。

[Add Dir] ディレクトリ全体をプロジェクトに追加します。**[Subdirs Also]** チェック ボックスがオンの場合、サブディレクトリ ツリー全体のすべてのファイルを再帰的に追加します。

[Show Dirs/Show Files] リスト ボックスに表示する内容 (ファイル名またはサブディレクトリ名) を切り替えます。

[Subdirs Also] このチェック ボックスがオンの場合、Source Insight は [Add] ボタンがクリックされたときにサブディレクトリのファイルも追加します。

このチェック ボックスがオフの場合、Source Insight は選択されているファイルまたは選択されているディレクトリのファイルのみ追加し、サブディレクトリを無視します。

[Browse] Windows の [ファイルを開く] ダイアログ ボックスを表示します。このダイアログ ボックスでファイルを選択すると、[File Name] テキスト ボックスにファイルがフル パスで表示されます。

[Remove] [Remove File] ダイアログ ボックスに切り替えます。

Add File List

入力ファイルで指定したファイル名とディレクトリをカレント プロジェクトに追加します。

プロジェクト管理者が Source Insight プロジェクトのビルドに使用できるソース ファイルやソース ディレクトリのリストを管理する際に便利です。手動でファイルを追加しないで済みます。

Advanced Options

Source Insight の最適化オプションを選択します。不具合を特定する際に便利です。不具合を報告すると、[Advanced Options] ダイアログ

ボックスのオプションを変更するように指示される場合があります。通常は、この機能を使用する必要はありません。

Back Tab

カーソルを 1 つ左のタブ ストップに移動します。

Backspace

挿入ポイントの左にある文字を削除します。選択範囲が拡張されている場合、選択範囲のテキストを削除します。

Beginning of Line

挿入ポイントをカレント行の行頭に移動します。

Beginning of Selection

選択範囲が拡張されている場合、挿入ポイントを現在の選択範囲の先頭に移動します。選択範囲が挿入ポイントの場合、何も起こりません。

Blank Line Down

挿入ポイントを次の空白行の行頭に移動します。

Blank Line Up

挿入ポイントを前の空白行の行頭に移動します。

Block Down

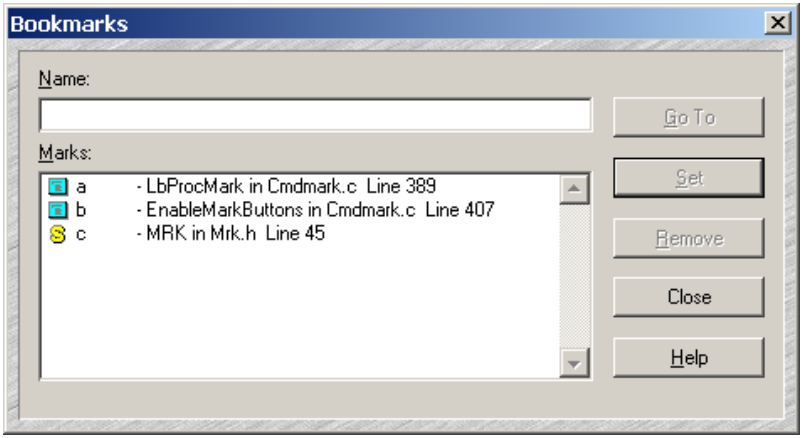
挿入ポイントを次の } 括弧に移動します。C/C++ や Java のような言語における現在のコード ブロックの最後に相当します。

Block Up

挿入ポイントを前の { 括弧に移動します。C/C++ や Java のような言語における現在のコード ブロックの先頭に相当します。

Bookmark

ブックマークの追加と削除、既存のマークの場所の移動を行います。ブックマークは、カレント ワークスペースの一部です。



[Name] ブックマークの名前を入力します。入力したブックマークが存在する場合、[Go To] ボタンがデフォルトのボタンになります。Enter を押すと、そのマークに移動します。入力したブックマークが存在しない場合、[Set] ボタンがデフォルトのボタンになります。Enter を押すと、新規ブックマークが作成されます。

[Marks] リスト 現在設定されているすべてのブックマークのリストを表示します。リストの各項目に、ブックマークの名前、ファイル名、および行番号が表示されます。[Marks] リストで項目を選択すると、ブックマークの名前が [Name] テキスト ボックスにロードされます。

[Go To] [Marks] リストで選択されているブックマークにジャンプします。

[Set] 新規ブックマークを作成します。[Name] テキスト ボックスの内容がブックマークの名前になります。ブックマークの名前がリストに既にある場合、位置が再定義されます。

[Remove] 選択しているブックマークを削除します。

Bottom of File

挿入ポイントをカレント ファイルの最後の行の行末に移動します。

Bottom of Window

挿入ポイントをアクティブなウィンドウに表示されている最後の行の行頭に移動します。

Browse Files

Windows の [ファイルを開く] ダイアログ ボックスを表示します。Source Insight の [Open] コマンド (プロジェクト ウィンドウにディレクトリに関係なくカレント プロジェクトに含まれるファイルのみを表示する) とは異なります。

Browse Project Symbols

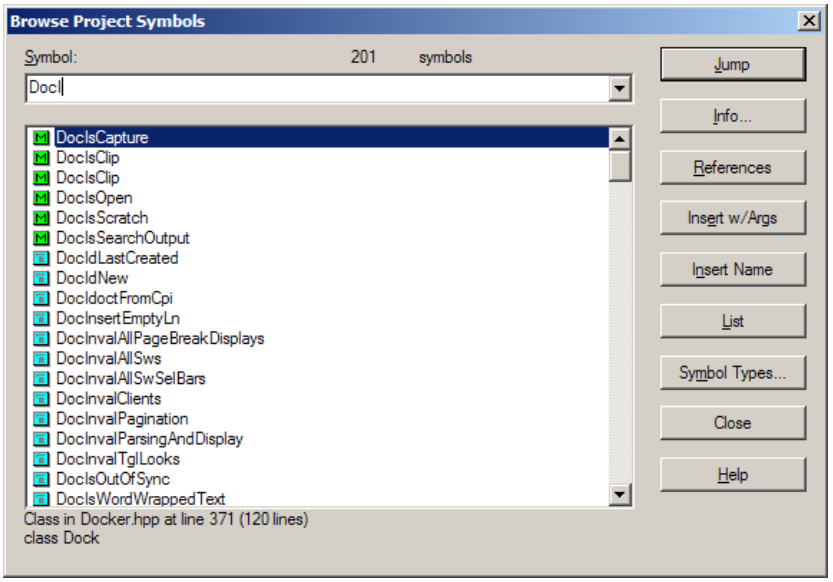
カレント プロジェクトのシンボルをすべてリストします。このダイアログ ボックスから、次の操作を実行できます。

- 名前の一部をベースとするシンボルを検索する。
- シンボル定義を参照する。
- シンボル定義にジャンプする。
- ソース ファイルの関数に呼び出しを挿入する。
- クロスリファレンス リストを作成する。

[Browse Project Symbols] コマンドを使用すると、選択範囲の最初の単語が自動的に選択されます。単語は、ダイアログ ボックスのシンボル名テキスト ボックスにもロードされます。[Insert] ボタンを使用してシンボル名を置換できます。

ヒント： このモーダル ダイアログ ボックスを使用する代わりに、プロジェクト ウィンドウのシンボル リスト ビューを使用することもできます。プロジェクト ウィンドウはモードレスで、アプリケーション ウィンドウと結合または分離できます。コンテキスト ウィンドウには、プロジェクト ウィンドウのシンボル リストで選択している項目の宣言が表示されます。

[Browse Project Symbols] ダイアログ ボックス



[Symbol] 検索するシンボルの名前です。選択範囲の最初の単語が自動的にロードされます。任意のシンボル名を入力できます。

[Symbol] リスト プロジェクトのすべてのシンボルのリストを表示します。[Symbol] テキスト ボックスで検索パターンが指定された場合、検索パターンと一致するすべてのシンボルがリストされます。シンボル リストの下部に、現在選択しているシンボルのタイプとシンボルを含むファイルが表示されます。

リストのシンボルのタイプは [Symbol Types…] ボタンをクリックして設定できます。

入力している内容に一致するシンボルが [Symbol] リストに表示されます。たとえば、「Pch」と入力すると、「Pch」で始まるリストの (並べ替え後の順序で) 最初の項目がリストで選択されます。名前の大文字と小文字は区別されません。また、先頭の下線 (_) は無視されます。

シンボル シラブル マッチングを ([Preferences: Typing] で) 有効にした場合、入力したシラブルと一致する項目も [Symbol] リストに表示されます。たとえば、「cre win」(途中にスペースが含まれている点に注意) と入力すると、名前に「Cre」および「Win」を含むすべてのシンボルが [Symbol] リストに表示されます。

シラブル マッチングのオンとオフを一時的に切り替えるには、エントリの先頭にスペースを追加します。

検索パターンの前に疑問符 (?) を入力し、正規表現を使用してシンボルを検索できます。

疑問符 (?)、検索パターンの順に入力して [Jump] ボタンをクリックすると、シンボルの検索に正規表現形式の検索パターンを指定できます。パターンと一致するシンボルはすべて、[Symbol] リストに表示されます。たとえば、「Delete」で開始して「Foo」を含むすべてのシンボルを検索するには、「?^Delete.*Foo」と入力します。

[Jump] 現在選択しているシンボルの定義にジャンプします。[Symbol] リストで項目が選択されている場合、その項目がカレント シンボルです。選択されていない場合、[Symbol] テキスト ボックスに入力したシンボルが使用されます。

[Symbol] テキスト ボックスの内容が疑問符 (?) で始まっている場合、疑問符に続く検索パターンと一致するすべてのシンボルがリストされます。ダイアログ ボックスは開いたままです。

[Info…] 選択しているシンボルについて [Symbol Info] コマンドを実行します。

[References] 選択しているシンボルへの参照を検索します。

[Insert w/Args] シンボルが関数の場合、シンボル定義で指定されているように、現在の選択範囲をシンボルの名前、パラメータに置換します。

[Insert Name] 現在の選択範囲をシンボルの名前に置換します。

[List] [Symbol] リストに現在リストされているシンボルのクロスリファレンス リストを作成します。新規ファイル Symlist.txt が作成されます。ファイルの各行には、シンボル名、そのシンボルが定義されているファイル名と行番号が含まれます。

[Symbol Types…] シンボル リストに含めるシンボルのタイプと [Symbol] テキスト ボックスで正規表現を使用した場合に検索するシンボルのタイプを指定します。

クロスリファレンス リストの作成

Source Insight では、シンボル名のリストを含む出力ファイルを作成できます。

シンボルのクロスリファレンス リストを作成するには

1. **[Browse Project Symbols]** コマンドを実行します。
2. **[List]** ボタンをクリックします。[Symbol] リストにリストされているすべてのシンボルのリスト、シンボルが見つかったファイルと行番号を含む新規ファイル Symlist.txt が作成されます。

一部のシンボルのクロスリファレンス リストを作成するには

1. **[Browse Project Symbols]** コマンドを実行します。
2. **[Symbol Types…]** ボタンをクリックして、リストに表示するシンボルのタイプを指定します。
3. **[Symbol]** テキスト ボックスに検索表現を入力します。パターンは、パターンであることを示す ? 文字で始める必要があります。たとえば、「?Word」は部分文字列「Word」を含むすべてのシンボルを検索します。
4. **[Jump]** ボタンをクリックして、リストの内容をパターンと一致するすべてのシンボルに置換します。検索が完了すると、一致したすべてのシンボルが [Symbol] リストに表示されます。
5. **[List]** ボタンをクリックしてシンボル リストを作成します。
6. 関数呼び出しの引数を適切に設定します。

名前で関数を検索するには

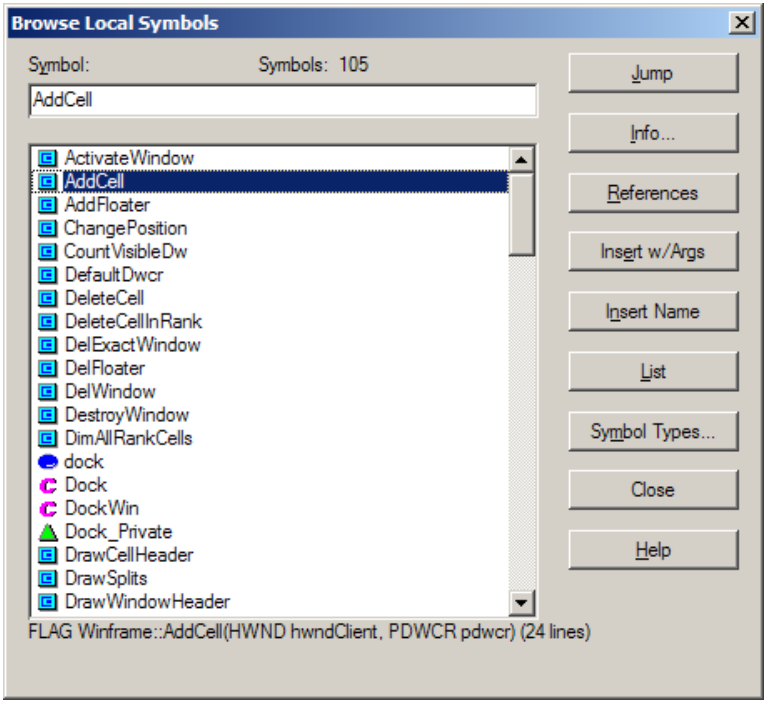
名前を完全に覚えていない関数を呼び出す場合を考えます。その関数の名前に「Insert」と「Char」が含まれているとします。この例では、プロジェクトを作成したときにシンボル シラブル マッチングを有効にしたと仮定しています。

1. 関数呼び出しを挿入する場所を選択します。
2. **[Browse Project Symbols]** コマンドを実行します。
3. [Symbol] テキスト ボックスに「Insert Char」と入力してしばらく待ちます。2 つの単語の間にスペースを入力することに注意してください。入力したシラブルで [Symbol] リストがフィルタされ、名前に「Insert」と「Char」を含むすべてのシンボルがリストされます。この機能は、大文字で始まる単語を入力した場合にのみ動作します。

Browse Local File Symbols

ファイル範囲にあるカレント ファイルに含まれるシンボルをすべてリストします。このダイアログ ボックスから、シンボル定義の参照、シンボルへのジャンプ、または現在の選択範囲へのシンボル定義のコピーの挿入を行うことができます。

[Browse Local File Symbols] コマンドを使用すると、選択範囲の最初の単語が自動的に選択されます。[Insert] ボタンを使用してシンボル名を置換できます。



[Symbol] 検索するシンボルの名前です。選択範囲の最初の単語が自動的にロードされます。任意のシンボル名を入力できます。

[Symbol] リスト プロジェクトのすべてのシンボルのリストを表示します。[Symbol] テキスト ボックスで検索パターンが指定された場合、検索パターンと一致するすべてのシンボルがリストされます。シンボル リストの下部に、現在選択しているシンボルのタイプとシンボルを含むファイルが表示されます。

リストのシンボルのタイプは [Symbol Types…] ボタンをクリックして設定できます。

入力している内容に一致するシンボルが [Symbol] リストに表示されます。たとえば、「Pch」と入力すると、「Pch」で始まるリストの (並べ替え後の順序で) 最初の項目がリストで選択されます。名前の大文字と小文字は区別されません。また、先頭の下線 (_) は無視されます。

疑問符 (?)、検索パターンの順に入力して [Jump] ボタンをクリックすると、シンボルの検索に正規表現形式の検索パターンを指定できます。パターンと一致するシンボルはすべて、[Symbol] リストに表示されま

す。たとえば、「Delete」で開始して「Foo」を含むすべてのシンボルを検索するには、「?^Delete.*Foo」と入力します。

[Jump] 現在選択しているシンボルの定義にジャンプします。[Symbol] リストで項目が選択されている場合、その項目がカレント シンボルです。選択されていない場合、[Symbol] テキスト ボックスに入力したシンボルが使用されます。

[Symbol] テキスト ボックスの内容が疑問符 (?) で始まっている場合、疑問符に続く検索パターンと一致するすべてのシンボルがリストされます。ダイアログ ボックスは開いたままです。

[Info…] 選択しているシンボルについて [Symbol Info] コマンドを実行します。

[References] 選択しているシンボルへの参照を検索します。

[Insert w/Args] シンボルが関数の場合、シンボル定義で指定されているように、現在の選択範囲をシンボルの名前、パラメータに置換します。

[Insert Name] 現在の選択範囲をシンボルの名前に置換します。

[List] [Symbol] リストに現在リストされているシンボルのクロスリファレンス リストを作成します。新規ファイル Symlist.txt が作成されます。ファイルの各行には、シンボル名、そのシンボルが定義されているファイル名と行番号が含まれます。

[Symbol Types…] シンボル リストに含めるシンボルのタイプと [Symbol] テキスト ボックスで正規表現を使用した場合に検索するシンボルのタイプを指定します。

Cascade Windows

ウィンドウを重ねて表示します。

Checkpoint

カレント ファイルをディスクに保存し、変更履歴と操作履歴を消去します。「クリーンな」保存操作であると考えられます。ファイルを保存して閉じ、再度開くことと同じです。このコマンドを使用した後で以前の変更を元に戻すことはできません。

Source Insight バージョン 3.0 よりも前のバージョンでは、ファイルを保存した後、操作履歴と変更履歴を保存していなかったため、このコマンドは [Save] として知られていました。

Checkpoint All

開いているすべてのファイルに対して [Checkpoint] コマンドを実行します。開いているすべてのファイルをディスクに保存して、変更履歴

と操作履歴を消去します。このコマンドを使用した後で以前の変更を元に戻すことはできません。

Clear Highlights

すべてのソース ウィンドウで単語のハイライト表示をクリアします。ハイライト表示を行うには、[Highlight Word] コマンドを使用します。

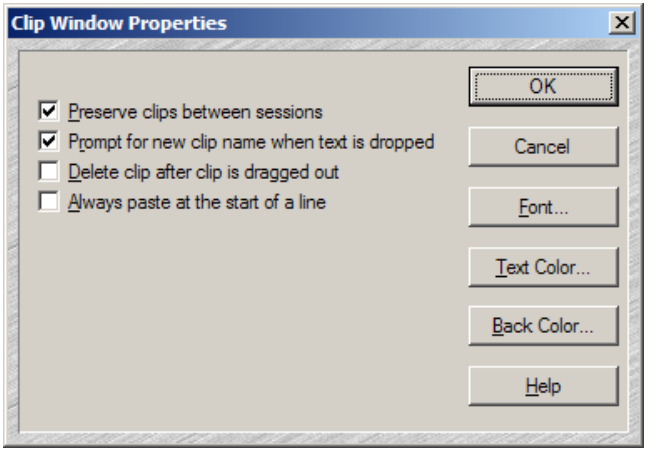
Clip Properties

(クリップ ウィンドウ ツール バーの右クリック メニュー)

クリップの名前を編集する [Clip Properties] ダイアログ ボックスを表示します。

Clip Window Properties

クリップ ウィンドウのオプションを設定する [Clip Window Properties] ダイアログ ボックスを表示します。



[Preserve clips between sessions] オンの場合、クリップは (Source Insight プログラム ディレクトリの) Clips ディレクトリに自動的に保存され、次回 Source Insight を実行するときにロードされます。オフの場合、Source Insight を終了するとクリップは破棄されます。

[Prompt for new clip name when text is dropped] オンの場合、Source Insight はテキストをクリップ ウィンドウにドロップしたときにクリップの名前を確認します。オフの場合、Source Insight はクリップの名前を自動的に生成します。

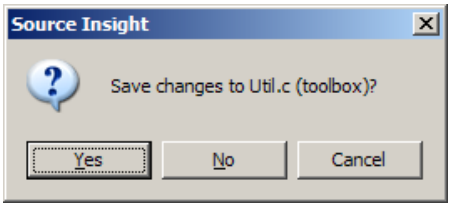
[Delete clip after clip is dragged out] オンの場合、クリップをクリップ ウィンドウの外にドラッグすると、クリップ ウィンドウから削除されます。オフの場合、クリップは削除されません。

[Always paste at the start of the line] オンの場合、カーソルの位置ではなく、カレント行の行頭にクリップを貼り付けます。クリップを特定の場所にドラッグした場合には適用されません。

[Font…]、[Text Color…]、[Back Color…] クリップ ウィンドウの表示オプションを設定します。

Close

カレント ファイルを閉じます。ファイルが変更され、保存されていない場合、Source Insight はファイルを閉じる前に変更を保存するかどうか確認します。



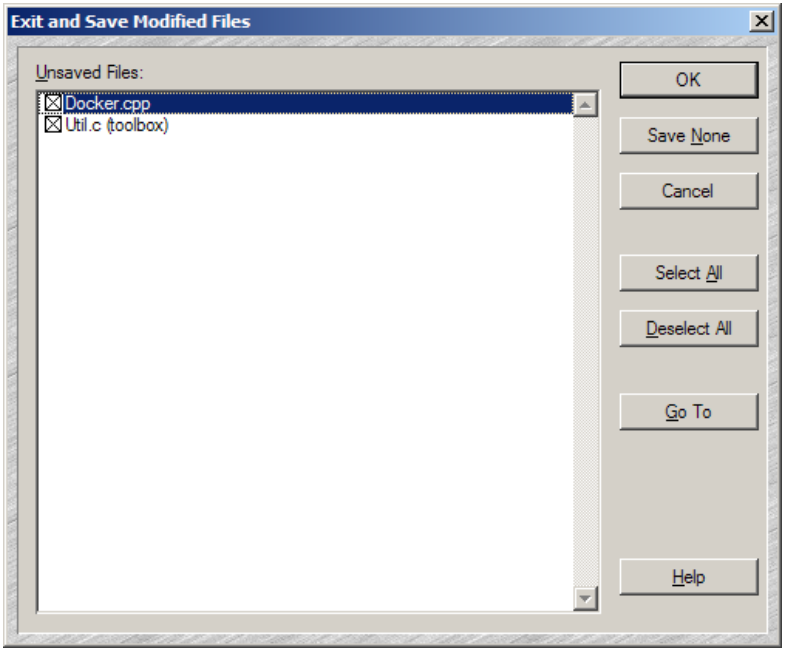
[はい] ファイルを保存して閉じます。

[いいえ] ファイルを保存しないで閉じます。行った変更は保存されません。

[キャンセル] [Close] コマンドをキャンセルします。ファイルは開かれたままで保存されません。

Close All

開いている各ファイルについて [Close] コマンドを実行します。変更後に保存されていないすべてのファイルについて、Source Insight はファイルを保存するかどうか確認します。



カスタム コマンド ウィンドウを開いてカスタム コマンドを実行している場合、これらのウィンドウは閉じられません。

Close Project

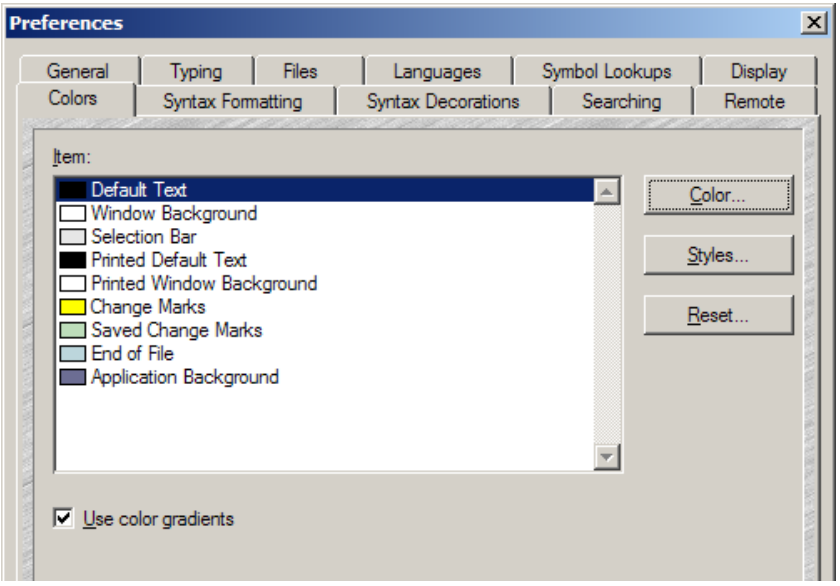
カレント プロジェクトを閉じます。プロジェクトを閉じると、開いていたすべてのファイルも閉じられます。Source Insight は、開いている各ファイルについて [Close] コマンドを実行します。プロジェクトのワークスペースと設定ファイルも保存されます。

Close Window

カレント ウィンドウを閉じます。ファイルは複数のウィンドウに表示できるため、ウィンドウを閉じてもファイル バッファを閉じる必要はありません。ファイルを表示しているウィンドウのみを閉じる場合、ファイルも閉じられます。

Color Options

ユーザー インターフェイス項目の色を指定する [Preferences: Colors] ダイアログ ボックスを表示します。



[Item] リスト 色を設定できる表示項目をリストします。リストには以下の項目が含まれます。

表 5.1: 色を設定できる表示項目

表示項目	説明
[Default Text]	他のスタイルが適用されていないプレーン テキストです。
[Window Background]	ウィンドウの背景色です。
[Printed Default Text]	プレーン テキストの印刷色です
[Printed Window Background]	背景の印刷色です。
[Change Marks]	変更後に保存していない行の左に表示される変更マークの色です。
[Saved Change Marks]	変更後に保存した行の左に表示される変更マークの色です。
[End of File]	ファイルの最後の下部に表示される領域の色です。
[Application Background]	ソース ファイル ウィンドウを含むメイン アプリケーション フレーム ウィンドウの色です。

[Use color gradients] オンの場合、ユーザー インターフェイスの一部で色のグラデーションが使用されます。オフの場合、単色が使用されます。

[Color…] 新しい色を選択します。

[Styles…] スタイル プロパティを編集します。

[Reset…] 色オプションを初期値に戻します。

Command Shell

Source Insight のシェル コマンド ボックスを起動するカスタム コマンドです。

Complete Symbol

入力時にシンボル名のエントリを補完します。名前全体を入力する時間を節約できます。

自動補完が有効な場合、[Complete Symbol] コマンドを実行すると自動補完機能が起動され、自動補完ウィンドウがポップアップします。

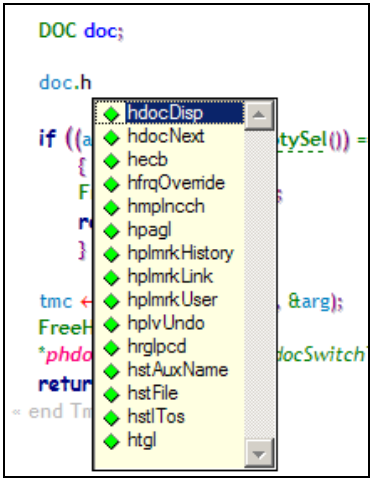


図 5.1 入力後に自動補完ウィンドウが表示されます。

自動補完が無効でコンテキスト ウィンドウが表示されている場合、シンボル名を入力すると、コンテキスト ウィンドウに入力した内容と部分的に一致するシンボルの名前が表示されます。

[Complete Symbol] コマンドを実行すると、入力している単語がシンボルの完全な名前に置換されます。

たとえば、InitWindowState という名前の関数があるとします。「InitWi」と入力すると、自動補完ウィンドウまたはコンテキスト ウィ

ンドウに該当する関数 (InitWindowState) が表示されます。[Complete Symbol] コマンドを実行すると、「InitWi」が「InitWindowState」に置換されます。

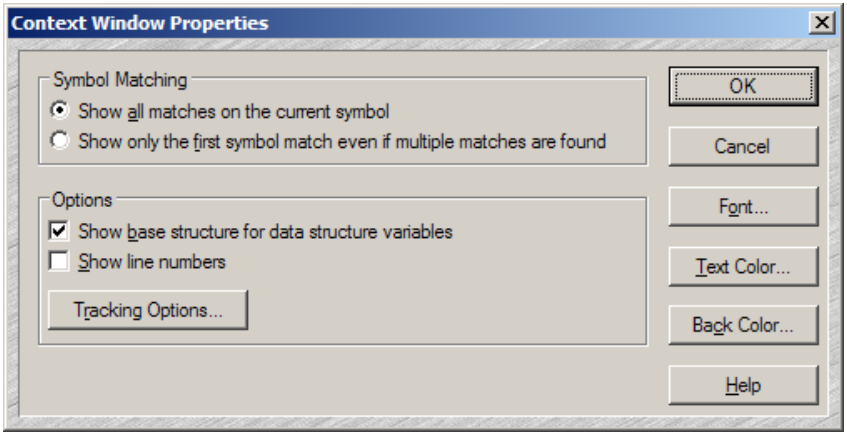
自動補完設定は補完機能の動作に影響します。関数名が挿入されたときに関数呼び出しパラメータを挿入するオプションが用意されています。[Preferences] コマンドを使用して自動補完オプションを変更できます。

Context Window

コンテキスト ウィンドウの表示を切り替えます。[Activate Context Window] コマンドを使用した場合も、コンテキスト ウィンドウが表示され、コンテキスト ウィンドウにフォーカスが変更されます。

Context Window Properties

コンテキスト ウィンドウのプロパティを指定するダイアログ ボックスを表示します。コンテキスト ウィンドウはテキスト ウィンドウで選択または入力した内容に加えて、プロジェクト ウィンドウ、リレーション ウィンドウ、クリップ ウィンドウで選択したファイルを追跡します。



[Show all matches on the current symbol] コンテキスト ウィンドウに、一致したすべてのシンボルの完全なリストを表示するには、このオプションを選択します。同じ名前の複数の定義がある場合、リストにすべて表示されます。

[Show only the first symbol match] 最初に一致したシンボルの定義のみを表示するには、このオプションを選択します。複数の場所で定義されている本質的に同じシンボルを使用している場合、このオプションをオンにします。

[Show base types for data structure variables] 変数宣言をデコードしてベース構造タイプ定義 (structs、unions、classes、その他) を表示するには、このオプションをオンにします。

例 :

```
struct S { int x; };
struct S *psvar;

psvar->...
```

psvar の内部を選択すると、Source Insight は psvar の宣言を確認して、struct S のポインタであることを理解し、struct S の宣言を表示します。

宣言の階層構造を確認しないで、各変数の宣言を表示するには、このオプションをオフにします。

[Show line numbers] コンテキスト ウィンドウに行番号を表示します。

[Tracking Options…] コンテキスト ウィンドウで追跡する表示オプションを設定する [Symbol Tracking Options] ダイアログ ボックスを表示します。

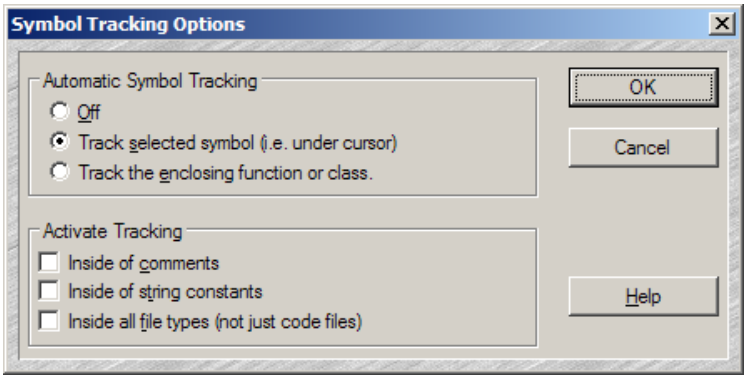
[Font…] コンテキスト ウィンドウでシンボル リストとソース コードの表示に使用するフォントを指定します。コンテキスト ウィンドウでリストが表示されている場合、リストのフォントが設定されます。コンテキスト ウィンドウでソース ファイルが表示されている場合、ソース コードのフォントが設定されます。

[Text Color…] コンテキスト ウィンドウのリスト項目のテキストの色を指定します。ソース コードの色は [Preferences] ダイアログ ボックスの [Syntax Formatting] オプションで指定します。

[Back Color…] コンテキスト ウィンドウのリスト項目の背景色を指定します。ソース コードの色は [Preferences] ダイアログ ボックスの [Syntax Formatting] オプションで指定します。

[Symbol Tracking Options]

コンテキスト ウィンドウで追跡する表示オプションを設定します。



[Automatic Symbol Tracking]

ソース ファイルでカーソルを移動すると、コンテキスト ウィンドウで、カーソルの下、またはカーソルの周りにあるシンボルが自動的に追跡されます。コンテキスト ウィンドウで追跡する項目を指定します。

[Off] 自動シンボル追跡を無効にします。

[Track selected symbol] (i.e. under cursor) 入力カーソルの下にあるシンボルの定義を調べます。

[Track the enclosing function or class] 入力カーソルを含む関数またはクラスの定義を表示します。関数を編集するときにコンテキスト ウィンドウで関数の定義とパラメータを確認するのに便利です。

[Activate Tracking] グループ

自動追跡が行われる条件を指定します。

[Inside of comments] カーソルがコメントの内部にあるときにシンボルを調べます。

[Inside of string constants] カーソルが引用文字列定数の内部にあるときにシンボルを調べます。

[Inside all file types] カーソルが (ソース コード ファイルだけでなく) 任意のファイルの内部にあるときにシンボルを調べます。

Copy

現在の選択範囲の内容をクリップボードにコピーします。クリップボードにコピーした後は、[Paste] コマンドを使用して他の場所に貼り付けることができます。このコマンドは、現在の選択範囲が拡張されている場合にのみ使用できます。

Copy Line

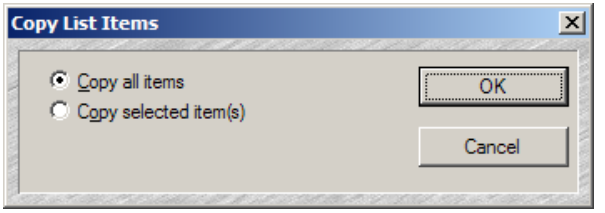
行全体を含むように現在の選択範囲を拡張して、選択範囲をクリップボードにコピーします。[Copy Line] コマンドを使用するたびに、選択範囲が 1 行下に拡張されます。

Copy Line Right

挿入ポイントからカレント行の行末までのテキストをクリップボードにコピーします。

Copy List

このコマンドは、リストを右クリックすると表示されます。[Copy List] コマンドは、リストの内容をクリップボードにコピーします。任意のリストのコピー、ファイルへの貼り付け、印刷に使用できます。



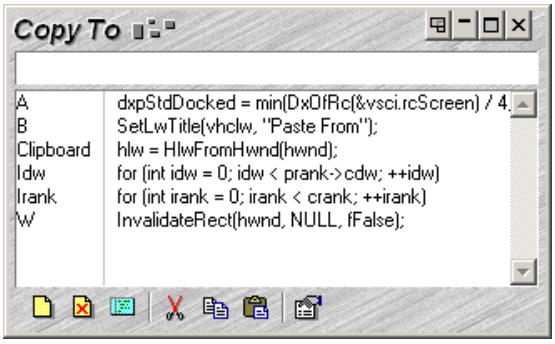
Copy Symbol

このコマンドは、シンボル ウィンドウを右クリックすると表示されます。

[Copy Symbol] コマンドは、選択しているシンボルと定義のボディをクリップボードにコピーします。たとえば、関数名をクリックして [Copy Symbol] コマンドを選択すると、関数全体がクリップボードにコピーされます。

Copy To Clip

現在の選択範囲の内容を指定したクリップ バッファにコピーします。このコマンドを使用すると、クリップ ウィンドウが表示されます。



リストのクリップ項目を選択するか、クリップ名を入力した後、Enter を押してコピー操作を完了します。

Create Key List

すべてのコマンドと各コマンドに割り当てられているキーストローク ショートカットのリストを含む Keylist.txt ファイルを生成します。Keylist.txt ファイルは通常のテキスト ファイルです。このファイルは編集できます。独自のキーボード クイックリファレンス ガイドを作成する場合に便利です。

Create Command List

すべてのコマンドと説明のリストを含む CommandList.txt ファイルを生成します。CommandList.txt ファイルは通常のテキスト ファイルです。このファイルは編集できます。独自のコマンド クイックリファレンス ガイドを作成する場合に便利です。

Cursor Down

挿入ポイントを 1 行下に移動します。

Cursor Left

挿入ポイントを 1 文字左に移動します。

Cursor Right

挿入ポイントを 1 文字右に移動します。

Cursor Up

挿入ポイントを 1 行上に移動します。

Custom Commands

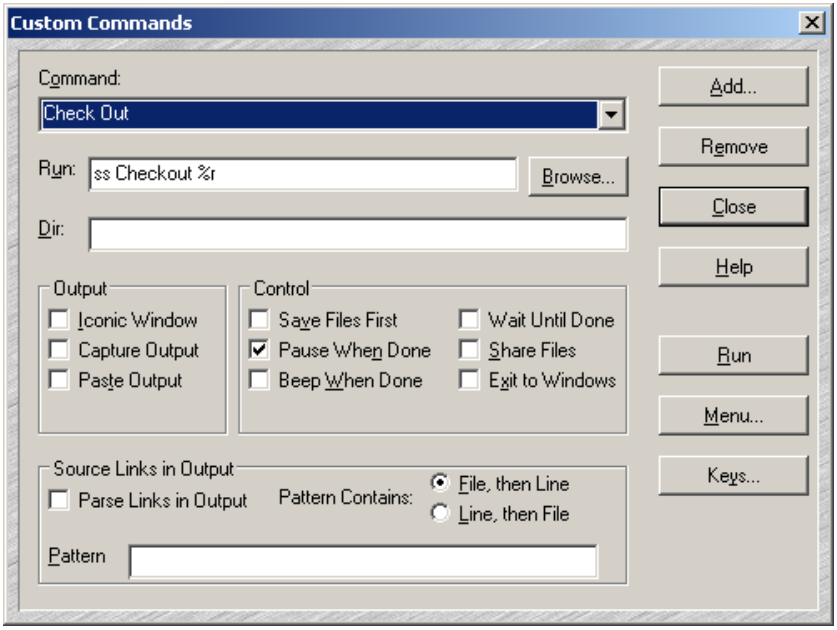
カスタム コマンドは、コマンド シェル バッチ ファイルに似ています。カスタム コマンドを使用すると、Source Insight からコマンド ライン ツールを起動して、その出力をキャプチャできます。Windows プログラムも実行できます。

カスタム コマンドを実行した後、Source Insight に戻ることができます。シェル カスタム コマンドの出力は、ファイルに保存するか、貼り付けることができます。カスタム コマンドは、現在の設定の一部として格納されます。

カスタム コマンドを使用して、コンパイラ、ソース管理プログラム、および sort のようなファイル フィルタを起動できます。

ヒント： カスタム コマンド編集のショートカットは、コマンドを選択する間 Ctrl キーを押したままにします。そのコマンドの [Custom Commands] ダイアログ ボックスを表示します。

[Custom Commands] ダイアログ ボックス



[Command] 現在選択しているコマンドの名前を表示します。このプルダウン リストには、定義済みのすべてのカスタム コマンドのリストが含まれます。

[Run] カスタム コマンドが起動されたときに実行するコマンド ラインです。[Run] テキスト ボックスには特殊なメタ文字を入力できます。179 ページの「Run フィールドのフォーマット」も参照してください。

[Dir] [Run] テキスト ボックスで指定されたスクリプトを実行するときに使用する作業ディレクトリです。Source Insight は、コマンドを実行する前にこの場所を作業ディレクトリに設定します。何も指定しない場合、プロジェクト ソース ディレクトリが作業ディレクトリに設定されます。

[Output] グループ コマンドの出力を制御します。

[Iconic Window] オンの場合、プログラムは最小化して起動されます。オフの場合、プログラムは通常通り起動されます。

[Capture Output] オンの場合、コマンドの標準出力がキャプチャされ、コマンドが完了したときに新規コマンド出力ウィンドウに表示されます。コマンド出力ウィンドウのタイトルは、カスタム コマンドの名前になります。オフの場合、標準出力はキャプチャされません。

[Paste Output] オンの場合、コマンドの標準出力が現在の選択範囲に貼り付けられます。

[Control] グループ コマンド実行の前後に行う操作を指定します。

[Save Files First] オンの場合、Source Insight はカスタム コマンドを実行する前に [Save All] コマンドを実行します。[Save All] コマンドを実行すると、編集後に保存していないファイルを保存するかどうか確認するメッセージが表示されます。

オフの場合、変更したファイルを保存しないでコマンドが実行されます。コマンドが完了したとき変更は保存されないままで Source Insight に戻ります。コマンドを実行してクラッシュが発生した場合、Source Insight はリカバリを行います。変更は失われません。128 ページの「クラッシュからのリカバリ」も参照してください。

[Pause When Done] オンの場合、Source Insight はコマンドが完了したとき次のメッセージを表示します。

```
[Press any key to return to Source Insight...]
```

オフの場合、コマンドが完了した後 DOS ボックスが閉じられます。

[Beep When Done] オンの場合、Source Insight はコマンドが完了したときビープを出力します。

[Wait Until Done] オンの場合、Source Insight はコマンドが完了するまで待ちます。

オフの場合、Source Insight はコマンドの完了を待たずに処理を続行します。Source Insight ウィンドウに戻った後、コマンドはバックグラウンドで処理されます。

[Exit to Windows] プログラムの起動後 Source Insight を終了します。

[Source Links in Output]

コマンドが完了した後、出力をどのように扱うか指定します。Source Insight が出力を解析する際に特定の警告またはエラー メッセージを検索するよう指定できます。

[Parse Links in Output] コマンド出力がソース リンク パターンで検索されます。パターンは通常、警告およびエラー メッセージと一致します。パターンが一致すると、Source Insight はその行にソース リンクを挿入します。ソース リンクは、警告またはエラー メッセージからターゲットのソース行へのリンクに使用されます。[Parse Links in Output] オプションがオンの場合、[Pattern] テキスト ボックスに有効な検索表現を指定する必要があります。

[Pattern Contains] [File, then Line] および [Line, then File]。パターン表現でのグループの順序を示します。

パターン表現の最初のグループがファイル名で 2 番目のグループが行番号の場合、[File, then Line] を選択します。この設定では、2 番目のグループ (たとえば、行番号) はオプションです。

パターン表現の最初のグループが行番号で 2 番目のグループがファイル名の場合、[Line, then File] を選択します。

[Pattern] コマンド出力のファイル名と行番号の検索に使用される正規表現です。[Parse Links in Output] オプションが無効な場合は無視されます。オプションがオンの場合、このテキスト ボックスにはファイル名と行番号の「グループ」を含む有効な正規表現が含まれている必要があります。113 ページの「正規表現」も参照してください。

[Add…] [Command] テキスト ボックスのコマンドを定義します。コマンドが既に存在する場合、再定義されます。

[Remove] コマンドを削除します。

[Run] コマンドを定義して実行します。

[Close] ダイアログ ボックスを閉じます。ダイアログ ボックスで行われた変更はそのまま残ります。

[Menu…] 現在のコマンドを定義して [Menu Assignments] ダイアログ ボックスを表示します。

[Keys...] 現在のコマンドを定義して [Key Assignments] ダイアログ ボックスを表示します。

Run フィールドのフォーマット

[Run] テキスト ボックスには、カスタム コマンドが起動されたときに実行するコマンドを指定します。[Run] テキスト ボックスには、複数のコマンドを指定できます。コマンドはセミコロンで区切ります。たとえば、

```
cat make.log;echotime
```

「cat make.log」を実行した後、「echotime」を実行します。

コマンド シェルの実行

「type」や「dir」のようなシェル コマンドを実行する場合や、バッチ ファイルを実行する場合は、最初に cmd.exe を実行する必要があります。たとえば、

```
cmd /c mybat.bat  または
cmd /c type foo.txt
```

メモ： Windows 9x/Me を使用している場合、cmd.exe の代わりに command.com を使用してください。

[Run] テキスト ボックスに (セミコロンで区切って) 複数のコマンドを指定した場合、cmd.exe を指定する必要はありません。Source Insight がバッチ ファイルを作成して、cmd.exe を自動的に実行します。たとえば、

```
cat readme.txt;dir
```

cmd.exe が既にシェル内部のバッチ ファイルで実行されているため、問題なく動作します。

cmd.exe で起動されたシェルの環境スペースが不足する場合があります。この場合、/e スイッチを使用して cmd.exe を起動します。たとえば、

```
cmd /e:1024 /c mybat.bat
```

cmd.exe で起動される新しいサブシェルに 1K バイトの環境スペースが割り当てられます。

コマンド ライン置換

[Run] テキスト ボックスには、メタ文字を指定できます。メタ文字は、次の項目に置換されます。

表 5.2: カスタム コマンドのメタ文字

文字	置換後	例
%f	カレント ファイルのフル パス名 *	c:\myproj\file.c
%r	プロジェクト ソース ディレクトリに対するカレント ファイルの相対パス名 *	file.c
%n	カレント ファイルのリーフ名 *	file.c
%d	カレント ファイルのディレクトリ パス	c:\myproj
%h	カレント ファイルのディレクトリ パス (ドライブ名なし)	\myproj
%b	カレント ファイルのリーフ名 (拡張子なし) *	file
%e	カレント ファイルの拡張子	c
%c	カレント ファイルのドライブ名	c:
%p	カレント プロジェクト名	c:\myproj\myproj
%j	カレント プロジェクトのソース ディレクトリ	c:\myproj
%J	カレント プロジェクトのデータ ディレクトリ	C:\Documents and Settings\Jim Smith\My Documents\Source Insight\Projects\Base
%v	カレント プロジェクトのソース ディレクトリのドライブ名	c:
%o	プロジェクトのリーフ名 (パスなし)	myproj
%l	カレント行の番号	任意の番号
%w	選択範囲の最初の単語、またはカーソルの下の単語	任意の単語
%s	カスタム コマンドの実行中に現在の選択範囲が保存される一時ファイルの名前	d:\tmp\vt0004

表 5.2: カスタム コマンドのメタ文字

文字	置換後	例
%a	現在の日付	05-12-02
%t	現在の時間	08:23
%1 - %9	カスタム コマンドの引数	任意の文字列

置換後の説明の最後に * があるメタ文字には、以下の修飾子文字のいずれかを続けて指定できます。

文字	置換後	例
%o	すべての開いているファイル	%f%o
%m	すべての変更したファイル	%f%m

ShellExecute コマンド

ShellExecute コマンドは、Windows シェル コマンドを起動します。

カスタム コマンドは、指定されたファイルに対して Windows シェルが操作を実行する「ShellExecute」関数をサポートしています。ShellExecute を使用する場合、ファイルの特定の種類を処理するために登録されているアプリケーションを知っておく必要はありません。技術的なバックグラウンド情報については、Windows シェル API ドキュメントで「ShellExecute」関数を参照してください。

この機能を使用するには、カスタム コマンドの [Run] 文字列で「ShellExecute」を起動します。フォーマットは次のようになります。

```
ShellExecute <verb> <filespec> <optional parameters>
```

たとえば、Web サイトをブラウズする場合、

```
ShellExecute open http://www.somedomain.com
```

verb には以下のいずれかの単語を指定します。

- **edit** ファイルのエディタを開きます。
- **explore** 指定されたフォルダを表示します。
- **open** 指定されたファイルを開きます。実行形式ファイルまたはドキュメント ファイルを指定できます。フォルダや URL も指定できます。
- **print** 指定されたドキュメント ファイルを印刷します。filespec がドキュメント ファイルでない場合、関数は動作しません。
- **properties** ファイルまたはフォルダのプロパティを表示します。
- **find** [スタート] - [検索] - [ファイルやフォルダ] を起動します。
- **""** (空の文字列) ShellExecute に verb パラメータを渡しません。

filespec パラメータには有効なパスを指定します。スペースを含むパス名は二重引用符で囲んでください。%f (カレント ファイル) のようなメタ文字も使用できます。実行ファイルの名前も指定できます。

optional parameters リストは filespec の後に指定するすべての項目です。実行するアプリケーションに渡されるパラメータを指定します。フォーマットは起動する verb、および実行するアプリケーションによって決定されます。ここでもカスタム コマンドのメタ文字を使用できます。

カスタム コマンドの作業ディレクトリは、ShellExecute が起動される前に適用されます。しかし、ShellExecute を使用する場合、出力はキャプチャまたは解析できません。

ShellExecute の例

ShellExecute の使用方法を示すいくつかの例を紹介します。

動作	カスタム コマンドの Run 文字列
Web サイトをブラウズする	ShellExecute open http://www.example.com
Windows 2000 のドキュメント ファイル フォルダを表示する	ShellExecute explore "C:\Documents and Settings"
Windows 98 のドキュメント ファイル フォルダを表示する	ShellExecute explore "C:\My Documents"
Internet Explorer を起動する	ShellExecute "" iexplore
Internet Explorer でファイルを表示する	ShellExecute "" iexplore %f
カレント プロジェクト フォルダのファイルを検索する	ShellExecute find %j

カスタム コマンドのバックグラウンドでの実行

Source Insight でカスタム コマンド シェル プログラムを起動すると、Sihook3.exe というプログラムが実行されます。Sihook3.exe は、コマンドを起動して出力をキャプチャします。カスタム コマンドを実行して Source Insight ウィンドウに戻ると、カスタム コマンドをバックグラウンドで実行しながら Source Insight で編集を続行できます。

コンパイル コマンドとビルド コマンドの作成

カスタム コマンドを使用して Source Insight からコンパイラを起動し、エラー メッセージをキャプチャして解析できます。次に、[Go To First Link] および [Go To Next Link] を使用して、ソース ファイル中のエラーを表示できます。

単純なコンパイル コマンドを作成するには

Microsoft® C++ コンパイラを使用する単純なコンパイラ コマンドを作成するには

1. **[Custom Command]** コマンドを実行します。
2. [Command] ドロップダウン リストで [Compile File] コマンドを選択します。
3. **[Run]** テキスト ボックスに、「`cl %f`」と入力します。この場合、カレント ファイルがコンパイルされます。代わりに「make」プログラムやバッチ ファイルを起動することもできます。バッチ ファイルを使用する場合、「`cmd /c mybatch.bat`」のように、コマンド プロセッサを最初に実行してください。
4. **[Parse Links in Output]** オプションをオンにします。デフォルトの解析パターンは、コンパイラの出力からコンパイラのエラー メッセージを解析するように設定されます。
5. **[Save Files First]** オプションをオンにします。コンパイラを実行する前にファイルが保存されます。
6. **[Add…]** ボタンをクリックして新規コマンドを保存します。
   [Menu…] または **[Keys…]** ボタンをクリックすると、新規コマンドが定義され、[Menu Assignments] または [Keyboard Assignments] ダイアログ ボックスでコマンドにメニューまたはキーストロークを割り当てることができます。

[Parse Links in Output] オプションをオンにすると、Source Insight はコンパイラの出力を検索して、各エラー メッセージにソース リンクを設定します。この場合、「リンク ソース」はコンパイラ出力ファイルの各エラー メッセージです。各リンクの「リンク ターゲット」は、各エラー メッセージで指定されたファイルと行番号です。

Microsoft® Developer Studio を使用してプロジェクトをビルドするには

1. **[Custom Command]** コマンドを実行します。
2. **[Command]** ドロップダウン リストで [Build Project] コマンドを選択します。[Build Project] カスタム コマンドは、Source Insight をインストールしたときに定義されます。
3. **[Run]** テキスト ボックスに、次のように入力します。

   ```
   C:\MsDevPath\msdev project.dsp /make /rebuild
   ```

   ここで、「MsDevPath」は msdev.exe プログラムのパスで、「project.dsp」は Developer Studio プロジェクトの名前です。

   msdev.exe が起動され、指定したプロジェクトがビルドされます。
4. **[Parse Links in Output]** オプションをオンにします。デフォルトの解析パターンは、コンパイラの出力からコンパイラのエラー メッセージを解析するように設定されます。
5. **[Save Files First]** オプションをオンにします。プロジェクトをビルドする前にファイルが保存されます。

コンパイラ エラーの表示

エラーとソース行を表示するには

1. 上記で定義した **[Compile File]** カスタム コマンドを実行します。
2. エラーがある場合、コンパイラを終了したときにエラー メッセージが「Compile File」ウィンドウに表示されます。Source Insight はソース リンクを自動的に設定して [Go To First Link] コマンドを実行します。 最初のエラー メッセージとソース行が選択され表示されます。
3. **[Go To Next Link]** コマンドを実行します。「Compile File」ウィンドウの次のエラー メッセージが選択され、最初のエラーと同様に、リンクのターゲットが表示されます。
4. リンク (エラー メッセージ) がすべて表示されるまで、**[Go To Next Link]** コマンドを使用して続行します。リンクがそれ以上ない場合、Source Insight はビープを出力して、「No links.」メッセージをステータス バーに表示します。

Cut

現在の選択範囲の内容をクリップボードにコピーして、選択範囲を削除します。[Cut] コマンドは選択範囲を削除しますが、テキストはクリップボードに存在していることに注意してください。[Cut] コマンドに続いて [Paste] コマンドを実行すると、削除を元に戻すことができます。

Cut Line

カレント行をクリップボードにコピーして、行を削除します。このコマンドを使用する場合、カーソルは行のどの位置にあってもかまいません。

Cut Line Left

カレント行の挿入ポイントの左にあるテキストをすべて切り取ります。

Cut Line Right

カレント行の挿入ポイントの右にあるテキストをすべて切り取ります。

Cut Selection] or Paste

現在の拡張範囲が拡張されている場合は [Cut] コマンドを実行し、選択範囲が挿入ポイントの場合は [Paste] コマンドを実行します。

このコマンドにキーまたはマウスのボタンを割り当てた場合、キーまたはマウスのボタンを押すだけでテキストを簡単に操作できます。

このコマンドを使用するには :

1. マウスのボタンをクリックおよびドラッグして移動するテキストを選択します。
2. テキストを切り取るマウスのボタンまたはキーを押します。
3. マウスのボタンをポイントおよびクリックして新しい場所を選択します。
4. テキストを貼り付けるマウスのボタンまたはキーを押します。

Cut Symbol

(シンボル ウィンドウの右クリック メニュー)

選択しているシンボルを切り取ります。

Cut To Clip

現在の選択範囲の内容を指定したクリップ バッファにコピーして、選択範囲を削除します。このコマンドを使用すると、クリップ ウィンドウが表示されます。

Cut Word

挿入ポイントから単語の最後まで (およびその後のスペース) を切り取ります。

Cut Word Left

挿入ポイントから単語の先頭までを切り取ります。

Delete

現在開いているファイルを含む、ディスク上のファイルを削除します。

Delete All Clips

クリップ ウィンドウからすべてのユーザー クリップを削除します。「Clipboard」は削除できない特殊なクリップです。

Delete Character

挿入ポイントの文字を削除します。現在の選択範囲が拡張されている場合、選択範囲全体を削除します。

Delete Clip

(クリップ ウィンドウ ツール バーの右クリック メニュー)

クリップ ウィンドウおよびディスクからクリップ ファイルを削除します。

Delete File

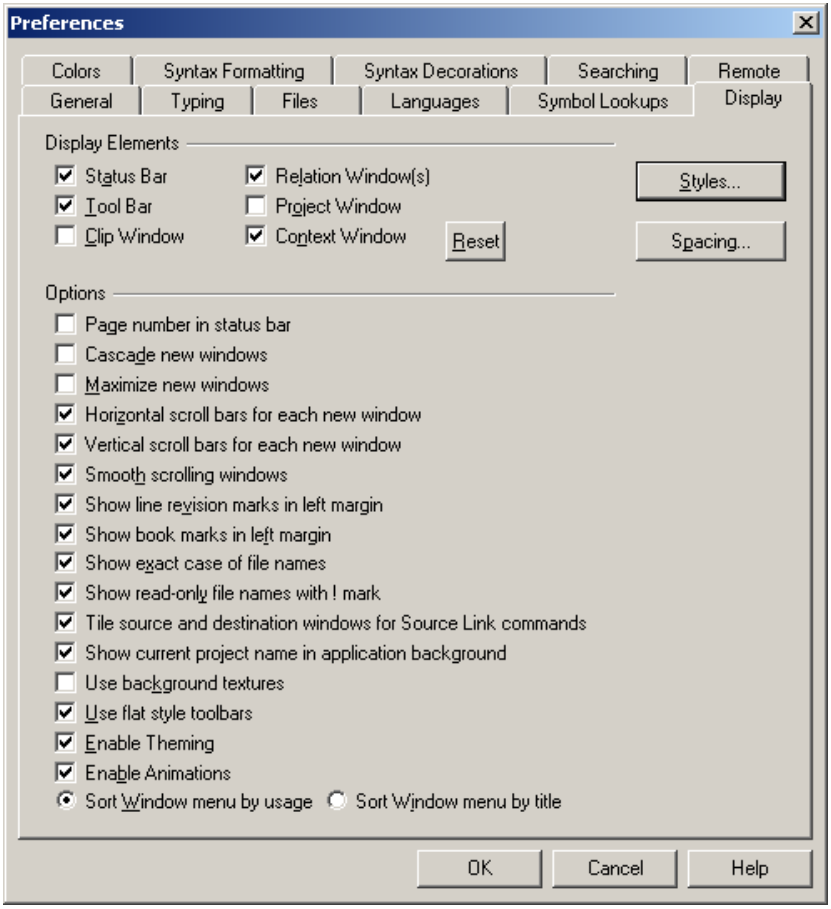
ディスクからファイルを削除し、そのファイルがプロジェクトの一部であった場合はカレント プロジェクトから削除します。カレント ファイルを指定した場合、ファイルは閉じられた後、削除されます。

Delete Line

カレント行を削除します。[Cut Line] コマンドと異なり、クリップボードの内容は変更されません。

Display Options

[Preferences] ダイアログ ボックスの [Display] タブを表示します。これらのオプションは、現在の設定の一部です。



[Display Elements] グループ ユーザー インターフェイスの表示を切り替えます。

[Status Bar] Source Insight アプリケーション ウィンドウの下部にあるステータス バーの表示を切り替えます。

[Tool Bar] Source Insight アプリケーション ウィンドウの上部にあるメイン ツール バーの表示を切り替えます。

[Clip Window] クリップ ウィンドウの表示を切り替えます。

[Project Window] プロジェクト ウィンドウの表示を切り替えます。

[Context Window] コンテキスト ウィンドウの表示を切り替えます。

[Relation Window(s)] リレーション ウィンドウの表示を切り替えます。

[Reset] すべての補助ウィンドウの位置をリセットしてメイン モニタに移動します。

[Options] グループ

プログラムの異なる表示要素の一般的なオプションを制御します。

[Page Number in status bar] オンの場合、現在の選択範囲を含むページの番号がステータス バーに表示されます。ページ番号は [Document Options] コマンドで選択されたプリンタ フォントのサイズ、構文フォーマット オプション、および [Page Setup] の設定から計算されます。

[Cascade new windows] オンの場合、Source Insight は新規ウィンドウを他のウィンドウに重ねて表示します。オフの場合、Source Insight は新規ウィンドウを現在のウィンドウと同じ場所に表示します。現在のウィンドウは新規ウィンドウに隠されます。

[Maximize new windows] オンの場合、Source Insight は新しく開いたウィンドウを自動的に最大表示します。オフの場合、Source Insight は通常の MDI 形式でウィンドウを開きます。

[Horizontal scroll bars for each new window] オンの場合、新規ソース ファイル ウィンドウに水平スクロール バーが表示されます。

[Vertical scroll bars for each new window] オンの場合、新規ソース ファイル ウィンドウに垂直スクロール バーが表示されます。

[Smooth scrolling windows] オンの場合、ウィンドウは 1 行ごとにジャンプする代わりに、スムースにスクロールされます。スムース スクロールはスクロールを連続に行うのではなく、一度に 1 行または 2 行スクロールする場合に有効になります。編集を開始するか、素早くスクロールすると、スムース スクロールもスピードアップします。オフの場合、ウィンドウは一度に 1 行スクロールします。使用しているビデオ カードでスクロール処理が遅い場合、このオプションをオフにしてください。

[Show line revision marks in left margin] オンの場合、Source Insight はファイルが保存または開かれた後、変更または削除された行の左余白をハイライト表示します。編集した行を素早く確認するのに便利です。[Go To Next Change] および [Go To Previous Change] コマンド (Alt+ キーパッドの + および Alt+ キーパッドの -) を使用すると、カレント ファイルの変更を移動できます。

[Show book marks in left margin] オンの場合、Source Insight は [Bookmark] コマンドでマークを追加した行の左余白にブックマーク アイコンを表示します。

[Show exact case of file names] オンの場合、Source Insight はファイル名をそのまま表示します。オフの場合、Source Insight はファイル名の最初の文字を大文字、その他の文字を小文字で表示します。ファイル名に大文字と小文字が含まれている場合、Source Insight はファイル名を変換しません。

このオプションは、Source Insight が表示するファイル名にのみ影響します。ファイル名は、ファイル システムで示されたとおりにデータベースに格納されます。

[Show read-only file names with ! mark] オンの場合、Source Insight はウィンドウ タイトルでファイル名の先頭に感嘆符 (!) を追加して読み取り専用ファイルを表示しされます。このオプションは、ウィンドウ タイトルでのファイル名の表示方法にのみ影響します。

[Show current project name in application background] オンの場合、カレント プロジェクトの名前が Source Insight アプリケーション ウィンドウの複数のドキュメント バックグラウンド領域に表示されます。

[Tile source and destination windows for Source Link commands] オンの場合、[Go To First Link]、[Go To Next Link] のようなソース リンク コマンドを使用すると、Source Insight はソース リンクのソースとターゲットを並べて表示します。ソース リンク コマンドは、ソース リンクの出力を解析するカスタム コマンド (「コンパイル」コマンドなど) を実行したときにも使用されます。オフの場合、Source Insight はソース リンク コマンドを使用したときにウィンドウを並べて表示しませんが、ソース リンクのターゲットを含むウィンドウをアクティブにします。

カレント ウィンドウが最大表示されている場合、ウィンドウは並べて表示されません。

[Use background textures] ツール バーやダイアログ ボックスのようなユーザー インターフェイス要素の背景に灰色のテクスチャを使用します。

[Use flat style toolbars] 新しい Windows プログラムのように、新しい「フラット」形式のツール バー ボタンを使用します。オフの場合、ツール バー ボタンは従来の形式で表示されます。

[Enable Theming] Windows XP およびそれ以降で、XP テーマの使用とダイアログ ボックス、メニュー、ウィンドウ フレーム、その他のユーザー インターフェイス要素のビジュアル スタイルを有効にします。設定の変更を有効にするには、Source Insight を再起動する必要があります。

[Enable Animations] 入力フォーカスがフローティング ツール ウィンドウに変更されたとき、およびフローティング ウィンドウが「拡大」または「縮小」されたときに表示するアニメーションを有効にします。

ビデオ カードのパフォーマンスが低い場合はアニメーションをオフにしてください。

[Sort Window menu by usage] および [Sort Window menu by title]

[Window] メニューに表示されるウィンドウ名をどのように並べ替えるかを指定します。[Sort Window menu by usage] が選択されている場合、最近使用したウィンドウから順に表示されます。[Sort Window menu by title] が選択されている場合、アルファベット順に表示されます。

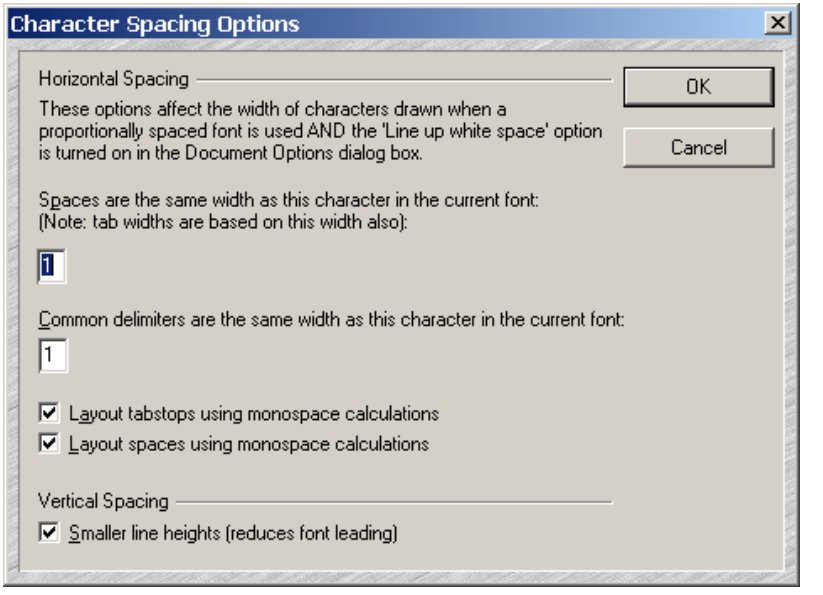
[Styles...] スタイル プロパティを編集します。311 ページの「Style Properties」も参照してください。

[Spacing...] 文字の間隔オプションを変更します。190 ページの「[Character Spacing Options] ダイアログ ボックス」も参照してください。

[Character Spacing Options] ダイアログ ボックス

文字の水平間隔オプションはプロポーショナル フォントで便利です。

文字の間隔オプションは、文字の水平間隔および垂直間隔の制御に使用します。このダイアログ ボックスでは、Source Insight が空白、タブ、および一般的な区切り文字の幅をどのように計算するか調整できます。[Document Options] ダイアログ ボックスの **[Line up white space]** オプションがオフの場合、最初の 2 つの設定は効果がありません。



[Horizontal Spacing] オプション

プロポーショナル フォントが使用され、[Document Options] ダイアログ ボックスの **[Line up white space]** オプションがオンの場合、表示する文字の幅を調整します。オンの場合、Source Insight はスペースとタブに固定の幅を使用します。スペースとタブは固定ピッチのフォントを使用した場合と同じ方法で表示されます。プロポーショナル フォントを使用している場合、このオプションをオンにするとプログラムがより見やすくなります。

スペース幅文字

スペース幅文字は単一スペースの幅を制御します。この結果、タブ (スペース数文字分) の幅も制御されます。Source Insight は、表示に使用されるすべてのフォントで、この文字と同じ幅になるようにスペースを計算します。たとえば、このダイアログ ボックスで文字「1」が指定されている場合、スペース文字の幅は文字「1」と同じ幅になります。(文字とその数値を混同しないでください。Source Insight が文字の数値に干渉することはありません。)

Source Insight はスペースの幅をスペース幅文字に基づいて計算するため、スペースは使用されているフォントやフォント サイズに関係なく正確に計測されます。

幅広フォントでの作業

通常の文字幅でないフォントを使用している場合、または空白とインデントの量を調整する場合は、この設定を変更します。幅の狭い文字では空白が縮小され、幅の広い文字では空白が拡大されます。この幅は [Document Options] ダイアログ ボックスで指定されたタブ幅の設定には依存しません。タブ幅は固定幅文字カラムの数として指定されます。

一般的な区切り文字の幅もスペースに似た方法で制御できます。影響を受ける区切り文字は、ほとんどのフォントで狭い文字である - | \ / および！ です。

[Layout tabstops using monospaced calculations] タブの幅を制御します。オンの場合、等幅フォントを使用していると仮定してタブの幅が計算されます。ソース コードの表示に可変ピッチのフォントを使用している場合でも、テキストのカラムをタブ付きにします。

[Layout spaces using monospaced calculations] スペース文字をモノスペースのフォントが使用された場合と同じように表示します。たとえば、4 つのスペースはタブ ストップと同じ幅になります (タブの幅が 4 文字の場合)。Source Insight は各行を確認して、スペース文字にいつこのルールを適用するか決定します。スペースを使用して手動でカラムを並べる場合は、このルールが適用されます。ルールは 2 つ以上の連続するスペースにのみ適用されます。その他の場合、指定された

フォントで自然な幅のスペースになるようにスペースの幅が計算されます。このオプションは、デフォルトでオンです。

このオプションを使用すると、タブとスペースを使用してカラムを並べる場合を除いて、スペースのサイズは自然なサイズになります。これは精密化学ではありません！

Source Insight は、構文フォーマットを使用している場合でも、単純に表示されるようにテキストを並べて表示します。[Draft View] コマンドを使用してテキストのアライメントを確認することもできます。

[Vertical Spacing] オプション

これらのオプションは行間隔を制御します。

[Smaller line heights (reduces font leading)] 行の高さを圧縮して画面に多くのテキスト行を表示するには、このオプションをオンにします。このオプションをオンにすると、にオペレーティング システムによってフォントに追加される「行間隔」が少なくなります。行間隔が追加されると印刷されたドキュメントは見やすくなります。しかし、ソース コードの編集には必要ありません。

間隔が問題となる理由

「なぜ間隔が問題になるのでしょう？タブ ストップを 1/2 インチのように設定すればいいだけでは?」と思われるかもしれません。答えは少し複雑です。問題は、ワード プロセッサと異なり、Source Insight は単純なテキスト エディタと同じようにテキスト ファイルを表示しようとする点にあります。

たとえば、Source Insight を使用していない人やソース コードに常に固定ピッチのフォントを使用する人が同じワーク グループにいるとします。その人がコードを見るときに単純な固定幅のタブ ストップで表示されなくなるように Source Insight でコードを編集しようとは思わないでしょう。

ワード プロセッサ プログラムは、プリンタに印刷するイメージでテキストを表示します。Source Insight は、別のエディタで固定ピッチのフォントを使用して表示した場合と同じようにテキストを表示します。

ワード プロセッサ プログラムでは、テキストのサイズは、インチやセンチメートルのように物理的な単位で測定されます。たとえば、タブ ストップを 1/2 インチに設定します。テキストを印刷するとき、ワード プロセッサはタブ ストップがプリンタ上で 1/2 インチ幅になることを確認します。

Source Insight では、タブ ストップは固定サイズの文字カラムで計測されます。Source Insight はタブ付きのカラムを固定ピッチのフォントが使用された場合と同じように整列します。

Source Insight が水平のピクセル位置に基づいて次のタブ位置に単純に移動した場合、固定ピッチのフォントでコードを見ると、異なる数のタブが画面に表示されます。

次に例を示します。メモ帳で Courier New (固定ピッチのフォント) を使用し、X と Y、および Q と R をタブで区切ってカラムを揃えたとします。単語 「narrow」と「wide」は、どちらもカラム 0 (1 タブ ストップ) に収まっています。

タブ ストップ :	0	1	2	3
行 1:	narrow	X	Q	
行 2:	wide	Y	R	

次に、Source Insight でリッチ フォーマットを使用した場合、「narrow」はタブの幅に収まりますが、「wide」は収まりません。Y は 1 つ以上のタブ ストップ分移動され、次のようになります。

タブ ストップ :	0	1	2	3
行 1:	narrow	X	Q	
行 2:	w i d e		Y	R

Y は Q と整列されました。残りのカラムは整列されません。実際、[Document Options] で **[Line up white space]** オプションをオフにするとこのようになります。

[Line up white space] オプションがオンになっていると、Source Insight はタブの位置を揃えて表示しようとします。

タブ ストップ :	0	1	2	3
行 1:	narrow	X	Q	
行 2:	w i d e Y		R	

コードが上記のように表示される場合、「M」や「W」のように多くのフォントで幅広で表示される文字に異なるスペース幅を指定します。すべてのタブ ストップの幅が広くなり、次のように表示されます。

タブ ストップ :	0	1	2
行 1:	narrow	X	Q
行 2:	w i d e	Y	R

この処理は、固定ピッチのフォントで表示したときよりもテキストが非常に狭くなる場合にも有効です。

Document Options

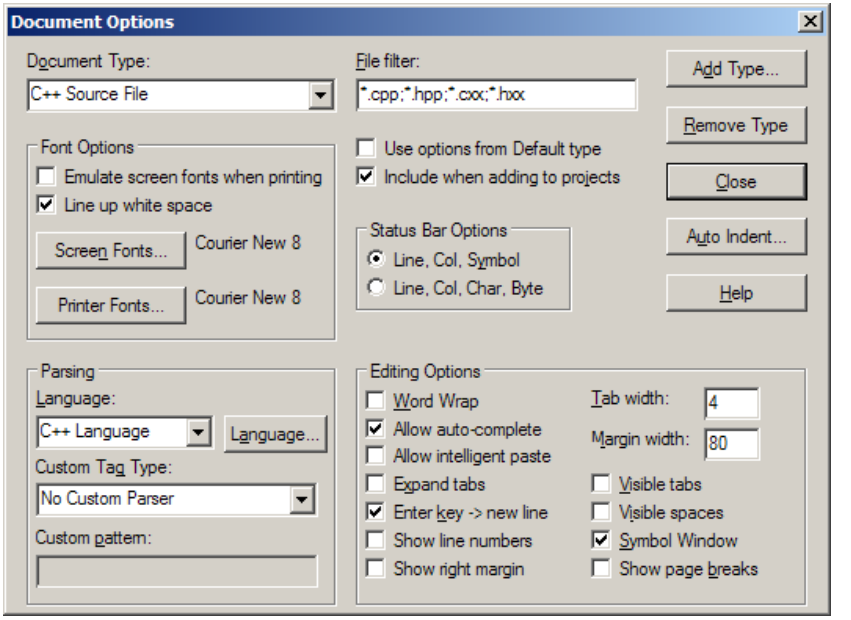
[Document Options] コマンドでは、現在編集しているファイルの名前または拡張子に基づいて編集および表示オプションを定義します。

ドキュメント タイプ

ドキュメント タイプは、ファイル タイプの言語と編集オプションを決定します。

ドキュメント タイプは [Document Options] コマンドで定義されるファイルの分類です。Source Insight は、ファイルの名前を使用してドキュメント タイプを決定します。ドキュメント タイプを使用して、Source Insight で異なる動作に異なる種類のソース ファイルを関連付けます。

[Document Options] ダイアログ ボックス



[Document Type] これまで定義したすべてのドキュメント タイプのリストです。このリストからドキュメント タイプを選択すると、ダイアログ ボックスの他のテキスト ボックスはそのドキュメント タイプのプロパティを反映するように更新されます。リストの最初のエントリは常に [Default] ドキュメント タイプです。エントリの内容は変更できますが、エントリを削除することはできません。

[Document Options] ダイアログ ボックスが表示されるとき、カレント ファイルのドキュメント タイプが自動的に選択されます。

[Add Type...] 新規ドキュメント タイプを追加します。ボタンをクリックすると、ドキュメント タイプの名前を確認するメッセージが表示されます。

[Remove Type] 選択しているドキュメント タイプを削除します。この操作は元に戻せません。

[Auto Indent...] 選択しているドキュメント タイプの自動インデント設定を変更します。199 ページの「自動インデント」も参照してください。

[File Filter] ファイル名の区切り付きリストです。リストのエントリはスペース、セミコロン、またはカンマで区切ります。各エントリは、ファイル名またはワイルドカードのいずれかで指定します。バックスラッシュを含むドライブ名またはフル パスは指定できません。

たとえば、

```
*.c;*.h
```

ワイルドカード フィルタを使用してファイルとドキュメント タイプを一致させます。

ファイル名を指定すると、Source Insight は [File Filter] テキスト ボックスのファイルと一致するすべての定義済みのドキュメント タイプを検索して、ファイルのドキュメント タイプを識別します。実際に、Source Insight はドキュメント タイプ レコードのセット全体を 2 つのパスに分けてファイルのドキュメント タイプを決定します。

1. 最初に、[File Filter] リストのすべてのファイル名と一致するファイルを検索します。
2. 一致するファイルがない場合、[File Filter] リストのすべてのワイルドカードと一致するファイルを検索します。
3. それでも一致するファイルがない場合、Source Insight はそのファイルが [Default] ドキュメント タイプであると仮定します。

これは、特殊な方法で一部のファイルを処理できることを意味します。たとえば、「*.inc」は通常 [x86 Asm Source File] ドキュメント タイプに含まれますが、「cmd.inc」ファイルを [C Source File] ドキュメント タイプとして指定できます。[C Source] ドキュメント タイプの [File Filter] テキスト ボックスに、「cmd.inc」を追加します。

```
*.c;*.h;cmd.inc
```

ドキュメント タイプのファイル フィルタとして * を追加すると、「Default」ドキュメント タイプの代わりにデフォルトの catchall タイプになります。catchall タイプはファイルが他のドキュメント タイプと一致していない場合にのみ適用されます。

新規ファイル拡張子の追加

標準ドキュメント タイプのファイル フィルタに新規エントリを追加することもできます。たとえば、デフォルトでは、[C Source File] ドキュメントの拡張子は「*.c」および「*.h」です。拡張子が .h2 の C ソース

ファイルがある場合、「*.h2」を [C Source File] ドキュメント タイプの [File Filter] リストに追加します。

```
*.c;*.h;*.h2
```

プロジェクトに追加されるドキュメント タイプを制御できます。

[Add Files] ダイアログ ボックスのようにファイルのリストが表示される場合は常に、Source Insight はすべての定義済みドキュメント タイプのすべての [File Filter] テキスト ボックスに含まれるファイルをリストします。つまり、新規ドキュメント タイプを追加すると、それらのファイルはファイル リストにも表示されます。

[Use options from Default type] オンの場合、[Editing Options] および [Status Bar Options] には [Default] ドキュメント タイプの設定が使用されます。多くのドキュメント タイプを定義して、1 つの場所 ([Default] ドキュメント タイプ レコード) からオプションを制御できます。パーサーの設定は影響を受けません。各ドキュメント タイプで一意のままです。

オフの場合、[Editing Options] および [Status Bar Options] には各ドキュメント タイプ レコードの設定が使用されます。

[Include when adding to projects] オンの場合、プロジェクトに追加する新規ファイルを検索するときに [Add File] コマンドおよび自動ファイル追加機能で選択しているドキュメント タイプが追加されます。

[Font Options] グループ

[Emulate screen fonts when printing] オンの場合、Source Insight は [Screen Fonts…] のフォント設定を印刷時に使用します。TrueType フォントを使用している場合に最適です。オフの場合、[Printer Fonts…] ボタンのフォント設定が印刷時に使用されます。

[Line up white space] このオプションは、プロポーショナル フォントを選択している場合にのみ適用されます。Courier New のような固定ピッチのフォントを選択している場合は影響を受けません。

プロポーショナル フォントを使用した場合でも正しくインデントできます。

オンの場合、Source Insight はスペースとタブに固定の幅を使用します。スペースとタブは固定ピッチのフォントを使用した場合と同じ方法で表示されます。プロポーショナル フォントを使用している場合、このオプションをオンにするとプログラムがより見やすくなります。コードの表示に Courier New (または他の固定ピッチのフォント) フォントを使用している場合は、このオプションを試してみてください。

オフの場合、Source Insight は文字の自然な幅を使用します。

固定ピッチのフォントを使用してソース コードを表示するツールを使用している人がチームにいる場合に便利です。プロポーショナル フォントを使用しながら、固定ピッチのフォントの有効な表現を維持できます。

[Preferences: Display] ダイアログ ボックスの [Spacing] ボタンをクリックして Source Insight がスペースの固定幅を計算する方法を制御できます。

詳細は、190 ページの「[Character Spacing Options] ダイアログ ボックス」を参照してください。

[Screen Fonts…] 画面の表示に使用するフォントを選択します。ボタンの右に、現在選択しているフォントの名前が表示されます。

[Printer Fonts…] 印刷に使用するフォントを選択します。ボタンの右に、現在選択しているフォントの名前が表示されます。この設定は、前述した [Emulate screen fonts when printing] オプションがオフの場合にのみ効果があります。

[Parsing] グループ

[Language] Source Insight で定義されているすべての言語のリストです。現在のドキュメント タイプの解析と表示に使用する言語をリストから選択します。たとえば、ドキュメントを Java ソース ファイルとして解析するには、リストから「Java Language」を選択します。解析を行わない場合は、「None」を選択します。

新規カスタム言語を追加するには、<New Language> を選択します。

[Language…] [Language Options] ダイアログ ボックスを開きます。このダイアログ ボックスで、新規言語の追加や言語のプロパティの編集を行います。たとえば、各言語に関連付けられている構文フォーマット キーワード リストを編集できます。各言語タイプには独自のキーワード リストがあります。228 ページの「Language Options」も参照してください。

[Custom Tag Type] [Custom pattern] テキスト ボックスのカスタム パーサー パターンを使用して検索するシンボルのタイプを指定します。リストには、すべての可能なシンボル タイプが含まれています。「No Custom Parser」エントリ が選択された場合、Source Insight はカスタム パターンを使用しません。他のエントリが選択された場合、シンボル名の解析に使用する正規表現パターンをカスタム パターンとして使用します。

[Custom pattern] 1 つのグループを含む有効な正規表現パターンを指定します。グループで、マッチング パターンのどの部分がシンボル タグと見なされるか説明します。このパターンを使用して解析されたシンボルが [Custom Tag Type] で示されているタイプで使用されます。[Custom Tag Type] が「No Custom Parser」に設定されている場合、このテキスト ボックスは無視されます。

カスタム パターンを使用すると、Source Insight にあらかじめ含まれていないファイルのシンボルを解析できます。たとえば、以下の文字列は、WIN.INI のような INI ファイルのセクションを解析します。

```
^\[\(.*\)\]
```

このカスタム解析パターンを使用する新規ドキュメント タイプ「INI File」を定義して WIN.INI のようなファイルを開くと、シンボル ウィンドウにセクション名が表示されます。

[Status Bar Options]

プログラム ウィンドウの下部のステータス バーの表示を制御します。

[Line, Col, Symbol] ステータス バーに挿入ポイントの行番号、カラム番号、およびシンボルの名前が表示されます。

[Line, Col, Char, Byte] ステータス バーに挿入ポイントの行番号、カラム番号、行の先頭からの文字数、およびファイルの先頭からのバイト数が表示されます。

[Editing Options] グループ

ドキュメント タイプがどのように編集されるか制御します。選択したドキュメント タイプのすべてのファイルで、次の編集オプションが有効になります。

[Word Wrap] オンの場合、挿入ポイントが余白の幅を超えると、Source Insight は自動的に次の行にワードラップします。このオプションは、新規テキストを入力しているときにのみ適用されます。オフの場合、Source Insight は入力したテキストを自動的にワードラップしません。

[Allow auto-complete] オンの場合、[Options] > [Preferences: Typing] ダイアログ ボックスで自動補完がグローバルで有効に設定されているドキュメント タイプでシンボルの自動補完が有効になります。

[Allow intelligent paste] オンの場合、[Options] > [Preferences: Typing] ダイアログ ボックスで拡張貼り付けがグローバルで有効に設定されているドキュメント タイプで拡張貼り付けが有効になります。

[Tab width] 文字スペースのタブ文字の幅です。

[Margin width] 自動的にワードラップが行われる前の文字スペースのテキストの幅です。

[Expand tabs] オンの場合、タブを入力したとき、Source Insight はタブ文字と等価な数のスペースを挿入します。テキストはタブが入力された場合と同じように見えますが、タブではなくスペースが挿入されます。オフの場合、タブを入力したとき、Source Insight はそのままタブ文字を挿入します。

[Enter key -> new line] オンの場合、入力中に Return または Enter キーが押されると、新しい行を挿入します。オフの場合、入力中に Return

または Enter キーが押されると、次の行の先頭に挿入ポイントを移動します。

[Show line numbers] 左余白に行番号を表示します。

[Show right margin] 右余白に薄い縦線を表示します。プロポーショナル フォントを使用している場合、行の途中に縦線が表示されることがあります。

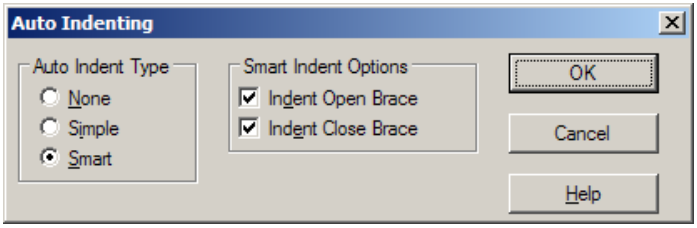
[Visible tabs] オンの場合、タブ文字を空白ではなく特殊なシンボルで表示します。

[Symbol Window] オンの場合、ソース ウィンドウの左側にシンボル ウィンドウが表示されます。

[Show page breaks] 改ページの位置に薄い横線を表示します。改ページの位置は構文フォーマットと選択されているプリンタ フォントに基づいて計算されます。

自動インデント

自動インデント機能は入力したテキストのインデントのレベルを制御します。Source Insight は、Simple と Smart の 2 種類の自動インデントをサポートしています。Smart はすべての言語ではサポートされていません。



[Auto Indent Type] 自動インデントの種類を指定します。自動インデントは新しい行を挿入するときに行われます。

- **[None]** 特殊なインデントは行われません。Source Insight は、新しい行を挿入したときまたはワードラップされたとき、挿入ポイントを次の行の先頭に移動します。
- **[Simple]** Source Insight は、前または次の行に合わせてテキストを自動的にインデントします。
- **[Smart]** Source Insight は、新しい行を挿入したときインデントのレベルを自動的に調整します。すべての言語ではサポートされていません。このオプションが選択されている場合、[Smart Indent Options] が適用されます。

[Smart Indent Options] スマート インデントが中括弧と閉じ中括弧にどのように影響するか決定します。

インデント スタイル	チェック ボックスの設定

`if (x)`
`{`
`}`
	両方のチェック ボックスをオフにします。

`if (x)`
`    {`
`    }`
	両方のチェック ボックスをオンにします。

`if (x) {` `}`	[Indent Open Brace] をオンにします。 [Indent Close Brace] をオフにします。

Draft View

テキストを素早く表示するには、[Draft View] コマンドを使用します。

[Style Properties] で [Draft View] のフォントを編集できます。

ドラフト モードの表示を切り替えるには、[View] メニューで [Draft View] コマンドを選択します。ドラフト モードがオンの場合、色の変更を除く、ほぼすべての構文フォーマットがオフになります。

すべてのテキストは、「Draft View」スタイルを使用して表示されます。スタイルは [Style Properties] ダイアログ ボックスで編集できます。「Draft View」スタイルはモノスペース フォント (Courier New) を使用するようにあらかじめ設定されています。311 ページの「Style Properties」も参照してください。

Source Insight の構文フォーマット機能は強力ですが、単一フォントを使用しているときにテキストが別のエディタや簡易表示モードでどのように表示されるか確認しておく必要があります。ドラフト モードはディスプレイを基本的な等幅フォントに素早く切り替える際、役に立ちます。これは、カラムを並べるのに、タブの代わりにスペースを使う場合は特に、役に立ちます。

ドラフト モードが有効な場合、[Preferences: Syntax Formatting] および [Preferences: Syntax Decorations] ダイアログ ボックスの設定よりも優先されます。

Drag Line Down

選択されているテキストを 1 行下に移動します。選択範囲をファイルの別の場所にドラッグするときに便利です。

Drag Line Down More

選択されているテキストを数行下に移動します。複数行移動することを除いて、[Drag Line Down] コマンドと同じです。

Drag Line Up

選択されているテキストを 1 行上に移動します。選択範囲をファイルの別の場所にドラッグするときに便利です。

Drag Line Up More

選択されているテキストを数行上に移動します。複数行移動することを除いて、[Drag Line Up] コマンドと同じです。

Duplicate

選択されているものすべての複製を作成します。

Duplicate Symbol

(シンボル ウィンドウの右クリック メニュー)

選択しているシンボルの複製を作成します。

Edit Condition

選択している解析条件変数の値を編集します。C、C++、およびリソース ファイルのような条件コンパイルをサポートしている言語に使用します。条件付きコードは #ifdef のような # ディレクティブ間に配置されます。

Source Insight は、指定する条件変数の値に条件付きで依存するコードのセクションを解析できます。[Edit Condition] コマンドを使用して、条件変数の値または条件変数のリストを編集できます。

このコマンドを使用するには、コードで条件変数である識別子を右クリックします。次に、[Edit Condition] を選択します。変数の値を指定します。

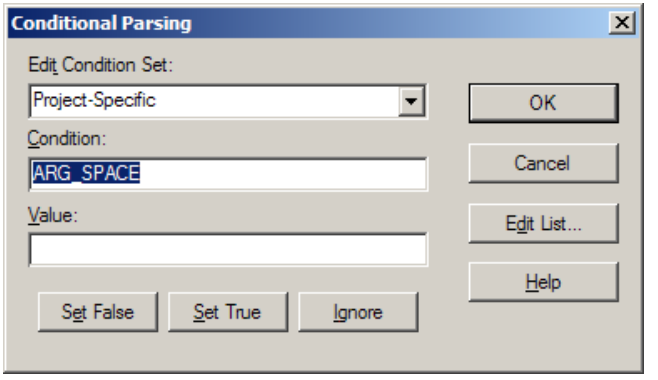
たとえば、MACOBJECTS の内部にカーソルを配置して [Edit Condition] を選択します。

```
#ifdef MACOBJECTS
int jklm;
#endif
```

プロジェクト条件と グローバル条件

2 つの条件変数セットがあります。1 つ目はプロジェクト固有のセットで、プロジェクトに格納されます。2 つ目はグローバルなセットで、すべてのプロジェクトに適用されます。条件が両方のリストに含まれる場合、プロジェクト固有の値が使用されます。

[Conditional Parsing] ダイアログ ボックス



[Edit Condition Set] 編集する条件セット (Project-Specific または Global Contidions) を選択します。条件の値に加えるすべての変更が、ここで選択するリストに追加されます。

[Condition] 条件変数の名前です。

[Value] 条件の値です。代表的な値は、「False」を示す 0 (ゼロ)、または「True」を示す 1 です。他の任意の値を変数に設定できます。値が空の場合、Source Insight はこの変数を参照する条件付きプリプロセッサ ディレクティブを無視します。

[Set False] [Value] フィールドを 0 (ゼロ) に設定します。

[Set True] [Value] フィールドを 1 に設定します。

[Ignore] [Value] フィールドを空にします。[Value] フィールドが空の場合、Source Insight は条件変数の値を指定しないと仮定します。値を指定しない場合、変数を参照するときに #if のようなプリプロセッサ ステートメントは無視されます。これは、任意の条件付き変数のデフォルトの動作です。

[Edit List...] 定義済みの条件をすべて表示する [Conditional Parsing] ダイアログ ボックスを表示します。編集するリストは、[Edit Condition Set] コントロールで選択されているセットに依存します。

Enable Event Handlers

マクロ イベント ハンドラを有効または無効にします。詳細は、395 ページの第 7 章「マクロ イベント ハンドラ」を参照してください。

End of Line

挿入ポイントをカレント行の行末に移動します。

End of Selection

挿入ポイントを選択範囲の最後に移動します。現在の選択範囲が拡張されていない場合、このコマンドは何も行いません。

Exit

Source Insight プログラムを終了します。変更後に保存していないファイルがある場合、Source Insight は保存するかどうか確認します。

Source Insight は終了時にカレント ワークスペース ファイルを保存するので、Source Insight を次に実行するとき、セッションは終了前の状態に戻ります。

Exit and Suspend

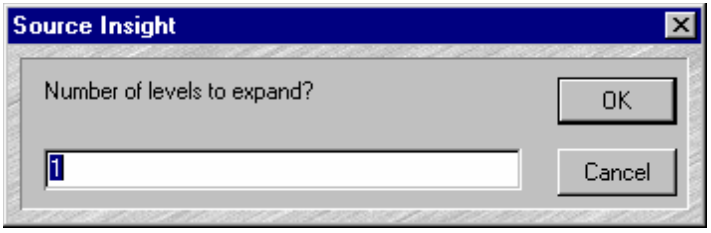
リカバリ ファイルに書き込んだ後、ファイルを保存しないで Source Insight を閉じます。ファイルを保存することなく、編集内容を保存して Source Insight を終了できます。Source Insight を次に実行すると、編集内容が回復されます。

警告！

警告！このコマンドを使用した後、Source Insight を実行して編集内容を回復する前に Source Insight で開いていたファイルを変更しないでください。Source Insight のリカバリ システムはオリジナル ファイルが変更されていないことを前提にしています。

Expand Special

ツリー リストの内部で、選択している項目を展開するレベルを指定します。

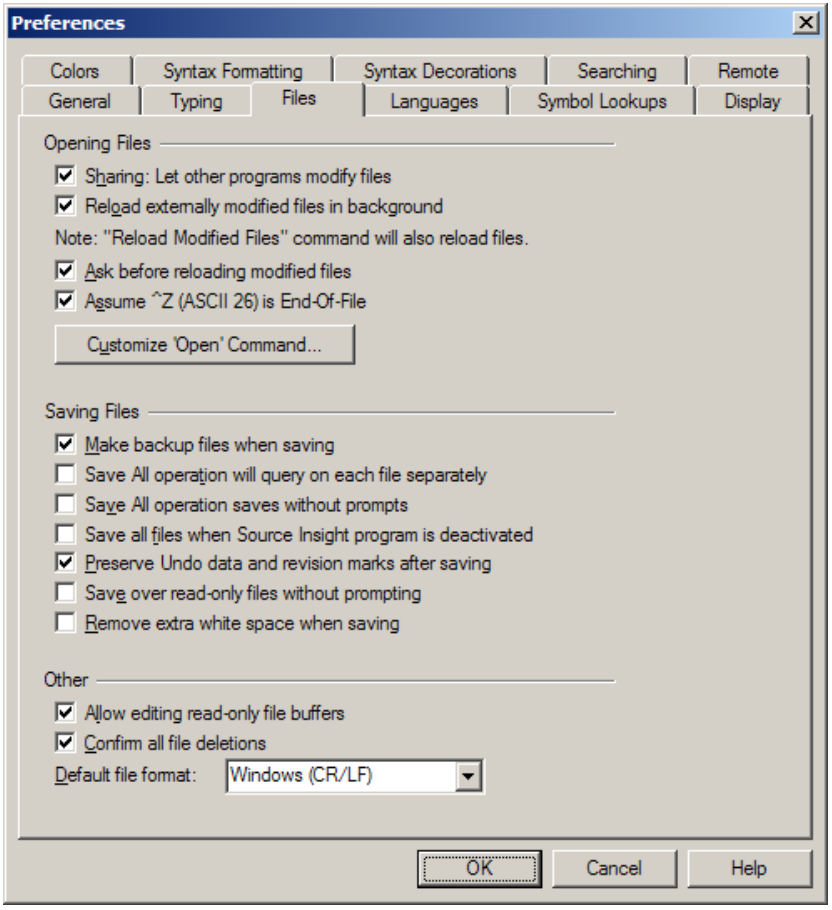


[Number of levels to expand] 選択しているノード以下のツリーを展開するレベルを入力します。たとえば、「1」と入力すると、選択しているノードの 1 レベル下が表示されます。

File Options

[Preferences] ダイアログ ボックスの [Files] タブを表示します。このタブで、ファイルのロードおよび保存オプションを設定します。

[Preferences] ダイアログ ボックスの [Files] タブ



[Sharing] Source Insight で開いているファイルを他のプログラムが修正できるようにします。つまり、Source Insight で開いているファイルを他のプログラムで上書きできるようになります。オンの場合、ファイルを開く操作と保存操作は少し遅くなります。また、開いているファイルに対して使用するディスク容量も少し増えます。

オフの場合、Source Insight が開いているファイルに他のプログラムが書き込むことはできません。ファイルは、保存中を除いて、読み取り専用モードで開かれます。しかし、他のプログラムが同じファイルを書き込み用に開くことはできません。操作は少し速くなります。

[Reload externally modified files in background] オンの場合、Source Insight はファイルが外部的に変更されたことを検出します。その場合、ファイルは自動的にリロードされます。ファイルは、数秒おきまたは Source Insight アプリケーションがアクティブになったときにチェック

されます。変更されたファイルは、[Ask before reloading modified files] オプションがオンの場合、またはそのファイルを Source Insight で編集した場合を除いて、確認なしで自動的にリロードされます。

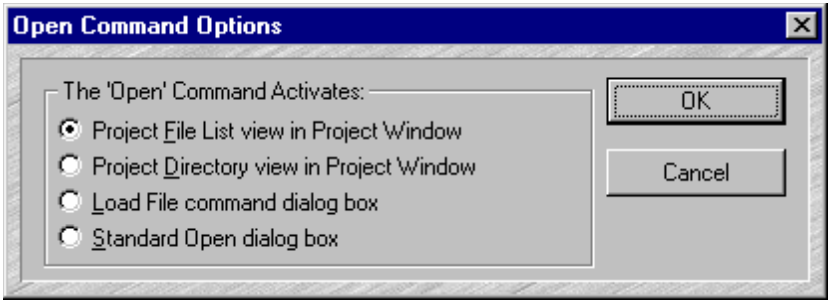
オフの場合でも、[Reload Modified Files] コマンドを使用して変更されたファイルを手動でリロードできます。

[Ask before reloading modified files] オンの場合、Source Insight は外部的に変更されたファイルをリロードする前に確認します。

オフの場合、Source Insight は、そのファイルを Source Insight で編集した場合を除いて、外部的に変更されたファイルを確認しないで自動的にリロードします。既にファイルを編集している場合、リロード操作は行われません。

[Assume ^Z (ASCII 26) is End-Of-File] オンの場合、Source Insight は EOF (ASCII 26) 文字が見つかるとファイルのロードを停止します。ファイルを保存するとき、EOF 文字がファイルの最後に追加されます。オフの場合、Source Insight は EOF 文字を無視してファイルをロードします。

[Customize the 'Open' Command...] [Open] コマンドで実行される動作を指定します。[Open] コマンドのショートカットは Ctrl+O で、ツールバーにもボタンがあります。



[Make backup files when saving] オンの場合、Source Insight はファイルを保存するときにファイルの以前のバージョンをバックアップ ディレクトリに移動します。バックアップ ディレクトリは、Source Insight プログラム ディレクトリのサブディレクトリ「Backup」になります。

オフの場合、Source Insight は以前のバージョンを移動しないでファイルを保存します。

[Save All operation will query on each file separately] オンの場合、Source Insight は [Save As] コマンドを実行したときに変更したファイルを保存するかどうか確認します。各ファイルについて、保存する、保存しない、またはキャンセルを選択できます。

オフの場合、Source Insight は変更したすべてのファイルを自動的に保存します。

[Save all files when Source Insight program is deactivated] オンの場合、Source Insight は Source Insight アプリケーション ウィンドウがアクティブでなくなったとき (異なるアプリケーションがアクティブになったとき)、[Save All] コマンドを自動的に実行します。別のエディタや IDE で同じファイルを開いて作業する場合、Source Insight で行った変更が常に反映されます。たとえば、IDE アプリケーションに切り替えると、Source Insight は編集されたすべてのファイルを自動的に保存します。オフの場合、Source Insight はアクティブでなくなったときにファイルを保存しません。

[Save over read-only files without prompting] オンの場合、Source Insight はファイルが読み取り専用の場合でも警告を表示しないでファイルを上書きします。実際には、セッションで読み取り専用ファイルを最初に上書きするときに警告が表示されます。

オフの場合、Source Insight は読み取り専用ファイルを上書きする前に警告を表示します。ファイル単位で読み取り専用ファイルを上書きするかどうか指定できます。

Source Insight が読み取り専用ファイルに上書きすると、ファイルの属性は読み書き可能に変更されます。

メモ： 何らかの理由により読み取り専用に設定されているファイルに間違って上書きすることがあるため、通常はこのオプションは使用しないでください。このオプションは、ソース管理システムが読み書き可能なファイルを「チェックアウトされたファイル」として扱う場合に役立ちます。チェックアウトする前にファイルを編集して保存できます。

[Preserve Undo data and revision marks after saving] オンの場合、ファイルを保存した後でも [Undo] を実行して改訂マークを確認できます。

[Allow editing read-only file buffers] 読み取り専用の場合でもファイル バッファを編集できるようにします。ファイルの属性を Source Insight の外部で読み書き可能に変更するか、保存中に「Overwrite」ボタンをクリックして読み取り専用ファイルを明示的に上書きしない限り、ファイルに保存することはできません。ソース管理システムによっては、ファイルのバージョンをチェックアウトしてから、Source Insight に戻ってファイルを保存できます。

Source Insight の内部でファイル バッファを編集しても、[Save] コマンドを使用するまでファイルは保存されない点に注意してください。ただし、[Save all files when Source Insight program is deactivated] オプションがオンの場合は、Source Insight から別のプログラムへの切り替えのような操作を行うとファイルは保存されます。

[Remove extra white space when saving] ファイルを保存するときに行末の空白を除去します。

[Confirm all file deletions] ソース ファイルを削除する前に確認します。Source Insight は、プロジェクトからソース ファイルを削除したとき、またはプロジェクトを削除したときにソース ファイルを削除しません。Source Insight が作成したプロジェクト データ ファイルのみ削除します。しかし、プロジェクト ウィンドウでソース ファイルを選択して削除することはできます。

[Default file format] Source Insight が新規ソース ファイルを保存するときにデフォルトのテキスト ファイル フォーマットを使用します。フォーマットは、行末文字の種類 (CR および LF) によって異なります。次のフォーマットがあります。

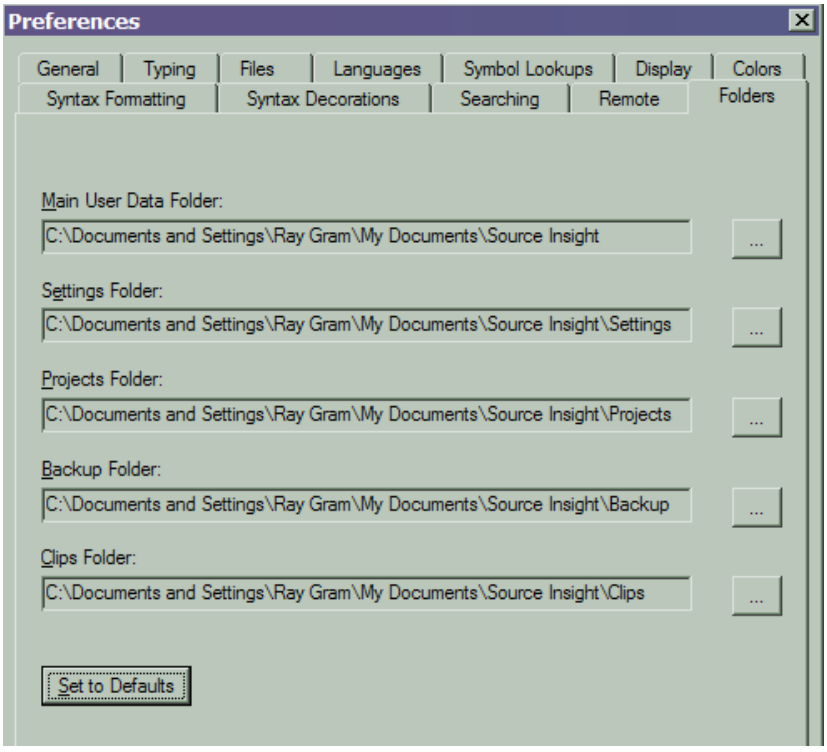
- Windows (CR/LF)
- Unix (LF)
- Mac (CR)

Source Insight が既存のファイルを開いて保存するとき、オリジナル ファイルのフォーマットを維持する点に注意してください。異なるフォーマットで保存するには、**[File] > [Save As]** コマンドを使用します。

Folder Options

[Preferences] ダイアログ ボックスの [Folders] タブを表示します。このタブで、Source Insight で使用される各種データ フォルダの場所を指定します。これらのオプションは、現在の設定の一部として保存されます。

[Preferences] ダイアログ ボックスの [Folders] タブ



[Main User Data Folder] Source Insight の情報が格納されるメイン フォルダです。Source Insight は、このフォルダにいくつかのサブフォルダを作成します。このフォルダを変更すると、サブフォルダも自動的に更新されます。

デフォルトでは、このフォルダは「My Documents\Source Insight」です。そのため、ユーザーごとに別のフォルダが使用されます。

ユーザー データ フォルダの変更

このフォルダの場所をネットワーク ドライブの「My Documents」フォルダに変更することもできます。ただし、ネットワークのデータにアクセスすると、処理が遅くなることがあります。この場合、フォルダの場所をローカル マシンのフォルダに変更してください。

[Settings Folder] 設定ファイルが格納されるフォルダです。

[Projects Folder] プロジェクト データ ファイルが格納されるフォルダです。プロジェクトごとに、このフォルダにサブフォルダが作成されます。

[Backup Folder] バックアップ ソース ファイルが格納されるフォルダです。

[Clips Folder] クリップ ウィンドウからアクセスするクリップが格納されるフォルダです。

Function Down

挿入ポイントをカレント ファイルで定義されている次の関数またはメソッドに移動します。

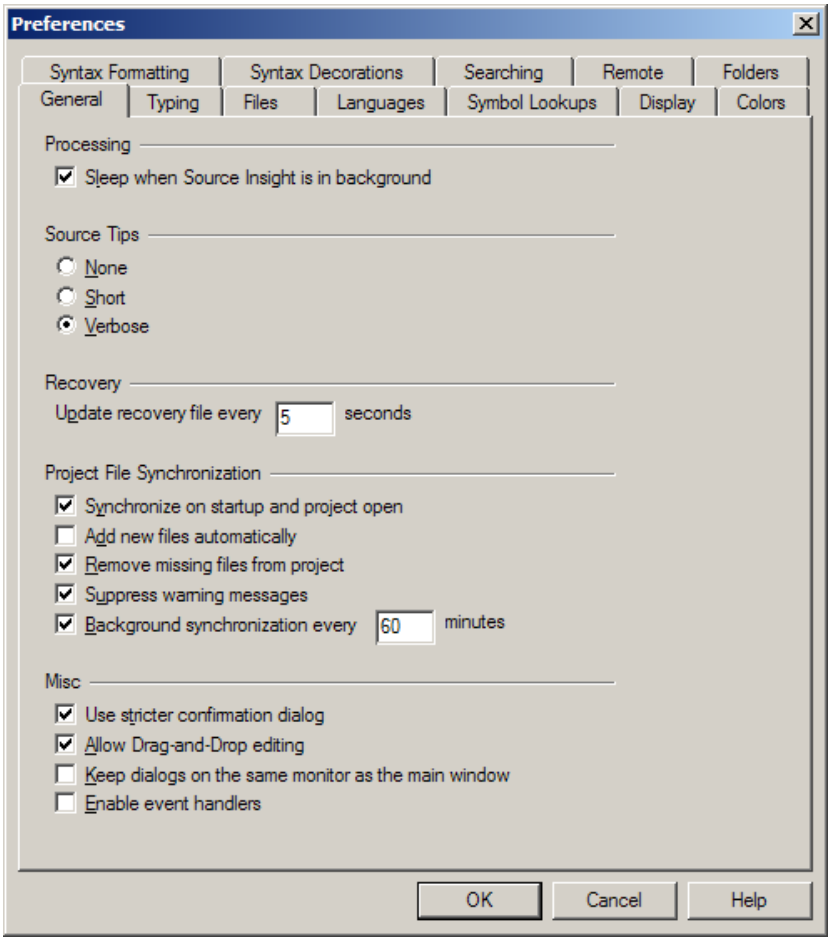
Function Up

挿入ポイントをカレント ファイルで定義されている前の関数またはメソッドに移動します。

General Options

[Preferences] ダイアログ ボックスの [General] タブを表示します。このタブで、さまざまな Source Insight のオプションを変更します。これらのオプションは、現在の設定の一部です。

[Preferences] ダイアログ ボックスの [General] タブ



[Sleep when Source Insight is in background] オンの場合、Source Insight アプリケーションが最小化されているとき、またはアクティブでないとき (カスタム コマンドを実行しているときなど)、Source Insight はバックグラウンド処理を行いません。

オフの場合、通常通りバックグラウンド処理が行われます。しかし、Source Insight アプリケーションがアクティブでないとき、Source Insight は処理の優先度を標準よりも下にします。

バックグラウンド タスク

Source Insight は、次のタスクを含む多くのタスクをバックグラウンドで実行します。

- ファイルを解析して開いているソース ウィンドウすべてについてシンボル ウィンドウの内容を更新します。
- カスタム コマンドの終了を確認します。
- バックグラウンド同期が有効な場合、プロジェクトのファイル (プロセスで新しく追加されます) をすべて同期させます。
- コンテキスト ウィンドウと現在のテキスト選択を同期させます。
- リレーション ウィンドウと現在のテキスト選択を同期させます。
- 開いているファイルが他のプログラムによって Source Insight の外部で変更された場合はリロードします。

通常は、行うことは何もありません。Source Insight はほとんど、またはまったくプロセッサ サイクルを使用しません。

[Source Tips] ポップアップ ソース ヒント ウィンドウで提供される情報のレベルを設定します。ソース ヒント ウィンドウは、マウス カーソルをシンボル識別子の上に移動した後、数秒すると表示されます。

クラッシュ リカバリ オプション

[Recovery: Update recovery file every NNN seconds] Source Insight がクラッシュ リカバリ ファイルを更新する間隔を指定します。デフォルト値は 15 秒です。リカバリ ファイルは最後に更新された後に編集された場合のみ更新されます。リカバリの更新は非常に高速です。実行中でも気づくことはほとんどありません。通常、リカバリ ファイルの保存によって処理が中断されることはありません。このため、この間隔はあまり長くしないことを推奨します。

[Project File Synchronization]

[Synchronize on start-up] このオプションとバックグラウンド同期がどちらもオンの場合、Source Insight を開始するとき、または新規プロジェクトを開くときにファイルのタイムスタンプが確認されます。プロジェクトに関連するファイルの日付が古いことがわかると、Source Insight はファイルを再同期対象としてマークします。後から、ファイルは再スキャンされ、バックグラウンドで同期されます。

[Add new files automatically] オンの場合、ファイルをすべて同期させる前に、Source Insight は、プロジェクトのソース ディレクトリとサブディレクトリに、新規ファイルを再帰的に追加します。しかし、スキャンされるのは既にプロジェクト ファイルに含まれているディレクトリのみです。プロジェクトのソース ディレクトリの派生ディレクトリでないディレクトリはスキャンされません。ソース管理システムなどを使用してプロジェクト ディレクトリに新規ファイルを追加する

と、Source Insight プロジェクトにそれらの新規ファイルが自動的に追加されます。

[Background project synchronization every NNN minutes] オンの場合、Source Insight は編集中に [Synchronize File] コマンドをバックグラウンドで実行します。このオプションがオンの場合、通常は [Synchronize Files] コマンドを実行する必要はありません。

NNN には、ファイルの再同期が必要かどうか判断するためにプロジェクト ファイルを確認する間隔を指定します。この間隔が過ぎると、Source Insight はファイルのタイムスタンプを確認して同期プロセスをバックグラウンドで開始します。

[Use stricter confirmation dialog] オンの場合、Source Insight が操作を確認するとき、「yes」と入力する必要があります。

[Keep dialogs on the same monitor as the main window] オンの場合、メインの Source Insight アプリケーション ウィンドウと同じモニタにダイアログ ボックスを表示します。オフの場合、Source Insight はダイアログ ボックスの位置を記憶します。

[Enable event handlers] オンの場合、マクロ イベント ハンドラが有効になります。詳細は、395 ページの第 7 章「マクロ イベント ハンドラ」を参照してください。

Go Back

挿入ポイントを選択履歴の前の場所に移動します。Source Insight は、最近選択した場所 (最大 100) の循環リストである選択履歴を保存します。選択履歴は、カレント ファイルだけではなく、開いているファイルすべてに対してグローバルです。

Source Insight の [Go Back] と [Go Forward] コマンドは、インターネット ブラウザの [戻る] と [進む] ボタンに似ています。

[Go Back] コマンドを使用して関数呼び出しチェーンを表示する

[Go Back] コマンドは [Jump To Definition] コマンドと一緒に使用すると便利です。関数定義にジャンプする場合、[Go Back] コマンドを使用して関数の呼び出し元に移動します。このプロセスは何度でも繰り返すことができます。[Go Back] および [Go Forward] コマンドを使用して、関数呼び出しチェーンを前後に移動できます。

選択履歴は循環リストなので、結局、始点で終了します。

リストを表示するには、[Selection History] コマンドを使用します。コマンドを使用すると、それぞれの位置と、その場所の関数またはその場所を囲むシンボルを表示します。

Go Back Toggle

[Go Back] および [Go Forward] コマンドの実行を切り替えます。[Go Back Toggle] コマンドを繰り返して使用すると、最後の 2 つの場所が切り替えられます。

Go Forward

挿入ポイントを選択履歴の次の場所に移動します。選択履歴は、選択履歴は、最近選択した場所 (最大 100) の循環リストです。詳細は、[Go Back] コマンドを参照してください。

Go To First Link

最初のソース リンクを検索して次の操作を行います。

1. リンク ソース ファイルのリンク行を選択します。
2. リンク ターゲット ファイルのリンク行を選択します。
3. 両方のファイルがウィンドウで表示されることを保証します。このコマンドが使用されたときにカレント ウィンドウが最大化されていた場合、リンク ターゲット ファイルのみ表示されます。

[Go To Next Link] および [Go To Previous Link] コマンドも、それぞれ、ソース リンクの次と前であることを除いて、同じ操作を行います。

最初のソース リンク

「最初のソース リンク」は、[Go To Link Location] コマンドが使用されたリンク ソース ファイルの最初のリンクです。たとえば、[Search Files] コマンドを使用してソース リンクを含む検索結果ファイルを作成した後、検索結果ウィンドウの行で [Go To Link Location] コマンドを使用すると、最初のソース リンクが検索結果ウィンドウの最初のリンクになります。

[Go To First Link]、[Go To Next Link]、および [Go To Previous Link] コマンドはリンクからリンクに素早くスキップするために使用します。特に、ソース リンクとコンパイラ エラー メッセージおよびプログラム ソースが接続されている場合に便利です。

リンクとコンパイラ エラーの使用

カスタム コマンドを使用して Source Insight からコンパイラを起動し、出力をキャプチャしてエラー メッセージを解析した場合、[Go To First Link] および [Go To Next Link] コマンドを使用してソース ファイルのエラーを表示できます。

「Compile File」カスタム コマンドを定義するとき、[Parse Links in Output] オプションをオンにすると、Source Insight はコンパイラの出力

を検索して、各エラー メッセージにソース リンクを設定します。この場合、「リンク ソース」はコンパイラ出力ファイルの各エラー メッセージです。各リンクの「リンク ターゲット」は、各エラー メッセージで指定されたファイルと行番号です。

エラーとソース行を表示するには

ビルドまたはコンパイル コマンドを実行して Source Insight がエラー メッセージの場所を指すようにするには :

1. 183 ページの「コンパイル コマンドとビルド コマンドの作成」の説明で定義した「Compile File」または「Build Project」カスタム コマンドを実行します。
2. エラーがある場合、コンパイラを終了したときにエラー メッセージがコマンド出力ウィンドウに表示されます。Source Insight はソース リンクを自動的に設定して [Go To First Link] コマンドを実行します。 最初のエラー メッセージとソース行が選択され表示されます。
3. [Go To Next Link] コマンドを実行します。コマンド出力ウィンドウの次のエラー メッセージが選択され、最初のエラーと同様に、リンクのターゲットが表示されます。
4. リンク (エラー メッセージ) がすべて表示されるまで、[Go To Next Link] コマンドを使用して続行します。リンクがそれ以上ない場合、Source Insight はビープを出力して、「No links.」メッセージをステータス バーに表示します。

リンクと検索出力の使用

[Search Files] コマンドは、その出力を検索結果ウィンドウに表示します。検索結果ウィンドウの各行のテキストはソース リンクです。この場合、「リンク ソース」は検索結果ウィンドウの各行です。各リンクの「リンク ターゲット」は、検索パターンが見つかったファイルと行番号です。

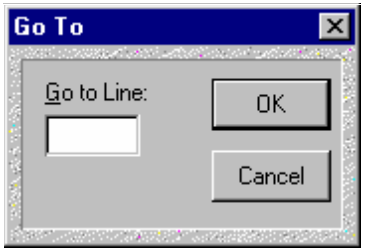
パターンが見つかった場所を表示するには :

検索を実行してパターンが見つかった場所を表示するには :

1. [Search Files] コマンドを実行します。
2. [Go To First Link] コマンドを使用して最初の一致を表示します。
3. [Go To Next Link] コマンドを使用して次の一致を表示します。
4. 「No Links.」メッセージが表示されるまで [Go To Next Link] コマンドを使用します。

Go To Line

挿入ポイントをファイルの指定した行に移動します。



[Go to Line] 行番号を入力します。

[OK] 入力した行番号に移動します。ファイルの最後よりも後の行番号が入力された場合、ファイルの最後に移動します。

[Cancel] [Go To Line] コマンドをキャンセルします。

Go To Next Change

カーソルを編集した行の次のブロックに移動します。カーソルは、変更マークの次のセットに移動されます。

Go To Previous Change

カーソルを編集した行の前のブロックに移動します。カーソルは、変更マークの最後のセットに移動されます。

Go To Next Link

[Go To Next Link] コマンドの動作は、次のリンクが使用されることを除いて、[Go To First Link] コマンドと同じです。[Go To First Link] コマンドも参照してください。

Go To Previous Link

[Go To Previous Link] コマンドの動作は、前のリンクが使用されることを除いて、[Go To Next Link] コマンドと同じです。[Go To First Link] コマンドも参照してください。

Help

Source Insight のヘルプを表示します。ダイアログ ボックスで F1 キーを押すと、現在のコマンドのヘルプが表示されます。

Help Mode

ヘルプ モードをオンにします。ヘルプ モードがオンのときにコマンドが起動されると、Source Insight は、コマンドを実行する代わりに、コマンドのヘルプを表示してヘルプ モードをオフにします。ヘルプ モードをキャンセルするには、[Help Mode] コマンドを再度実行します。

たとえば、[File] > [Open] コマンドのヘルプを表示するには、Ctrl+F1 を押してヘルプ モードをオンにします。ステータス バーにヘルプ モードがアクティブであることを示すメッセージが表示されます。[File] メニューから [Open] コマンドを選択します。ヘルプ ウィンドウが開き、[Open] コマンドのヘルプが表示されます。

ダイアログ ボックスで F1 キーを押すと、現在のコマンドのヘルプが表示されます。

Highlight Word

すべてのソース ウィンドウで、カーソルの場所の単語をハイライト表示します。紙面で蛍光ペンを使用するのに似ています。単語を選択して [Highlight Word] コマンドを使用すると、ソースで単語が含まれるすべての場所がハイライト表示されます。

デフォルトでは、背景は明るい黄色、テキストは黒の太字で表示されます。ハイライト効果は、[Highlight] スタイルを編集して設定できます。スタイルを編集するには、[Style Properties] コマンドを使用します。

Incremental Search

[Increment Search] および [Incremental Search Backward] コマンドはインクリメンタル サーチ モードを起動します。デフォルトでは、F12 が [Incremental Search] コマンドに割り当てられています。

インクリメンタル サーチ モードを開始すると、Source Insight は現在のカーソルの場所から入力した文字の検索を開始します。多くの文字を入力するほど、検索はより明確になります。入力した文字は、ステータス バーの下部に表示されます。

インクリメンタル サーチ モードを終了するには、任意のコマンド キー (矢印キーなど) または Esc を押します。検索を再度実行するには、F12 を 2 回押します。前回のパターンがロードされ、検索が実行されます。

Incremental Search Mode

F12 を押すと、インクリメンタル サーチ モードが開始します。インクリメンタル サーチ モードでは次のことが行えます。

- F12 は現在の文字列で再度検索を実行します。現在の文字列が空の場合、前回の文字列をロードします。
- Shift+F12 は逆方向に検索を実行します。
- Backspace は検索文字列を 1 文字削除します。
- Esc は検索をキャンセルして最初の場所に戻ります。
- Enter は検索を停止します。現在の場所の選択範囲はそのままです。
- コマンドにマップされている任意のキーが押されると、インクリメンタル サーチ モードは終了し、現在の場所の選択範囲はそのままで、コマンドが実行されます。
- その他のキーが押されると、現在の検索パターンに文字が追加されます。現在の検索パターンはステータス バーの下部に表示されます。
- 検索バッファには、最後に成功した検索パターンが残されます。

インクリメンタル サーチ パターンは、大文字を入力した場合を除いて、大文字と小文字を区別しません。

Incremental Search Backward

カレント ファイルをインクリメンタルに逆方向に検索します。216 ページの「Incremental Search」も参照してください。

Horizontal Scroll Bar

ソース ファイル ウィンドウの水平スクロール バーの表示を切り替えます。

HTML Help

HTML ヘルプ ファイルで現在選択している単語をキーワード検索します。HTML ヘルプ ファイルは、[Setup HTML Help] コマンドを使用して指定します。306 ページの「Setup HTML Help」も参照してください。

Indent Left

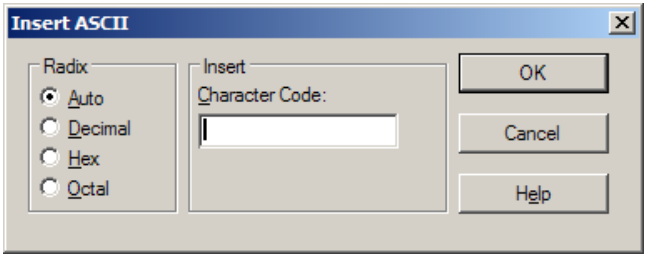
現在の選択範囲の行を 1 タブ ストップ分左にアウトデントします。# で始まる行はインデントされません。

Indent Right

現在の選択範囲の行を 1 タブ ストップ分右にインデントします。# で始まる行はインデントされません。Source Insight は、タブ文字を挿入して行を右にインデントします。現在のドキュメント タイプで [Expand tabs to spaces] オプションを有効にしている場合、タブ文字の代わりにタブ ストップと等価な数のスペースが挿入されます。

Insert ASCII

文字を ASCII コードで挿入します。

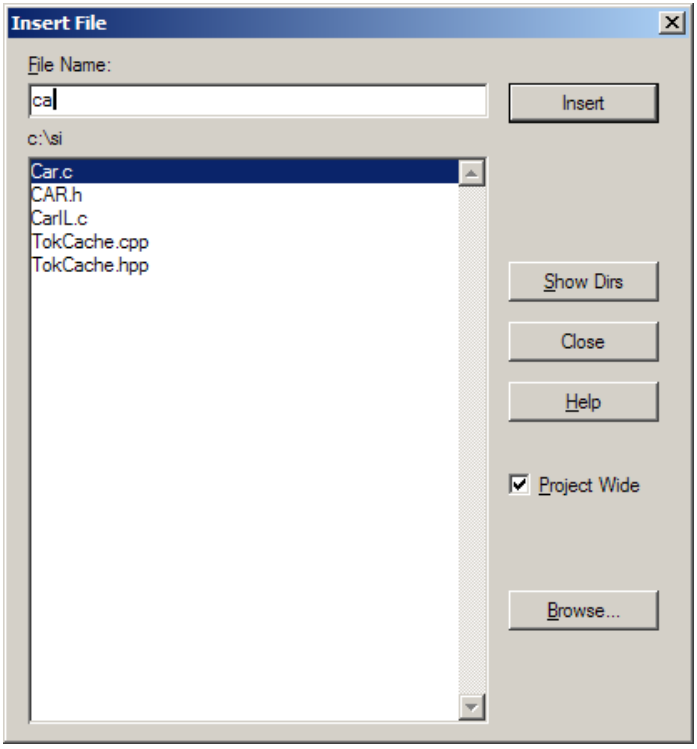


[Radix] 入力する文字コードの基数を選択します。[Auto] の場合、基数は入力した内容によって決定されます。たとえば、0x20 と入力すると、基数は Hex であると判断され、ASCII 32 が挿入されます。

[Character Code] 挿入する文字の ASCII コードです。

Insert File

別のファイルのテキストを現在の選択範囲に貼り付けます。



[File Name] 挿入するファイルの名前です。ワイルドカードを入力して [Insert] ボタンをクリックすると、Source Insight はワイルドカード拡張の結果をファイル リストに表示します。ワイルドカードはカレント ディレクトリで拡張されます。

ファイル リスト [Project Wide] オプションがオンの場合、カレント プロジェクトに含まれるすべてのファイルが表示されます。オフの場合、現在の作業ディレクトリに含まれるすべてのファイルが表示されます。現在の作業ディレクトリのパスがリスト ボックスの上に表示されます。

[Insert] [File Name] テキスト ボックスのファイルの内容を挿入します。[File Name] テキスト ボックスに 1 つ以上のワイルドカードが含まれている場合、Source Insight はワイルドカード拡張の結果をファイル リストに表示します。ワイルドカードはカレント ディレクトリで拡張されます。

[Show Dirs/Show Files] リスト ボックスに表示する内容 (ファイル名またはサブディレクトリ名) を切り替えます。

[Browse] Windows の [ファイルを開く] ダイアログ ボックスを表示します。

[Project Wide] オンの場合、ファイル リストにカレント プロジェクトに追加されているファイルがすべて表示されます。オフの場合、ファイル リストに現在の作業ディレクトリにあるファイルのみが表示されます。プロジェクトが開かれていない場合、このオプションは常にオフです。

Insert Line

選択している行の前に、空の新しい行を挿入します。カーソルが行の先頭または最後にある必要はありません。

[Auto Indent] がオンの場合、新しい行は下の行に合わせてインデントされます。

Insert Line Before Next

選択している行の後に、空の新しい行を挿入します。[Auto Indent] がオンの場合、新しい行は上の行に合わせてインデントされます。

Insert New Line

挿入ポイントに新しい行を挿入します。カーソルが次の行に移動しないことを除いて、Enter を押すことと同じです。

Join Lines

挿入ポイントがある行と次の行を結合して 1 行にします。選択範囲が拡張されている場合、選択範囲に含まれている行がすべて結合されます。

Jump To Base Type

選択している変数またはタイプの基本構造タイプにカーソルを移動します。たとえば、以下のコードを考えてみます。

```
struct MyStruc
    {
    int afield;
    int anotherfield;
    };

// MS type is defined as struct MyStruct
typedef struct MyStruct MS;

MS ms;// declare ms with a type of "MS"
x = ms.afield;
```

割り当てステートメントの ms (または ms 変数が表示される場所) にカーソルを移動して [Jump To Base Type] コマンドを使用すると、変数の基本構造タイプである struct MyStruc の定義にジャンプします。MS の typedef では停止しません。

Jump To Caller

選択している関数がある場合、その関数の呼び出し元にジャンプします。たとえば、関数名にカーソルを移動して [Jump To Caller] コマンドを使用すると、その関数を呼び出している関数にジャンプします。複数の関数がその関数を呼び出している場合、関数のリストが表示されます。

Jump To Definition

現在の選択範囲の最初の単語からシンボルを取得して、その定義にジャンプします。[Go Back] および [Go Forward] コマンドはジャンプスポットを前後に移動するのに便利です。

このコマンドを使用するには :

5. ソース ファイルでシンボル名を選択します。
6. Alt+= と入力して実際のシンボル定義にジャンプします。

マウス ショートカット

デフォルト設定では、マウスを使用してコマンドを起動することもできます。ファイルでシンボルをポイントして、Ctrl を押しながらマウスをクリックすると、[Select Word] コマンドが実行されます。Ctrl を押しながらマウスをダブルクリックすると、[Jump To Definition] コマンドが実行されます。

マウスでこのコマンドを使用するには、Ctrl キーを押したままで、シンボルをポイントしてダブルクリックします。

ヘッダー ファイルを開く

カーソルが #include ステートメントのようにファイル名をポイントしているときに [Jump To Definition] コマンドを使用すると、ヘッダー ファイルを開くことができます。

Jump To Link

ソース リンクのリンク先へ移動します。213 ページの「Go To First Link」も参照してください。

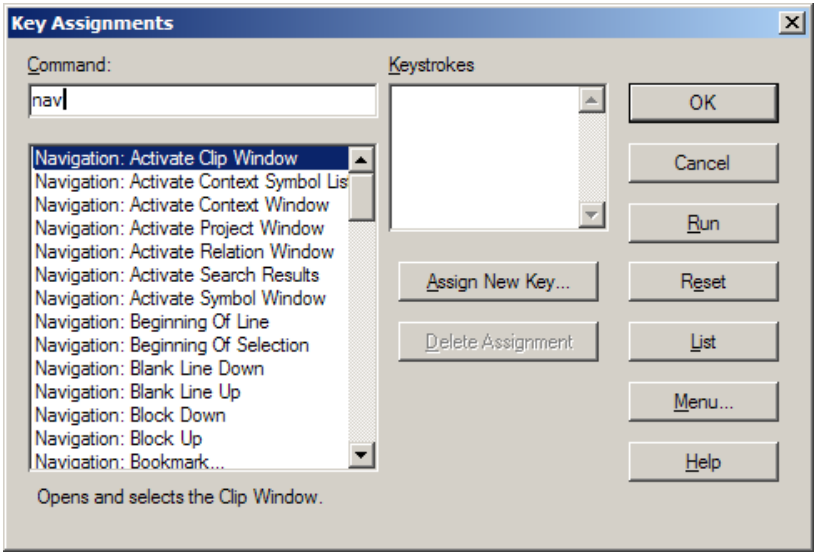
Jump To Prototype

選択している関数のプロトタイプの宣言にジャンプします。このコマンドは、選択しているシンボルが関数の場合にのみ動作します。

Key Assignments

コマンドにキーストロークの組み合わせを割り当てます。または、割り当てを削除します。マウス ボタンをコマンドに割り当てることもできます。キー割り当ては、現在の設定の一部です。

[Key Assignments] ダイアログ ボックス



[Command] コマンド名を入力します。入力した内容に応じて該当するコマンドのみがリストされ、コマンドを簡単に見つけることができます。シラブル マッチングが使用されるため、コマンド名に含まれている単語の一部を入力するだけでかまいません。

コマンド リスト ユーザーが定義したマクロとカスタム コマンドを含む、すべての Source Insight コマンドがリストされます。ここでコマンドを選択すると、そのコマンドに現在割り当てられているすべてのキーストロークを含むキーストローク リストがロードされます。

[Keystrokes] リスト 選択しているコマンドに割り当てられているすべてのキーストロークをリストします。キーストロークを削除するには、キーストロークを選択してから、[Delete Assignment] ボタンをクリックします。

[OK] 現在の設定に新しいキー割り当てを保存します。

[Cancel] コマンドをキャンセルします。現在の設定は変更されません。

[Assign New Key…] 新しいキーストロークまたはマウス クリックを [Command] リストで選択しているコマンドに割り当てます。キーの入力を指示するウィンドウがポップアップします。

[Delete Assignment] [Command] リストで選択しているコマンドから [Keystrokes] リストで選択しているキーストロークの割り当てを削除します。

[Run] 選択しているコマンドを実行します。行った変更も保存します。

[Reset] キー割り当てを初期設定に戻します。Source Insight は確認を表示します。

[List] キー割り当てリスト ファイルを作成します。行った変更も保存します。リスト ファイルは、コマンドのリストとキー割り当てを含むテキスト ファイルです。

[Menu…] 現在の設定に新しいキー割り当てを保存して、[Menu Assignments] コマンドを実行します。248 ページの「Menu Assignments」も参照してください。

数値キーパッド キー

数値キーパッド キー / * - + にはデフォルトで以下のコマンドが割り当てられています。

キー	コマンド
/	Scroll Half Page Up
*	Scroll Half Page Down
-	Function Up
+	Function Down

これらのキーを数値キーとして使用するには、コマンドからキー割り当てを削除する必要があります。

[Options] > [Key Assignments] ダイアログ ボックスを使用して、これらのコマンドを検索し、コマンドからキー割り当てを削除します。キー割り当てが削除されると、これらのキーは数値キーとして動作します。

キーとマウス クリックの割り当て

キーとマウス クリックの割り当ての手順を以下に説明します。

キーストロークを割り当てるには

マウス ボタンを含む。Alt、Ctrl、および Shift キーと任意のキーの組み合わせを追加できます。

コマンドに新しいキーストロークの組み合わせを割り当てるには :

1. [Command] リストでコマンドを選択します。
2. [Assign New Key…] ボタンをクリックします。
3. 割り当てるキーストロークを入力します。割り当てをキャンセルするには、Esc を押します。入力したキーストロークが異なるコマンドに既に割り当てられている場合、Source Insight は割り当てを変更するかどうか確認します。

マウス クリックを割り当てるには

コマンドにマウス クリックを割り当てるには :

1. [Command] リストでコマンドを選択します。
2. [Assign New Key…] ボタンをクリックします。
3. 割り当てるマウス ボタンをクリックします。Alt、Shift、Ctrl のようなキーを追加する場合は、マウス ボタンをクリックする前にキーを押します。マウスの右ボタンを変更するためにマウスの左ボタンを使用することもできます。割り当てをキャンセルするには、Esc を押します。

キーの割り当てを削除するには

コマンドからキーストロークの割り当てを削除するには :

1. [Command] リストでコマンドを選択します。
2. [Keystrokes] リストで削除するキーストロークを選択します。
3. [Delete Assignment] ボタンをクリックします。

Keyword List

現在の言語の構文フォーマットで使用される言語キーワードを編集する [Language Keywords] ダイアログ ボックスを表示します。

[Language Keywords] ダイアログ ボックスにはキーワードとスタイルがリストされます。ダイアログ ボックスのタイトルには、作業している

言語の種類が示されます。Source Insight は、非常に高速なハッシュ テクニックを使用して、優れたパフォーマンスを保ちながら大規模なキーワード リストを管理します。

Keyword および Style

キーワード リストには、構文フォーマットでハイライト可能な言語キーワードがすべて含まれています。リストの各キーワードはスタイル名と関連付けられています。各スタイルのフォーマット オプションを設定するには、[Style Properties] コマンドを使用します。

たとえば、C 言語キーワード リストでは、単語「NULL」は「Null Value」スタイルと関連付けられています。

指定された単語のフォーマットを決定するため、Source Insight は適切な言語タイプのキーワード リストで単語を検索します。キーワード リストには、スタイルと関連付けられているフォーマットを示すスタイル名が含まれます。

つまり、ファイル名およびファイル内の単語で始まります。Source Insight は、この情報を使用して単語のスタイルを決定します。

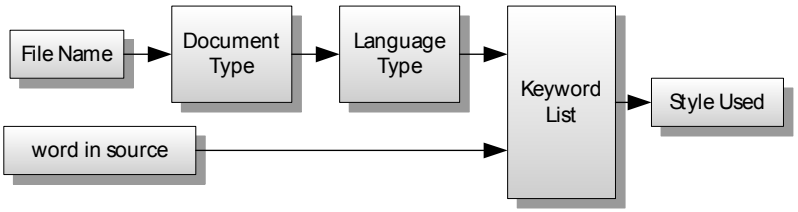
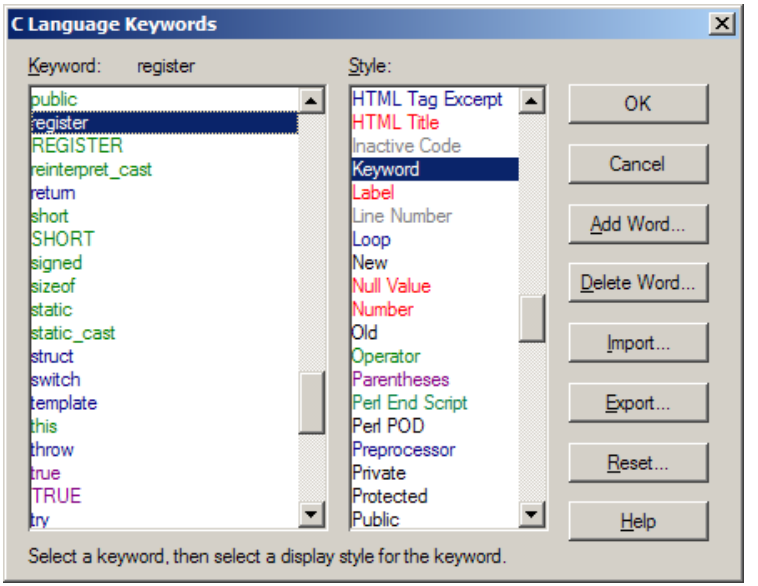


図 5.2 ソース テキストの単語に使用されるスタイルは、ファイルのドキュメント タイプで指定されている言語のキーワード リストにより決定されます。

キーワードをフォーマット スタイルと関連付けることで、[Style Properties] コマンドを使用してスタイルを変更することで、構文フォーマットを素早く変更できます。変更したスタイルと関連付けられているすべての言語キーワードに、新しいスタイル フォーマットが反映されます。

[Language Keywords] ダイアログ ボックス



[Keyword] その言語のキーワード リストです。リストからキーワードを選択すると、そのキーワードに関連付けられているスタイルが [Style] リストで選択されます。

[Style] すべての構文フォーマット スタイルのリストです。このリストからスタイルを選択すると、[Keyword] リストで選択しているキーワードと関連付けられているスタイルを編集します。スタイル名をダブルクリックしてスタイルを編集することもできます。

[OK] 変更を保存します。

[Cancel] コマンドをキャンセルします。変更は保存されません。

[Add Word…] 新しい単語をキーワード リストに追加します。スペースを含まない単一の単語を入力できます。Source Insight は、入力された単語をキーワード リストに追加します。単語を追加した後、関連付けるスタイルを選択してください。

[Delete Word…] 現在キーワード リストで選択している単語を削除します。

[Import…] 外部テキスト ファイルから新規キーワード リスト エントリをインポートします。詳細は、この後のセクションを参照してください。

[Export…] キーワード リストをテキスト ファイルにエクスポートします。

[Reset...] 言語キーワード リストを初期設定に戻します。

キーワード リストのインポートとエクスポート

キーワードとスタイルの関連付けをテキスト ファイルにエクスポート、またはテキスト ファイルからインポートするには、[Language Keyword] ダイアログ ボックスの [Import] と [Export] ボタンを使用します。

テキスト ファイルには、次のように、行ごとにキーワード、スタイル名のペアが記録されます。

```
<keyword> , <style-name>          または
"<keyword>", "<style-name>"
```

各キーワードは、空白を含まない単一の単語です。スタイル名は、[Language Keywords] ダイアログ ボックスの [Sylye] リストまたは [Style Properties] ダイアログ ボックスの [Style Name] リストにリストされている定義済みスタイル名の 1 つです。キーワードまたはスタイル名は二重引用符で囲んでもかまいません。

インポートの際、重複キーワードは無視されます。

[Import Options]

[Language Keywords] ダイアログ ボックスで [Import] ボタンをクリックしたとき、キーワード リストを置換するかマージするか、いずれかのオプションを選択します。

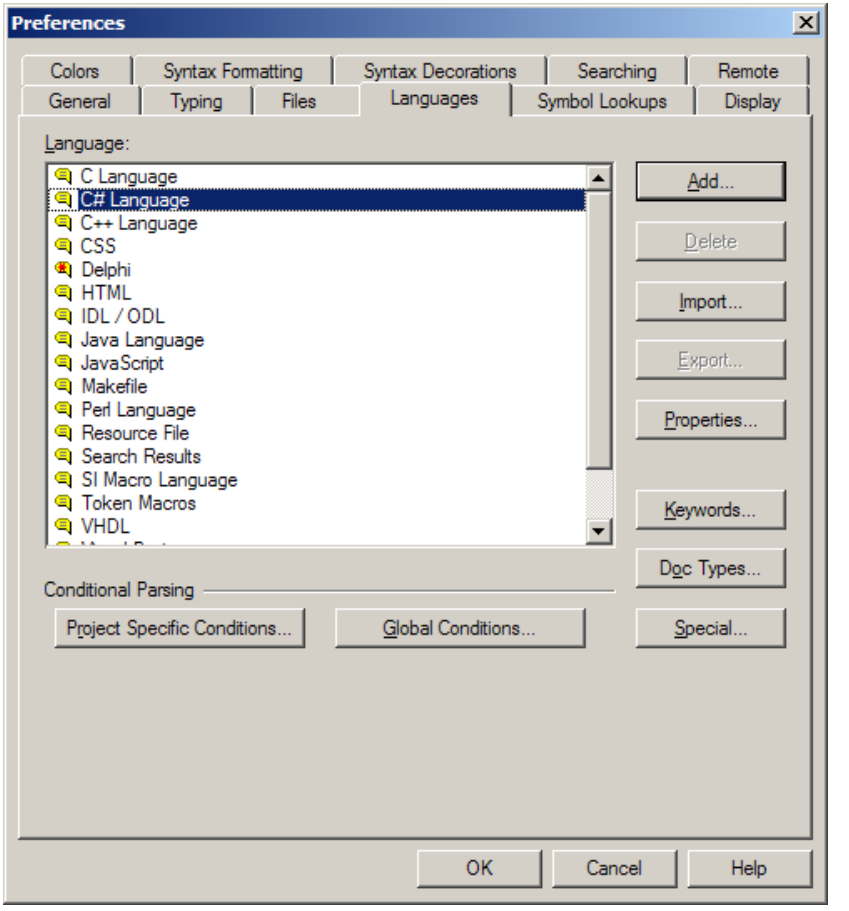
[Replace current keyword list with imported list] 現在のキーワード リストをインポートするリストで完全に置換するには、このオプションを選択します。リストをインポートした後でも、[Language Keywords] ダイアログ ボックスで [Cancel] ボタンをクリックすることで、キーワード リストに対して行った変更を無効にできます。

[Merge current keyword list with imported list] 現在のキーワード リストとインポートするリストをマージするには、このオプションを選択します。マージは、インポートしたリストを既存のリストに追加します。インポートしたキーワードは現在のリストのキーワードと置換されます。

Language Options

[Preferences] ダイアログ ボックスの [Languages] タブを表示します。このタブで、言語キーワード リストのような言語固有のオプションを編集します。

Source Insight は、ビルトインとカスタムの 2 種類の言語をサポートします。ビルトイン言語のいくつかのオプションは変更できます。カスタム言語については、汎用言語のパラメータをすべて制御できます。



[Language] インストールされているすべての言語のリストです。カスタム言語はアイコンに赤のアスタリスクが表示されます。

[Document Options] コマンドを使用して言語と特定のドキュメント タイプを関連付けることができます。194 ページの「Document Options」も参照してください。

[Add…] 新規カスタム言語を追加します。231 ページの「Language Properties」も参照してください。

[Delete] 選択しているカスタム言語を削除します。カスタム言語のみ削除できます。ビルトイン言語は削除できません。

[Import…] カスタム言語ファイル (.CLF) からカスタム言語をリストにインポートします。カスタム言語ファイルには、単一カスタム言語のプロパティがすべて含まれます。

[Export…] 選択しているカスタム言語をカスタム言語ファイル (.CLF) にエクスポートします。カスタム言語ファイルには、単一カスタム言語のプロパティがすべて含まれます。他のユーザーは、このカスタム言語ファイルを各自の Source Insight 設定にインポートできます。

[Properties…] [Language Properties] ダイアログ ボックスを表示します。このダイアログ ボックスを使用してカスタム言語プロパティを制御します。231 ページの「Language Properties」も参照してください。

[Keywords…] 選択している言語タイプと関連付けられているキーワード リストを編集します。[Language Keywords] ダイアログ ボックスを表示します。

[Doc Types…] [Document Options] ダイアログ ボックスを開きます。

[Special…] 選択している言語固有の特殊なオプションを表示します。特殊なオプションがない言語もあります。

[Conditional Parsing]

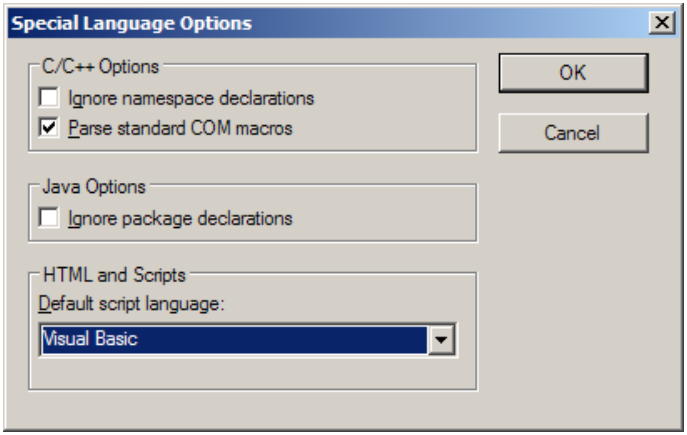
[Project Specific Conditions…] カレント プロジェクトのみに固有な定義済みの条件を編集します。これらの条件はカレント プロジェクトが開かれている場合にのみ (プロジェクトに属しているファイルに対してのみ) 有効です。

[Global Conditions…] グローバル条件セットを編集します。グローバル条件は、すべてのプロジェクトに対して定義されます。プロジェクト固有の条件セットとグローバル条件セットの両方を組み合わせたセットが、指定したプロジェクトの定義済み条件の合計セットになります。プロジェクト固有の条件は、同じ名前のグローバル条件よりも優先されます。201 ページの「Edit Condition」も参照してください。

[Special Language Options]

[Preferences: Languages] ダイアログ ボックスで [Special…] ボタンをクリックすると、[Special Language Options] ダイアログ ボックスが表示

されます。このダイアログ ボックスで、ビルトイン言語の特殊なオプションを制御します。



[C/C++ Options]

[Ignore namespace declarations] オンの場合、C++ コードの名前空間宣言は無視されます。名前空間で宣言されたすべてのシンボルは、名前空間宣言を記述していないものとして、ファイル スコープにあると見なされます。

オフの場合 (デフォルト)、名前空間で宣言されたすべてのシンボルは、名前空間スコープにあると見なされます。

[Parse standard COM macros] オンの場合、STDMETHOD のような標準 COM ヘルパー プリプロセッサ マクロが認識され解析されます。c.tom トークン マクロ ファイルにこれらのマクロのエントリが既に含まれている場合、このオプションは効果がないことに注意してください。

[Java Options]

[Ignore package declarations] オンの場合、Java ファイルのパッケージ宣言は無視されます。パッケージ ステートメントの後に宣言されたすべてのシンボルは、「global」パッケージ スコープにあると見なされます。つまり、同じ仮想パッケージにあると見なされます。

オフの場合 (デフォルト)、パッケージ ステートメントの後に宣言されたすべてのシンボルは、パッケージ スコープにあると見なされます。

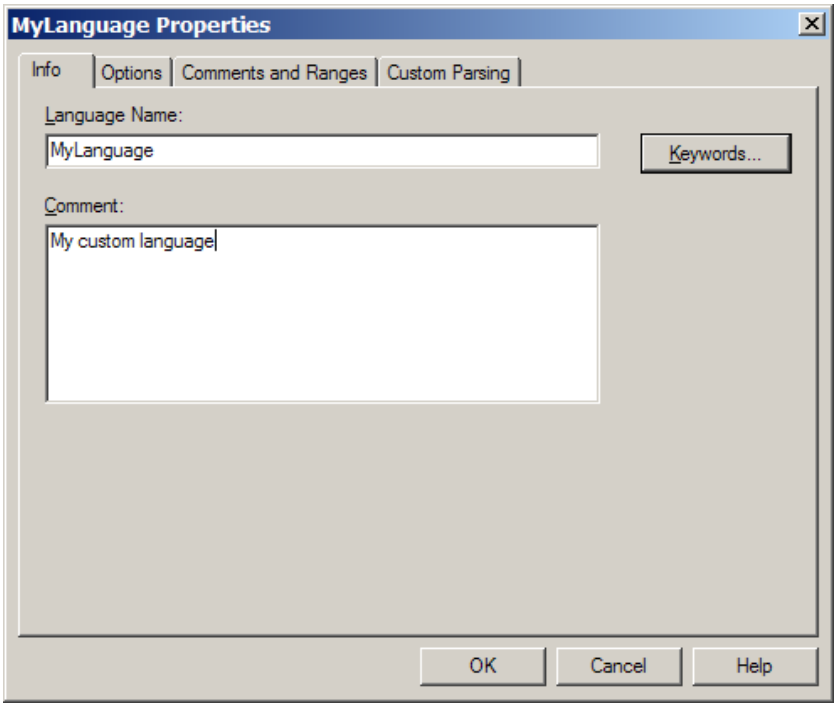
[HTML and Scripts] オプション

[Default script language] このコントロールを使用する HTML または ASP で使用するデフォルトのスクリプト言語を指定します。デフォルトのスクリプト言語は、スクリプトで別の言語が指定されていない場合にのみ使用されます。

Language Properties

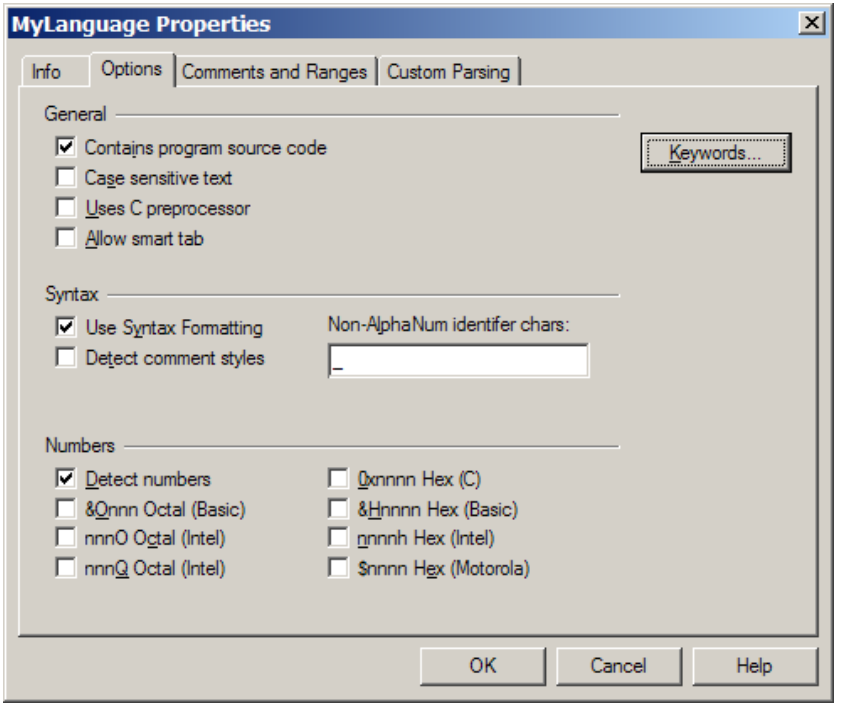
現在選択している言語のプロパティを表示します。[Preferences: Language] ダイアログ ボックスで [Properties…] ボタンをクリックすると、[Language Properties] ダイアログ ボックスが表示されます。

Source Insight は、ビルトインとカスタムの 2 種類の言語をサポートします。ビルトイン言語のいくつかのオプションは変更できます。カスタム言語については、汎用言語のパラメータをすべて制御できます。



[Options] タブ

各言語には、Source Insight がその言語のファイルを処理する際に利用する基本的なオプションがあります。C/C++ のようなビルトイン言語では、カスタム言語よりもオプションは少なくなります。



[Contains program source code] オンの場合、Source Insight はこの言語をプログラミング言語であると見なします。単純なテキスト形式言語ではなく、プログラミング言語が使用された場合、特定の機能は変更されます。たとえば、宣言済みシンボルへの参照は、「Ref to …」スタイルで表示されます。

[Case sensitive text] 言語で大文字と小文字を区別するかどうか示します。キーワードの一致とシンボル ルックアップ エンジンにおけるシンボル名の解決に影響します。

[Uses C preprocessor] オンの場合、Source Insight は #if および #ifdef プリプロセッサ ディレクティブを認識します。

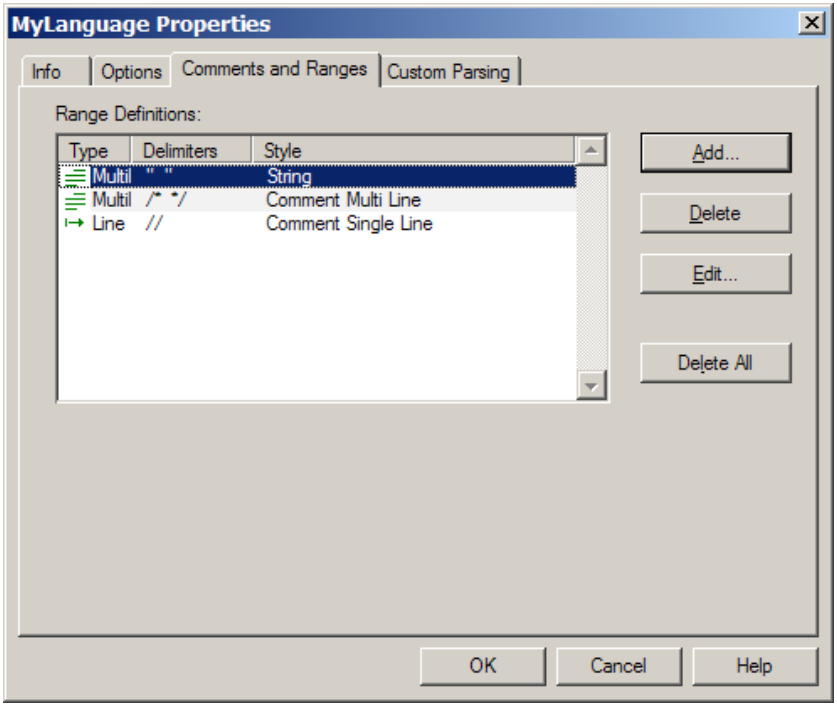
[Allow smart tab] オンの場合、この言語を編集するときに Smart Tab 機能が有効になります。オフの場合、Smart Tab は単純なタブのように動作します。

[Use Syntax Formatting] オンの場合、この言語のファイルを表示するときに構文フォーマットが使用されます。

[Detect comment styles] オンの場合、特殊なコメント スタイルが検出されます。107 ページの「コメント スタイル」も参照してください。

[Comments and Ranges] タブ

コメントと他の複数行範囲要素の解析方法を指定します。引用文字列は非コメント複数行範囲要素の例です。



[Add…] [Range Definition] ダイアログ ボックスで新規範囲要素を追加します。234 ページの「[Range Definition]」も参照してください。

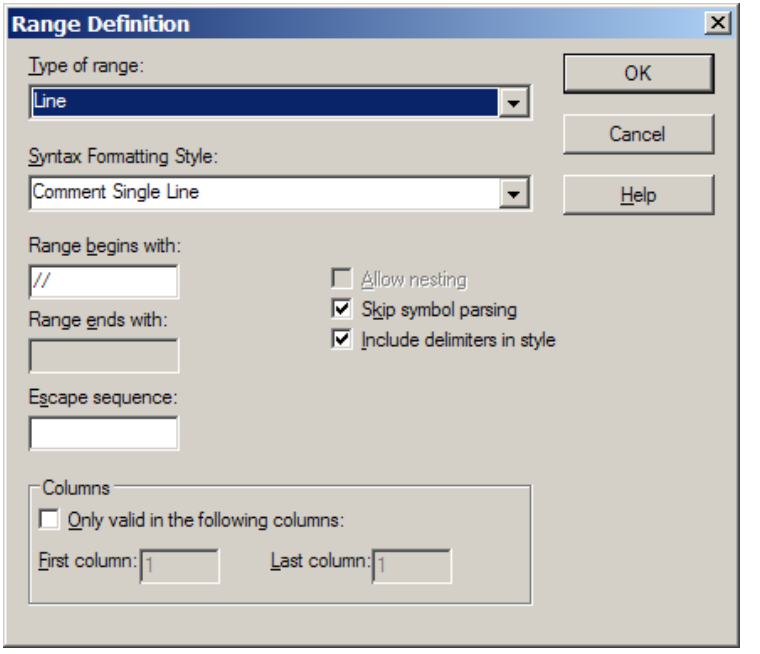
[Delete] 選択しているドキュメント タイプを削除します。

[Edit…] [Range Definition] ダイアログ ボックスで範囲要素のプロパティを編集します。234 ページの「[Range Definition]」も参照してください。

[Delete All] すべての範囲要素を削除します。

[Range Definition]

[Language Properties] ダイアログ ボックスの [Comments and Ranges] タブで新規範囲要素を追加、または範囲要素を編集すると、[Range Definition] ダイアログ ボックスが表示されます。このダイアログ ボックスで、範囲のプロパティをすべて制御できます。範囲定義は、コメントと他の複数行範囲要素の解析方法を指定します。引用文字列は非コメント複数行範囲要素の例です。



[Type of range] リストから範囲要素の種類を選択します。2 種類の範囲要素があります。

- **[Line]** 範囲は区切り文字で開始し、行の最後で終了します。複数行に分割できません。
- **[Multiline]** 範囲は区切り文字で開始し、別の区切り文字で終了します。範囲は複数行に分割できますが、単一行に含めることもできます。(引用符のように) 同じ区切り文字を開始と終了に指定できます。

リストには、引用文字列範囲と、いくつかのコメント スタイルのプリセットが含まれています。プリセットの 1 つを選択すると、そのプリセットのパラメータがダイアログ ボックスの他のテキスト ボックスにロードされます。

[Syntax Formatting Style] 範囲に適用する構文フォーマット スタイルを指定します。通常は、コメント スタイルを選択します。しかし、任

意のスタイルを選択してかまいません。範囲要素で引用文字列を記述している場合、「String」スタイルを選択します。311 ページの「Style Properties」も参照してください。

メモ： スタイルは範囲全体に適用されます。スタイルは、「Reference To…」スタイルのような自動的に適用される他のスタイルよりも優先されます。

[Range begins/ends with] 範囲を開始および終了する区切り文字トークンを指定します。トークンは 15 文字まで指定できます。[Line] 範囲を指定した場合、1 つのテキスト ボックスのみ有効になります。[Multiline] 範囲を指定した場合、開始と終了を同じにできます。引用文字列の場合はこれに該当します。

[Escape sequence] 開始または終了区切り文字の前にこのエスケープ シーケンスが記述されている場合、区切り文字は無視されます。たとえば、バックスラッシュ (\) を引用文字列のエスケープ シーケンスとして指定すると、「a string with \」 embedded\」 quotes」のように文字列の内部に引用文字を記述できます。

[Allow nesting] [Multiline] 範囲が指定されている場合にのみ適用されます。このオプションがオンの場合、範囲は入れ子にできます。ただし、開始と終了区切り文字が同じ場合は、入れ子にできません。

[Skip symbol parsing] ファイルがシンボル定義用に解析されたとき、範囲の内容を無視します。

[Include delimiters in style] 開始と終了区切り文字を選択したスタイルでフォーマットします。このオプションがオフの場合、区切り文字は「Delimiter」スタイルでフォーマットされます。

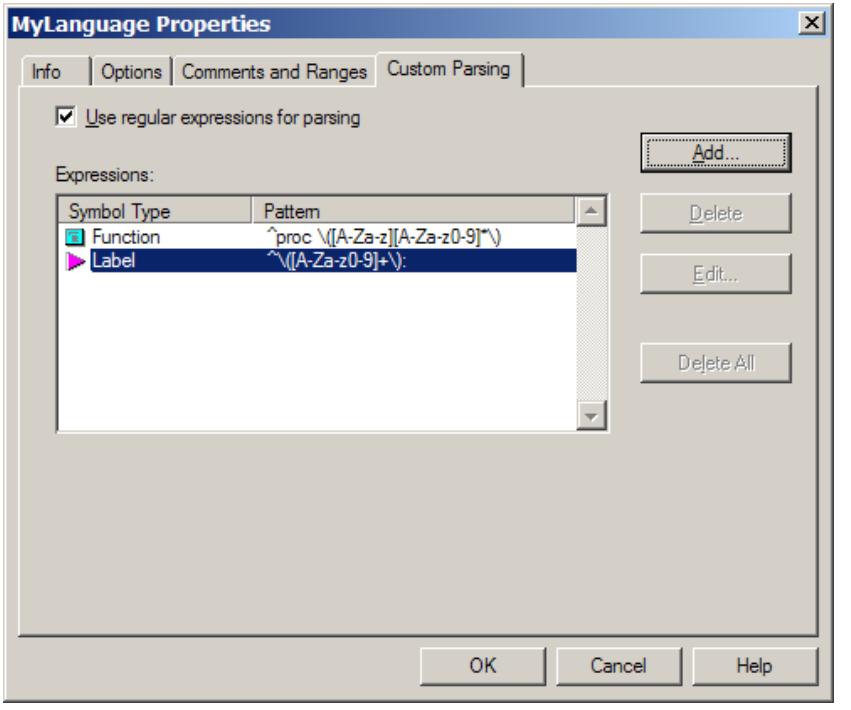
[Columns] グループ

範囲要素が含まれる範囲を制御します。

[Only valid in the following columns] オンの場合、範囲は開始区切り文字が **[First column]** で開始して **[Last column]** までの範囲にある場合にのみ認識されます。オフの場合、カラムは無視されます。

[Custom Parsing] タブ

正規表現のセットを入力して、言語ソース ファイルで単純な解析操作を実行できます。



[Use regular expressions for parsing] オンの場合、カスタム解析表現を有効にします。オフの場合、カスタム解析の使用を無効にします。

[Expressions] 各解析表現とシンボルのタイプをリストします。Source Insight がファイルを解析するとき、すべての表現がファイル全体に適用されます。

[Add...] 新しい解析表現をリストに追加します。237 ページの「Custom Parsing Expression」も参照してください。

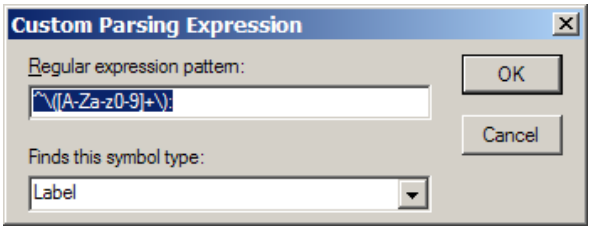
[Delete] 選択している表現を削除します。

[Edit...] 表現を編集します。237 ページの「Custom Parsing Expression」も参照してください。

[Delete All] リストから表現をすべて削除します。

Custom Parsing Expression

[Language Properties] ダイアログ ボックスの [Comments and Ranges] タブで [Add…] または [Edit…] ボタンをクリックすると、[Custom Parsing Expression] ダイアログ ボックスが表示されます。



[Regular expression pattern:] ファイルのシンボル定義の解析に使用する正規表現を指定します。表現には 1 つのグループが含まれている必要があります。グループで、マッチング パターンのどの部分がシンボル名であるか説明します。カスタム パターンを使用すると、Source Insight にあらかじめ含まれていないファイルのシンボルを解析できます。たとえば、以下の文字列は、WIN.INI のような INI ファイルのセクションを解析します。

```
^\[\(.*\)\]
```

詳細は、113 ページの「正規表現」を参照してください。

[Finds this symbol type:] どのタイプのシンボルがパターンの解析によって見つかるか指定します。リストには、すべての可能なシンボル タイプが含まれています。シンボル タイプは固定で拡張できません。シンボル タイプは、シンボルの表示に使用する構文フォーマット スタイルも決定します。

カスタム解析シンボルのスタイル

カスタム解析表現に指定するシンボル タイプは、シンボルの宣言と参照に使用するスタイルを決定します。各シンボル タイプ X には、対応する「Ref to X」および「Declare X」スタイルがあります。

シンボル定義が解析されると、シンボル名を表示するときに対応する「Declare…」スタイルが使用されます。たとえば、シンボル定義が [Function] の場合、関数名は「Declare Function」スタイルで表示されます。311 ページの「Style Properties」も参照してください。

Line Numbers

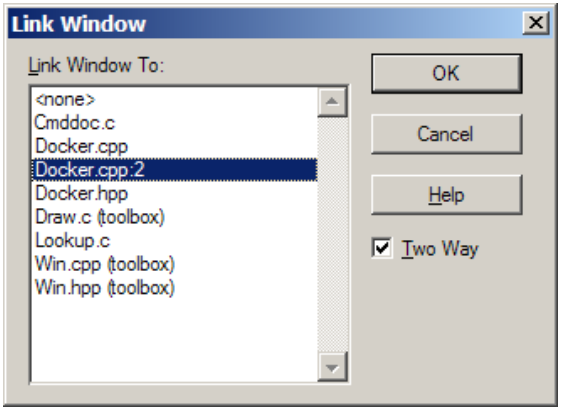
行番号の表示を切り替えます。行番号は、各行の左に表示されます。行番号は、「Line Number」スタイルで表示されます。[Style Properties] コマンドを使用して、このスタイルを編集できます。

Link All Windows

[Link All Windows] モードを切り替えます。[Link All Windows] モードがオンの場合、任意のウィンドウをスクロールすると、他のウィンドウもすべてスクロールします。[Link All Windows] モードは、[Link Window] コマンドでセットアップされる標準のウィンドウ リンクよりも優先されます。

Link Window

カレント ウィンドウと別のウィンドウをリンクして、同時にスクロールします。[Link All Windows] コマンドと異なり、このコマンドでは、リンクするウィンドウを指定します。



カレント ウィンドウと別のウィンドウをリンクしたら、カレント ウィンドウをスクロールするとリンクしたウィンドウもスクロールします。

[Two Way] チェック ボックスがオンの場合、ウィンドウは互いにリンクされるので、いずれかのウィンドウがスクロールすると、別のウィンドウもスクロールします。しかし、チェック ボックスがオフの場

合、片方向のみのリンクになります。リンクしたウィンドウをスクロールしても、別のウィンドウはスクロールしません。

リンク内容	結果
A -> B にリンク	ウィンドウ A からウィンドウ B にリンクされます。ウィンドウ A をスクロールすると、ウィンドウ A とウィンドウ B がスクロールします。ウィンドウ B をスクロールしてもウィンドウ A はスクロールしません。
A -> B -> C にリンク	ウィンドウから他のウィンドウに順番にリンクされます。ウィンドウ A をスクロールすると、すべてのウィンドウがスクロールします。ウィンドウ B をスクロールすると、ウィンドウ B とウィンドウ C がスクロールします。

ウィンドウ リンクは循環にできます。つまり、ウィンドウから、そのウィンドウに間接的にリンクされることがあります。すべてのウィンドウをスクロールする際に便利です。スクロールしているウィンドウに関係なく、すべてのウィンドウをスクロールできます。すべてのウィンドウをスクロールする別の方法は、**[Link All Windows]** コマンドを使用して [Link All Windows] モードをオンにするか、**[Two Way]** チェック ボックスをオンにします。

[Line Window To] カレント ウィンドウ以外のすべてのウィンドウがリストされます。ここでリンクするウィンドウを選択します。ウィンドウをリンクしない場合は、<none> を選択します。

[Two Way] オンの場合、両方向にリンクします。リンクされているいずれかのウィンドウをスクロールすると、他のウィンドウもスクロールします。 オフの場合、逆方向にリンクしません。

Load Configuration

新規設定ファイルを現在の設定にロードします。設定ファイル全体、または一部をロードできます。

メモ： すべてのカスタマイズ設定を含むグローバル設定ファイルのバックアップをとることを推奨します。[Load Configuration] コマンドを使用するか、Source Insight 内のカスタマイズ設定を変更すると、設定ファイルは自動的に変更されます。

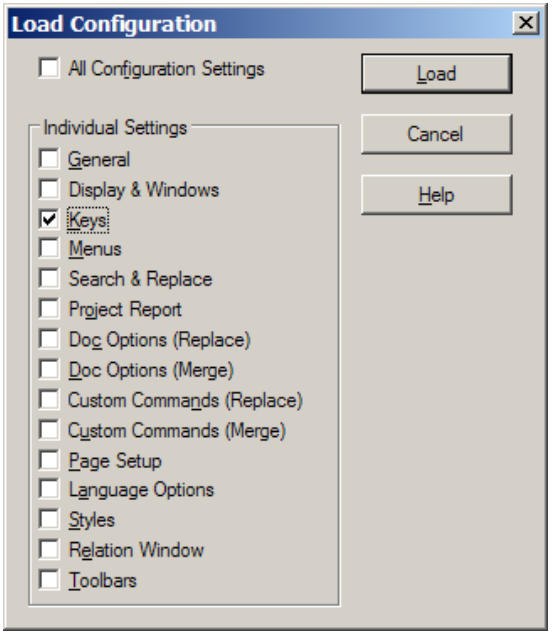
Source Insight を更新する場合、あらかじめバックアップをとることを推奨します。Source Insight の新しいビルドは以前の設定ファイルと互換性がありますが、以前のビルドで新しい設定ファイルを開けない場合があります。以前のビルドに戻す場合は、古い設定ファイルを使用してください。

グローバル設定

通常、Source Insight は Global.cf3 という名前の設定ファイルを Source Insight プログラム ディレクトリに保存して管理します。このファイルを手動でロードまたは保存する必要はありません。処理は自動的に行われます。設定に変更を加えると保存され、プロジェクトを開くときにロードされます。

一部の設定

[Save Configuration] コマンドを使用して一部の設定を保存できます。たとえば、キーの組み合わせのみを保存します。一部の設定を含む設定ファイルをロードすると、ファイルに存在する設定のみロードされます。残りの設定は変更されません。一部の設定をロードおよび保存して、一部の設定を組み合わせることができます。



[All Configuration Settings] オンの場合、設定ファイルのすべての部分をロードします。オフの場合、[Individual Settings] グループで定義された部分のみをロードします。

[Individual Settings] [All Configuration Settings] チェック ボックスをオフにすると、ロードする設定の部分を選択できます。たとえば、画面の色や画面のサイズなどの表示設定のみをロードして、他の設定を変更しないようにできます。

このグループには、設定の各項目用のチェック ボックスが含まれます。ここでロードする項目をチェックします。ドキュメント タイプやカスタム コマンドのような名前付きの項目のリストからなる設定部分を、ロードまたはマージできます。

[Doc Options (Replace)] 現在のドキュメント オプションを設定ファイルのドキュメント オプションに置換するには、このオプションをオンにします。

[Doc Options (Merge)] 現在のドキュメント オプションに設定ファイルのドキュメント オプションをマージするには、このオプションをオンにします。マージとは、ロードしたファイルのドキュメント タイプと同じ名前のドキュメント タイプを置換して、ロードしたファイルのみに存在するドキュメント タイプを現在のドキュメント タイプのリストに追加することです。他の既存のドキュメント タイプは変更されません。

[Custom Commands (Replace)] 現在のカスタム コマンドを設定ファイルのカスタム コマンドに置換するには、このオプションをオンにします。

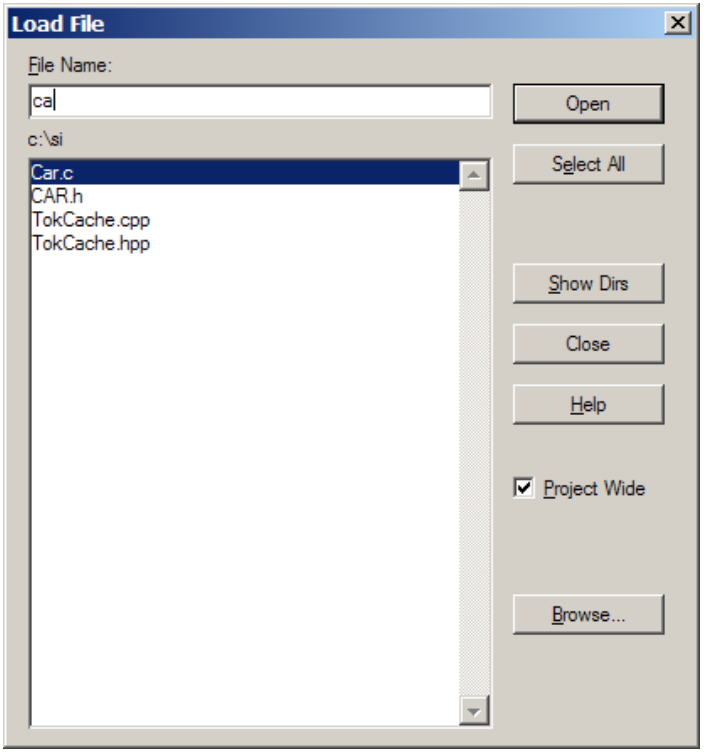
[Custom Commands (Merge)] 現在のカスタム コマンドに設定ファイルのカスタム コマンドをマージするには、このオプションをオンにします。マージとは、ロードしたファイルのカスタム コマンドと同じ名前のカスタム コマンドを置換して、ロードしたファイルのみに存在するカスタム コマンドを現在のカスタム コマンドのリストに追加することです。他の既存のカスタム コマンドは変更されません。

[Load] [File Open] ダイアログ ボックスを表示します。このダイアログ ボックスを使用して、ロードする設定ファイルを選択します。

Load File

編集するファイルを開きます。ディレクトリに関係なく、プロジェクト内のすべてのファイルのリストを含むダイアログ ボックスを開きます。

[Open] コマンドを使用して、または Windows エクスプローラからファイルを Source Insight ウィンドウにドラッグしてファイルを開くこともできます。



[File Name] 開くファイルの名前、または名前の一部を入力します。ワイルドカードを入力して [Open] ボタンをクリックすると、Source Insight はワイルドカード拡張の結果をファイル リストに表示します。ワイルドカードはカレント ディレクトリで拡張されます。

このテキスト ボックスに入力した内容に応じて、ファイル リストは更新されます。

ファイル リスト [Project Wide] オプションがオンの場合、ディレクトリに関係なく、カレント プロジェクトに含まれるすべてのファイルが表示されます。

[Project Wide] オプションがオフの場合、現在の作業ディレクトリに含まれるすべてのファイルが表示されます。現在の作業ディレクトリのパスがリスト ボックスの上に表示されます。現在の作業ディレクトリに表示されるファイルは、[Document Options] ダイアログ ボックスで指定されたワイルドカード文字列により拡張されます。

[Open] ファイル リストで選択しているファイルを開きます。ファイル リストで選択しているファイルがない場合、Source Insight は [File Name] テキスト ボックスのファイルを開きます。[File Name] テキスト ボックスに 1 つ以上のワイルドカードが含まれている場合、Source Insight はワイルドカード拡張の結果をファイル リストに表示します。[Project Wide] オプションがオンの場合、カレント プロジェクトに含まれるファイルのリスト全体で拡張されます。オフの場合、カレント ディレクトリで拡張されます。指定したファイルが存在しない場合、Source Insight はその名前で新規ファイルを作成します。

[Select All] ファイル リストにリストされているすべてのファイルを選択します。

[Show Dirs/Show Files] リスト ボックスに表示する内容 (ファイル名またはサブディレクトリ名) を切り替えます。

[Project Wide] オンの場合、ディレクトリに関係なく、ファイル リストにカレント プロジェクトに追加されているすべてのファイルが表示されます。オフの場合、ファイル リストに現在の作業ディレクトリにあるファイルのみが表示されます。プロジェクトが開かれていない場合、このオプションは常にオフです。

[Browse] Windows の [ファイルを開く] ダイアログ ボックスを表示します。

Load Search String

現在の選択範囲の内容を現在の検索パターンにロードします。検索パターンは、[Search] および [Replace] コマンドの [Find] テキスト ボックスに表示されます。

Lock Context Window

コンテキスト ウィンドウをロックして、ウィンドウが変更されないようにします。コンテキスト ウィンドウがロックされている場合、アクションは追跡されません。

Lock Relation Window

リレーション ウィンドウのロックを切り替えます。ロックされている場合、ウィンドウは自動的に更新されません。その場合でも、[Show Relations] コマンドを使用してリレーション ウィンドウを手動で更新できます。リレーション ウィンドウ ツールバーの [Refresh Relation Window] ボタンをクリックしてウィンドウを更新することもできます。

Lookup References

カレント プロジェクトで選択したシンボルへの参照を検索します。たとえば、「BeginPaint」をクリックし、[Lookup References] コマンドを実行すると、Source Insight は検索結果ウィンドウを開いて、プロジェクトで「BeginPaint」が使用されているすべての場所をリストします。

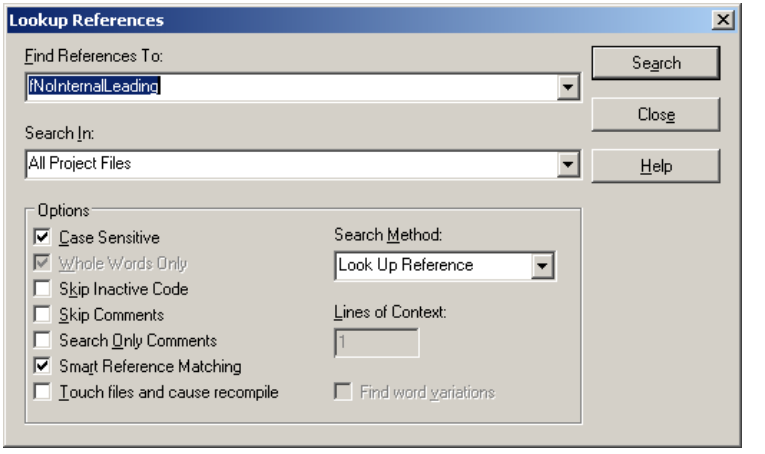
Source Insight はシンボル インデックスを使用して検索を高速化します。

参照は、コメント、および使用されない #ifdef 分岐を含む、すべてのソース コード テキストに含まれます。しかし、これらの場所を検索するかどうか制御できます。

メモ： [Search Project] コマンドは、オプションの状態を除いて [Lookup References] と同じです。301 ページの「Search Project」も参照してください。

[Lookup References] ダイアログ ボックス

このコマンドは、[Search Project] コマンドに似ています。唯一の違いは、それぞれのダイアログ ボックスに独自の継続状態があることです。



[Find References To] 検索するシンボル名を入力します。カーソル位置の単語がこのテキスト ボックスに自動的にロードされます。Source Insight は、正確なシンボル インスタンスを決定するため、カーソル位置のコンテキストを使用します。シンボル ダイアログ ボックスまたはウィンドウから [Lookup References] を起動すると、Source Insight は、このテキスト ボックスとともに正確なシンボル参照を維持します。

通常は、プログラムの識別子の名前を入力しますが、任意の文字列を入力してプロジェクト全体を検索することもできます。単一の単語のみを入力した場合、検索は非常に高速です。

[Search In] ドキュメント タイプのリストが含まれます。このリストを使用して、特定の種類のファイルのみ、またはカレント ファイルのみを検索するように制限できます。プロジェクト ウィンドウが表示されている場合、このリストを使用してプロジェクト ウィンドウで選択するファイルを指定することもできます。

[Search Method] 使用する検索メソッドを選択します。4 種類の検索メソッドが利用できます。

- **Simple String**
- **[Regular Expression]** パターンを正規表現として解釈します。
- **[Keyword Expression]** インターネット検索クエリに似ています。
- **[Lookup Reference]** シンボル参照を検索します。

[Lines of Context] [Keyword Expression] 検索メソッドを選択した場合にのみ適用されます。指定したすべてのキーワードを検索する対象とする行数を指定します。246 ページの「キーワード表現」も参照してください。

[Find word variations] オンの場合、Source Insight は指定したキーワードの最後の形式が異なるものも検索します。たとえば、キーワード「open」を指定した場合、Source Insight は「opens」、「opened」、「opening」も検索します。このオプションは、[Keyword Expression] 検索メソッドを選択した場合にのみ利用できます。

検索オプション

[Case Sensitive] 検索で大文字と小文字を区別するかどうか指定します。

[Whole Words Only] [Lookup References] モードの場合、このオプションは常にオンです。その他の検索メソッドを選択した場合、完全に一致する単語のみを検索します。

[Skip Inactive Code] オンの場合、条件付きコンパイルでアクティブなコードのみを検索します。Source Insight がアクティブな条件を判断できるように、最初に [Preferences: Languages] ダイアログ ボックスで条件を指定する必要があります。条件付きコンパイルは、一部の言語でのみ利用できます。

[Skip Comments] オンの場合、コメントを検索しません。

[Search Only Comments] オンの場合、コメントのみを検索します。[Skip Comments] オプションと同時に指定できません。これらのコメント オプションをオンにした場合、検索は多少遅くなります。

[Smart Reference Matching] Source Insight のスマート参照マッチング機能を有効にします。Source Insight は、見つかった参照が実際に検索しているシンボルを参照しているかどうか判断します。

正確な参照のマッチング処理を行うと、参照検索プロセスは遅くなります。

[Smart Reference Matching] オプションは、検索結果に指定したシンボルへの参照と一致するもののみが含まれることを意味します。たとえば、構造体のメンバを選択してその参照を検索すると、検索結果には、単なる等価な文字列ではなく、その特定の構造体の特定なメンバへの参照のみが含まれます。同じ文字列の検索をシンボル ルックアップで行う必要があるため、このオプションを使用すると処理は遅くなる点に注意してください。

[Touch files and cause recompile] オンの場合、各ファイルの「更新日時」が現在の日時に設定されます。識別子を使用してコンパイル時の依存関係を確認している場合に便利です。このオプションをオンにして、このコマンドを使用して参照を検索するだけで、識別子が参照される場所が「タッチ」され、make プログラムや開発システムは次回プログラムをビルドするときに、それらのファイルを再コンパイルします。

キーワード表現

キーワード表現検索はインターネット検索エンジン クエリに似ています。Source Insight は、プロジェクトで指定された行数内に含まれるキーワードのセットを検索します。[Lines of Context] テキスト ボックスは、互いのキーワードを検索する最大の範囲を示します。

たとえば、「cat food」と入力すると、Source Insight はそれぞれ X 行内の「cat」および「food」を検索します。

キーワード間には、暗黙の論理 AND 演算子があります。つまり、複数のキーワードを入力すると、両方のキーワードが一致しなければなりません。その他のブール演算も含めることができます。以下の表は、利用可能な演算子の一覧です。

表 5.3: キーワード検索演算子

演算子	例	動作
AND または +	cat and dog	両方の用語を含む
OR	cat or dog	いずれかの用語を含む
NOT または - または !	-cat	用語を含まない
=	=Betty	大文字と小文字を区別する
? 「regexp」	? 「^Ich」	正規表現

括弧を使用して表現をグループ化できます。例 :

```
(cat or kitty) and food
(file or buffer) and (save or write) and !error
```

キーワード バリエーション

[Find word variations] オプションがオンの場合、Source Insight は指定したキーワードの最後の形式が異なるものも検索します。たとえば、キーワード「open」を指定した場合、Source Insight は「opens」、「opened」、「opening」も検索します。以下の表現を入力した場合と同じ効果があります。

```
(open or opens or opening)
```

ワード バリエーションは、各キーワードに適用されます。たとえば、以下のキーワードを指定した場合 :

```
save write
```

「save」と「write」が含まれる必要があります。ワード バリエーションが有効な場合、以下の検索と同じになります。

```
(save or saves or saving) and (write or writes or writing)
```

キーワード検索結果

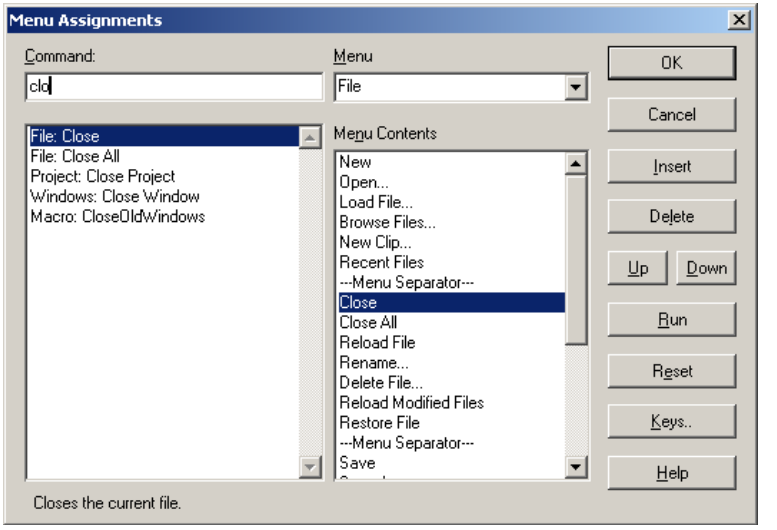
キーワード検索を実行すると、検索結果にキーワードを含む行のブロックがリストされます。 一致の周りのコンテキストが少しわかります。

Make Column Selection

マウスを使用してカラムを選択します。Alt を押しながらマウスをクリックしてドラッグし、長方形の選択範囲を作成します。選択範囲は、切り取りまたはコピーして貼り付けることができます。

Menu Assignments

コマンドにメニューを割り当てます。または、割り当てを削除します。メニュー割り当ては、現在の設定の一部です。



[Command] コマンド名を入力します。入力した内容に応じて該当するコマンドのみがリストされ、コマンドを簡単に見つけることができます。

コマンド リスト ユーザーが定義したマクロとカスタム コマンドを含む、すべての Source Insight コマンドがリストされます。コマンドはカテゴリ別に分類してリストされます。ここでコマンドを選択すると、コマンドが既にメニューに割り当てられている場合、そのメニューが [Menu] リストに表示され、そのメニューの内容が [Menu Contents] にロードされます。

リストに、「--Menu Separator--」と呼ばれる特別なコマンドがあることに注意してください。メニューに区切り線を追加するには、この項目をメニューに挿入します。

[Menu] リスト すべてのメニューのタイトルがリストされます。ここでメニューを選択すると、メニューの内容が [Menu Contents] にロードされます。

[Menu Contents] [Menu] リストで選択しているメニューのメニュー項目がリストされます。削除する項目、または新しい項目を追加する場所を指定するには、ここでメニュー項目を選択します。

[OK] 現在の設定に新しいメニュー割り当てを保存します。

[Cancel] コマンドをキャンセルします。それまでに行った変更は現在の設定に保存されません。

[Insert] 選択しているメニューに選択している項目を挿入します。項目は、[Menu Contents] でハイライト表示されているメニュー項目の前に挿入されます。[Insert] をクリックする前に、コマンド、メニュー、メニュー項目を選択する必要があります。

[Delete] 選択しているメニュー項目を削除します。[Delete] をクリックする前に、[Menu Contents] でメニュー項目を選択する必要があります。

[Up] 選択しているメニュー項目をメニューで 1 つ上に移動します。

[Down] 選択しているメニュー項目をメニューで 1 つ下に移動します。

[Run] 現在の設定に新しいメニュー割り当てを保存して、選択しているコマンドを実行します。

[Reset] メニュー割り当てを初期設定に戻します。Source Insight は確認を表示します。

[Keys...] 現在の設定に新しいメニュー割り当てを保存して、[Key Assignments] コマンドを実行します。222 ページの「Key Assignments」も参照してください。

New

新規の未保存ファイル バッファを作成します。[Save] または [Save As] コマンドを使用するまで、作成したファイルはディスクに保存されません。

[New] コマンドを実行すると、ファイル名を確認するメッセージが表示されます。デフォルトでは、新規ファイルには New0001.ext 形式で名前が設定されます。[New] コマンドを再度使用すると、ファイル名は New0002.ext になります。拡張子には、カレント ファイルと同じ拡張子が使用されます。入力した名前と拡張子により、そのファイルで有効なドキュメント オプションが決定されます。

新規ファイルを最初に保存すると、Source Insight は、ファイル名と、ファイルをカレント プロジェクトに追加するかどうか確認します。

新規ファイルを作成するもう 1 つの方法は、[Open] コマンドを使用して、存在しないファイル名を指定する方法です。Source Insight は、そのファイルを作成するかどうか確認します。Source Insight は、指定した名前で新規の未保存ファイルを作成します。

New Clip

新規クリップ バッファを作成します。クリップの名前を確認するメッセージが表示されます。クリップ名には拡張子を追加しないでください。

New Relation Window

新規リレーション ウィンドウを作成します。リレーション ウィンドウはいくつでも作成できます。各ウィンドウに、独自のオプションを設定できます。

たとえば、関数名を選択して、その関数が呼び出している他の関数を表示するリレーション ウィンドウと、その関数を呼び出している他の関数を表示する別のリレーション ウィンドウを作成できます。

または、現在の選択範囲を追跡するリレーション ウィンドウと、囲まれている関数を追跡する別のリレーション ウィンドウを作成することもできます。

New Project

新規プロジェクトを作成して開きます。

Source Insight は一度に 1 つのプロジェクトのみ開くことができるため、カレント プロジェクトがある場合、Source Insight は続行前にカレント プロジェクトを閉じるかどうか確認します。カレント プロジェクトを閉じない場合、[New Project] コマンドはキャンセルされます。

プロジェクトを保存する場所

[New Project] ダイアログ ボックスでファイルを選択するとき、プロジェクトのデータ ファイルを保存する場所を指定します。ソース ファイルと同じディレクトリにすることも、全く別の場所にすることもできます。

マシンにローカルに格納されているソース ファイル用のプロジェクトを作成する場合、ソース ファイルと同じディレクトリにプロジェクト ファイルを作成することに問題はありません。

共有サーバー上、または書き込み許可がない場所のファイルを参照するプロジェクトを作成する場合、ローカル マシンにプロジェクトを作成してください。後で [Project Settings] ダイアログ ボックスを使用し

てプロジェクト ソース ディレクトリが指すソース ファイルの場所を設定できます。

プロジェクトを作成したディレクトリは、プロジェクトのデフォルト ルート、または「ホーム」ディレクトリになります。[Project Settings] ダイアログ ボックスで、プロジェクト ソース ディレクトリに異なるパスを指定できます。プロジェクト ソース ディレクトリは通常、ソース ファイルを含む最上位のディレクトリのパスです。Source Insight がファイル名を表示するとき、ファイルはプロジェクト ソース ディレクトリへの相対パスで表示されます。大部分のソース ファイルを含むディレクトリをプロジェクト ソース ディレクトリに設定すると、冗長なパス情報が表示されないようになります。

さらに、「Add new files automatically」機能 ([Preferences: General]) は、ファイルがプロジェクト ソース ファイルまたはその派生ディレクトリにある場合にのみ、新規ファイルをプロジェクトに自動的に追加します。

詳細は、319 ページの「Synchronize Files」および 263 ページの「Project Settings」を参照してください。

New Window

新規ウィンドウを開きます。カレント ウィンドウに表示されているファイルが新規ウィンドウにも表示されます。

Next File

[Close] コマンドを実行してから、[Open] コマンドを実行します。カレント ファイルを変更した場合、[Preferences: General] ダイアログ ボックスで [Save Quietly] オプションをオンにした場合を除いて、カレント ファイルを保存するかどうか指定できます。

Next Relation Window View

リレーション ウィンドウの表示モード (アウトライン ビューとグラフ ビュー) を変更します。

Open

プロジェクト ウィンドウをアクティブにして、プロジェクト ウィンドウのテキスト ボックスにフォーカスを設定します。プロジェクト ウィンドウが表示されていない場合、プロジェクト ウィンドウを表示して、開くファイルを選択した後、非表示にします。

[Open] コマンドの動作は [Preferences: Files] ダイアログ ボックスでカスタマイズできます。プロジェクト ウィンドウをアクティブにする代わりに、別のダイアログ ボックスやウィンドウを表示できます。

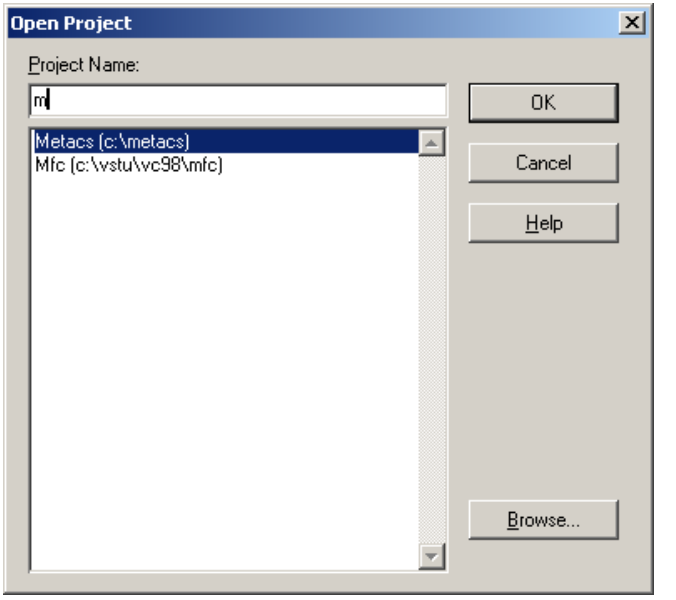
開いているプロジェクトがない場合、このコマンドを実行すると、Windows の [ファイルを開く] ダイアログ ボックスが表示されます。

Source Insight のアプリケーション ウィンドウにファイルをドラッグ アンド ドロップしてファイルを開くこともできます。

Open Project

新規カレント プロジェクトを開きます。Source Insight は一度に 1 つのプロジェクトのみ開くことができるため、開いているカレント プロジェクトがある場合、まず最初のそのプロジェクトを閉じます。

Windows エクスプローラから Source Insight ウィンドウにプロジェクト ファイル (.PR) をドラッグして、または [Open] ダイアログでプロジェクト ファイルを指定してプロジェクトを開くこともできます。



[Project Name] 選択するプロジェクトの名前、または名前の一部を入力します。

プロジェクト リスト マシンで作成されたまたは開かれたすべてのプロジェクトがリストされます。ここで開くプロジェクトを選択します。このリストは、既知のプロジェクトの記録です。ここにリストされていない他のプロジェクトを開くこともできます。その場合、[Browse…] ボタンをクリックしてプロジェクト ファイルを選択します。

[Browse] Windows の [ファイルを開く] ダイアログ ボックスを表示します。プロジェクト ファイル (拡張子 .PR) を選択します。ファイルを

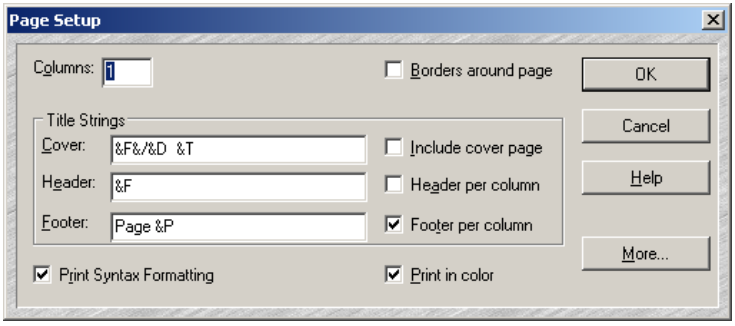
選択して [ファイルを開く] ダイアログ ボックスを閉じると、選択したファイルが [Project Name] テキスト ボックスにロードされます。次に、[OK] をクリックしてファイルを開きます。

Page Down

アクティブなウィンドウを 1 画面下にスクロールします。スクロール前の最後の行がスクロール後の最初の行になります。

Page Setup

[Print] コマンドで印刷するページのテキストのレイアウトを制御します。



[Columns] 用紙の 1 シートに印刷するカラム数を指定します。各カラムは、独自のページ番号を使用します。たとえば、カラムが 2 つある場合、用紙の各シートで 2 ページのソース コードが印刷されます。横向き (用紙の幅が高さよりも長い) モードで印刷する場合に便利です。

[Borders around page] オンの場合、ページのテキストの周りに罫線が印刷されます。

[More...] システムの [ページ設定] ダイアログ ボックスを開きます。このダイアログ ボックスで、用紙のサイズ、余白、向き (縦または横) を設定できます。このダイアログ ボックスから現在のプリンタの設定を変更することもできます。

[Title Strings] グループ

このグループの項目は、カバー ページ、ページの上部 (ヘッダー)、およびページの下部 (フッター) に印刷されるタイトルに影響します。

[Cover]、[Header]、[Footer] これらのテキスト ボックスで、各ページのカバー ページ、ヘッダーおよびフッターに使用されるフォーマット コードを指定します。文字列のフォーマットの詳細は、254 ページの「ヘッダーとフッター コード」を参照してください。

[Include cover page] オンの場合、カバー ページがすべての他のページの前に印刷されます。[Cover] テキスト ボックスに名前を設定すれば、印刷した人を簡単に識別できます。オフの場合、カバー ページは印刷されません。

[Header per column] オンの場合、ヘッダーが各カラムの上に印刷されます。カラム数が 2 以上に設定されている場合にのみ有効です。オフの場合、単一のヘッダーがページの上部に印刷されます。

[Footer per column] オンの場合、フッターが各カラムの下に印刷されます。カラム数が 2 以上に設定されている場合にのみ有効です。オフの場合、単一のフッターがページの下部に印刷されます。

フォーマット オプション

フォーマット オプションは、構文フォーマットをプリンタに出力する方法を制御します。これらのオプションは、[Preferences: Syntax Formatting] ダイアログ ボックスで制御する画面の設定には依存しません。

[Print Syntax Formatting] オンの場合、画面に表示されるのと同じように、すべての構文フォーマットが印刷されます。

[Print in color] オンの場合、カラーで印刷します。プリンタがカラー プリンタでない場合、灰色の濃淡で印刷されます。オフの場合、すべてのテキストは、画面の色に関係なく、白黒で印刷されます。灰色に近いカラーは灰色で表示されます。

ヘッダーとフッター コード

ヘッダーは、ページの上部に印刷されるタイトルです。フッターは、ページの下部に印刷されるタイトルです。

[Page Setup] コマンドを使用して、印刷ページのヘッダーとフッターをカスタマイズできます。カスタマイズするには、[Header] および [Footer] テキスト ボックスで以下のコードを使用します。

表 5.4: ヘッダーとフッター コード

コード	結果
&L	後に続く文字を左揃えにします。デフォルトの設定です。
&C	後に続く文字を中央揃えにします。
&R	後に続く文字を右揃えにします。
&/	次の行に移動します。
&B	後に続く文字を太字で印刷します。
&I	後に続く文字を斜体で印刷します。
&U	後に続く文字を下線付きで印刷します。
&D	現在の日付を印刷します。

表 5.4: ヘッダーとフッター コード

コード	結果
&T	現在の時間を印刷します。
&F	ファイル名を印刷します。
&P	ページ番号を印刷します。
&N	ドキュメントのページの総数を印刷します。たとえば、ドキュメントが 12 ページの場合、Page &P of &N と指定すると、Page 1 of 12、Page 2 of 12、Page 3 of 12、のように印刷されます。
&&	単一アンパサンド (&) を命令としてではなくリテラル文字として印刷します。

たとえば、左の余白に「Confidential」、中央にページ番号、右の余白に現在の日付を印刷するには、以下のように入力します。

```
&LConfidential&C&P&R&D
```

Page Up

アクティブなウィンドウを 1 画面上にスクロールします。スクロール前の最初の行がスクロール後の最後の行になります。

Paren Left

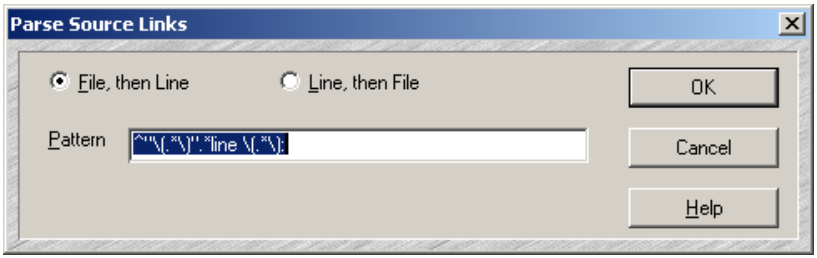
挿入ポイントを次の括弧の左に移動します。

Paren Right

挿入ポイントを次の括弧の右に移動します。

Parse Source Links

指定したパターンでカレント ファイルを検索します。検索がマッチすると、その場所にソース リンクを作成します。ソース リンクは、カレント ファイルのソース行とパターンを使用して解析したファイルおよび行番号をリンクします。



[File, then Line] および [Line, then File] パターン表現の最初のグループがファイル名で 2 番目のグループが行番号の場合、[File, then Line] を選択します。この設定では、2 番目のグループ (たとえば、行番号) はオプションです。

パターン表現の最初のグループが行番号で 2 番目のグループがファイル名の場合、[Line, then File] を選択します。

[Pattern] コマンド出力のファイル名と行番号の検索に使用される正規表現です。このテキスト ボックスには、ファイル名と行番号の「グループ」を含む有効な正規表現が含まれている必要があります。このテキスト ボックスの内容は、カレント ワークスペースに保存されます。

[Parse Source Links] コマンドは、コンパイラ出力およびエラー メッセージを含むログ ファイルがある場合に便利です。ログ ファイルを開いて、[Parse Source Links] コマンドを実行するだけです。リンクがエラー メッセージを含むログ ファイルの各行に設定されます。118 ページの「検索とソース リンク」.

複数の解析パターンの管理

2 つ以上の解析パターンを使用する場合、[Custom Commands] を使用して、出力からソース リンクを解析する各タイプ用のカスタム コマンドを定義できます。各カスタム コマンドには独自の解析パターンを設定できるので、この方法で多くの解析パターンを保存できます。

たとえば、

1. [Options] メニューから、**[Custom Commands]** を選択します。
2. **[Add]** ボタンをクリックして、[Name] テキスト ボックスに「Parse Type 1」と入力します。
3. **[Run]** テキスト ボックスに、「`command /c type %f`」と入力します。このコマンド ラインは、カレント ソース ファイルの内容を出力します。
4. **[Parse Link in Output]** チェック ボックスをオンにします。
5. **[Pattern]** テキスト ボックスに解析パターンを入力します。

このコマンドを実行すると、Source Insight によってカレント ファイルがキャプチャされ、カスタム コマンドを使用して格納された解析パターンを使用してソース リンクが解析されます。

Paste

クリップボードの内容を現在の選択範囲にコピーします。現在の選択範囲が拡張されている場合、貼り付けの前に削除されます。

Paste From Clip

クリップ バッファの内容を現在の選択範囲にコピーします。このコマンドを使用すると、クリップ ウィンドウが表示されます。

クリップ ウィンドウでクリップをダブルクリックして、このコマンドを実行することもできます。

Paste Line

挿入ポイントをカレント行の行頭に移動して、[Paste] コマンドを実行します。[Cut Line]、[Copy Line]、および [Paste Line] と一緒に使用することで、ファイルの行を素早く移動できます。

Paste Symbol

(シンボル ウィンドウの右クリック メニュー)

クリップボードの内容を、シンボル ウィンドウで選択しているシンボルの直前に貼り付けます。

Play Recording

記録したコマンドを再生します。コマンドは、[Start Recording] コマンドを使用して記録します。[Play Recording] コマンドを使用したときにレコーダがオンだった場合、レコーダは自動的にオフになります。

Preferences

各種ユーザー オプションを指定します。ダイアログ ボックスには、[Display]、[Files]、および [Syntax Formatting] のような各種オプション用のタブがあります。

詳細は、以下を参照してください。

209 ページの「General Options」も参照してください。

327 ページの「Typing Options」も参照してください。

203 ページの「File Options」も参照してください。

228 ページの「Language Options」も参照してください。

315 ページの「Symbol Lookup Options」も参照してください。

187 ページの「Display Options」も参照してください。

169 ページの「Color Options」も参照してください。

320 ページの「Syntax Decorations」も参照してください。

322 ページの「Syntax Formatting」も参照してください。

301 ページの「Searching Options」も参照してください。

282 ページの「Remote Options」も参照してください。

Print

カレント ファイルを印刷します。Windows の [印刷] ダイアログ ボックスが表示されます。

ファイルの印刷に使用されるフォントを制御するには、[Document Options] コマンドを使用します。[Document Options] コマンドでは、印刷に使用するフォントと画面の表示に使用するフォントを指定できます。または、[Emulate screen fonts when printing] オプションをオンにして、印刷時に画面の表示に使用するフォントをエミュレートすることもできます。TrueType フォントを選択した場合、フォントをエミュレートしても問題ありません。「MS Sans Serif」や「Courier」のようなフォントを選択した場合、フォントのエミュレートが正しく行われないことがあります。プリンタ ドライバによっては、似たフォントに置換されることがあります。

カラー印刷

ファイルをカラーで印刷するには、[Page Setup] ダイアログ ボックスで [Print in color] オプションをオンにする必要があります。

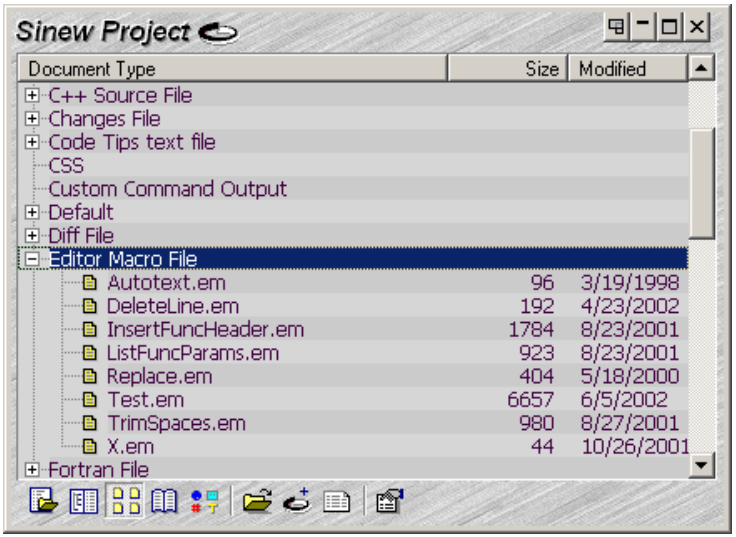
Print Relation Window

リレーション ウィンドウのグラフを印刷します。このコマンドは、リレーション ウィンドウを右クリックしてアクセスすることもできます。

Source Insight は、複数のページ グラフを印刷できます。各ページの下部には、ページの座標が印刷されます。たとえば、グラフが 4 x 3 ページに分かれている場合、最初のページの下部に、「Cell 1,1 of 4 x 3 square」のように印刷されます。

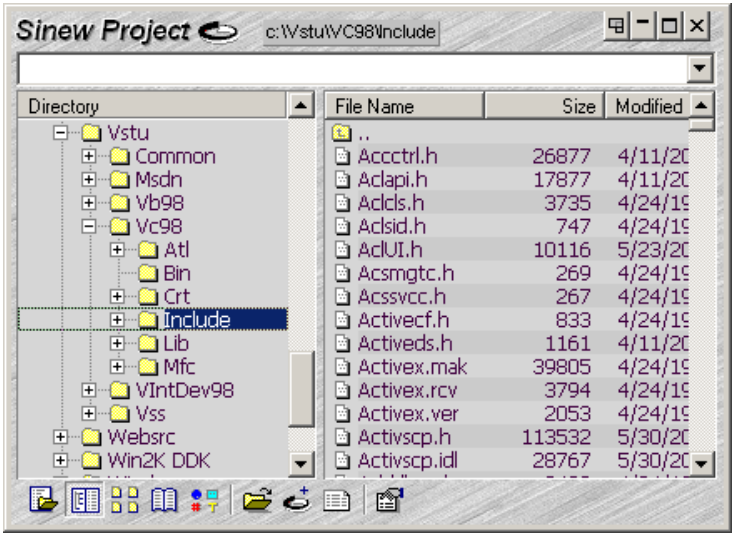
Project Document Types

プロジェクト ウィンドウに、プロジェクト ファイルのリストをドキュメント タイプ別に分類して表示します。



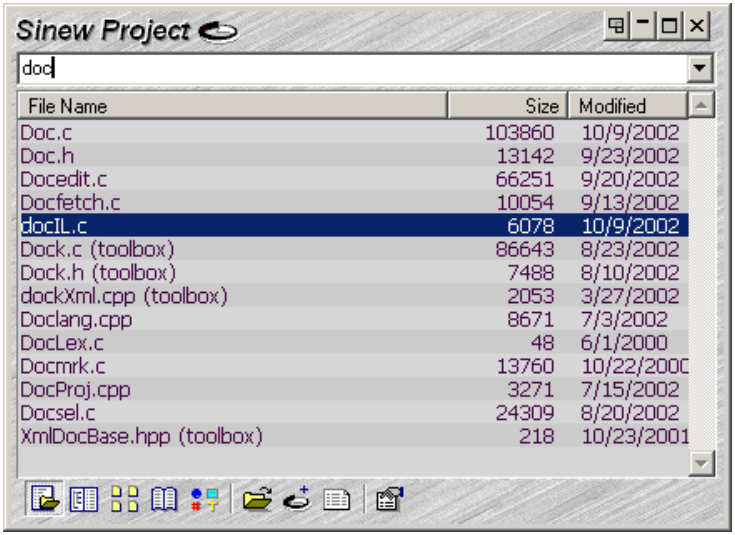
Project File Browser

プロジェクト ウィンドウに、ファイル ブラウザ ビューを表示します。このビューで、ディスクに含まれるファイルを参照できます。



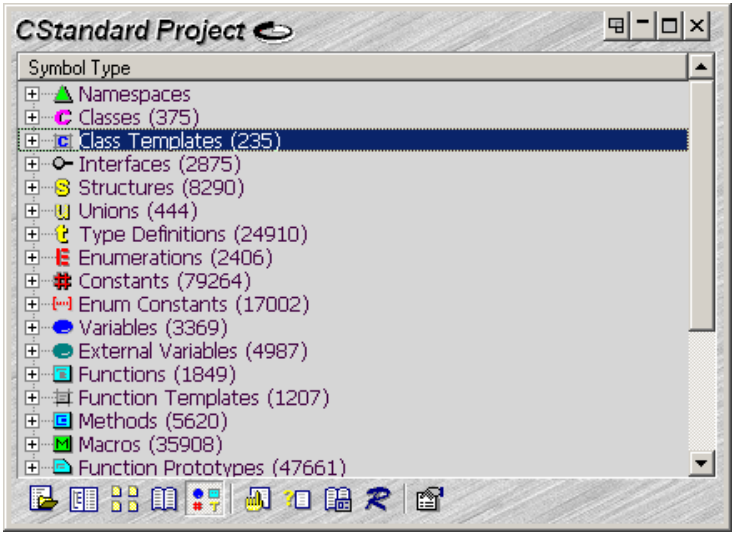
Project File List

プロジェクト ウィンドウに、すべてのプロジェクト ファイルを表示します。ディレクトリに関係なく、カレント プロジェクトのすべてのファイルがリストされます。



Project Symbol Classes

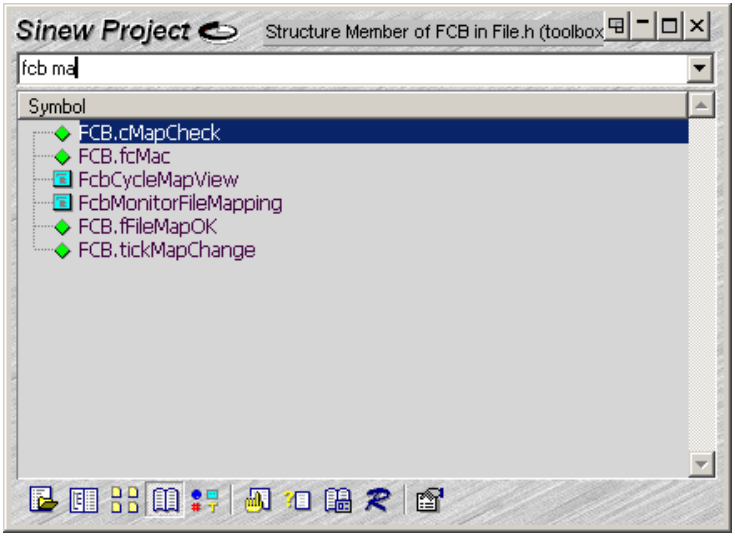
プロジェクト ウィンドウに、カテゴリ別にプロジェクト シンボルを表示します。



Project Symbol List

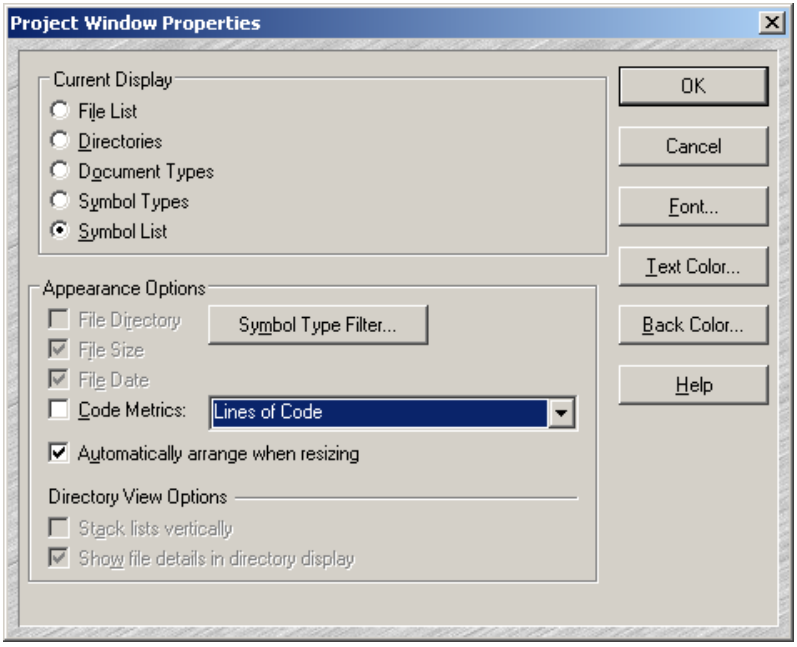
プロジェクト ウィンドウに、すべてのプロジェクト シンボルを表示します。このビューは、モーダル ダイアログ ボックスが表示される [Browse Project Symbols] コマンド (F7) よりも便利です。プロジェクト

ウィンドウはモードレス ダイアログ ボックスなので、コンテキスト ウィンドウやリレーション ウィンドウと同時に使用できます。



Project Window Properties

プロジェクト ウィンドウのプロパティを編集します。



[Current Display] プロジェクト ウィンドウの表示モードを選択します。プロジェクト ウィンドウで Ctrl+Tab を使用して表示モードを選択することもできます。また、ツールバーのボタンを使用して表示モードを選択することもできます。

[Appearance Options]

プロジェクト ウィンドウに表示する内容を制御します。

[File Size]、[File Date] ファイルを表示するモードで、これらのオプションをオンにすると、各ファイルのサイズ、日付が別々のカラムに表示されます。

[Code Metrics:] 選択したコード メトリック カラムを表示します。右のドロップダウン リストから、カラムに表示するコード メトリック値を選択します。

[Symbol Type Filter…] プロジェクト ウィンドウに表示されるシンボル リストに表示するシンボルのタイプを選択します。

[Automatically arrange when resizing] オンの場合、ウィンドウのサイズを変更すると、プロジェクト ウィンドウはカラムのサイズを自動的に変更します。オフの場合、カラムの幅を変更した場合を除いて、カラムの幅は固定です。

[Directory View Options]

[Stack list vertically] オンの場合、ファイル ブラウザ ビューで、ディレクトリ リストをファイル リストの上に表示します。オフの場合、リストを横に並べて表示します。[Automatically arrange when resizing] オプションがオンの場合、プロジェクト ウィンドウの縦横比に応じて向きが自動的に切り替えられます。

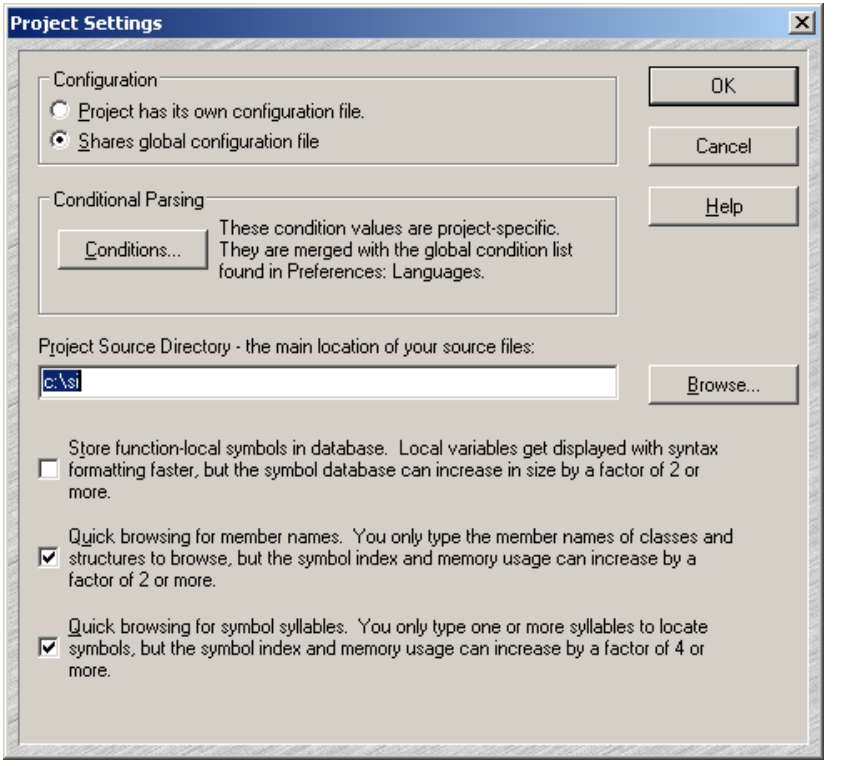
[Show file details in directory display] File Directory ビューでのみ使用されます。オンの場合、ファイルのサイズのような詳細な情報がファイル リストに表示されます。

[Font…]、[Text Color…]、[Back Color…] それぞれ、フォント、テキストの色、背景色を選択します。

Project Settings

[Project Settings] コマンドを使用して、カレント プロジェクトを管理するさまざまなオプションを設定できます。開いているプロジェクトがない場合、[Project Settings] コマンドはデフォルトのオプションを設定します。次回以降作成される新規プロジェクトでは、このオプションが使用されます。

設定は、プロジェクト ディレクトリのプロジェクト データ ファイルに保存されます。これらの設定は、設定ファイルに依存しません。



[Configuration] オプション

プロジェクトを開くときに使用する設定を指定します。設定には、キー バインドのような、ユーザーがカスタマイズした項目のほとんどが保存されます。

[Project has its own configuration file] プロジェクトは、プロジェクト ディレクトリに保存された独自の設定ファイルを使用します。プロジェクト設定ファイルの名前は、<プロジェクト名 >.CF3 です。

[Shares global configuration file] プロジェクトは、他のプロジェクトとグローバル設定ファイルを共有します。グローバル設定ファイルは、Source Insight プログラム ディレクトリに保存されます。名前は Global.cf3 です。

[Conditional Parsing]

条件付きコンパイル値の設定を制御します。

[Conditions...] カレント プロジェクトのみに固有な定義済みの条件を編集します。これらの条件はカレント プロジェクトが開かれている場

合にのみ (プロジェクトに属しているファイルに対してのみ) 有効です。[Edit Condition] コマンドを使用して条件を編集することもできます。

ソース ディレクトリ

[Project Source Directory] ソース ファイルのメインの場所のパスを指定します。このディレクトリは、プロジェクトの「ホーム」ディレクトリです。Source Insight がファイル名を表示するとき、ファイルはこのディレクトリへの相対パスで表示されます。このテキスト ボックスを空白にした場合、Source Insight はメイン プロジェクト ファイル (拡張子 .PR) が作成されたプロジェクト データ ディレクトリを使用します。

プロジェクト ソース ディレクトリは、ソース ファイルのホーム ディレクトリです。

プロジェクト データ ファイルをソース ファイルと別のディレクトリに保存する場合は、異なるプロジェクト ソース ディレクトリを指定します。プロジェクト データ ファイルは .PR ファイルで指定された場所に格納されますが、ソース ファイルは別の場所に保存できます。たとえば、ワークステーション上にローカルに格納されるプロジェクトを作成して、プロジェクトにリモート ネットワーク ドライブからファイルを追加できます。ソース ファイルへのネットワーク ドライブ パスをプロジェクト ソース ディレクトリとして指定した場合、ファイル名に追加のパス情報は含まれません。

この機能は、プロジェクト データ ファイルをソース ファイルと同じディレクトリに作成できない場合に便利です。プロジェクト マネージャによっては、ソース コードの共有ポイントへの読み取り専用アクセスを許可し、Source Insight ファイルの保存を許可しないことがあります。

プロジェクト ソース ディレクトリ設定は、プロジェクトを作成した後、いつでも [Project Settings] ダイアログ ボックスで変更できます。

プロジェクト ソース ディレクトリ パスは相対フォーマットで格納されます。パスは .PR ファイルを含むディレクトリへの相対パスです。相対パスを使用することにより、あるマシン上で作成されたプロジェクトを、論理ドライブを混同することなく、別のマシンからリモートに開くことができます。プロジェクト ディレクトリ ツリー全体を新しい場所にコピーすることもできます。

一部のカスタム コマンド文字列メタ文字の置換も、このパス設定によって影響を受けます。%j (プロジェクト ソース ディレクトリ) および %v (プロジェクト ソース ディレクトリのボリューム文字) は、.PR ファイルを含む場所ではなく、このパス値を参照します。

シンボル データベース オプション

これらのオプションは、プロジェクトのシンボル データベースに格納される内容に影響を与えます。プロジェクトにファイルを追加する前に、これらのオプションを選択してください。プロジェクトを構築し

た後にこれらのオプションを変更した場合、プロジェクトを再構築する必要があります。Source Insight を使用して再構築します。

[Store function-local symbols in the database] 関数の本体内部で宣言されたローカル変数をシンボル データベースに格納します。データベースのサイズは大きくなりますが、ファイルを開くと変数の構文フォーマットが直ちに表示されます。

[Quick browsing for member names] オンの場合、構造体またはクラス名を入力するだけで名前の部分一致が実行されます。しかし、シンボル インデックスのサイズとメモリの使用量は 2 倍以上増加します。オブジェクト指向言語を主に使用している場合、このオプションを推奨します。クラス名を入力することなく、メンバ関数と変数を検索できます。

[Quick browsing for symbol syllables] オンの場合、シンボルの 1 つまたは 2 つのシラブルを入力するだけで名前の部分一致が実行されます。しかし、シンボル インデックスのサイズとメモリの使用量は 4 倍以上増加します。シラブル インデックスを使用すると、シンボル名の最初の文字がわからない場合でも、シラブル マッチングを使用してシンボルを素早く検索できます。

このオプションは、プロジェクト シンボル パス経由でのみ参照する外部コモン プロジェクトには推奨しません。この場合、シラブル インデックスを使用する利点はなく、使用するスペースが増加するだけです。

プロジェクトが非常に大きく、システムの RAM が少ない (ディスクのスワップ領域のサイズが小さい) 場合、このオプションは推奨しません。

インデックスのパフォーマンス

上記の条件に当てはまる場合を除いて、シラブル マッチングを有効にすることを推奨しました。パフォーマンスに影響を受けていると感じ

る場合、インデックスが大きすぎるために、以下のような現象が起きています。

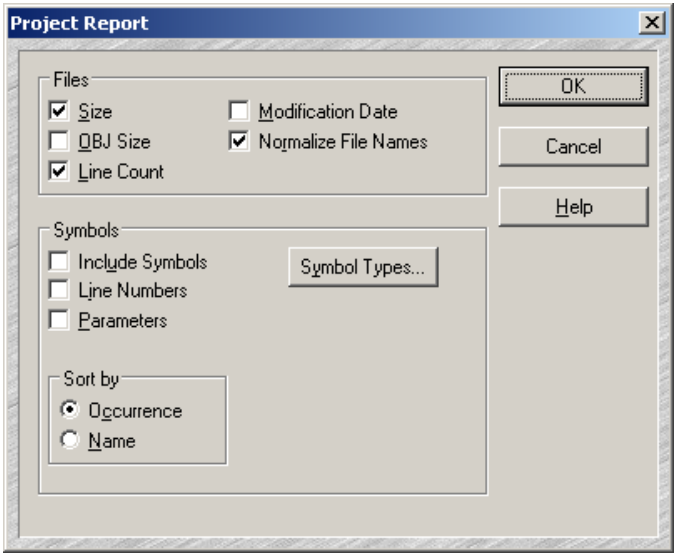
- 大規模なプロジェクトのビルドまたはリビルド中にディスク スラッシングが発生します。
- プロジェクトを開くまたは閉じるのに時間がかかります。
- 個々のファイルの同期処理が遅くなります。
- [Browse Project Symbols] ダイアログ ボックス (F7) の表示に時間がかかります。最初に使用するときの多少の遅延は正常です。
- プロジェクトに数百万のインデックス エントリが含まれます。この制限は、RAM の量に依存します。
- ハード ドライブのアクセス ランプが点灯したままになります。またはディスクがフラッシュされている間システムが休止します。

大規模なプロジェクト (200,000 シンボル以上) でシステムが遅い場合、[Quick browsing for symbol syllables] をオフにしてください。[Project] > [Rebuild Project] を選択してダイアログ ボックスの下部に表示される情報を確認することでデータベースのサイズが予測できます。情報を確認したら、[Cancel] ボタンをクリックしてダイアログ ボックスを閉じてください。インデックス エントリの数が百万を超えると、システムは遅くなります。ただし、メモリ 256MB の Pentium III クラスのマシンでも、このサイズのプロジェクトを問題なく処理できます。

マシンにメモリを追加すると、パフォーマンスが向上します。

Project Report

カレント プロジェクトのファイルとシンボルのリストを含む出力レポート ファイル (<プロジェクト >.RPT) を生成します。



[Files] 対応する情報を含めるには、これらのチェック ボックスをオンにします。

[Include Symbols] オンの場合、シンボルが各ファイルの下にリストされます。オフの場合、プロジェクト レポートにシンボル情報はリストされません。

[Line Numbers] オンの場合、シンボルが定義された行番号がシンボル名順にリストされます。

[Sort by] [Occurrence] が選択された場合、シンボルは行番号順にリストされます。[Name] が選択された場合、シンボルはシンボル名順にリストされます。

[Symbol Types…] 含めるシンボルのタイプを指定します。

Project Window コマンド

プロジェクト ウィンドウは、カレント プロジェクトのファイルのリストを表示するフローティング / 結合可能なウィンドウです。プロジェクト ウィンドウのエントリをダブル N リックすると、ファイルが開きます。エクスプローラまたはファイル マネージャからファイルをプロジェクト ウィンドウにドラッグすると、ファイルがプロジェクトに追

加されます。プロジェクト ウィンドウの下部には、プロジェクトに関連する他のコマンド用のツールバーがあります。

プロジェクト ウィンドウは、[Project Window] コマンドを実行して、オンとオフを切り替えることができます。プロジェクト ウィンドウにフォーカスを移すには、[Activate Project Window] コマンドを実行します。このコマンドは、非表示のプロジェクト ウィンドウがあると、ウィンドウを表示します。

プロジェクトが開かれている場合、[Open] コマンドを実行するとプロジェクト ウィンドウが表示されます。

コンテキスト ウィンドウが開かれている場合、プロジェクト ウィンドウで選択されているファイルの内容が表示されます。

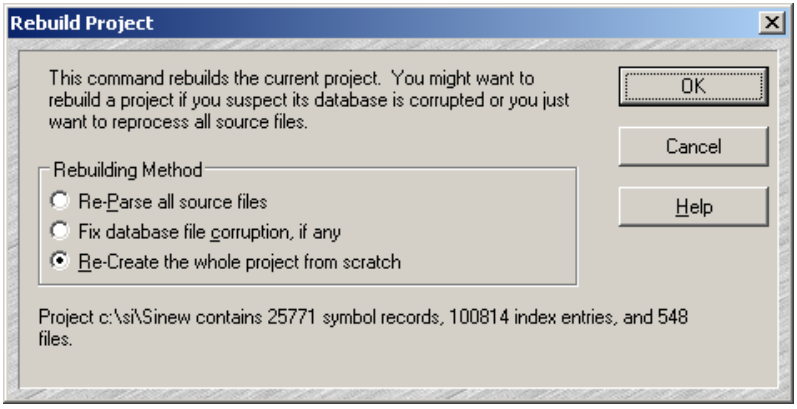
Rebuild Project

プロジェクト データベース ファイルをリビルドします。

大規模な変更の後にすべてのファイルを再解析する場合、またはプロジェクト データが正しくないと思われる場合は、プロジェクトをリビルドします。

プロジェクトを閉じないで Source Insight が異常終了した場合、プロジェクトは壊れることがあります。

[Rebuild Project] ダイアログ ボックスは、プロジェクトに関するいくつかの統計もリストします。シンボル データベース レコード、シンボル インデックス エントリ、およびファイルの数が表示されます。この情報は、[Project Report] コマンドを実行した場合にも出力されます。



プロジェクトをリビルドするメソッドは 3 つあります。最後のメソッドを推奨します。

[Re-Parse all source files] プロジェクトのすべてのファイルを単純にスキャンして再解析し、シンボル データベースを更新します。最も遅い

メソッドです。しかし、操作をキャンセルすると、シンボル データベースは多少日付が古くなりますが、完全なままです。キャンセルしても、正常に編集を続行できます。再解析はバックグラウンドで続行されます。シンボル データベースは、少なくともリビルド プロセスを開始したときと同程度に最新になります。

[Fix database file corruption, if any] データベースをスキャンし、データベースが正当で自己矛盾がない状態であることを保証します。しかし、最近プロジェクトにファイルを追加した場合やプロジェクトからファイルを削除した場合、リビルドが完了した後、一部のファイルまたはシンボルが見つからないことがあります。高速なメソッドです。プロジェクトが壊れていると思われる場合、このコマンドを使用してください。通常、Source Insight はプロジェクトを開くときにプロジェクトが適切に閉じられたかどうか検出できます。プロジェクトが壊れていることを検出すると、Source Insight はプロジェクトをリビルドするかどうか確認します。Source Insight が、壊れたプロジェクトを故意に開くことはありません。

[Re-Create the whole project from scratch] (推奨) プロジェクトをリビルドする最も安全なメソッドです。このメソッドを使用することを推奨します。プロジェクトに関するすべてのシンボル情報を削除し、すべてのソース ファイルを再スキャンして、シンボル データベースを最初から再生成します。[Fix database file corruption, if any] メソッドより時間がかかります。途中でキャンセルすると、シンボル データベースの一部のみ再生成されます。この場合、スキャンしていないシンボルの情報はわかりません。バックグラウンド同期により解析は完了します。または、[Project] > [Synchronize Files] コマンドを使用してプロセスを完了します。

Record New Default Properties

(シンボル ウィンドウの右クリック メニュー)

ウィンドウの現在の設定を新規ウィンドウのデフォルト設定にします。

シンボル ウィンドウの場合、ウィンドウの幅、シンボルのソート、シンボルのタイプのフィルタリングが記録され、以降に作成される新規ウィンドウのデフォルト値として使用されます。

Redo

[Undo] コマンドの効果を逆にします。ファイルを編集すると、Source Insight は各ファイルに加えた変更のリストを作成します。[Undo] コマンドはリストを戻し、[Redo] コマンドはリストを進めます。

[Redo All] コマンドを使用してすべての [Undo] アクションを逆にすることもできます。

Source Insight は、開いている各ファイルについて、[Undo]/[Redo] の履歴を保存します。

Redo All

すべての [Undo] アクションの影響を逆にします。ファイルは、最後に変更された状態になります。[Redo] できなくなるまで [Redo] コマンドを繰り返し実行することと等価です。

Redraw Screen

Source Insight の画面全体を再表示します。

Reform Paragraph

段落内のすべての行が、現在のドキュメント タイプの余白の幅に収まる範囲で、できるだけ多く表示されるように、選択しているテキストの段落を再フォーマットします。

選択範囲が指定されていない場合、挿入ポイントを含む一連の段落が再フォーマットされます。複数の段落が選択されている場合、選択範囲のすべての段落が再フォーマットされます。

段落の最初の行がインデントされている場合、各段落の後の以降も同じようにインデントされます。

テキストの段落は、黒線で囲まれた行のグループであると見なされます。

余白の幅は、カレント ファイルのドキュメント タイプの [Document Options] コマンドで指定できます。

たとえば、[Reform Paragraph] を実行する前の段落が以下の場合 :

```
The quick brown
fox jumped
over the
lazy dog,
and other
mysterious sentences.
```

[Reform Paragraph] を実行した後、行が余白に収まる範囲で、できるだけ多く表示されるように、次のように再フォーマットされます。

```
The quick brown fox jumped over the lazy dog, and other
mysterious sentences.
```

Refresh Relation Window

リレーション ウィンドウの内容を再計算して更新します。

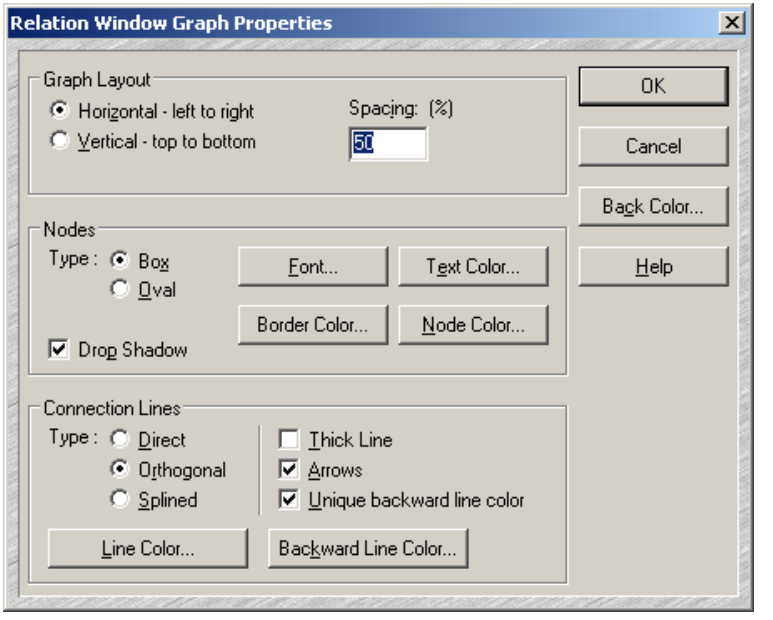
通常、リレーション ウィンドウは選択範囲に基づいて自動的に更新されます。

リレーション ウィンドウを自動的に更新しない場合、[Lock Relation Window] コマンドを使用してロックするか、[Relation Window Properties] コマンドを使用して「Automatic Symbol Tracking」を「None」に設定します。その場合でも、[Refresh Relation Window] コマ

ンドを使用してリレーション ウィンドウの内容を手動で計算できます。

Relation Graph Properties

プロジェクト ウィンドウのグラフ プロパティを表示します。グラフ プロパティは、リレーション ウィンドウが (アウトライン モードではなく) グラフ モードの場合に適用されます。



[Back Color…] グラフ ウィンドウの背景色を選択します。

[Graph Layout] グラフのレイアウト モードを選択します。上から下、または左から右のいずれかを選択できます。

[Spacing] ノード間グラフの間隔の割合をパーセントで入力します。値が大きくなると、ノードの間隔は広くなります。

[Nodes] オプション

グラフのノード要素の概観を制御します。

[Type] ノードの影を選択します。

[Font…] 各ノードの内部で使用するフォントを選択します。

[Border Color…]、[Text Color…]、[Node Color…] ノードの罫線、テキストおよびノードに使用する色を選択します。

[Drop Shadow] 各ノードにドロップ シャドウを挿入します。

[Connection Lines] オプション

ノードを接続する線を制御します。

[Type] ノードを接続する線のスタイルを選択します。

- **[Direct]** ノード間を直線で接続します。
- **[Orthogona]l** ノード間を直交線で接続します。
- **[Splined]** ノード間を曲線で接続します。

[Line Color…] 接続線に使用する色を選択します。

[Thick Line] 接続線を太くします。

[Arrows] 接続線を矢印付きにします。[Spacing] の値が小さい場合、内部ノード空間が不足するため矢印は省略されます。

[Unique backward line color] 逆の向きの線を別の色にします。色を選択するには、[Backward Line Color] ボタンを使用します。

[Backward Line Color…] 逆の向きの線に使用する色を選択します。

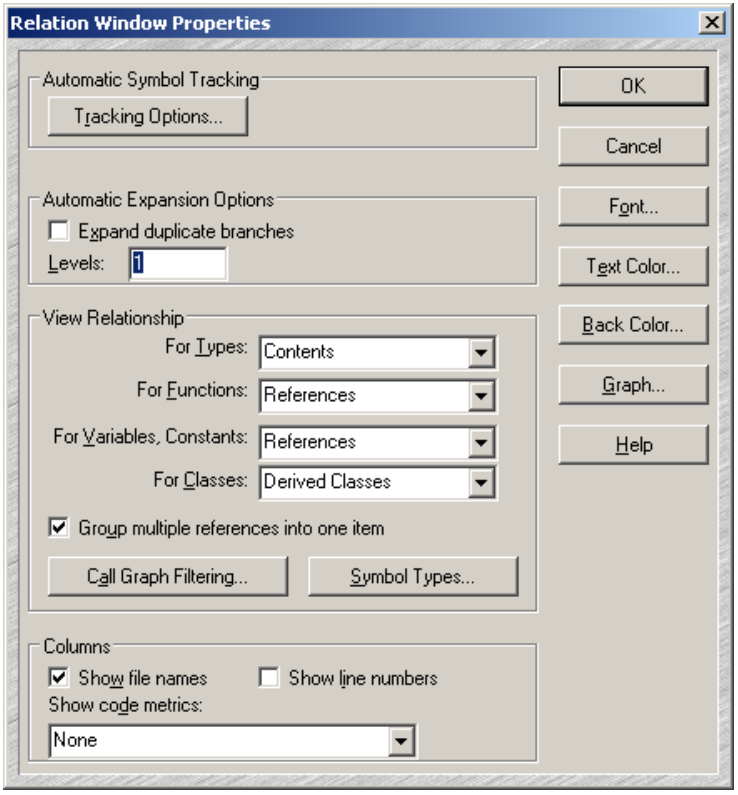
Relation Window

リレーション ウィンドウの表示を切り替えます。すべてのリレーション ウィンドウの表示が切り替わります。

Relation Window Properties

[Relation Window Properties] コマンドは、リレーション ウィンドウ ツールバーまたは、ショートカット メニューからアクセスできます。このダイアログ ボックスから、表示するリレーションシップ、およびウィンドウの表示方法を制御できます。

[Relation Window Properties] ダイアログ ボックス



[Font…]、[Text Color…]、[Back Color…] それぞれ、リレーション ウィンドウのフォント、テキストの色、背景色を選択します。アウトライン ビューにのみ適用されます。

[Graph…] [Relation Graph Properties] ダイアログ ボックスを開きます。このダイアログ ボックスから、リレーション ウィンドウのグラフ ビューのオプションを調整できます。

[Automatic Symbol Tracking]

リレーション ウィンドウの追跡オプションを制御するには、**[Tracking Options]** ボタンをクリックします。カーソル位置のシンボル、または選択範囲の囲まれた関数やデータ構造を追跡できます。

[Automatic Expansion Options]

リレーション ウィンドウが自動的に展開するレベルを指定します。リレーション ウィンドウでノードを右クリックして **[Expand Special]** を選択し、個別の設定よりも優先させることができます。

[Levels] ルート ノード以下で展開するレベルの数です。

[Expand duplicate branches] 既に展開されている場合でも、重複する子の分岐を展開します。オフの場合、重複する子は折り畳まれた状態で挿入されます。

[View Relationship] グループ

リレーション ウィンドウが異なるタイプのシンボルを追跡するときに使用するリレーションシップ展開ルールを指定します。

[View Relationship] 表示されるリレーションシップはシンボルのタイプによって決まります。異なるシンボル タイプに表示するリレーションシップを指定できます。たとえば、「Calls」では関数のリレーションシップ、「Inheritance」ではクラスのリレーションシップを表示するように指定できます。つまり、関数を選択すると、リレーション ウィンドウはコール ツリーを表示し、クラス名を選択すると、リレーション ウィンドウはクラス継承階層を表示します。276 ページの「リレーションシップのルール」も参照してください。

[Group multiple references into one item] オンの場合、リレーション ウィンドウでリレーションシップの「参照」タイプを表示するときに、同じオブジェクト内部からの重複する参照を抑制します。たとえば、関数 B が関数 A の内部から 3 回呼び出されている場合、関数 A からの最初の参照のみリストされます。オフの場合、3 つの参照がすべて、3 つのノードとしてリストされます。

オフの場合、より多くの参照情報が提供されます。グラフ ビューでは、複数の参照が単一グラフ ノードの内部に表示され、オブジェクト間で参照がどのように集中しているか簡単に確認できます。

[Call Graph Filtering…] [Call Graph Filter] ダイアログ ボックスを表示します。このダイアログ ボックスで、コール ツリー計算に関係するシンボルを制御できます。277 ページの「[Call Graph Filter]」も参照してください。

[Symbol Types…] [Symbol Type Filter] ダイアログ ボックスを表示します。リレーション ウィンドウから特定のタイプのシンボルをフィルタして出力します。278 ページの「[Symbol Type Filter]」も参照してください。

[Columns] グループ

アウトライン リスト ビューに表示するカラムを指定します。リレーション ウィンドウは、任意のカラム ヘッダーをクリックしてソートできます。

[Show file names] ファイル名カラムを表示します。

[Show line numbers] 行番号カラムを表示します。ほとんどの場合、シンボルは行番号で参照されます。

[Show code metrics] 選択したコード メトリック カラムを表示します。1 つのコード メトリック カラムのみ表示できます。

リレーションシップのルール

リレーション ウィンドウに表示されるリレーションシップはシンボルのタイプによって決まります。異なるシンボル タイプに表示するリレーションシップを指定できます。たとえば、「Calls」では関数のリレーションシップ、「Inheritance」ではクラスのリレーションシップを表示するように指定できます。つまり、関数を選択すると、リレーション ウィンドウはコール ツリーを表示し、クラス名を選択すると、リレーション ウィンドウはクラス継承階層を表示します。

リレーション ウィンドウがシンボルを拡張して新規レベルを表示する場合、拡張で表されるリレーションシップは拡張するシンボルのタイプに基づきます。これは、各リレーション ウィンドウが複数のリレーションシップを潜在的に表示できることを意味します。たとえば、クラスは、構成されるメンバ関数を表示します。各メンバ関数は、その参照を表示します。

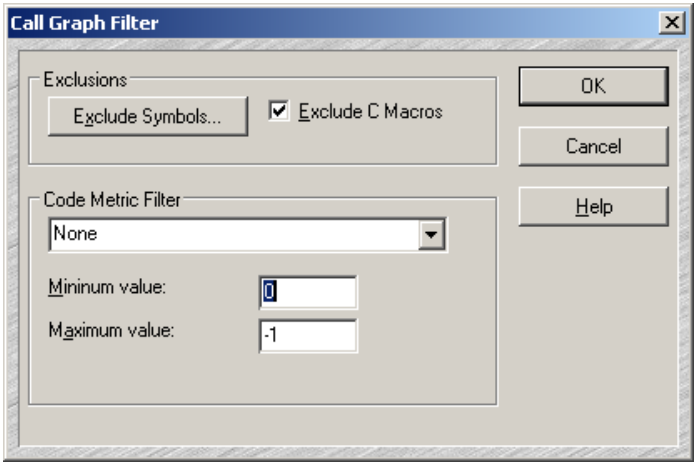
「Type of」リレーションシップ

「Type of」という名前の変数リレーションシップがあります。「Type of」リレーションシップは、変数のタイプを生成します。次に、タイプのリレーションシップ ルールが適用されます。これは、間接リレーションシップの種類です。

たとえば、変数リレーションシップを「Type of」、タイプ リレーションシップを「References」に設定します。特定の構造体またはクラス タイプの変数を選択すると、リレーション ウィンドウは変数のタイプをデコードし、「References」リレーションシップ ルールを適用して、変数のタイプへの参照を表示します。

[Call Graph Filter]

[Call Graph Filter] ダイアログ ボックスで、コール ツリー計算に関係するシンボルを制御できます。



[Exclusions] コール ツリーのフィルタされるシンボルを制御します。

[Exclude Symbols…] [Exclude Call Graph Symbols] リストを表示します。固有のシンボルをこのリストに追加できます。シンボルが除外リストに含まれる場合、Source Insight はコール グラフ表示でシンボルを展開しません。

[Exclude C Macros] オンの場合、コール グラフで C 関数形式のマクロが省略されます。オフの場合、コール グラフで関数形式のマクロが実関数のように展開されます。

[Code Metric Filter] シンボルをフィルタするコード メトリック基準を指定します。有効な場合、シンボルは、シンボルのコード メトリック値が許容可能な範囲内にある場合にのみコール グラフに含まれます。

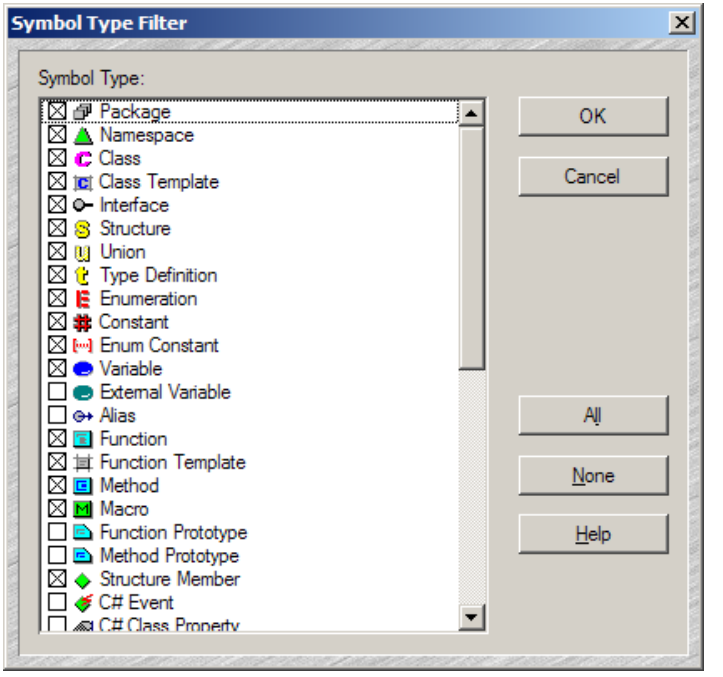
[Code Metric Filter] ドロップダウン リストから、基準として使用するコード メトリックを選択します。このオプションを使用しない場合は、「None」を選択します。

[Minimum value] このコード メトリック値以上のシンボルが含まれます。

[Maximum value] このコード メトリック値以下のシンボルが含まれます。値が -1 に設定された場合、最大値はありません。

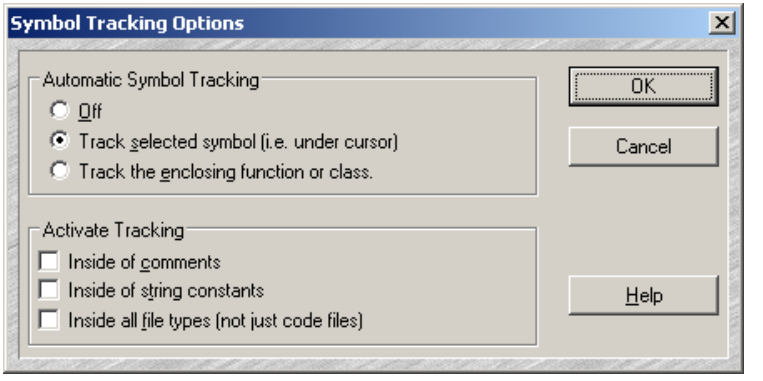
[Symbol Type Filter]

[Symbol Type Filter] ダイアログ ボックスは、操作またはリスティングをフィルタするために使用されるシンボル タイプを指定するときに表示されます。



[Symbol Tracking Options]

リレーション ウィンドウで追跡する表示オプションを設定します。



[Automatic Symbol Tracking]

ソース ファイルでカーソルを移動すると、リレーション ウィンドウで、カーソルの下、またはカーソルの周りにあるシンボルが自動的に追跡されます。リレーション ウィンドウで追跡する項目を指定します。

[Off] 自動シンボル追跡を無効にします。

[Track selected symbol] (i.e. under cursor) 入力カーソルの下にあるシンボルの定義を調べます。

[Track the enclosing function or class] 入力カーソルを含む関数またはクラスの定義を表示します。関数を編集するときにリレーション ウィンドウで関数の定義とパラメータを確認するのに便利です。

[Activate Tracking] グループ

自動追跡が行われる条件を指定します。

[Inside of comments] カーソルがコメントの内部にあるときにシンボルを調べます。

[Inside of string constants] カーソルが引用文字列定数の内部にあるときにシンボルを調べます。

[Inside all file types] カーソルが (ソース コード ファイルだけでなく) 任意のファイルの内部にあるときにシンボルを調べます。

Reload File

カレント ファイルをディスクからリロードします。保存後に行った変更はすべて失われます。保存しないでファイルを閉じ、再度ファイルを開くのと同じです。変更履歴と操作履歴も失われます。

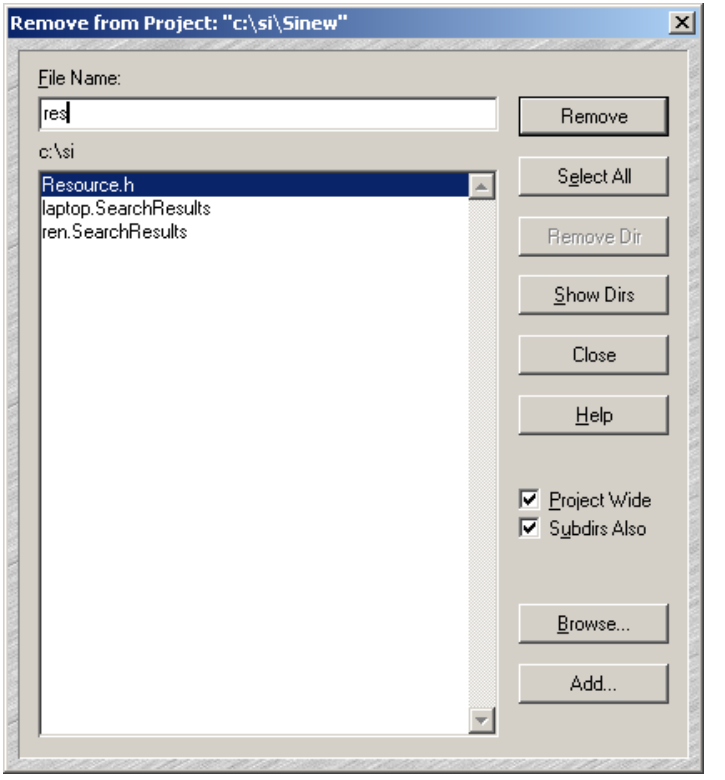
Reload Modified Files

開いている各ファイルのタイムスタンプを確認して、エディタの外部で変更されたファイルをリロードします。この処理は、**[Preferences: Files]** ダイアログ ボックスで [Sharing...] オプションをオンにして開かれたファイルにのみ適用されます。

[Preferences: Files] ダイアログ ボックスでオプションを選択して、この処理をバックグラウンドで自動的に実行できます。

Remove File

1 つ以上のソース ファイルをカレント プロジェクトから削除します。実際のファイルはディスクから削除されない点に注意してください。ファイルはプロジェクトから削除されるだけです。



[File Name] プロジェクトから削除するファイルの名前です。ワイルドカードを追加して Enter を押すと、ファイル リストがワイルドカード拡張されます。

ファイル リスト 現在の作業ディレクトリにあるすべてのプロジェクト ファイルのリストを表示します。このリスト ボックスからファイルを選択すると、そのファイル名が [File Name] テキスト ボックスにロードされます。

[Remove] 選択しているファイルをプロジェクトから削除します。[File Name] テキスト ボックスにワイルドカード文字が含まれている場合、該当するファイルが [File] リスト ボックスに表示されます。ダイアログ ボックスは開いたままです。

[Select All] [File] リスト ボックスに含まれているすべてのファイルを選択します。

[Remove Dir] 選択しているディレクトリの内容をプロジェクトから削除します。.

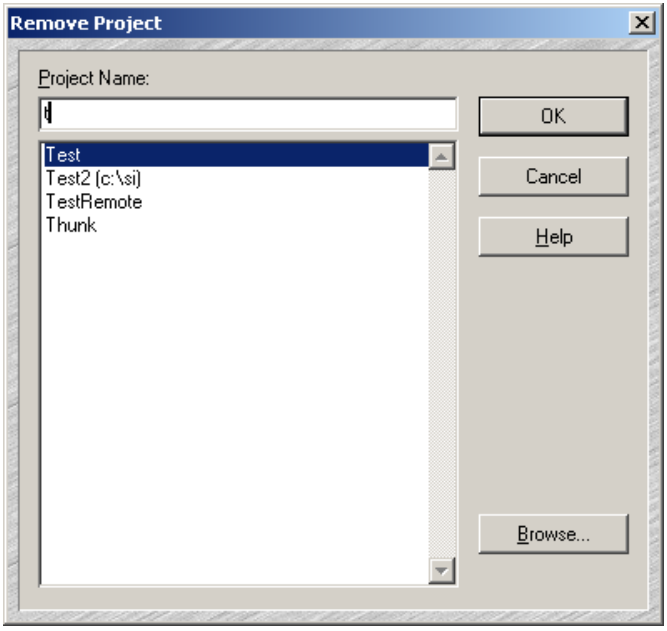
[Show Dirs/Show Files] リスト ボックスに表示する内容 (ファイル名またはサブディレクトリ名) を切り替えます。

[Browse] Windows の [ファイルを開く] ダイアログ ボックスを表示します。

[Add] [Add File] ダイアログ ボックスを表示します。

Remove Project

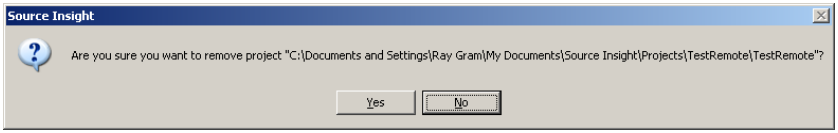
既存のプロジェクトを削除します。プロジェクトが削除されると、Source Insight で作成されたすべてのプロジェクト データが削除されます。ソース ファイルは削除されません。



[Project Name] 選択するプロジェクトの名前、または名前の一部を入力します。

プロジェクト リスト マシンで開かれたまたは作成されたすべてのプロジェクトがリストされます。ここで削除するプロジェクトを選択します。

大規模なプロジェクトを作成してソース ファイルを追加するプロセスは時間がかかるため、Source Insight はプロジェクトを削除するかどうか確認します。



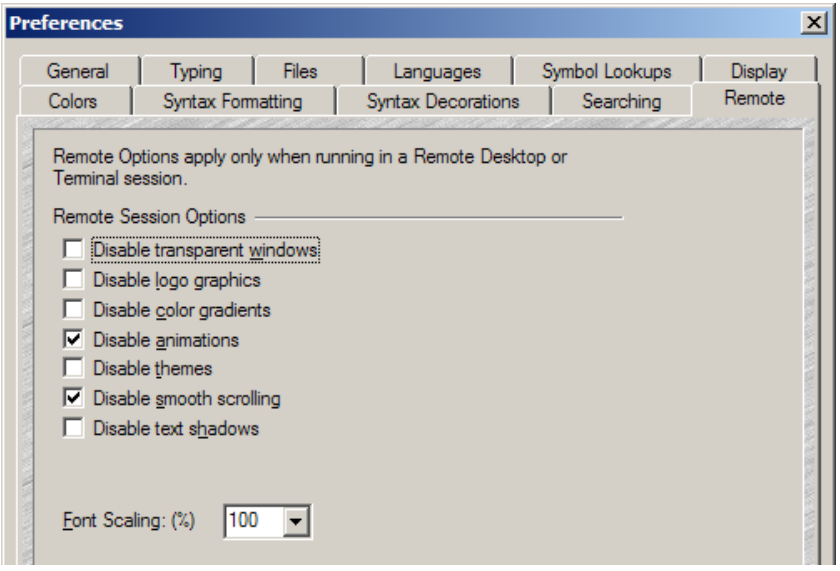
[Remove Project] コマンドをキャンセルするには、[いいえ] をクリックします。プロジェクトは削除されません。削除するように選択したプロジェクトが以前開かれていた場合、ここで閉じられます。

プロジェクトを削除するには、[はい] をクリックします。

Remote Options

[Remote Options] ダイアログ ボックスでは、Source Insight をターミナル サーバーまたはリモート デスクトップ セッションで実行するときのオプションを設定できます。ターミナル サーバー セッションで、Source Insight はリモート マシン上で実行されますが、デスクトップはローカル マシン上に表示されます。

このダイアログ ボックスで行った設定は、ターミナル サーバーまたはリモート デスクトップ セッションで実行した場合にのみ適用されます。



[Remote Session Options] これらのチェック ボックスは、リモート接続が高速でない場合に、表示が遅くなる動作を無効にします。

[Font Scaling] プログラム全体のテキストの表示比率を設定します。ソース コードとリストを含むほとんどのテキストは、この比率で表示されます。

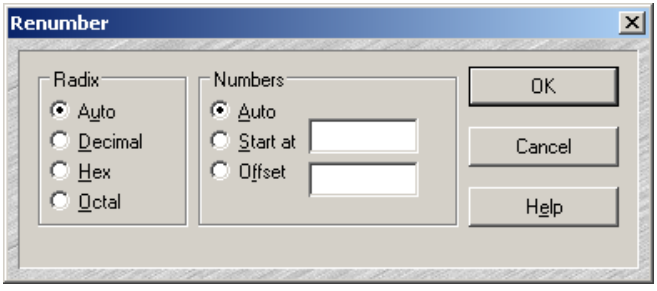
Rename

カレント ファイルの名前を変更します。ディスク上にファイルを保存する必要はありません。ファイルがカレント プロジェクトの一部である場合、プロジェクトは新しい名前を反映するように修正されます。このコマンドを使用してファイルを新しいディレクトリに移動することもできますが、ファイルを異なるドライブに移動することはできません。

[Rename] コマンドを実行してもファイルは保存されません。

Renumber

カレント ファイルまたは現在の選択範囲で見つかった数を振り直します。[Renumber] コマンドを使用したときに選択範囲が拡張されていた場合、選択範囲のみ処理されます。現在の選択範囲が挿入ポイントの場合、ファイル全体が処理されます。また、カラムを選択している場合にも処理されます。



[Radix] [Renumber] コマンドで生成する番号の基数を指定します。

[Auto] Source Insight によって決定された基数を使用します。

[Decimal] 基数 10 を使用します。

[Hex] 基数 16 を使用します。

[Octal] 基数 8 を使用します。

[Numbers] ファイルで見つかった数に対して実行するアクションを指定します。基数 10、8、または 16 で、[Start at] および [Offset] テキスト ボックスに値を設定できます。

[Auto] 最初に見つかった数の値から、昇順に数を振り直します。

[Start at] この値から、昇順に数を振り直します。

[Offset] この値を加えて数を振り直します。

基数の決定

Source Insight は、以下のように番号の基数を決定します。

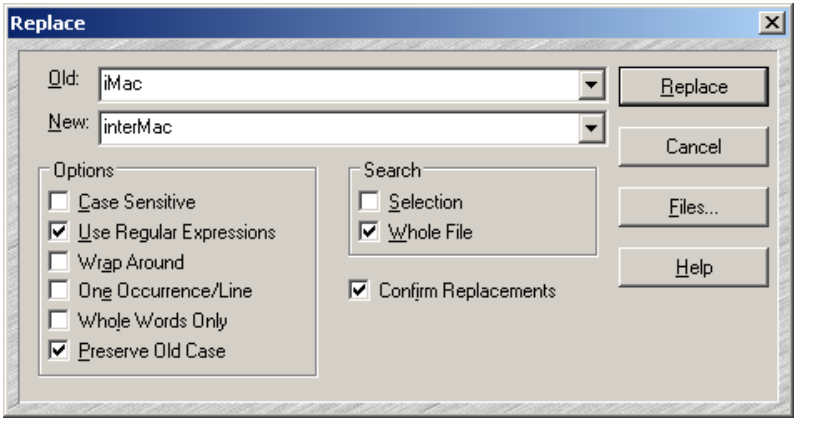
- 数が 0x で開始する場合、16 進数であると仮定します。
- 数が 0 と数字で開始する場合、8 進数であると仮定します。
- その他の場合、数は 10 進数であると仮定します。

Repeat Typing

入力した最後の文字を繰り返します。たとえば、何かを選択して abc と入力した後 [Repeat Typing] コマンドを実行すると、別の abc が自動的に挿入されます。

Replace

指定されたパターンで検索し、新規パターンに置換します。置換できるのはカレント ファイルのみです。置換の範囲は、ファイル全体または現在の選択範囲のみです。



[Old] 置換する古いパターンを入力します。正規表現を使用できます。

[New] 古いパターンと置換する新規パターンを入力します。

[Replace] 置換を開始します。

[Files...] 複数のファイルで置換を実行できる、[Replace Files] コマンドに変更します。

[Options] グループ

[Case Sensitive] オンの場合、Source Insight は大文字と小文字を区別して検索します。

[Use Regular Expressions] オンの場合、[Old] および [New] パターンは正規表現であると見なされます。

[Wrap Around] オンの場合、ファイルの最後に達したときにファイルの先頭から検索を続行します。この動作は一度だけ行われます。オフの場合、検索はファイルの最後に達したときに停止します。

[One Occurrence/Line] オンの場合、各行の最初の古いパターンのみ置換されます。オフの場合、各行のすべての古いパターンが置換されます。

[Whole Words Only] オンの場合、Source Insight は完全に一致する単語のみを検索します。オフの場合、Source Insight は入力されたテキストが単語の一部になっている場合も一致と見なします。

[Preserve Old Case] オンの場合、Source Insight は置換後のテキストの大文字と小文字をオリジナル テキストに合わせます。オフの場合、[New] テキスト ボックスに入力されたとおりに (大文字と小文字を変更しないで) テキストを置換します。このオプションは、[Case Sensitive] がオフの場合に便利です。

大文字と小文字を区別しないで検索し、置換後のテキストの大文字と小文字をオリジナル テキストに合わせることができます。たとえば、「abc」と「ABC」をそれぞれ、「xyz」と「XYZ」に置換するには、以下のようにします。[Old] テキスト ボックスに「abc」、[New] テキスト ボックスに「xyz」と入力します。[Case Sensitive] をオフにして、[Preserve Old Case] をオンにします。

[Confirm Replacements] オンの場合、Source Insight は置換前に確認します。

[Search] グループ

検索範囲を指定します。

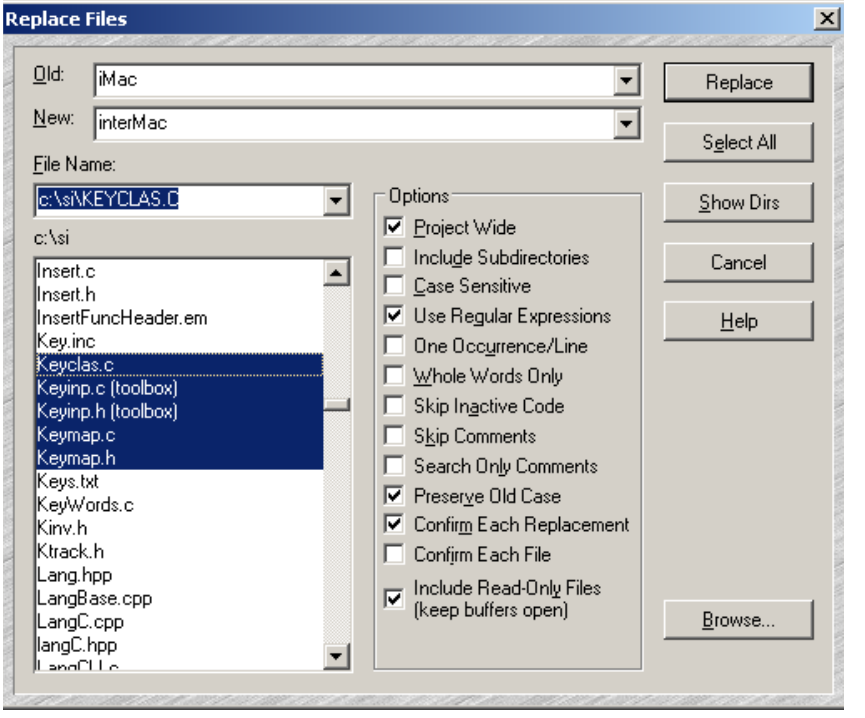
[Selection] 現在選択されているテキストのみを検索します。このチェック ボックスは、[Replace] コマンドが起動されたときに現在の選択範囲が拡張されている場合は自動的にオンになります。

[Whole File] 最初の行から最後の行まで、ファイル全体を検索します。このチェック ボックスは、[Replace] コマンドが起動されたときに現在の選択範囲が挿入ポイントだった場合は自動的にオンになります。

このグループで何もチェックされていない場合、現在の選択範囲からファイルの最後まで検索します。

Replace Files

複数のファイルで指定されたパターンを検索し、新規パターンに置換します。



[Replace] 置換を開始します。

[Select All] ファイル リストにリストされているすべてのファイルを選択します。

[Show Dirs/Show Files] リスト ボックスに表示する内容 (ファイル名またはサブディレクトリ名) を切り替えます。

[Old] 置換する古いパターンを入力します。正規表現を使用できます。

New 古いパターンと置換する新規パターンを入力します。

[File Name] 検索するファイルの名前です。ワイルドカードを追加して [Replace] ボタンをクリックすると (または Enter を押すと)、Source Insight はワイルドカード拡張の結果をファイル リストに表示します。[Project Wide] オプションがオンの場合、カレント プロジェクトに含まれるファイルのリスト全体で拡張されます。オフの場合、カレント ディレクトリで拡張されます。

[Project Wide] オプションがオンの場合、Source Insight は [File Name] テキスト ボックスのファイル名のプロジェクト シンボルを検索します。これらのファイルのディレクトリを指定する必要はありません。

ファイル リスト [Project Wide] オプションがオンの場合、カレント プロジェクトに含まれるすべてのファイルが表示されます。

[Project Wide] オプションがオフの場合、現在の作業ディレクトリに含まれるすべてのファイルが表示されます。現在の作業ディレクトリのパスがファイル リストの上に表示されます。Source Insight は、カレント ディレクトリにある既知のドキュメント タイプのファイルのみを表示します。ドキュメント タイプは、[Document Options] コマンドを使用して指定します。

[Options] グループ

[Project Wide] オンの場合、ファイル リストにプロジェクトのすべてのファイルが表示されます。オフの場合、現在の作業ディレクトリのファイルのみが表示されます。

[Include Subdirectories] オンの場合、選択されているディレクトリは再帰的に検索されます。このオプションと [Project Wide] オプションは同時にオンにできません。

ディレクトリのセットを再帰的に検索するには :

1. **[Project Wide]** チェック ボックスをオフにします。
2. **[Include Subdirectories]** チェック ボックスをオンにします。
3. ファイル リストで 1 つ以上のディレクトリを選択 μ ます。

[File Name] テキスト ボックスにワイルドカードを入力して、検索を特定のファイル拡張子または名前に制限できます。

[Case Sensitive] オンの場合、Source Insight は大文字と小文字を区別して検索します。

[Use Regular Expressions] オンの場合、[Old] および [New] パターンは正規表現であると見なされます。

[One Occurrence/Line] オンの場合、各行の最初の古いパターンのみ置換されます。オフの場合、各行のすべての古いパターンが置換されます。

[Whole Words Only] オンの場合、Source Insight は完全に一致する単語のみを検索します。オフの場合、Source Insight は入力されたテキストが単語の一部になっている場合も一致と見なします。

[Skip Inactive Code] オンの場合、条件付きコンパイルでアクティブなコードのみを検索します。Source Insight がアクティブな条件を判断できるように、最初に [Preferences: Languages] ダイアログ ボックスで条

件を指定する必要があります。条件付きコンパイルは、一部の言語でのみ利用できます。

[Skip Comments] オンの場合、コメントを検索しません。

[Search Only Comments] オンの場合、コメントのみを検索します。[Skip Comments] オプションと同時に指定できません。これらのコメント オプションをオンにした場合、検索は多少遅くなります。

[Preserve Old Case] オンの場合、Source Insight は置換後のテキストの大文字と小文字をオリジナル テキストに合わせます。オフの場合、[New] テキスト ボックスに入力されたとおりに (大文字と小文字を変更しないで) テキストを置換します。このオプションは、[Case Sensitive] がオフの場合に便利です。

大文字と小文字を区別しないで検索し、置換後のテキストの大文字と小文字をオリジナル テキストに合わせることができます。たとえば、「abc」と「ABC」をそれぞれ、「xyz」と「XYZ」に置換するには、以下のようにします。[Old] テキスト ボックスに「abc」、[New] テキスト ボックスに「xyz」と入力します。[Case Sensitive] をオフにして、[Preserve Old Case] をオンにします。

[Confirm Each Replacement] オンの場合、Source Insight は置換前に確認します。

[Confirm Each File] オンの場合、Source Insight はファイルを変更する前に確認します。

[Include Read-Only Files (keep buffers open)] オンの場合、読み取り専用ファイルに対しても置換が行われます。Source Insight は、置換中にファイルを保存しません。ユーザーがファイルを保存することはできます。オフの場合、読み取り専用ファイルはスキップされます。このオプションは、[Preferences: Files] の [Allow editing read-only file buffers] オプションとは独立して動作する点に注意してください。

[Preferences: Files] の [Save over read-only files without prompting] オプションがオンの場合、Source Insight は置換中に読み取り専用ファイルを自動的に保存します。

Restore File

カレント ファイルの内容を、最初に開かれた状態に戻します。

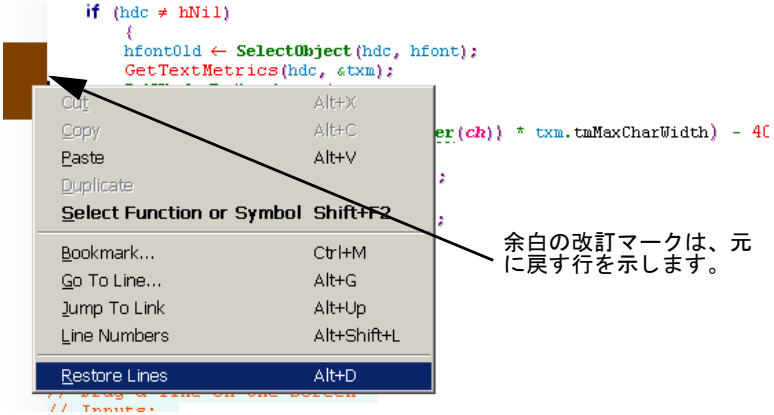
ファイルを保存した場合でも、ファイルを最初に開いた後に行ったすべての変更が元に戻されます。ディスクに保存されているファイルの内容は変更されません。ファイル バッファの内容のみが元に戻されます。

ファイルに対して行ったすべての保存が元に戻されるため、このコマンドは注意して使用してください。

[Undo] コマンドを使用して [Restore File] 操作を元に戻すことができます。

Restore Lines

編集行のブロックをオリジナルの状態に戻します。行はファイルを最初に開いた状態に戻されます。



ファイルを保存した場合でも、ファイルを最初に開いた後に行ったすべての変更が元に戻されます。ディスクに保存されているファイルの内容は変更されません。ファイル バッファの内容のみが元に戻されます。

[Restore Line] コマンドは元に戻すことができます。このコマンドは、パワフルなアウトオブオーダーの (Undo (元に戻す) 機能を提供します。

変更した行の左余白を右クリックすることで、[Restore Lines] コマンドに素早くアクセスできます。行改訂マークをオンにして、変更された行を確認できます。([Preferences: Display] コマンドを参照)

Save

カレント ファイルをディスクに保存します。ファイルはカレント名で保存されます。保存の前に、カレント ファイルに行ったすべての変更は、編集していた未保存のバージョンにのみ表示されます。ディスク上のファイルは、[Save]、[Save As]、または [Save All] コマンドを使用してファイルを保存するまで変更されません。

ファイルは、変更したファイルを閉じるときに「Save changes to file?」メッセージで「はい」を選択した場合にも保存されます。

ファイルが新規で保存されていない場合、またはファイルが読み取り専用の場合、[Save] コマンドを実行すると代わりに [Save As] コマンド

が実行されます。[Save As] コマンドでは、保存するファイルの名前を指定できます。

Save A Copy

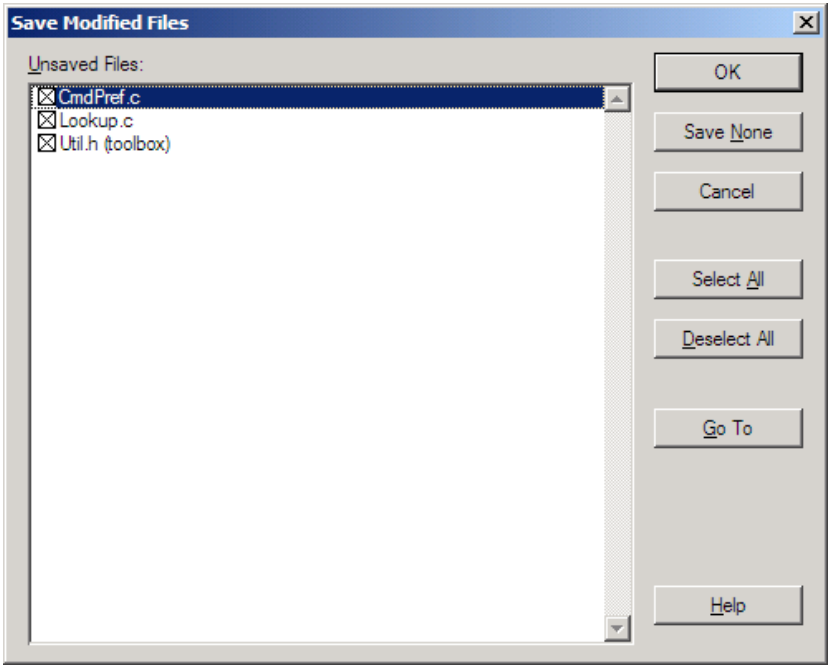
カレント ファイルを新しいファイルに保存します。カレント ファイルは上書きまたは変更されません。新しく保存されたファイルは、別のファイル バッファと同じように開かれたままです。ファイルを重複する便利な方法です。

Save All

開かれていて、最後に保存された後に変更されたすべてのファイルを保存します。

[Save Modified Files] ダイアログ ボックス

保存が必要なすべてのファイルが、[Save Modified Files] ダイアログ ボックスに表示されます。保存するファイルを選択して、[OK] をクリックします。このダイアログ ボックスは、保存が必要なファイルがあるときに [Close All] コマンドを使用したとき、または Source Insight を終了したときにも表示されます。



確認しないで保存

[Save Modified Files] ダイアログ ボックスを表示しないで、すべてのファイルを保存するには、**[Preferences: Files]** ダイアログ ボックスで [Save All operation saves without prompts] オプションをオンにします。

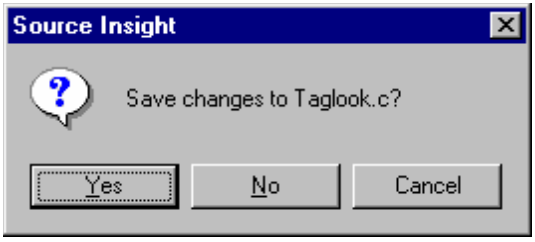
別のプログラムに切り替えたときに保存

別のプログラムに切り替えたときに、変更したファイルを自動的に保存するには、**[Preferences: Files]** ダイアログ ボックスで [Save all files when Source Insight program is deactivate] オプションをオンにします。

ファイルごとの確認

ファイルごとに「はい、いいえ、キャ塔 Z ル」メッセージを使用して保存を確認するには、**[Preferences: Files]** ダイアログ ボックスで [Save All operation will query on each file separately] オプションをオンにします。

変更され、保存が必要なファイルごとに、以下のダイアログ ボックスが表示されます。



[はい] ファイルを保存します。

[いいえ] ファイルを保存しません。

キャンセル [Save All] コマンドをキャンセルします。

Save All Quietly

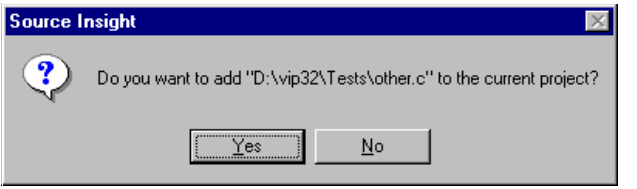
開かれていて、最後に保存された後に変更されたすべてのファイルを保存します。Source Insight は各ファイルを保存するかどうか確認しません。ファイルは自動的に保存されます。

Save As

カレント ファイルを指定した名前でディスクに保存します。

新規ファイルのカレント プロジェクトへの追加

保存するファイルがプロジェクトを開いた後に保存されていない場合、Source Insight はファイルをカレント プロジェクトに追加するかどうか確認します。



[はい] ファイルをカレント プロジェクトに追加します。

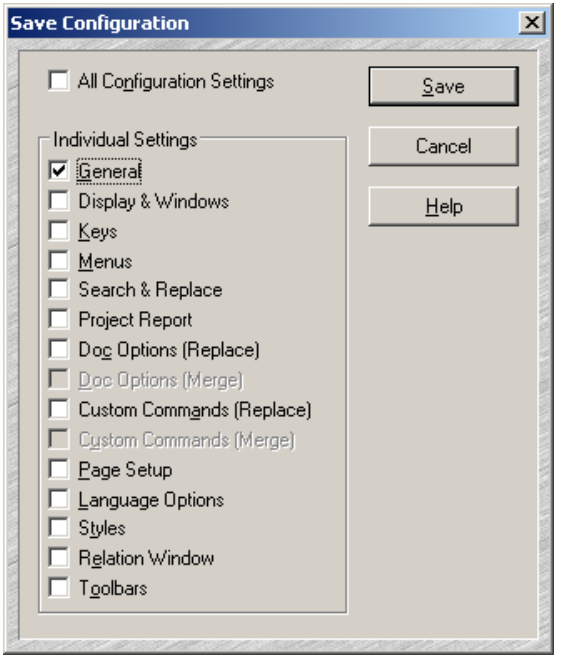
[いいえ] ファイルをカレント プロジェクトに追加しません。ファイルは保存されます。

Save Configuration

現在の設定を指定した設定ファイルに保存します。現在の設定全体、または設定の一部のみを保存できます。

設定の一部を含む設定ファイルがロードされると、そのファイルに含まれる設定のみが影響を受けます。たとえば、キーボード設定のみを設定ファイルに保存して、ファイル名を「MyKeyboard」に変更しま

す。「MyKeyboard」がロードされると、キーボードの設定のみが影響を受けます。



[All Configuration Settings] オンの場合、設定ファイルのすべての部分が保存されます。[Individual Settings] グループで定義された部分のみを保存するには、このチェック ボックスをオフにします。

[Individual Settings] **[All Configuration Settings]** チェック ボックスをオフにすると、保存する設定の部分を選択できます。たとえば、画面の色や画面のサイズなどの表示設定のみを保存して、他の設定を変更しないようにできます。

このグループには、設定の各項目用のチェック ボックスが含まれます。ここで保存する項目をチェックします。

[Save] [Save] ダイアログ ボックスを表示します。このダイアログ ボックスを使用して、保存する設定ファイルを選択します。

複数の設定の保存

複数の設定を保存できます。Source Insight で設定を行った後、[Save Configuration] コマンドを使用して、それぞれの設定を異なるファイルに保存します。設定を変更するには、[Load Configuration] コマンドを使用して設定ファイルの名前を指定します。設定ファイルが開かれると、現在の設定が置換されます。

メモ： 設定ファイルをロードすると、有効な現在の設定ファイルに自動的に保存されます。デフォルトでは、Source Insight プログラム ディレクトリの Global.cf3 に保存されます。設定を残しておくには、Global.cf3 のバックアップを作成してください。

239 ページの「Load Configuration」も参照してください。

Save Selection

現在選択しているテキストをファイルに保存します。ファイルは開かれた状態になります。新規ファイル、または既存のファイルを指定できます。Source Insight で既に開いているファイルも指定できます。ファイルが既に開かれている場合、Source Insight は新規テキストでファイルを置換するか、新規テキストをファイルに追加するか確認します。

Save Workspace

カレント ワークスペースを指定したワークスペース ファイルに保存します。

カレント ワークスペースは Source Insight を終了するときに自動的に保存され、次回 Source Insight を実行するときにリロードされます。

複数のワークスペースでの作業

個々のファイルではなくファイルのセットを使用して作業している場合、ファイルの各セットを異なるワークスペース ファイルに保存できます。別のセットのファイルを変更する場合、[Open] コマンドを使用して開くワークスペース ファイルの名前を指定します。ワークスペース ファイルが開かれると、現在のワークスペースが置換されます。つまり、すべてのファイルが閉じられ、新しいワークスペースのファイルが開かれます。

Scroll Half Page Down

アクティブなウィンドウを画面の半分下にスクロールします。

Scroll Half Page Up

アクティブなウィンドウを画面の半分上にスクロールします。

Scroll Left

アクティブなウィンドウを 1 タブ サイズ左にスクロールします。

Scroll Line Down

カレント ウィンドウをファイルで 1 行下にスクロールします。

Scroll Line Up

カレント ウィンドウをファイルで 1 行上にスクロールします。

Scroll Right

アクティブなウィンドウを 1 タブ サイズ右にスクロールします。

SDK Help

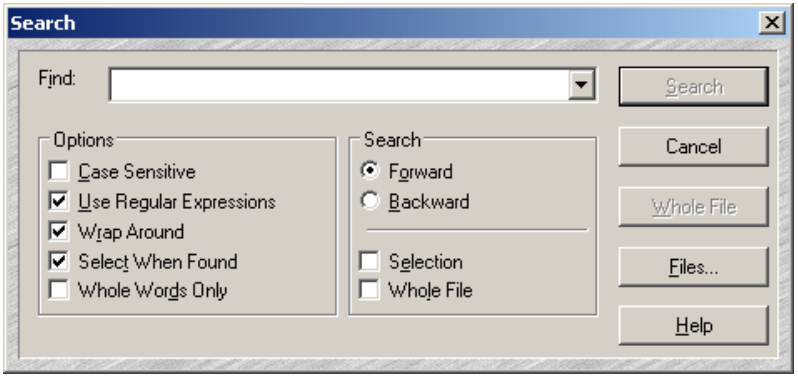
現在の選択範囲の単語に関する Windows SDK (ソフトウェア開発キット) ヘルプを表示します。たとえば、プログラムで「TextOut」を選択して [SDK Help] コマンドを実行すると、TextOut 関数のヘルプ ウィンドウが開きます。

このコマンドを使用するには、マシンに Windows SDK ヘルプ ファイルがインストールされている必要があります。

WinHelp 3.1 以降の任意のヘルプ ファイルも使用できます。この場合、SDK ヘルプ ファイルである必要はありません。Source Insight から異なるヘルプ ファイルの内容を検索する場合に便利です。

Search

指定したパターンでカレント ファイルまたは選択範囲を検索します。



[Find] 検索するパターンを入力します。

[Search] 検索を開始します。

[Cancel] コマンドをキャンセルします。

[Whole File] ファイル全体を検索して、検索結果を検索結果ウィンドウに表示します。

[Files...] 検索するファイルを指定する [Search Files] ダイアログ ボックスを開きます。

[Case Sensitive] 大文字と小文字を区別して検索します。

[Use Regular Expressions] 検索パターンは正規表現であると見なされます。113 ページの「正規表現」も参照してください。

[Wrap Around] オンの場合、ファイルの最後に達したときにファイルの先頭から検索を続行します。この動作は一度だけ行われます。オフの場合、検索はファイルの最後に達したときに停止します。

[Select When Found] オンの場合、Source Insight は一致したときにその文字を選択します。オフの場合、Source Insight は一致したテキストの最初の文字の前に挿入ポイントを移動します。

[Whole Words Only] オンの場合、Source Insight は完全に一致する単語のみを検索します。オフの場合、Source Insight は入力されたテキストが単語の一部になっている場合も一致と見なします。

[Search] グループ 検索方向と検索範囲を指定します。

[Forward] 現在の選択範囲から順方向に検索します。[Selection] または [Whole File] がオンの場合、検索は常に順方向に行われます。

[Backward] 現在の選択範囲から逆方向に検索します。

[Selection] 選択範囲のみ、順方向に検索します。

[Whole File] ファイル全体を、順方向に検索します。

[Selection] と [Whole File] の両方がオフの場合、現在の選択範囲から順方向または逆方向にファイル全体を検索します。

Search Backward

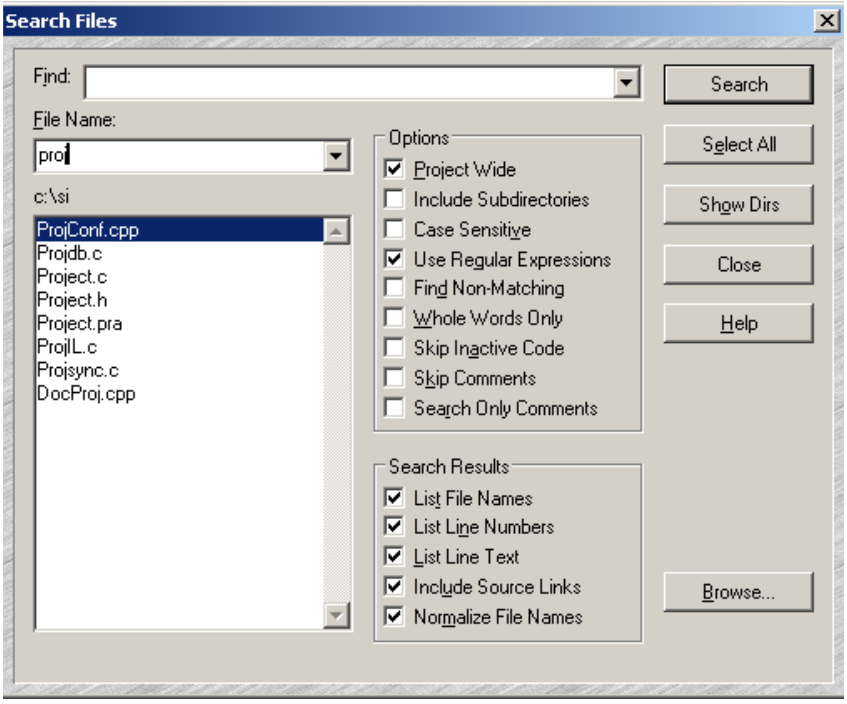
以前検索したパターンでカレント ファイルを逆方向に検索します。検索パターンは、[Search] コマンドを使用して追加します。検索は現在の挿入ポイントから行われます。

Search Backward for Selection

現在の選択範囲にある最初の単語の前の候補を検索します。このコマンドを使用するには、検索する単語内に挿入ポイントを移動してから、コマンドを起動します。Source Insight はその単語の前の候補を検索します。

Search Files

複数ファイルで検索を行います。新しい検索結果出力ウィンドウが作成されます。Source Insight は、ファイルで一致する行を見つけると、検索結果にエントリを追加します。検索結果ファイルの各行には、別のファイルの一致するテキストの場所をリンクしたソース リンクが含まれます。



[Search] 選択しているファイル、または [File Name] テキスト ボックスに表示されているファイルで検索を開始します。

[Select All] ファイル リストにリストされているすべてのファイルを選択します。

[Browse] Windows の [ファイルを開く] ダイアログ ボックスを表示します。このダイアログ ボックスでファイルを選択すると、[File Name] テキスト ボックスにファイルがフル パスで表示されます。

[Find] 検索するパターンを入力します。正規表現を使用できます。

[File Name] 検索するファイルの名前です。ワイルドカードを追加して [Search] ボタンをクリックすると、Source Insight はワイルドカード拡張の結果をファイル リストに表示します。[Project Wide] オプションがオンの場合、カレント プロジェクトに含まれるファイルのリスト全体

で拡張されます。オフの場合、カレント ディレクトリで拡張されます。

[Project Wide] オプションがオンの場合、Source Insight は [File Name] テキスト ボックスのファイル名のプロジェクト シンボルを検索します。これらのファイルのディレクトリを指定する必要はありません。

ファイル リスト [Project Wide] オプションがオンの場合、カレント プロジェクトに含まれるすべてのファイルが表示されます。

[Project Wide] オプションがオフの場合、現在の作業ディレクトリに含まれるすべてのファイルが表示されます。現在の作業ディレクトリのパスがファイル リストの上に表示されます。Source Insight は、カレント ディレクトリにある既知のドキュメント タイプのファイルのみを表示します。ドキュメント タイプは、[Document Options] コマンドを使用して指定します。

[Show Dirs/Show Files] リスト ボックスに表示する内容 (ファイル名またはサブディレクトリ名) を切り替えます。

[Options] グループ

[Project Wide] オンの場合、ファイル リストにプロジェクトのすべてのファイルが表示されます。オフの場合、現在の作業ディレクトリのファイルのみが表示されます。

[Include Subdirectories] オンの場合、選択されているディレクトリは再帰的に検索されます。このオプションと [Project Wide] オプションは同時にオンにできません。

ディレクトリのセットを再帰的に検索するには :

1. **[Project Wide]** チェック ボックスをオフにします。
2. **[Include Subdirectories]** チェック ボックスをオンにします。
3. ファイル リストで 1 つ以上のディレクトリを選択します。

[File Name] テキスト ボックスにワイルドカードを入力して、検索を特定のファイル拡張子または名前に制限できます。

[Case Sensitive] オンの場合、Source Insight は大文字と小文字を区別して検索します。

[Use Regular Expressions] オンの場合、検索パターンは正規表現であると見なされます。113 ページの「正規表現」も参照してください。

[Find Non-Matching] オンの場合、Source Insight はパターンが一致しないすべての行を検索します。

[Whole Words Only] オンの場合、Source Insight は完全に一致する単語のみを検索します。オフの場合、Source Insight は入力されたテキストが単語の一部になっている場合も一致と見なします。

[Skip Inactive Code] オンの場合、条件付きコンパイルでアクティブなコードのみを検索します。Source Insight がアクティブな条件を判断できるように、最初に [Preferences: Languages] ダイアログ ボックスで条件を指定する必要があります。条件付きコンパイルは、一部の言語でのみ利用できます。

[Skip Comments] オンの場合、コメントを検索しません。

[Search Only Comments] オンの場合、コメントのみを検索します。[Skip Comments] オプションと同時に指定できません。これらのコメント オプションをオンにした場合、検索は多少遅くなります。

[Search Results] 検索が完了した後、検索結果に表示する項目を指定します。

[List File Names] オンの場合、パターンが見つかったファイルの名前を検索結果に表示します。このオプションがオンで、[List Line Numbers] と [List Line Text] の両方がオフの場合、ファイル名はパターンがファイルで見つかったときに一度だけ検索結果に追加されます。つまり、ファイルごとに 1 つの一致のみ検索結果に表示されます。

[List Line Numbers] パターンが見つかったファイルの行番号を検索結果に表示します。

[List Line Text] パターンが見つかった行のソース テキストを検索結果に表示します。

[Include Source Links] オンの場合、検索結果に追加される各行にソース リンクが作成されます。ソース リンクを使用すると、検索結果の行とパターンが見つかった行の間でジャンプできます。ソース リンクは編集中に自動的に調整され、リンク内容が維持されます。

オフの場合、テキストのみ検索結果に追加されます。ソース リンクが多くなるとメモリの使用量が増加するため、非常に多くの一致があると考えられる場合、このオプションをオフにしてください。

[Normalize File Names] オンの場合、ファイル名は正規化されて検索結果に表示されます。オフの場合、ファイル名はフル パスで表示されます。71 ページの「ファイル名の正規化」も参照してください。

ファイルのセットを検索するには

[File Name] テキスト ボックスにワイルドカードを入力して [Search] ボタンをクリックすると、ワイルドカード リストが拡張され、ファイル リストのすべてのファイルが自動的に選択されます。**[Project Wide]** チェック ボックスがオンの場合、ワイルドカードはプロジェクトのすべてのファイルに拡張されます。

このため、たとえば、プロジェクトのすべての .h ファイルを検索する場合、[File Name] テキスト ボックスに *.h と入力し、Enter を押して

[Search] ボタンをクリックし、[Search] ボタンを再度クリックしてファイル リストのすべてのファイルを検索します。[Project Wide] オプションがオンの場合、ディレクトリに関係なく、Source Insight はプロジェクトに含まれるすべての .h ファイルを表示します。

286 ページの「Replace Files」も参照してください。

Search Forward

以前検索したパターンでカレント ファイルを順方向に検索します。検索パターンは、[Search] コマンドまたは [Search Forward for Selection] を使用して追加します。検索は現在の挿入ポイントから行われます。

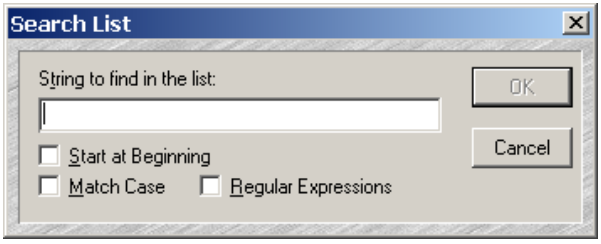
Search Forward for Selection

現在の選択範囲にある最初の単語の次の候補を検索します。このコマンドを使用するには、検索する単語内に挿入ポイントを移動してから、コマンドを起動します。Source Insight はその単語の次の候補を検索します。

Search List

このコマンドは、リストを右クリックすると表示されます。文字列または正規表現でファイルを検索できます。

ヒント： リストで入力フォーカスがオンになったら、F4 を押すと次の候補を検索できます。



[String to find in the list] 検索する文字列パターンを入力します。

[Start at Beginning] オンの場合、リストの最初の項目から検索を開始します。オフの場合、リストで選択している項目の後から検索を開始します。

[Match Case] オンの場合、大文字と小文字を区別して検索します。

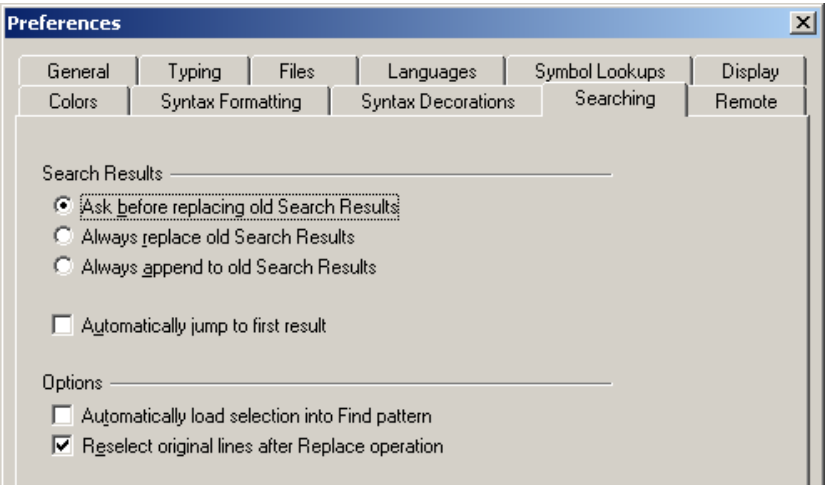
[Regular Expressions] オンの場合、検索文字列を正規表現として解釈します。113 ページの「正規表現」も参照してください。

Search Project

すべてのプロジェクト ファイルでテキストまたはキーワードを検索します。このコマンドは、[Lookup References] コマンドに似ています。唯一の違いは、それぞれのダイアログ ボックスに独自の継続状態があることです。244 ページの「Lookup References」も参照してください。

Searching Options

[Search] コマンドのオプションを指定します。



[Search Results] テキストを検索結果ウィンドウに追加する方法を制御します。

[Ask before replacing old Search Results] オンの場合、既存の検索結果を置換または追加するか、新しい検索結果ファイルを作成するか確認するメッセージが表示されます。

[Always replace old Search Results] オンの場合、既存の検索結果が新しい結果に置換されます。

[Always append to old Search Results] オンの場合、新しい検索結果が現在の検索結果の最後に新しい検索セットとして追加されます。

[Automatically jump to first result] オンの場合、検索が完了すると、Source Insight は最初に一致した結果にジャンプします。オフの場合、検索結果ウィンドウで最初に一致した結果が選択されます。

[Automatically load selection into Find pattern] オンの場合、カーソル位置の単語が [Search] ダイアログ ボックスの [Find] テキスト ボックスに自動的にロードされます。オフの場合、検索パターンは変更されません。

[Reselect original lines after Replace operation] オンの場合、置換操作が完了した後、オリジナルの選択範囲が再度選択されます。

Select All

カレント ファイルのすべてのテキストを選択します。

Select Block

現在の選択範囲を囲んでいる最小の C ブロックを選択します。[Select Block] コマンドを使用するたび、次に大きな C ブロックが選択されます。

Select Char Left

現在の選択範囲を 1 文字左に拡張します。

Select Char Right

現在の選択範囲を 1 文字右に拡張します。

Select Function or Symbol

囲まれている関数のような、囲まれているシンボル全体を選択します。このコマンドは、左の余白でマウスをダブルクリックして起動することもできます。

Select Line

カレント行をすべて選択します。

Select Line Down

現在の選択範囲を 1 行下に拡張します。

Select Line Up

現在の選択範囲を 1 行上に拡張します。

Select Match

対応する括弧まで選択します。たとえば、挿入ポイントが開き括弧の直前にある場合、対応する閉じ括弧まで選択します。

Select Next Window

アクティブなウィンドウ フォーカスを次のウィンドウに変更します。このコマンドは、すべての開いているウィンドウのフォーカスを順に変更します。

このコマンドが使用されたときにアクティブ ウィンドウが最大化されていた場合、次のウィンドウも最大化して表示されます。

Select Paragraph

囲まれている段落全体を選択します。テキストの段落は、黒線で囲まれた行のグループであると見なされます。

Select Sentence

文全体 (次のピリオドまで) を選択します。

Select Symbol

囲まれているシンボル全体を選択します。たとえば、現在の選択範囲が関数の内部である場合、[Select Symbol] コマンドを実行すると、関数に先行する行を含め、前のシンボルまで、関数全体が選択されます。

Select To

マウスを使用して新しいポイントまで既存の選択範囲を拡張します。

このコマンドを使用するには、Shift キーを押したままで、マウスをクリックします。指した場所まで選択範囲が拡張されます。選択範囲内の位置を指した場合、選択範囲はその場所まで縮小されます。

Select To End Of File

挿入ポイントからファイルの最後まで選択範囲を拡張します。

Select To Top Of File

挿入ポイントからファイルの先頭まで選択範囲を拡張します。

Select Word

挿入ポイントの単語全体を選択します。

マウスを使用してこのコマンドを実行するには :

Ctrl キーを押したままで、単語をポイントしてクリックします。

単語全体が選択されます。この状態でマウスをドラッグすると、選択範囲を拡張できます。

Select Word Left

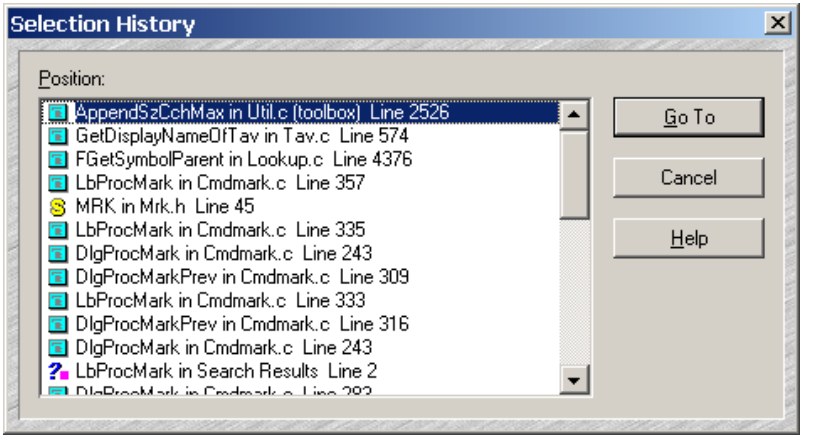
挿入ポイントから現在の単語の先頭まで選択範囲を拡張します。

Select Word Right

挿入ポイントから現在の単語の最後まで選択範囲を拡張します。

Selection History

現在開いているファイルで最近選択した位置のリストを表示します。選択履歴は、カレント ワークスペースの一部です。



[Position] すべての選択履歴の位置のリストを表示します。リストの各項目に、ファイルと行番号が表示されます。位置がシンボルの内部である場合、シンボルも表示されます。たとえば、関数の内部を選択していた場合、関数名も表示されます。

[Go To] 選択している位置にジャンプします。

Setup Common Projects

ビルドするコモン プロジェクトの種類を指定します。

このコマンドは、最初に Source Insight をインストールした後、自動的に実行されます。また、[Preferences: Symbol Lookups] ダイアログ ボックスから直接このコマンドを実行することもできます。

コモン プロジェクトとは

ほとんどの Source Insight ユーザーは、C/C++ ランタイム ライブラリや標準 Java パッケージのような標準ライブラリを利用します。Source Insight で標準ライブラリにシンボル補完などのシンボル機能を提供するには、それらのライブラリ用に別々のプロジェクトを設定する必要があります。カレント プロジェクトにシンボルが見つからない場合、Source Insight はコモン プロジェクトを検索するためソートしなおします。

[Setup Common Projects] コマンドは、これらのライブラリ用のプロジェクトをビルドする作業を支援します。ビルドするプロジェクトはプロジェクト シンボル パスに追加されるので、Source Insight は独自の

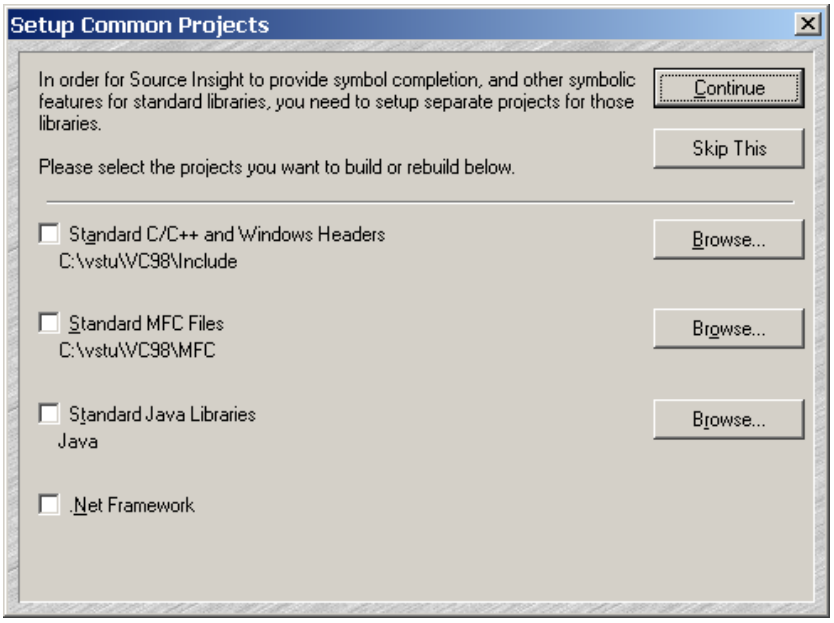
プロジェクト内からこれらのライブラリにシンボル補完などのシンボル機能を提供できます。

[Setup Common Projects] ダイアログ ボックス

それぞれのコモン プロジェクトについて、対応するファイルを保存するディレクトリを指定します。ディスクにライブラリのソース コードをインストールした場合、Source Insight で、プロジェクトの基本としてそのソース コードを使用できます。たとえば、C ランタイム ライブラリの関数 strtok への呼び出しをクリックすると、Source Insight は strtok のソース コードを表示します。

ここで作成されるコモン プロジェクトはそれぞれ、プロジェクト シンボル パスに追加されます。パスは [Preferences: Symbol Lookups] ダイアログ ボックスで編集できます。

メモ： このダイアログ ボックスでリビルドするプロジェクトを選択すると、既存のプロジェクトは置換されます。C/C++ ランタイムや Windows ヘッダー プロジェクトのような大規模なプロジェクトのビルドには時間がかかります。



[Standard C/C++ and Windows Headers] このプロジェクトには、標準 Windows、C、C++ インクルード ファイルとソース コードが含まれます。Source Insight は、レジストリを参照してプロジェクトを格納する

場所を表示します。この場所は変更できます。[Browse] ボタンをクリックして、ソース ファイルを含むフォルダを指定します。

[Standard MFC Files] このプロジェクトには、MFC (Microsoft Foundation Classes) インクルード ファイルとソース コードが含まれます。[Browse] ボタンをクリックして、ソース ファイルを含むフォルダを指定します。

[Standard Java Libraries] このプロジェクトには、java.lang のような標準 Java パッケージ用の Java 開発キット ソース コードが含まれます。Source Insight は、レジストリを参照してプロジェクトを格納する場所を表示します。この場所は変更できます。[Browse] ボタンをクリックして、ソース ファイルを含むフォルダを指定します。

[.Net Framework] このプロジェクトは、C# のシンボル自動補完に使用します。C# を使用していない場合、このプロジェクトを選択する必要はありません。Source Insight がディレクトリを作成するため、このプロジェクトのディレクトリを指定する必要はありません。

[Continue] [Continue] ボタンをクリックすると、選択したコモン プロジェクトが作成されます。選択したプロジェクトについて、Source Insight はプロジェクトに追加するファイルを確認します。[Add and Remove Project Files] ダイアログ ボックスが表示されます。

Setup HTML Help

[HTML Help] コマンドで使用する HTML ヘルプ ファイルを指定します。Microsoft® MSDN® または Microsoft® Developer Studio® ツールをインストールしている場合、開発ツールのメイン ヘルプ ファイルであるコンパイル済み HTML ヘルプの「コレクション」を選択できます。このファイルの拡張子は .col です。これで、Source Insight の内部から Windows 開発 API の HTML ヘルプを起動できるようになります。

Setup WinHelp File

[SDK Help] コマンドで使用する WinHelp ヘルプ ファイルを指定します。Windows の [ファイルを開く] ダイアログ ボックスで使用する .HLP ファイルを選択してください。

Show Clipboard

クリップボードの内容を表示するウィンドウを開きます。クリップボードの内容は編集または選択できません。

Show File Status

カレント ファイルのサイズ (行数とバイト数) をステータス バーに表示します。最後に保存された後に変更されている場合、および読み取り専用の場合、それらの情報も表示します。

Simple Tab

通常のタブを挿入します。Smart Tab モードよりも優先されます。Smart Tab オプションを有効にしている場合に便利です。Smart Tab モードでは、通常の Tab キーの動作が変更されます。このため、Smart Tab の結果が想定した結果と異なることがあります。このコマンドを使用することで、特殊な効果のない、通常のタブを挿入できます。

Smart End of Line

カーソルをカレント行の最後などに移動します。動作は次のようになります。

カーソルが行の途中にある場合、行の最後の空白でない文字の後にカーソルを移動します。

カーソルが行の最後の空白でない文字の後にある場合、行の最後にカーソルを移動します。

カーソルが行の最後にある場合、ファイルの最後にカーソルを移動します。

Smart Beginning of Line

カーソルをカレント行の先頭などに移動します。動作は次のようになります。

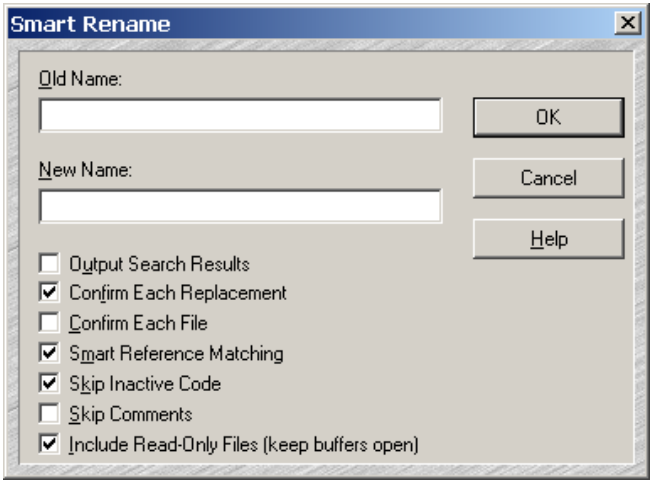
- カーソルが行の途中にある場合、行の最初の空白でない文字の前にカーソルを移動します。
- カーソルが行の最初の空白でない文字の前にある場合、行の先頭にカーソルを移動します。
- カーソルが行の先頭にある場合、ファイルの先頭にカーソルを移動します。

Smart Rename

シンボル名を変更します。[Smart Reference Matching] オプションがオンの場合、Smart Rename は正しいスコープ コンテキストに含まれるシンボル名のみ変更します。すべてのプロジェクト ファイルのシンボル名を変更できます。関数のローカル変数、クラス、構造体メンバ、および関数の名前の変更に使用できます。

Smart Rename は大文字と小文字を区別します。

Smart Rename は、特別な形式のグローバル検索と置換です。Source Insight は、高速化のためにシンボル データベース インデックスを使用します。



[Old Name] 名前を変更する識別子の名前を入力します。カーソル位置の単語がこのテキスト ボックスに自動的にロードされます。Source Insight はローカル スコープ コンテキストに基づいて名前を変更するシンボルを決定するため、カーソルの位置は重要です。

このテキスト ボックスには任意の文字列を入力できますが、名前の変更操作は単一の単語文字列用に最適化されています。また、このテキスト ボックスに文字列を入力した場合、Source Insight は、初期カーソル位置に基づいて名前を変更するシンボルを再確立する必要があります。

メンバ変数、またはローカル変数の名前を変更する場合、[Old Name] テキスト ボックスにはコンテナ シンボルを含む、シンボルの完全な名前が表示されます。たとえば、「DocDraw.paintStruc」と表示されている場合、「DocDraw」が関数名で「paintStruc」がローカル変数です。ある意味では、「paintStruc」は「DocDraw」関数のメンバです。

[New Name] 変更後の名前を入力します。メンバの場合、新しいメンバ名のみを入力し、シンボル コンテナ修飾語は省略してください。

[Output Search Results] オンの場合、検索結果が検索結果ウィンドウに表示されます。名前を変更した記録です。検索結果ウィンドウには、[New Name] 文字列で置換された前のテキストが表示されます。

[Confirm Each Replacement] オンの場合、Source Insight は置換前に確認します。

[Confirm Each File] オンの場合、Source Insight はファイルを変更する前に確認します。

[Smart Reference Matching] オンの場合、Source Insight は、言語情報とカーソル スコープ コンテキストを使用して、名前を変更するシンボルを決定し、厳格な参照のみの名前を変更していることを確認します。

[Skip Inactive Code] オンの場合、条件付きコンパイルでアクティブなコードのみを検索します。Source Insight がアクティブな条件を判断できるように、最初に [Preferences: Languages] ダイアログ ボックスで条件を指定する必要があります。条件付きコンパイルは、一部の言語でのみ利用できます。

[Skip Comments] オンの場合、コメント内のシンボル参照の名前は変更されません。

[Include Read-Only Files (keep buffers open)] オンの場合、読み取り専用ファイルに対しても置換が行われます。Source Insight は、置換中にファイルを保存しません。ユーザーがファイルを保存することはできます。オフの場合、読み取り専用ファイルはスキップされます。このオプションは、[Preferences: Files] の [Allow editing read-only file buffers] オプションとは独立して動作する点に注意してください。

[Preferences: Files] の [Save over read-only files without prompting] オプションがオンの場合、Source Insight は名前変更中に読み取り専用ファイルを自動的に保存します。

Smart Tab

[Smart Tab] コマンドがソース コードで使用されると、Source Insight は選択範囲を次の「フィールド」に移動します。フィールドは現在のコンテキストに応じて決まります。Smart Tab を使用すると、特に新しい関数呼び出しを入力しているときに、カーソルを簡単に移動できます。

([Preferences: Typing] で) [Smart Tab] オプションがオンの場合、Tab キーを押すと [Smart Tab] コマンドが実行されます。[Simple Rab] コマンドはタブを挿入するコマンドで、Smart Tab よりも優先されます。このため、[Smart Tab] オプションがオンの場合にタブを挿入するには、[Simple Tab] コマンドを使用します。

Smart Tab の例

以下の 3 つの例では、初期の選択範囲の状態をステップ 1、Smart Tab を使用した後の選択範囲をステップ 2 以降で示しています。現在の挿入ポイントは ^、テキストの選択範囲はこのように下線付きで表示しています。

例 1:

```
1. Beg^inPaint(hwnd, pps);
2. BeginPaint(hwnd, pps);
```

例 2:

```
1.  BeginPaint^(hwnd, pps);
2.  BeginPaint(hwnd, pps);
3.  BeginPaint(hwnd, pps);
4.  BeginPaint(hwnd, pps^);
5.  BeginPaint(hwnd, pps)^;
```

例 3:

```
1.  ResetAbc(^)
2.  ResetAbc()^
```

Smart Tab は、行の先頭と最後、または関数呼び出しがない行で使用した場合は通常のタブのように動作します。

Smart Tab は、関数呼び出しの自動補完を使用している場合も問題なく動作します。ポップアップ自動補完ウィンドウから関数呼び出しを挿入すると、関数のパラメータのタイプと名前が挿入され、最初のパラメータが選択されます。最初のパラメータ上で入力を開始するだけでかまいません。Smart Tab を使用すると、次のパラメータが選択されます。

Sort Symbol Window

カレント ファイル ウィンドウのシンボル ウィンドウをソートします。以下の項目でソートできます。

- 名前
- 行番号 (デフォルト)
- タイプ + 名前

Sort Symbols By Line

シンボル ウィンドウにリストされているシンボル エントリを行番号でソートします。ファイルの各シンボルが行番号順にリストされます。

デフォルトでは、シンボル ウィンドウは行番号でソートされます。

この方法ですべてのシンボル ウィンドウをデフォルトでソートするには、シンボル ウィンドウを右クリックして、右クリック ショートカット メニューで [Record New Default Properties] コマンドを実行します。

Sort Symbols by Name

シンボル ウィンドウにリストされているシンボル エントリをシンボル名でアルファベット順にソートします。

デフォルトでは、シンボル ウィンドウは行番号でソートされます。

この方法ですべてのシンボル ウィンドウをデフォルトでソートするには、シンボル ウィンドウを右クリックして、右クリック ショートカット メニューで [Record New Default Properties] コマンドを実行します。

Sort Symbols By Type

シンボル ウィンドウにリストされているシンボル エントリをシンボル タイプでソートします。たとえば、すべての構造体、すべての関数、その他のようにリストされます。

デフォルトでは、シンボル ウィンドウは行番号でソートされます。

この方法ですべてのシンボル ウィンドウをデフォルトでソートするには、シンボル ウィンドウを右クリックして、右クリック ショートカット メニューで [Record New Default Properties] コマンドを実行します。

Source Dynamics on the Web

Web ブラウザで Source Dynamics の Web サイトを開きます。

Start Recording

コマンド レコーダをオンにします。実行したコマンドも記録されます。コマンド レコーダを使用すると、1 つの一連のコマンドを記録できます。記録したコマンドを再生するには、[Play Recording] コマンドを使用します。

記録を停止するには、[Stop Recording] コマンドを使用します。[Play Recording] コマンドを使用して再生を開始した場合も記録は停止されます。

記録は一度に 1 つのみ利用できます。記録はワークスペースに保存されます。

Stop Recording

コマンド レコーダをオフにします。レコーダを開始するには、[Start Recording] コマンドを使用します。コマンド レコーダを使用すると、1 つの一連のコマンドを記録できます。記録したコマンドを再生するには、[Play Recording] コマンドを使用します。

Style Properties

表示スタイルのフォーマット プロパティを設定します。スタイルの詳細は、101 ページの「構文フォーマットとスタイル」を参照してください。

フォーマット プロパティ

スタイルのプロパティは親スタイルと組み合わされます。

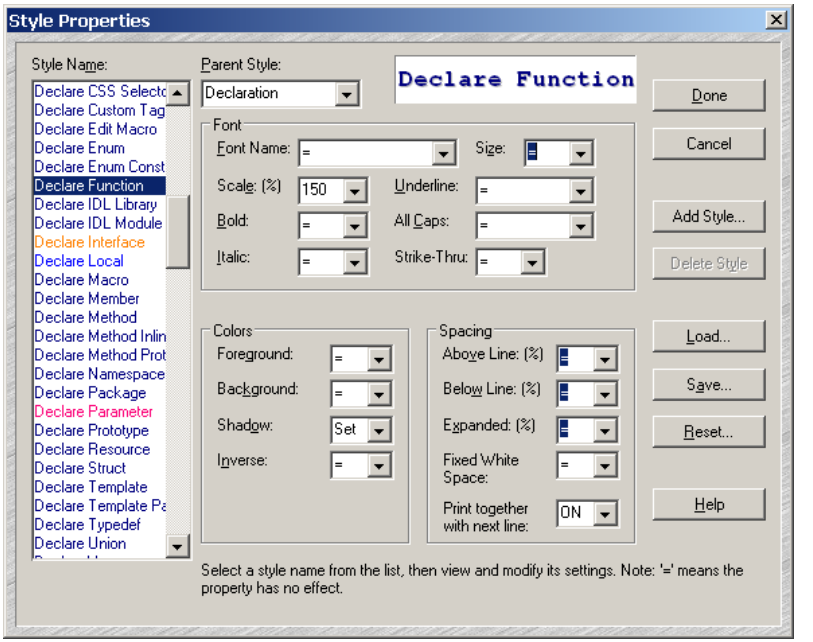
スタイルにはそれぞれ、多くのフォーマット プロパティが含まれています。スタイルは階層に存在するため、各フォーマット プロパティは親スタイルのプロパティと組み合わされて最終的なプロパティが生成されます。

たとえば、bold が「ON」の場合、bold フォーマットが追加されます。bold が「OFF」の場合、bold フォーマットが親スタイルのプロパティから削除されます。

このダイアログ ボックスの多くのフォーマット コントロールは、以下のいずれかの値になります。

- ON – プロパティが親スタイルのフォーマットに追加されます。
- OFF – プロパティが親スタイルのフォーマットから削除されます。
- 数字 – 親スタイルのプロパティがこの値に置換されます。
- =(イコール) - 親スタイルのプロパティと同じ値が継承されます。

[Style Properties] ダイアログ ボックス



[Style Name] リスト すべての構文フォーマット スタイルのリストです。このリストでスタイルを選択すると、そのプロパティが右のコントロールにロードされます。また、サンプル ボックスにスタイルのサンプルが表示されます。

[Parent Style] スタイル階層の親スタイルです。カレント スタイルはフォーマットを親スタイルから継承します。

[Add Style…] 新規ユーザー定義スタイルを追加します。

[Delete Style] ユーザー定義スタイルを削除します。標準のビルトイン スタイルは削除できません。

[Load…] 設定ファイルから新規スタイル シートをロードします。

[Save…] 現在のスタイル シート設定を新規設定ファイルに保存します。このファイルにはスタイル プロパティのみ含まれます。設定ファイルに保存できる他の要素は含まれません。この設定ファイルをロードすると、スタイル プロパティのみ変更されます。

[Reset…] すべてのスタイルを初期設定に戻します。Source Insight をインストールした後に行った変更はすべて失われます。

[Font] オプション

[Font Name] 現在選択しているフォントを示します。

[Size] フォント サイズをポイントで選択します。フォント サイズを相対的に親スタイルのフォント サイズの比率で指定するには、[Scale] プロパティを使用します。

[Scale] フォント サイズを親スタイルのフォント サイズの比率で指定します。たとえば、50% に指定すると、フォント サイズは親スタイルのフォント サイズの半分になります。

[Bold] カレント スタイルの太字プロパティを選択します。

[Italic] カレント スタイルの斜体プロパティを選択します。

[Underline] カレント スタイルの下線プロパティを選択します。

[All Caps] カレント スタイルのすべて大文字プロパティを選択します。

[Strike-Thru] カレント スタイルの取り消し線プロパティを選択します。

[Colors] オプション

[Foreground] カレント スタイルの前景色を選択します。

[Background] カレント スタイルの背景色を選択します。

[Shadow] カレント スタイルのドロップシャドウの色を選択します。

[Inverse] カレント スタイルの反転プロパティを選択します。反転とは、前景色と背景色を逆にすることです。

[Spacing] オプション

[Above Line] 行の上に追加する垂直間隔の比率を選択します。

[Below Line] 行の下に追加する垂直間隔の比率を選択します。

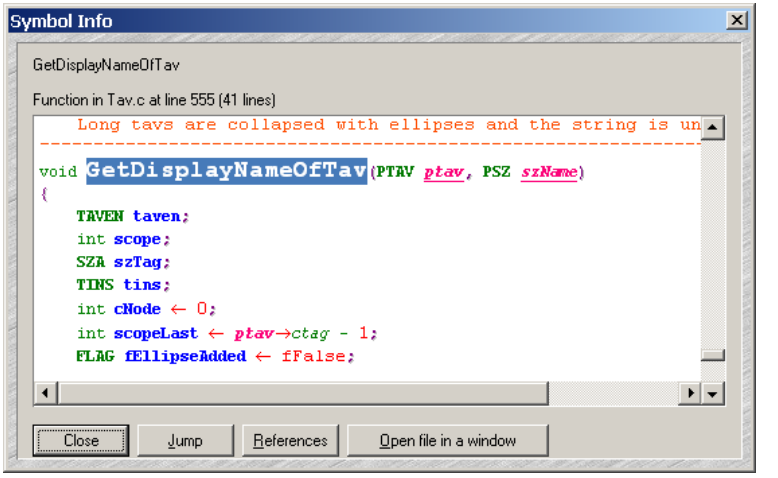
[Expanded] 文字に追加する水平間隔の比率を選択します。

[Fixed White Space] このオプションは、プロポーショナル フォントを選択している場合にのみ適用されます。Courier New のような固定ピッチのフォントを選択している場合は影響を受けません。オンの場合、Source Insight はスペースとタブに固定の幅を使用します。スペースとタブは固定ピッチのフォントを使用した場合と同じ方法で表示されます。プロポーショナル フォントを使用している場合、このオプションをオンにするとプログラムがより見やすくなります。190 ページの「[Character Spacing Options] ダイアログ ボックス」も参照してください。

[Print together with next line] オンの場合、Source Insight は印刷時にテキストを次の行と分離しないで同じページに印刷します。

Symbol Info

カーソル位置のシンボルの定義を示すポップアップ ウィンドウを表示します。識別子の定義を素早く確認できます。



シンボル名、タイプ、場所 シンボルの名前、タイプ、場所がウィンドウの上部に表示されます。

ソース ファイル シンボルが定義されているソース ファイルの名前と行番号がシンボル名の下に表示されます。シンボルのサイズがわかっている場合、シンボルのサイズが行数で表示されます。

テキスト ウィンドウ シンボルが定義されているソース ファイルの内容が表示されます。ウィンドウはスクロールできます。

[Close] ウィンドウを閉じます。

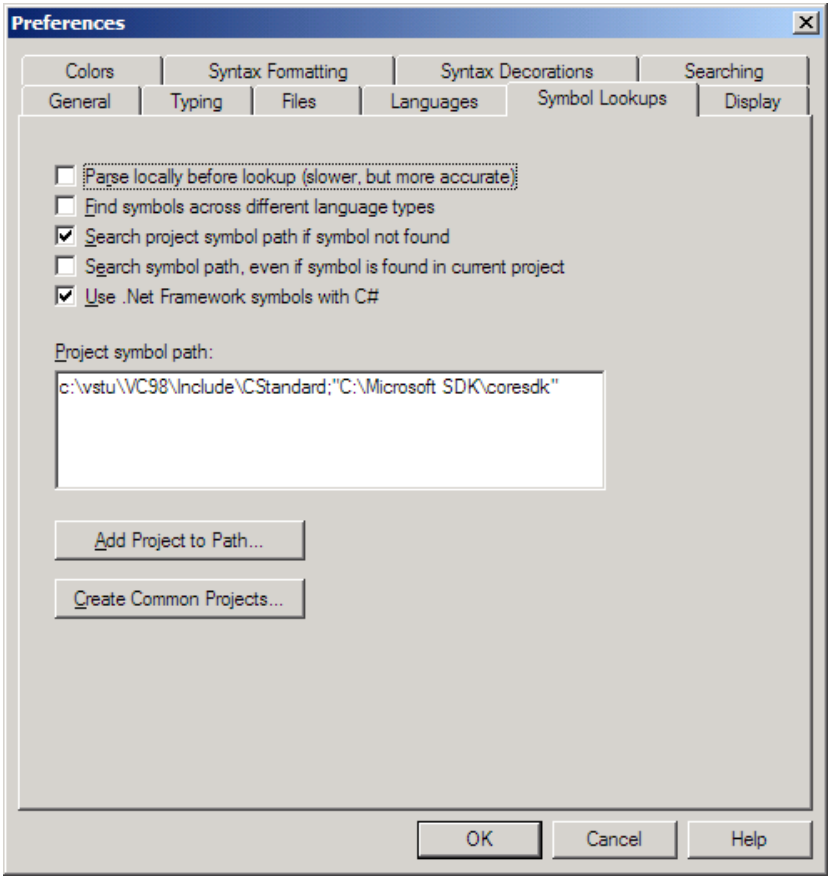
[Jump] ウィンドウを閉じて、シンボル定義に直接ジャンプします。

[References] シンボルへのプロジェクト全体の参照を検索します。

[Open file in a window] ファイルをウィンドウで開きます。ファイルが既に開かれている場合、このボタンは無効になります。

Symbol Lookup Options

Source Insight がシンボル定義を参照する方法を設定します。[Preferences] ダイアログ ボックスの [Symbol Lookups] タブを表示します。



[Parse locally before lookup] Source Insight は、カレント ファイルのシンボル情報がシンボルを参照する前に最新であることを保証します。このオプションは、編集時に作用します。文字を入力するたび、Source Insight はそのファイルのシンボル データが「古く」なると見なします。このオプションがオンの場合、ファイルは入力後に解析されます。このオプションがオフの場合、Source Insight は古い可能性のあるシンボル データをカレント ファイルに使用します。このオプションをオンにすると、シンボル参照はより正確になりますが、速度は遅くなります。また、自動補完ウィンドウも遅くなります。ほとんどの場合、このオプションをオフにしていてもシンボル参照は問題なく動作します。

[Find symbols across different language types] オンの場合、Source Insight は、ソース言語に関係なく、すべての言語のシンボル定義を検索します。オフの場合 (デフォルト)、同じ言語で定義されているシンボルのみを検索します。

[Search project symbol path if symbol not found] カレント プロジェクト、または開いているファイルでシンボルが見つからない場合、Source Insight はプロジェクト シンボル パスにリストされているプロジェクトを検索します。シンボル パスを検索した後、シンボル パスのすべてのプロジェクトを検索します。

[Search symbol path, even if symbol is found in current project] このオプションがオンの場合、シンボルが開いているファイルやカレント プロジェクトで既に見つかっている場合でも、Source Insight がシンボルを参照するたびに、シンボル パスのすべてのプロジェクトが検索されます。多くの同じシンボル名を含むプロジェクトの代替バージョンで作業している場合に便利です。カレント プロジェクトの特定のシンボルを検索すると、シンボル パスの他のプロジェクトで一致する項目も表示されます。

オフの場合、Source Insight は、シンボルが開いているファイルやカレント プロジェクトで見つからなかった場合にのみシンボル パスを検索します。

[Use .Net Framework symbols with C#] オンの場合、Source Insight は .NET Framework のシンボル情報にアクセスして C# ファイルのシンボルを自動補完します。この場合、特別なプロジェクト NetFramework が使用されます。オフの場合、NetFramework プロジェクトは使用されません。

メモ： プロジェクト シンボル パスに NetFramework プロジェクトを追加する必要はありません。Source Insight は、NetFramework プロジェクトを自動的に検索します。

[Project symbol path] プロジェクト シンボル パスは、Source Insight がシンボルの検索に使用するプロジェクトのリストです。プロジェクト

シンボル パスを使用すると、小さな自己完結型のプロジェクトが作成できますが、他のプロジェクトのシンボルを検索することもできます。

リストの各項目はプロジェクトのフル パス名です。プロジェクト パスはセミコロンで区切ります。プロジェクト ファイルのあるディレクトリに加えて、プロジェクト ファイルの名前をインクルードすることを忘れないでください。

例 :

```
c:\include\include;c:\windev\include\include
```

プロジェクト シンボル パスは、カレント プロジェクト外部のシンボルの定義を検索するためにのみ使用されます。シンボルの参照の検索や複数プロジェクトの検索には使用されません。

[Add to Project Path…] 既存のプロジェクト シンボル パスに追加するプロジェクトを選択します。

[Create Common Projects…] [Create Common Projects] ダイアログ ボックスを開きます。このダイアログ ボックスで、プロジェクト シンボル パス経由で参照する外部コモン プロジェクトを作成できます。

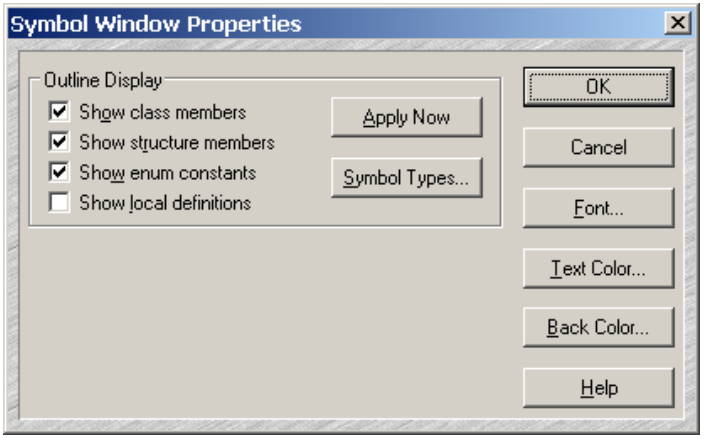
Symbol Window コマンド

カレント ファイル ウィンドウのシンボル ウィンドウを開きます / 閉じます。シンボル ウィンドウは、グローバルな範囲で通常表示される、ファイルに定義されているシンボルのリストです。シンボル ウィンドウは、[Sort Symbol Window] コマンドを使用してソートできます。

シンボル ウィンドウは、ファイルのドキュメント タイプに解析した言語が含まれている場合にのみ利用できます。

Symbol Window Properties

ソース ウィンドウの左に表示されているシンボル ウィンドウのプロパティを表示します。



[Show class members] オンの場合、シンボル ウィンドウにクラスのメンバ コンテンツが表示されます。オフの場合、クラス名のみ表示されます。

[Show structure members] オンの場合、シンボル ウィンドウに構造体フィールド メンバが表示されます。オフの場合、構造体の名前のみ表示されます。

[Show enum constants] オンの場合、シンボル ウィンドウに enum 定数が表示されます。オフの場合、enum タイプの名前のみ表示されます。

[Show local definitions] オンの場合、各関数のシンボル ウィンドウに関数ローカル宣言も表示されます。

[Apply Now] このダイアログ ボックスで行った変更をシンボル ウィンドウに適用します。

[Symbol Types…] [Symbol Type] ダイアログ ボックスを開きます。このダイアログ ボックスで、シンボル ウィンドウから異なるタイプのシンボルをフィルタして出力できます。

[Font…] シンボル ウィンドウの表示に使用するフォントを選択します。

[Text Color…] シンボル ウィンドウのテキストの色を選択します。

[Back Color…] シンボル ウィンドウの背景色を選択します。

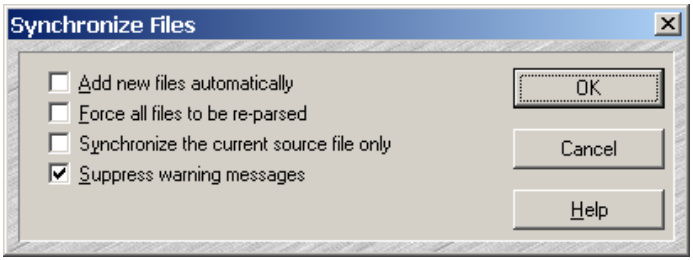
Sync File Windows

カレント ファイルを表示するすべてのウィンドウをカレント ウィンドウと同じ場所にスクロールします。

Synchronize Files

カレント プロジェクトとプロジェクトのすべてのソース ファイルを同期します。プロジェクトの各ファイルをスキャンし、Source Insight が最後にファイルを開いた後に変更されたファイルのシンボル データベースを更新します。存在しなくなったプロジェクトの一部であるファイルはプロジェクトから削除されます。

[Preferences: General] ダイアログ ボックスの [Background project synchronization] オプションをオンにして、編集中にバックグラウンドでファイルを同期することもできます。



[Add new files automatically] オンの場合、ファイルをすべて同期させる前に、Source Insight は、プロジェクトのソース ディレクトリとサブディレクトリに、新規ファイルを再帰的に追加します。しかし、スキャンされるのは既にプロジェクト ファイルに含まれているディレクトリのみです。プロジェクトのソース ディレクトリの派生ディレクトリでないディレクトリはスキャンされません。プロジェクト ディレクトリに新規ファイルを追加した後、[Synchronize Files] を実行すると、Source Insight プロジェクトにそれらの新規ファイルが自動的に追加されます。

[Force all files to be re-parsed] オンの場合、Source Insight はタイムスタンプを無視し、プロジェクトのすべてのファイルを日付が古いファイルと見なして、すべてのファイルのシンボル データベースを更新します。Source Insight のプロジェクト データ ファイルを完全にリビルドする、簡単で比較的高速な方法です。

オフの場合、プロジェクトで日付の古いファイルのみ更新されます。

[Synchronize the current source file only] 現在アクティブなソース ファイルのみ、シンボル データベースと同期されます。カレント ソース ファイルで知られていたすべてのシンボル データベース情報を置換する効果があります。

[Suppress warning messages] オンの場合、Source Insight は、ファイルを開けない / 読み取れない、などの警告メッセージを表示しません。

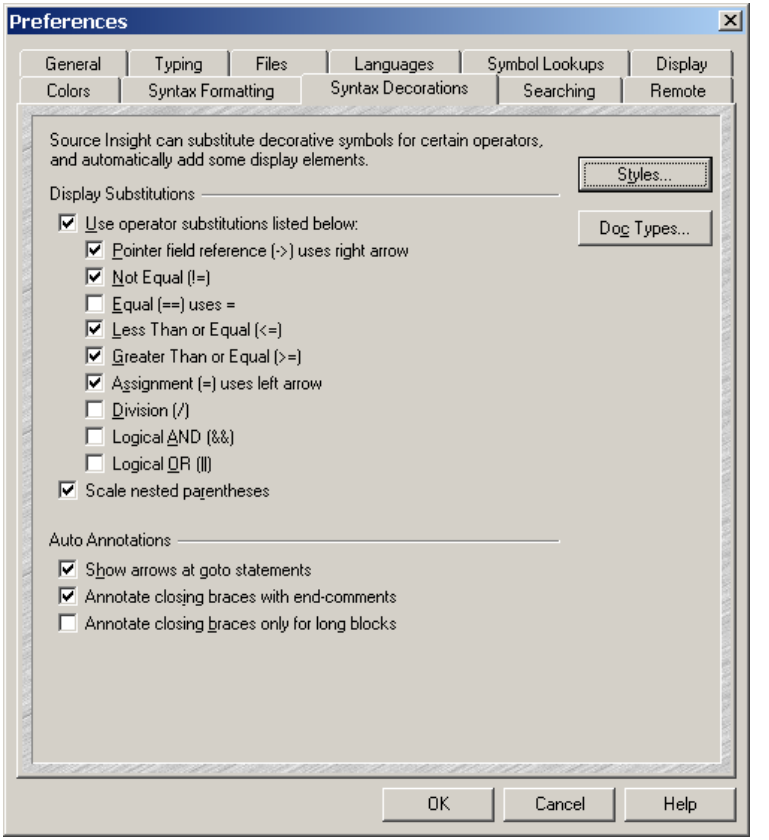
Syntax Decorations

ソース ファイル表示の構文修飾オプションを指定します。
[Preferences] ダイアログ ボックスの [Syntax Decorations] タブを表示します。

Source Insight では、一般的な演算子とシンボル文字を置換できます。[Syntax Decorations] コマンドを使用すると、使用される装飾を制御できます。

シンボル装飾やシンボル置換を行ってもソース ファイルのテキストは変更されません。画面の表示が特別なシンボルを使用するように変更されるだけです。このため、コードを編集または検索する場合は、演算子を入力する必要があります。

320 ページの「Syntax Decorations」も参照してください。



[Display Substitutions]

特別なシンボルに置換される演算子を制御します。

[User operator substitutions listed below] 以下の演算子置換のグループ全体を有効または無効にします。

[Pointer field reference uses right arrow] -> ポインタ演算子を → のような実際の矢印に置換します。

[Not Equal] != 不等号演算子を ≠ に置換します。

[Equal] == 等号演算子を = に置換します。

[Less Than or Equal] <= 演算子を ≤ に置換します。

[Greater Than or Equal] >= 演算子を ≥ に置換します。

[Assignment uses left arrow] = 代入演算子を ← に置換します。

[Division] / 除算演算子を ÷ に置換します。

[Logical AND] && 論理積を ∩ に置換します。

[Logical OR] || 論理和を ∪ に置換します。

[Scale nested parentheses] 入れ子の括弧で、外部レベルを内部レベルより大きく表示します。一致する括弧が見やすくなります。

```
i ← (ITIR) (((UINT)iMin + (UINT)iLim) >> 1);
```

[Auto Annotations]

Source Insight では、特定の注釈をソース コード表示に自動的に追加できます。以下のオプションで注釈の表示を制御します。

[Show arrows at goto statements] goto ステートメントのラベル名の隣に、上または下矢印シンボルを表示します。矢印は、ラ x ルが goto ステートメント行の上にあるか下にあるかを示します。

```
krel ← krelCalls;
goto ↑LSet;
```

[Annotate closing braces with end-comments] 閉じ中括弧の後に終了コメント注釈を表示します。終了コメントには、ブロックの開始の説明が含まれます。例 :

```
        } « end switch *pch »
    } « end if stcNewWord!=stcString... »
} « end if !FWhiteCh(*pch) »  // !fWhiteCh
```

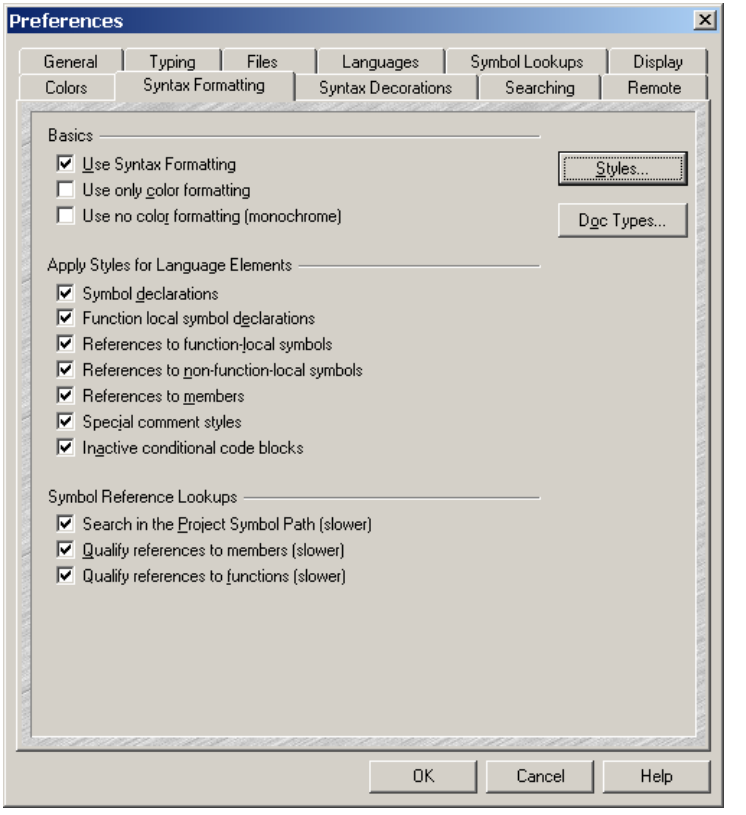
[Annotate closing braces only for long blocks] (括弧を含めて) 19 行以上のブロックのみ終了コメント注釈を表示します。

Syntax Formatting

ソース ファイル表示の構文フォーマット オプションを指定します。[Preferences] ダイアログ ボックスの [Syntax Formatting] タブを表示します。

Source Insight は、言語パーサーが収集した情報を使用してソース コードの書式を設定します。識別子は、太字や斜体のような効果に加えて、異なるフォントやフォント サイズで表示できます。

フォーマットは「スタイル」を使用して適用されます。スタイルはフォーマット プロパティのセットです。たとえば、スタイルで太字 + 斜体を指定します。各スタイルのフォーマット プロパティを編集するには、[Style Properties] コマンドを使用します。



[Styles…] スタイル プロパティを編集します。

[Doc Types…] ドキュメント タイプを編集します。

Basic オプション

以下の Basic オプションがあります。

[Use Syntax Formatting] オンの場合、Source Insight は構文フォーマットを使用してソース コードを表示します。オフの場合、ソース コードは色やフォントを変更しないで表示されます。

[Use only color formatting] フォント サイズの変更、太字や斜体のような色以外のすべてのフォーマット プロパティは表示されません。色の変更のみ表示されます。Source Insight の以前のバージョンでテキストを表示していた方法と同じです。

[Use no color formatting (monochrome)] オンの場合、Source Insight はすべての色の変更を表示しません。テキストは白黒および灰色で表示されます。

[Apply Styles for Language Elements]

Source Insight は、語彙と文脈の意味に基づいてトークンを表示するようにスタイルを適用します。このグループの各オプションを使用すると、より精巧なフォーマットが可能になります。

[Symbol declarations] シンボルの宣言を、適切な「Declare…」スタイルでフォーマットします。たとえば、関数名を宣言すると、「Declare Function」スタイルでフォーマットされます。

[Function local symbol declarations] ローカル関数スコープ変数と他のシンボルの宣言を、適切な「Declare…」スタイルでフォーマットします。ローカル変数と関数パラメータも含まれます。

[References to function-local symbols] ローカル関数スコープ変数とシンボルへの参照を、適切な「Ref to…」スタイルでフォーマットします。たとえば、ローカル変数への参照 (使用) は、「Ref to Local」スタイルでフォーマットされます。

[References to non-function-local symbols] クラス スコープやグローバル スコープのような、関数スコープの外部で宣言されたシンボルへの参照を、適切な「Ref to…」スタイルでフォーマットします。このオプションをオンにすると、多くの作業が行われるため、表示は遅くなります。参照情報はキャッシュされるため、コードの一部が処理されると、その後の表示は速くなります。

[References to members] 構造体とクラス メンバへの参照を、「Ref to Member」スタイルでフォーマットします。メンバ参照の正確さは、[Qualify references to members] オプションで制御できます。

[Special comment styles] Source Insight は、特別な //1-4 コメント見出しトークンで制御される特別なコメント スタイルとコメントの配置を

サポートしています。このオプションがオンの場合、Source Insight は、これらの特別なコメントに適切なコメント スタイルを適用します。

コメント見出し

コメント見出しスタイルは、1 から 4 の数字で始まるコメントです。例 :

```
//1 This is heading one.
//2 This is heading two.
```

コメント スタイルが使用されると、コメントの先頭の //x が非表示になり、見出しスタイル フォーマットが行の残りに適用されます。

[Inactive conditional code blocks] 非アクティブ条件付きコード ブロックのコードを、「Inactive Code」スタイルでフォーマットします。非アクティブ コード ブロックとは、非アクティブな #ifdef、#if、#elif、または #else 分岐に含まれているコードのことです。条件の状態は、[Edit Condition] コマンドを使用して制御できます。

[Symbol Reference Lookups]

潜在的な参照に遭遇すると、Source Insight はシンボルがどこで宣言されているか確認します。このグループのオプションは、Source Insight がプロジェクトで宣言されたシンボルへの参照をどのように解決するか制御します。

[Search in the Project Symbol Path] Source Insight は、カレント プロジェクトだけでなく、プロジェクト シンボル パスにあるすべてのプロジェクトでも宣言を検索します。

[Qualify references to members] オンの場合、Source Insight はメンバ宣言を「Ref to Member」スタイルでフォーマットする前に、そのメンバ宣言が存在するかどうか確認します。

オフの場合、Source Insight は、構文的にメンバ参照と思われるトークンを「Ref to Member」スタイルでフォーマットします。メンバが実際に存在するという保証はありません。例 :

```
PtrFoo->somemember  //  looks like a member reference
FooThing.somemember //  looks like a member reference
```

[Qualify references to functions] オンの場合、Source Insight は関数宣言を「Ref to Function」スタイルでフォーマットする前に、その関数宣言が存在するかどうか確認します。

オフの場合、Source Insight は、構文的に関数呼び出しと思われるトークンを「Ref to Function」スタイルでフォーマットします。関数が実際に存在するという保証はありません。例 :

```
SomeFunction(x) // looks like a function reference
```

Tile Horizontal

すべてのウィンドウを重ならないように整列します。ウィンドウは横長になるように (上下に並べて) 表示されます。複数のファイルを上下に並べて表示する際に便利です。

Tile One Window

カレント ウィンドウ以外のウィンドウを最小化します。カレント ウィンドウは Source Insight ウィンドウのワークスペース領域でほぼ最大化されます。

Tile Two Windows

2 つのウィンドウを分割して表示します。最初のウィンドウにカレント ウィンドウが含まれます。別のウィンドウには、以前のカレント ウィンドウ (最後に表示したウィンドウ) が含まれます。このコマンドは、2 つ以上のウィンドウを開いている場合にのみ使用できます。

Tile Vertical

すべてのウィンドウを重ならないように整列します。ウィンドウは縦長になるように (左右に並べて) 表示されます。複数のファイルを左右に並べて表示する際に便利です。

[Toggle Extend Mode]

拡張モードのオン、オフを切り替えます。拡張モードがオンの場合、[Cursor Up]、[Cursor Down]、[Top of Window]、および [Go To Line] のような移動コマンドを使用すると、現在の選択範囲が新しい場所に拡張されます。拡張モードがオフの場合、移動コマンドを使用すると、挿入ポイントが指定された場所に移動します。

Toggle Insert Mode

挿入モードと上書きモードを切り替えます。挿入モードでは、入力した文字は挿入ポイントの前に挿入されます。上書きモードでは、入力した文字は挿入ポイントの文字と置換されます。

Top of File

挿入ポイントをカレント ファイルの最初の行の行頭に移動します。

Top of Window

挿入ポイントをアクティブなウィンドウに表示されている最初の行の行頭に移動します。

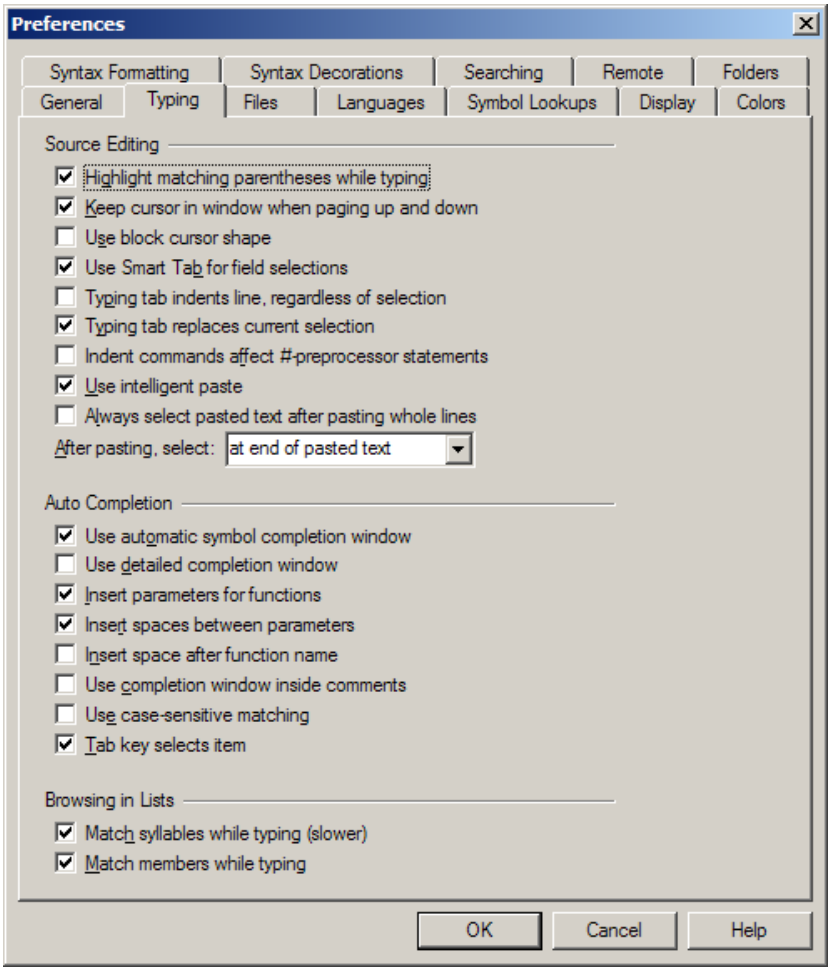
Touch All Files in Relation

このコマンドは、リレーション ウィンドウの右クリック メニューから選択できます。リレーション ウィンドウに現在表示されているすべてのファイルをタッチ (ファイルの更新日時タイムスタンプを更新) します。

リレーション ウィンドウに表示されているシンボルを含むすべてのファイルを再コンパイルする場合に役立ちます。リレーション ウィンドウに参照が表示されている場合に特に便利です。

Typing Options

入力と編集オプションを指定します。[Preferences] ダイアログ ボックスの [Typing] タブを表示します。



[Highlight matching parentheses while typing] オンの場合、閉じ括弧や閉じ中括弧を入力するときに対応する括弧または中括弧がハイライト表示されます。新しいソース コードを入力するときに対応する括弧や中括弧を確認するのに便利です。

[Keep cursor in window when paging up and down] オンの場合、ウィンドウをスクロールした場合でも挿入ポイントのカーソルをアクティブ

なウィンドウ内に表示します。オフの場合、スクロールに関係なく、挿入ポイントの位置は変更されません。

[Use block cursor shape] オンの場合、挿入ポイントのカーソルは I ビームではなくブロックで表示されます。

[Use Smart Tab for field selections] オンの場合、Tab キーを押したときに [Smart Tab] コマンドを起動します。Smart Tab を参照してください。

[Typing tab indents line, regardless of selection] オンの場合、Tab キーを押すと行全体がインデントされます。[Back Tab] コマンドは逆にインデントします。オフの場合、Tab キーを押すとタブ ストップが挿入されます。

[Typing tab replaces current selection] オンの場合、Tab キーを押すと現在の選択範囲が置換されます。たとえば、単語を選択している場合、タブを入力すると、その単語がタブに置換されます。オフの場合、タブは選択範囲の左に挿入されます。1 行または複数行が選択されている場合、Tab キーを押すと行全体がインデントされます。

[Indent commands affect #-preprocessor statements] オンの場合、[Indent Right] および [Indent Left] コマンドを実行すると、#ifdef のような C プリプロセッサ ステートメントで開始する行がインデントまたはアウトデントされます。オフの場合、コマンドを実行してもこれらの行では効果がありません。

[Use intelligent paste] 拡張貼り付けを有効にします。テキストを貼り付けるときの動作は、状況に応じて次のように変更されます。

- 挿入ポイントが行末以外にある場合、カレント行の上に行全体が挿入されます。挿入ポイントが行末にある場合、カレント行の下に新しい行が挿入されます。
- 挿入されたテキストは、他の行と一致するように自動的にインデントされます。拡張貼り付けは、[Paste]、[Paste Line]、[Drag Line Up/Down] コマンド、クリップ ウィンドウ、およびドラッグ アンド ドロップで使用できます。

[Document Options] ダイアログ ボックスで、ドキュメント タイプごとに拡張貼り付けを有効にすることもできます。194 ページの「Document Options」も参照してください。

[Always select pasted text after pasting whole lines] オンの場合、行全体を貼り付けた後、貼り付けたテキストを選択します。

[After pasting, select:] 貼り付けコマンドを使用した後、貼り付けたテキストのどの部分を選択するか指定します。

[Auto Completion]

自動補完機能の動作をカスタマイズします。

文字の入力を開始すると、自動補完ウィンドウが表示されます。この機能の目的は、入力する可能性のある識別子のリストを提供することで、入力する文字数を減らすことです。補完ウィンドウに表示されるシンボルは、カーソルの位置、参照している変数のタイプのように、コンテキストに依存します。

[Use automatic symbol completion window] 自動補完ウィンドウを有効にします。このオプションがオフの場合でも、[Complete Symbol] コマンドを手動で起動することで自動補完機能を使用できます。

[Use detailed completion window] 自動補完ウィンドウに、パラメータリストのような関数に関する詳細を表示します。

[Insert parameters for functions] 関数の名前に加えて、関数のパラメータ リストを挿入します。リストの挿入は、挿入ポイントの右にパラメータの括弧付きリストが存在しない場合にのみ行われます。

[Insert spaces between parameters] スペース文字が関数パラメータの間に挿入されます。オフの場合、パラメータはスペースなしで挿入されます。

[Insert space after function name] スペース文字が関数名と開き括弧の前に挿入されます。

[Use completion window inside comments] コメントの入力中に自動補完機能を有効にします。このオプションがオフの場合でも、[Complete Symbol] コマンドを手動で起動することで自動補完機能を使用できます。

[Use case-sensitive matching] シンボルの大文字と小文字を区別して自動補完ウィンドウにリストします。このオプションがオフの場合、入力したシンボルは大文字と小文字を区別しないでリストされます。しかし、Source Insight は入力した内容と一致する項目を選択しようとします。

[Tab key selects item] オンの場合、自動補完ウィンドウの表示中に Tab キーを押すと、リストで選択している項目が挿入されます。このオプションがオフの場合、Tab キーを押すとタブが挿入されます。どんな場合でも、Enter キーを押すと、選択している項目が挿入されます。

[Browsing in Lists]

シンボル リストおよびファイル リストで入力しているときの自動補完の動作をカスタマイズします。

[Match syllables while typing] デフォルトでシラブル マッチングを有効にします。このオプションがオフの場合でも、スペース文字を最初に入力した後、入力を続けることで、シラブル マッチングを使用できます。

[Match members while typing] クラスと構造体メンバが入力に従って一致されます。フル シンボル名は、シンボルの最上位のコンテナで始ま

るパスのようなものです。たとえば、クラス メンバの名前は、MyClass.member のようになります。このオプションがオンの場合、「member」のみ入力できます。オフの場合、完全なシンボル名のみ一致されます。このオプションをオフにしている場合でも、次のように先頭にドット (.) を入力することで、この機能を使用できます。

```
.member
```

Undo

カレント ファイルで最後に行ったコマンドの動作を元に戻します。たとえば、テキストを入力して [Undo] を実行すると、入力したテキストが削除されます。逆に、テキストを削除して [Undo] を実行すると、削除したテキストが元に戻されます。

ファイルを編集すると、Source Insight は各ファイルに加えた変更のリストを作成します。[Undo] コマンドはリストを戻し、[Redo] コマンドはリストを進めます。

操作履歴は、開いている各ファイルについて別々に保存されます。

カーソルの移動を元に戻す

カーソルの移動を元に戻すには、[Go Back] コマンドを使用します。

[Undo] コマンドは、他のいくつかのエディタと同様に、カーソルの移動を元に戻しません。Source Insight で、カーソルの移動の履歴を利用する最適な方法は、[Go Back] および [Go Forward] コマンドを使用することです。

すべての変更を元に戻す

[Undo] コマンドを連続して使用すると、すべての変更を元に戻すことができます。また、[Undo All] コマンドを使用すると、カレント ファイルを最後に開いた後に行ったすべての変更を元に戻すことができます。

操作履歴

ファイルを開いている限り、ファイルを保存した後、操作履歴が管理されます。操作履歴は、ファイルを閉じたとき、または [Checkpoint] コマンドを使用して最終保存を行うと破棄されます。[Options > Preferences: Files] ダイアログ ボックスを使用して、ファイル保存後の操作を元に戻せるかどうか制御できます。

ファイルを保存した後、いくつかの操作を元に戻すことは可能です。この場合、ディスクに保存されたファイルは、カレント バッファよりも多くの操作が行われた状態になります。この違いが問題となることがあります。この問題を防ぐため、Source Insight では、ロードされたバッファがディスク上のファイルと異なる場合、ファイル名がアスタリスク付きで表示されます。ファイルが保存された時点よりも前の状態に戻そうとすると、Source Insight は確認メッセージを表示します。

行を元に戻す

行を元に戻すには、[Restore Lines] コマンドを使用します。このコマンドは、選択している行の内容を最初にファイルを開いたときの状態に戻します。さらに、[Restore Lines] コマンドは [Undo] コマンドを元に戻すこともできます。[Undo] と [Restore Lines] コマンドを組み合わせて使用すると非常に便利です。

Undo All

カレント ファイルを最後に開いた後に行ったすべての変更を元に戻します。[Undo]、[Redo]、[Redo All] コマンドも参照してください。

Vertical Scroll Bar

カレント ウィンドウの垂直スクロール バーの表示を切り替えます。

すべてのウィンドウの垂直スクロール バーの表示を設定するには、[Preferences: Display] ダイアログ ボックスの [Vertical scroll bars for each new window] オプションを使用します。

View Relation Outline

リレーション ウィンドウのツリー データをアウトライン形式で表示します。59 ページの「リレーション ウィンドウ」も参照してください。

View Relation Window Horizontal Graph

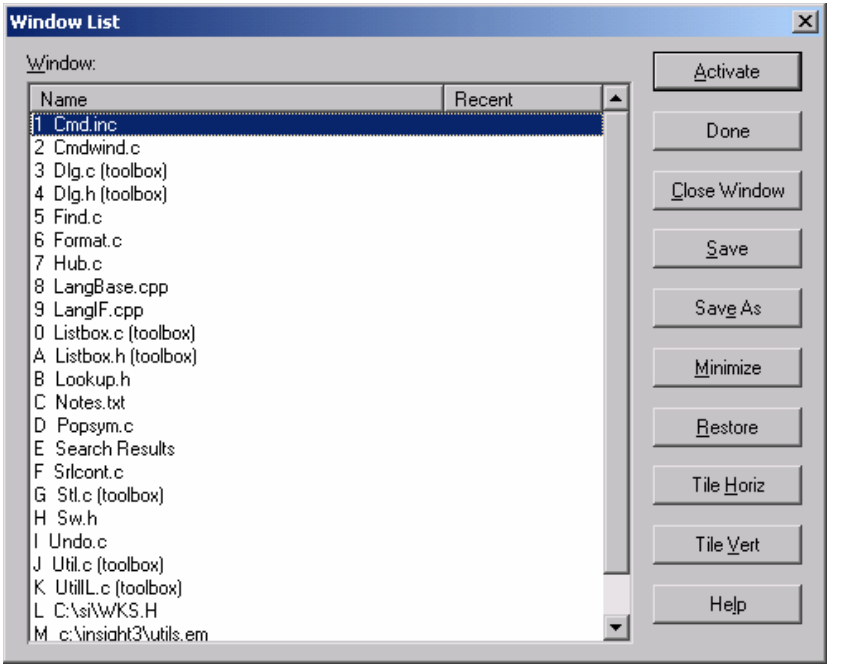
リレーション ウィンドウのツリー データを横のツリー グラフ ビューで (左から右に) 表示します。59 ページの「リレーション ウィンドウ」も参照してください。

View Relation Window Vertical Graph

リレーション ウィンドウのツリー データを縦のツリー グラフ ビューで (上から下に) 表示します。59 ページの「リレーション ウィンドウ」も参照してください。

Window List

開いているソース ウィンドウとファイル バッファのリストを管理します。ウィンドウに表示されていないファイル バッファもリストします。



Window リスト 開いているソース ウィンドウと、表示されずに「バックグラウンドで」開かれているものを含むファイル バッファのリストです。ウィンドウ リストは、名前順または最近使用した順にソートできます。リストをソートするには、リストの上部にある見出しタイトルをクリックします。

[Activate] 選択しているウィンドウを表示します。

[Close Window] 選択しているウィンドウを閉じます。

[Save] 選択しているファイルを保存します。

[Save As] 選択している各ファイルについて、[Save As] コマンドを実行します。

[Minimize] 選択しているウィンドウを最小化します。

[Restore] 選択しているウィンドウを元に戻します。

[Tile Horiz] 選択しているウィンドウを上下に並べて表示します。

[Tile Vert] 選択しているウィンドウを左右に並べて表示します。

Word Left

挿入ポイントを 1 単語左に移動します。

Word Right

挿入ポイントを 1 単語右に移動します。

Zoom Window

カレント ウィンドウが最大化されていない場合、ウィンドウを最大化します。カレント ウィンドウが既に最大化されている場合、ウィンドウを最大化する前のサイズに戻します。

第 6 章

マクロ言語ガイド

この章では、Source Insight のマクロ言語について説明します。Source Insight のマクロ言語の構文は C 言語に似ています。この章は、プログラミングの基本概念について理解していることを前提にしています。

マクロ言語の概要

Source Insight では、コマンド スクリプトの記述、特殊なフォーマット テキストの挿入、および編集操作の自動化に便利な C 言語に似たマクロ言語が用意されています。

マクロは拡張子 .EM のテキスト ファイルに保存されます。このファイルは、プロジェクト、プロジェクト シンボル パスにあるプロジェクト、またはベース プロジェクトに追加されます。プロジェクトにマクロ ファイルが追加されると、ファイル内のマクロ関数は、[Key Assignments] ダイアログ ボックスまたは [Menu Assignments] ダイアログ ボックスでユーザーレベル コマンドとして利用できるようになります。

基本的な構文規則

Source Insight のマクロ言語では大文字と小文字は区別されません。他のほとんどの言語と同様に、空白は無視されます。セミコロンは必要なく、無視されます。変数名は、数字ではなく、文字で始める必要があります。

マクロ関数

マクロ関数は、C 関数に似たフォーマットで宣言されます。空白は無視されます。関数名と変数名は大文字と小文字が区別されません。

マクロの形式 :

```
macro my_macro()
{
    /* comments */
    ..statements..   // comments
}
```

マクロ関数はパラメータを持つことができます。また、他のマクロを呼び出すこともできます。「return n」(ここで n はリターン値) を使用して、マクロから値を返すことができます。例 :

```
macro add2(n)
{
    return n + 2
}
```

マクロの範囲と参照

すべてのマクロ関数は「グローバル」です。

開いているファイルで宣言されているマクロ、プロジェクトに保存されているマクロ、プロジェクト シンボル パスにあるプロジェクトのマクロなど、すべてのマクロが同じグローバル スコープにあります。つまり、マクロの前方参照を行うことができます。マクロを呼び出す前に宣言する必要はありません。

Source Insight は、シンボル参照エンジンを使用して、マクロの実行時およびユーザーによるマクロ コマンドの呼び出し時に、マクロ名への参照を解決します。そのため、シンボル参照規則は、マクロ名のバインディングに適用されます。シンボル参照についての詳細は、「Source Insight の概念」の章の「シンボルおよびプロジェクトについて」を参照してください。

また、さまざまなシンボル参照手法を使用して、編集中にマクロ関数を検索することもできます。(「Source Insight の概要」の章の「プロジェクト シンボル パスの検索」を参照してください。) たとえば、[Browse Project Symbols] ダイアログ ボックスでマクロ名を入力して、その定義にジャンプします。

マクロの実行

マクロは、キーストロークを入力するか、またはメニュー項目を選択し、直接マクロ コマンドを呼び出して実行します。また、[Run Macro] コマンドを使用して、現在のカーソル位置で macro ステートメントを実行することもできます。

コマンドとしてのマクロ

ユーザーレベル コマンドは、パラメータがないマクロ関数です。

Source Insight は、パラメータがないマクロ関数をユーザーレベル コマンドと見なします。マクロ コマンドは、[Key Assignments] ダイアログ ボックスおよび [Menu Assignments] ダイアログ ボックスのコマンド リストに表示されます。ここで、メニューやキーマップにマクロ コマンドを割り当てることができます。また、これらのダイアログ ボックスから直接コマンドを実行することもできます。マクロ関数にパラメータがある場合は、コマンドとしてコマンド リストに表示されません。また、キーストロークを割り当てたり、メニューに割り当てることもできません。

ヒント： ユーザーレベル マクロ コマンド編集のショートカットは、コマンドを選択しながら、[Ctrl] キーを押したままの状態にします。そのコマンドのマクロ関数ソース コードが表示されます。

マクロ ファイルで新規マクロ関数を作成する場合、[Menu Assignments] および [Key Assignments] ダイアログのコマンド リストでマクロ コマンドを表示する前に、マクロ ファイルを保存して、Source Insight がマクロ ファイルとプロジェクト データベース ファイルを同期できるようにする必要があります。

マクロは、ベース プロジェクトまたはプロジェクト シンボル パスのプロジェクトに保存することもできます。マクロ名を解決するとき、Source Insight はこれらのプロジェクトを検索します。

メモ： Source Insight のマクロ ファイル (拡張子 .EM) に保存されていない限り、Source Insight はマクロ関数をユーザーレベル コマンド リストに追加しません。

インライン マクロ ステートメントの実行

ソース ファイルのコメントでマクロ ステートメントを実行できます。

[Run Macro] コマンドは、カーソルのある行からマクロ ステートメントの実行を開始します。このコマンドを使用すると、任意のファイルのオープン コード マクロ ステートメントを実行できます。これは、マクロ コードのテストとデバッグを行うときに便利です。

インライン マクロ ステートメントの実行は、プログラム コメント内にある小さなユーティリティ マクロ スクリプトの実行にも便利です。最後に必ず Stop ステートメントを使用してください。これを行わない

と、マクロ インタープリタはコメントの最後に達した後も実行し続けようとします。

たとえば、C ファイルのコメント内にある次のインライン マクロは、識別子 ucmMax への参照を検索し、識別子 ucmMax を参照しているすべてのファイルを再コンパイルします。このインライン マクロを実行するには、最初の行にカーソルを配置し、[Run Macro] コマンドを呼び出します。

```
#define ucmMax120

/* Macro: touch all ucmMax references:

// to run, place cursor on next line and invoke "Run Macro"
hbuf = NewBuf("TouchRefs") // create output buffer
if (hbuf == 0)
    stop
SearchForRefs(hbuf, "ucmMax", TRUE)
SetCurrentBuf(hbuf) // put search results in a window
SetBufDirty(hbuf, FALSE); // don't bother asking to save
Stop
*/
```

ステートメント

マクロは、個々のステートメントで構成されます。2 つ以上のステートメントのグループが中括弧 { } で囲まれています。

ステートメントは以下のいずれかです。

- `if` ステートメントや `while` ステートメントなど、マクロ言語のステートメント要素。これらのステートメントについては、後で説明します。
- ユーザー定義マクロ関数の呼び出し。マクロ関数は、マクロ ソース ファイルで定義し、保存します。
- GetCurrentBuf など、内部マクロ関数への呼び出し。Source Insight では、多くのビルトイン関数が用意されています。ビルトイン関数については、後のセクションで説明します。
- Line_Up or Copy など、Source Insight のユーザー コマンドの呼び出し。マクロ言語では、Source Insight のすべてのユーザーレベル コマンドを使用できます。ユーザー コマンドを呼び出すには、コマンド名だけ (パラメータなし) を使用します。括弧は使用しません。コマンド名にスペースが含まれている場合、スペースを下線 (_) に置換してください。たとえば、[Select All] コマンドを呼び出すには、次のようなステートメントを使用します : Select_All。

関数名と変数名は大文字と小文字が区別されません。

ステートメント構文は、C または Java と同じです。ただし、セミコロンは必要なく、無視されます。C または Java に慣れている場合、同じ要領で記述することができます。

いくつかのステートメントの例を示します。

```
hbuf = OpenBuf("file.c")// call internal function
SaveBufAs(hbuf, "filenew.c")// call internal function
Select_All// call user-level command "Select All"
Copy        // call user-level command "Copy"
Line_Up     // call user-level command "Line Up"
x = add2(n)// call user-defined macro function add2.
```

stop ステートメントを使用して実行を停止できます。

マクロからダイアログ ボックスのある Source Insight コマンドを実行すると、ダイアログ ボックスが表示されます。

基本的なマクロ言語ステートメントを次の表に要約します。

表 6.1: マクロ ステートメント

ステートメント	説明
`break`	while ループを終了します。
`if (cond)… else`	条件をテストします。
`continue`	while ループの先頭から続行します。
`return n`	関数から n をリターンします。
`stop`	マクロの実行を停止します。
`while (cond)`	cond が true の間ループします。

変数

変数に値を割り当てて定義します。変数を宣言する必要はありません。しかし、var ステートメントを使用して、変数を明示的に宣言することもできます。

変数名は大文字と小文字が区別されません。変数名は、数字ではなく、文字または下線 (_) で始める必要があります。

変数には型がありません。

すべての変数は文字列です。C のような型はなく、変数を宣言する必要はありません。そのため、簡単に変数を使用することができます。変換やキャストは必要ありません。さらに、文字列のメモリ管理も不要です。

変数の宣言

var ステートメントは 1 つのローカル変数を宣言します。

```
var variable_name
```

変数を宣言する必要はありませんが、変数の宣言にはいくつかの利点があります。

- 変数への参照の構文フォーマットが有効になる。
- 変数が空の文字列 (nil) に初期化される。そのため、初期化せずに使用しても、リテラル値と区別することができます。

例 :

```
macro SomeFunction()
{
    var localx

    // "localx" is displayed with "Ref to Local" style
    localx = 0
}
```

ローカル変数は、その変数が定義されている関数の外部からはアクセスできません。

変数の初期化

変数は、値を割り当てるだけで初期化されます。空の文字列に初期化すると便利です。この場合、特別な定数 nil を使用します。

例 :

```
S = nil // s is set to the empty string
S = ""  // same thing
```

グローバル変数

global ステートメントは、グローバル変数の宣言に使用します。例 :

```
macro SomeFunction()
{
    global last_count
    ...
}
```

var ステートメントで宣言された変数や割り当てによって作成された変数は、そのステートメントが含まれている関数のローカル変数です。global ステートメントは、ローカルではなく、グローバル スコープの変数を宣言します。つまり、複数の関数からアクセスすることができます。

global ステートメントの変数は、グローバル スコープ変数テーブルに追加されます。グローバル変数は、「nil」に初期化されます。この例では、グローバルなカウンタ変数の宣言方法と使用方法を紹介します。

```
macro SomeFunction()
{
    global last_count

    // this initializes it the first time through
    if last_count == nil last_count = 0
    ...
}
```

グローバル変数の値は、Source Insight のセッションの開始から終了まで保持されます。

グローバル変数は、Source Insight のセッションが有効な限り、有効です。つまり、マクロ関数やイベント関数の呼び出し間の情報を保持することができます。Source Insight を終了すると、グローバル変数は破棄されます。

global ステートメントに到達すると global ステートメントが実行されます。global ステートメントはステートメントの実行パスになければなりません。global ステートメントは関数内に配置してください。

グローバル変数は、カウンタの追加および他の継続状態に便利です。すべてのハンドルは、マクロの終了時に破棄されるため、ハンドルは*保持できません*。たとえば、次のコードは*使用できません*。

```
global hbuf
hbuf = OpenBuf("abc.txt")
```

上記の例では、マクロが終了すると、hbuf 変数には正しくないハンドルが保持されます。

変数名の拡張

識別子が定義されている変数名の場合、識別子はそれらの文字列値に拡張されます。そうでない場合、そのまま使用されます。また、二重引用符で囲まれている場合も、そのまま使用されます。例 :

```
s = abc  // same as s = "abc" if abc is not defined
```

.. または ..

```
abc = Hello
s = abc  // same as s = "Hello" (if Hello is not defined!)
s = "abc"    // s equals "abc"
```

誤って変数の値として変数名を使用しないように、var ステートメントを使用して変数の宣言を行ってください。

文字列内の変数の拡張

@ 文字を使用して、変数を別の文字列に挿入することができます。リテラル文字列内に @ 文字で囲まれた変数名がある場合、Source Insight は @variable@ をその変数の値に置換します。

例 :

```
S = "Hey, @username@, don't break the build again!"
```

この例は、文字列内の @username@ を変数 username の値に置換します。

\@ または @@ を使用して、@ 文字をエスケープできます。例 :

```
S = "Mail info@@company.com for information."
```

変数の演算

変数は文字列として表されますが、演算を行うことができます。例 :

```
s = "1"
x = s + 2   // x now contains the string "3"
y = 2 * x + 5// x now contains "11"
```

演算の前に、変数は数値に変換されます。

変数を数値として使用するために、何もする必要はありません。変数がどのように保存されているのかを考慮する必要もありません。しかし、変数に数値として評価されない文字列が含まれている場合は、注意が必要です。この場合、エラーが生成されます。例 :

```
s = "hello"
x = s + 1// error
```

IsNumber 関数を呼び出して、文字列が数値がどうかを確認できます。

```
if (IsNumber(x))
    x = x / 4  // okay to do arithmetic
```

浮動小数点数はサポートされていません。

文字列のインデックス

変数のインデックスを作成すると、1 文字の新規文字列が作成されます。

変数のインデックスを文字配列として作成できます。インデックスはゼロベースです。例 :

```
s = "abc"
x = s[0]    // x now contains the string "a"
```

変数のインデックスを作成すると、1 文字の新規文字列が作成されます。

最後の文字の次のインデックスは、空の文字列です。これは、C の 0 終端文字列に似ています。\0 文字の代わりに空の文字列が使用されます。

```
s = "abc"
length = strlen(s)
ch = s[length] // ch now contains the empty string
if (ch == "")
    msg "End of string."
```

レコード変数

レコード変数は構造体に似ています。

C のデータ構造はサポートされていませんが、集合型レコード変数はサポートされています。レコード変数は、実際には「name=value」のリストです。レコード変数の使用方法は、C の構造体と同じです。

レコード変数は、複数の値を返すのに便利なため、いくつかの内部マクロ関数によって返されます。

レコード フィールドは、ドット (.) 演算子を使用して <recordvar>.<fieldname> 形式で参照されます。

たとえば、次の例は シンボル参照レコード変数からファイル内のシンボルの場所を読み取ります。

```
Filename = slr.file // get file field of slr
LineNumber = slr.lnFirst  // get lnFirst field of slr
```

これに似た方法で、レコード変数に値を割り当てます。

```
userinfo.name = myname; // set "name" field of userinfo
```

nil を割り当てて、空のレコードを初期化します。

```
userinfo = nil// make a new empty record
userinfo.name = "Jeff"// begin adding fields
```

レコード変数の保存

レコード変数は、長い文字列です。

文字列として保存されます。各フィールドは、「fieldname=value」としてセミコロンで区切って保存されます。

例 :

```
name="Joe Smith";age="34";experience="guru"
```

レコード変数の文字列を一から作成する場合、次のように文字列を二重引用符で囲み、文字列内の各二重引用符には \" を使用してエスケープします (C と同じです)。

```
rec = "name=\"Joe Smith\";age=\"34\";experience=\"guru\""
```

ただし、フィールドごとに割り当てたほうが簡単です。例 :

```
Rec = nil  // initializes as an empty string
Rec.name = "Joe Smith"
Rec.age = "34"
Rec.experience = "guru"
```

フィールドに特定の順序はありません。事前に宣言され、レコード変数に関連付けらている構造体はありません。そのため、値を割り当てるだけで、新規フィールを追加できます。

例 :

```
Location = GetSymbolLocation(symname)
Location.myOwnField = xyz// append a field when you feel like
   it!
```

配列手法

ファイル バッファを配列として使用できます。

Source Insight のマクロ言語は、配列変数をサポートしていません。しかし、ファイル バッファを動的配列として使用することができます。バッファ内のそれぞれの行は、配列要素を表します。また、レコード変数を各行に保存してレコード配列とすることもできます。

バッファ関数については、後のセクションで説明します。バッファの作成と破棄を行う NewBuf および CloseBuf や、バッファ ライン関数の GetBufLine、PutBufLine、InsBufLine、AppendBufLine、DelBufLine、GetBufLineCount など、いくつかの便利な関数があります。また、NewWnd は、ウィンドウに配列バッファを配置し、配列の内容を表示することができます。

次の例は、レコードの配列バッファを作成します。

```
hbuf = NewBuf()
rec = "name=\"Joe Smith\";age=\"34\";experience=\"guru\""
AppendBufLine(hbuf, rec)
rec = "name=\"Mary X"\";age=\"31\";experience=\"intern\""
AppendBufLine(hbuf, rec)
// hbuf now has 2 records in it
…
rec = GetBufLine(hbuf, 0) // retrieve 0th element
CloseBuf(hbuf)
```

特別な定数

マクロを実行中に、いくつかの定数値が定義されます。他のすべての識別子と同様に、定数名は大文字と小文字が区別されません。

表 6.2: ランタイム定数

定数	値
True	"1"
False	"0"
Nil	"" – 空の文字列。
hNil	"0" – 無効なハンドル値。
invalid	"-1" – 無効なインデックス値。

演算子

ほとんどのバイナリ演算子は C と同じです。演算子の優先順位も C と同じです。括弧を使用して、式をグループ化することもできます。

表 6.3: 演算子

演算子	意味
+ および -	加算および減算
* および /	乗算および除算
!	否定または "Not"。たとえば、!True は False と同じです。
++i および i++	前置および後置インクリメント
--i および i--	前置および後置デクリメント

表 6.3: 演算子

演算子	意味
||	論理和演算
&&	論理積演算
!=	論理不等価演算
==	論理等価演算
<	より小さい
>	より大きい
<=	以下
>=	以上
#	文字列連結
"@var@"	変数式。文字列内の変数を拡張するために、引用符で囲まれた文字列内で使用されます。

変数には非数値が格納されていることがあるため、関係演算子は次のように処理されます。

表 6.4: 文字列の関係演算子

演算子	意味
==	文字列は同じです (大文字と小文字が区別されます)。
!=	文字列は同じではありません (大文字と小文字が区別されます)。
< > <= >=	文字列は数値に置換されてから比較されます。空の文字列または非数値の文字列は、ランタイム エラーになります。

条件およびループ : if-else と while

Source Insight のマクロ言語は、if ステートメントおよび while ステートメントをサポートしています。

if ステートメント

if ステートメントは、条件をテストし、条件に基づいて 1 つまたは複数のステートメントを実行するのに使用します。if ステートメントの構文は次のとおりです。

```
if (condition)
    statement      // condition is true
```

上記の例では、条件が true の場合、ステートメントが実行されます。{} を使用して 2 つ以上のステートメントをグループ化することもできます。例 :

```
if (condition)
    {
    statement1
    statement2
    }
```

if ステートメントに else 節を追加することができます。

```
if (condition)
    statement1     // condition is true
else
    statement2     // condition is false
```

while ステートメント

while ステートメントは、条件が true である間、ループします。while ステートメントの構文は次のとおりです。

```
while (condition)
    statement  // keeps executing statement until condition is
   false
```

上記の例では、条件が true である限り、while ブロック内のステートメントが実行されます。条件は、毎回ループの最初にテストされます。

{} を使用して 2 つ以上のステートメントをグループ化することもできます。例 :

```
while (condition)
    {
    statement1
    statement2
    }
```

break と continue

break ステートメントは while ループを終了するのに使用します。複数の while ループが入れ子されている場合、break ステートメントは最内ループを終了します。

```
while (condition)
    {
    if (should_exit)
        break   // exit the while loop
    …
    }
```

continue ステートメントは、ループの先頭 (条件式の評価) に移動します。

```
while (condition)
    {
    if (skip_rest_loop)
        continue   // continue at the top of the while loop
    …
    }
```

条件の評価

Source Insight は、毎回条件式全体を評価します。これは、C の条件式の評価方法とは異なる重要な点です。C では式の一部だけが評価されることがあります。次の条件式について考えてみましょう。

```
if (hbuf != hNil && GetBufLineCount(hbuf) > x)
```

hbuf が hNil の場合、Source Insight ではこの条件式はエラーになります。C では、hbuf != hNil を評価した後に評価は終了されます。Source Insight では式全体が評価されます。この場合、hNil がバッファ ハンドルとして GetBufLineCount へ渡されるためエラーになります。

Source Insight では、このステートメントは次のように記述する必要があります。

```
if (hbuf != hNil)
    if (GetBufLineCount(hbuf) > x)
```

命名規則

このマクロ言語ガイドの変数名と関数パラメータ名には、次の命名規則が使用されています。

表 6.5: 識別子の命名規則

名前	意味
s および sz	文字列
ch	単一文字。文字列に複数の文字がある場合、最初の 1 文字のみ使用されます。
ich	文字列内または行内の文字へのゼロベースのインデックス
ichFirst	文字範囲内の最初のインデックス
ichLast	文字範囲内の最後のインデックス (包括的)
ichLim	限界インデックス - 範囲内の最後のインデックスの次
cch	文字数
fThing	「f」はフラグまたは論理値を示します。fThing は「Thing」が True であることを意味します。
TRUE	非ゼロ値。例 :「1」
FALSE	ゼロ値。例 :「0」
ln	ゼロベースの行番号
lnFirst	範囲内の最初の行番号
lnLast	範囲内の最後の行番号 (包括的)
lnLim	限界 - 範囲内の最後の行番号の次
hbuf	ファイル バッファへのハンドル
hwnd	ウィンドウへのハンドル
hprj	プロジェクトへのハンドル
hsyml	シンボル リストへのハンドル
その他の名前	一般的な文字列変数

標準のレコード構造

レコード構造は C のデータ構造に似ています。レコード変数は、実際には「name=value」のリストです。レコード変数の使用方法は、C の構造体と同じです。

レコード変数は、複数の値を返すのに便利なため、いくつかの内部マクロ関数によって返されます。

このセクションでは、Source Insight の内部マクロ関数で使用されるレコード構造について説明します。

Bookmark レコード

ブックマークとは、ファイル バッファの位置です。Bookmark レコードには次のフィールドがあります。

フィールド	説明
Name	ブックマーク名。
File	ブックマークの位置のファイル名。
ln	行番号の位置。
ich	行にある文字インデックス。

Bufprop レコード

Bufprop レコードには、ファイル バッファのプロパティが格納されています。Bufprop レコードは、GetBufProps によって返されます。Bufprop には次のフィールドがあります。

フィールド	説明
Name	バッファ名 (例 : ファイル名)
fNew	保存されていない新規バッファの場合は True。
fDirty	バッファを保存または開いた後に編集した場合は True。
fReadOnly	バッファが読み取り専用の場合は True。
fClip	バッファがクリップ ウィンドウに表示されるクリップの場合は True。
fMacro	バッファがマクロ ファイルの場合は True。
fRunningMacro	バッファが現在実行中のマクロ ファイルの場合は True。
fCaptureResults	バッファにカスタム コマンド出力が格納されている場合は True。

フィールド	説明
fSearchResults	バッファに検索結果が格納されている場合は True。
fProtected	バッファが内部使用のために保護および予約されている場合は True。
lnCount	バッファの行数。
Language	バッファのプログラミング言語。ファイルのドキュメント タイプにより決定されます。
DocumentType	バッファのドキュメント タイプ。

DIM レコード

DIM レコードは、水平および垂直のピクセル サイズを指定します。

フィールド	説明
Cxp	X ピクセル数 (水平サイズ)
Cyp	Y ピクセル数 (垂直サイズ)

Link レコード

Link レコードは、ソース リンクによって指定されるファイル内の場所を指定します。Link レコードには次のフィールドがあります。

フィールド	説明
File	ファイル名
ln	行番号 - ファイルが変更されない限り有効です。ファイルに行が挿入されたり、ファイルから行が削除されると、行番号は無効になります。GetSourceLink を再度呼び出し、更新された行番号を取得できます。

ProgEnvInfo レコード

ProgEnvInfo レコードには、Source Insight の実行環境の情報が格納されています。

フィールド	説明
ProgramDir	Source Insight のプログラム ディレクトリ。プログラム ファイルが保存されている場所です。
TempDir	テンポラリ ファイル ディレクトリ。

フィールド	説明
BackupDir	バックアップ ディレクトリ。保存したファイルのバックアップが格納されている場所です。
ClipDir	クリップが保存されているディレクトリ。
ProjectDirectoryFile	プロジェクト ディレクトリ ファイルの名前。プロジェクト ディレクトリ ファイルには、開かれたすべてのプロジェクトのリストが含まれています。
ConfigurationFile	現在アクティブな設定ファイルの名前。
ShellCommand	コマンド シェルの名前。コマンド シェルは、実行中のオペレーティング システムのバージョンに依存します。
UserName	登録されているユーザー名。
UserOrganization	登録されているユーザーの組織名。
SerialNumber	登録されているシリアル番号。

ProgInfo レコード

ProgInfo レコードは、実行中の Source Insight のバージョンを指定します。次のフィールドがあります。

フィールド	説明
ProgramName	プログラムの名前 (例 : 「Source Insight」)
versionMajor	メジャー バージョン番号。バージョン番号が 1.02.0003 の場合、1 がメジャー バージョンです。
versionMinor	マイナー バージョン番号。バージョン番号が 1.02.0003 の場合、2 がマイナー バージョンです。
versionBuild	バージョンのビルド番号。バージョン番号が 1.02.0003 の場合、3 がビルド番号です。
CopyrightMsg	Source Dynamics の著作権情報。
fTrialVersion	プログラムが評価版の場合は True。
fBetaVersion	プログラムがベータ版の場合は True。
ExeFileName	実行ファイルの名前。
cchLineMax	ソース ファイル行で許可されている最大文字数。
cchPathMax	パス名でサポートされている最大文字数。

フィールド	説明
cchSymbolMax	シンボルのフル ネームで許可されている最大文字数。
cchCmdMax	コマンド、カスタム コマンド、またはマクロ コマンドの名前で許可されている最大文字数。
cchBookmarkMax	ブックマークの名前で許可されている最大文字数。
cchInputMax	ダイアログ ボックスのテキスト入力フィールドで許可されてい最大文字数。
cchMacroStringMax	マクロ文字列変数で許可されている最大文字数。
lnMax	ソース ファイルでサポートされている最大行数。
integerMax	サポートされている最大整数値。
integerMin	サポートされている最小整数値 (負の数)。

Rect レコード

Rect レコードは、矩形の座標を指定します。Rect レコードには次のフィールドがあります。

フィールド	説明
Left	矩形の左座標のピクセル。
Top	矩形の上座標のピクセル。
Right	矩形の右座標のピクセル。
Bottom	矩形の下座標のピクセル。

Selection レコード

Selection レコードは、ウィンドウ内のテキスト選択の状態を指定します。Selection レコードには次のフィールドがあります。

フィールド	説明
lnFirst	最初の行番号
ichFirst	lnFirst 行の最初の文字のインデックス
lnLast	最後の行番号
ichLim	lnLast 行の最後の文字の限界インデックス (最後の次)

フィールド	説明
fExtended	複数の文字を含めるように選択が拡張されている場合は TRUE。選択が単純な挿入ポイントの場合は FALSE。 これは、次の式と同じです。 (sel.fRect || sel.lnFirst != sel.lnLast || sel.ichFirst != sel.ichLim)
fRect	選択が矩形 (ブロック形式) の場合は TRUE。 選択が文字列の線形範囲の場合は FALSE。
次のフィールドは、fRect が TRUE の場合のみ適用されます。	
xLeft	ウィンドウ座標における矩形の左のピクセルの位置。
xRight	ウィンドウ座標における矩形の右のピクセルの位置。

Symbol レコード

Symbol レコードはシンボル宣言です。シンボルの場所と種類を指定します。プロジェクト内または開いているファイル バッファ内にあるシンボルを一意に指定するのに使用します。

Symbol レコードは、いくつかの関数によって返されます。また、いくつかの関数への入力として使用されます。Symbol レコードには次のフィールドがあります。

フィールド	説明
Symbol	シンボルのフル ネーム。シンボル名は、実際にはパスです。各シンボル名は、ドット (.) 文字で区切られたパスのコンポーネントに分類されます。たとえば、「myclass.member1」というシンボル名では、「member1」は「myclass」に含まれています。
Type	シンボルの種類 (例 :「関数」、「クラス」など)
Project	シンボルが見つかったプロジェクトのフル パス。
File	シンボルが見つかったファイルのフル パス。
lnFirst	シンボル宣言の最初の行番号
lnLim	シンボル宣言の限界行番号
lnName	宣言内のシンボル名が記述されている行番号
ichName	宣言内の lnName 行にあるシンボル名の文字インデックス。
Instance	File 内のシンボルのインスタンス番号パス。たとえば、シンボルの最初の記述はインスタンス 0、2 番目はインスタンス 1、というようになります。

SYSTIME レコード

SYSTIME レコードはシステム時間を指定します。GetSysTime 関数によって返されます。

フィールド	説明
time	文字列形式の時刻。
date	文字列形式の曜日、日、月、年。
Year	現在の年。
Month	現在の月 (数値)。1 月は 1 です。
DayOfWeek	曜日 (数値)。日曜日は 0、月曜日は 1 というようになります。
Day	日
Hour	現在の時間。
Minute	現在の分。
Second	現在の秒。
Milliseconds	現在のミリ秒。

内部マクロ関数

Source Insight では、多くの内部マクロ関数が用意されています。これらは、文字列、ファイル バッファ、ウィンドウ、プロジェクト、およびシンボル情報を操作するための関数です。

内部マクロ関数の呼び出し構文は、ユーザー定義マクロ関数の呼び出しと同じです。例 :

```
hbuf = GetCurrentBuf()   // call GetCurrentBuf function
```

文字列関数

文字列関数は、文字列の操作のために提供されています。C とは異なり、文字列のメモリ管理や文字列を格納するためのバッファの宣言は必要ありません。

AsciiFromChar (ch)

指定された文字 ch の ASCII 値を返します。ch が複数の文字を含む文字列の場合、最初の文字のみテストされます。

cat (a, b)

文字列 a と b を連結して返します。

CharFromAscii (ascii_code)

ascii_code の ASCII コードに対応する単一文字を含む文字列を返します。

islower (ch)

指定された文字 ch が小文字の場合、TRUE を返します。ch が複数の文字を含む文字列の場合、最初の文字のみテストされます。

IsNumber (s)

文字列 s が数字列の場合、TRUE を返します。非数字列を演算式で使用すると、ランタイム エラーになります。

isupper (ch)

指定された文字 ch が大文字の場合、TRUE を返します。ch が複数の文字を含む文字列の場合、最初の文字のみテストされます。

strlen (s)

文字長を返します。

strmid (s, ichFirst, ichLim)

ichFirst から ichLim までの範囲 (ichLim は含まれない) の s の中間文字列を返します。つまり、s[ichFirst] から s[ichLim - 1] までです。ichFirst と ichLim が同じ場合、空の文字列範囲が指定されます。

strtrunc (s, cch)

文字列 s を cch 文字に切り捨てたものを返します。

tolower (s)

指定された文字列を小文字に変換したものを返します。

toupper (s)

指定された文字列を大文字に変換したものを返します。

ユーザー入力および出力関数

ユーザー入力および出力関数は、ユーザーからの入力の取得、またはメッセージ ウィンドウへの出力の表示を行います。

Ask (prompt_string)

メッセージ ボックス ウィンドウに文字列 prompt_string を表示します。Ask メッセージ ボックスには、[OK] ボタンと [Cancel] ボタンがあります。[Cancel] ボタンをクリックすると、マクロが停止します。

AssignKeyToCmd(key_value, cmd_name)

cmd_name によって指定されたコマンドに key_value を割り当てます。以降、key_value を押すとコマンドが呼び出されます。

key_value は、GetKey および KeyFromChar によって返されるキーボードのキーの数値です。CharFromKey を使用して、key_value を文字に変換できます。

cmd_name は、コマンド名の文字列です。

例 :

```
key = GetKey();
AssignKeyToCmd(key, "Open Project");
```

Beep ()

ビープ音を 1 回鳴らします。

CharFromKey (key_code)

指定されたキー コードに関連付けられている文字を返します。key_code が標準文字のキー コードでない場合、0 を返します。

key_code は、GetKey および KeyFromChar によって返されるキーボードのキーの数値です。CharFromKey を使用して、key_code を文字に変換できます。

CmdFromKey(key_value)

現在 key_value に割り当てられている Source Insight コマンド名の文字列を返します。これは、key_value を押したときに呼び出されるコマンドの名前です。

key_value は、GetKey および KeyFromChar によって返されるキーボードのキーの数値です。

CharFromKey を使用して、key_value を文字に変換できます。

例 :

```
key = GetKey();
cmd_name = CmdFromKey(key);
msg("That key will invoke the @cmd@ command.");
```

EndMsg ()

StartMsg によって開始されたメッセージ ボックスを閉じます。

FuncFromKey (key_code)

ファンクション キー コードから、ファンクション キーの数字 (F1 - F12 に対応する 1 - 12) を返します。key_code がファンクション キー コードでない場合、0 を返します。

key_code は、GetKey および KeyFromChar によって返されるキーボードのキーの数値です。CharFromKey を使用して、key_code を文字に変換できます。

GetChar ()

キーが押されるのを待機し、単一文字を返します。

GetKey ()

キーが押されるのを待機し、キー コードを返します。キー コードは、Source Insight が各キーに関連付けている特別な数値です。CharFromKey を使用して、文字にキー コードを割り当てることができます。

GetSysTime(fLocalTime)

日時を表す文字列が格納されている SYSTIME レコードを返します。355 ページの「SYSTIME レコード」も参照してください。

fLocalTime が非ゼロの場合、ローカルの時間が返されます。そうでない場合、システム時間 (UTC 形式) が返されます。

IsAltKeyDown (key_code)

key_code に対して [ALT] キーが押されている場合、TRUE を返します。key codes には、[CTRL] キーおよび [ALT] キーの状態が含まれています。

key_code は、GetKey および KeyFromChar によって返されるキーボードのキーの数値です。CharFromKey を使用して、key_code を文字に変換できます。

IsCtrlKeyDown (key_code)

key_code に対して [CTRL] キーが押されている場合、TRUE を返します。key codes には、[CTRL] キーおよび [ALT] キーの状態が含まれています。

key_code は、GetKey および KeyFromChar によって返されるキーボードのキーの数値です。CharFromKey を使用して、key_code を文字に変換できます。

IsFuncKey (key_code)

key_code がファンクション キーの場合、TRUE を返します。そうでない場合、FALSE を返します。

key_code は、GetKey および KeyFromChar によって返されるキーボードのキーの数値です。CharFromKey を使用して、key_code を文字に変換できます。

KeyFromChar(char, fCtrl, fShift, fAlt)

指定された文字と修飾子のキーの状態で、キー値を返します。キー値は、GetKey によって返されるキーボードのキーの数値です。CharFromKey を使用して、キー値を文字に変換できます。

入力 :

char - キーストロークの文字部分。大文字と小文字は区別されません。

fCtrl - [CTRL] キーが含まれる場合は非ゼロ。

fShift - [Shift] キーが含まれる場合は非ゼロ。

fAlt - [ALT] キーが含まれる場合は非ゼロ。

char パラメータはいくつかの特殊な値を保持することができます。

表 6.6:「char」パラメータ値と意味

Char 値	意味
a-z	アルファベット文字
Fx	ファンクション キーの数字 x (例 : F10)
Nx	テンキーボードの文字 x (例 : "+" キーの場合 N+)

表 6.6:「char」パラメータ値と意味

Char 値	意味
Up、Down、Left、Right	矢印キー
Page Up、Page Down Insert、Delete Home、End、 Tab、Enter	その他の特殊キー

キーの割り当て例:

```
// assign Ctrl+C to Page Down command:
key = KeyFromChar("c", 1, 0, 0);  // Ctrl+C
AssignKeyToCmd(key, "Page Down");


// assign F9 to Close Project command:
key = KeyFromChar("F9", 0, 0, 1);  // Alt+F9
AssignKeyToCmd(key, "Close Project");
```

入力関数例:

```
// input a keypress and decode it
key = GetKey()
if (IsFuncKey(key))
    Msg cat("Function key = ", FuncFromKey(key))
if (IsAltKeyDown(key))
        Msg "Alt key down"
if (IsCtrlKeyDown(key))
    Msg "Ctrl key down"
ch = CharFromKey(key)
if (Ascii(ch) == 13)
    Msg "You pressed Enter!"
else if (toupper(ch) == 'S')
    Search_Files
...
```

Msg (s)

文字 s を表示するメッセージ ウィンドウを表示します。メッセージ ボックスには、[Cancel] ボタンがあり、マクロを停止することができます。

StartMsg (s)

文字 s を表示するメッセージ ウィンドウを表示します。メッセージ ボックスには、[Cancel] ボタンがあり、マクロを停止することができます。メッセージ ウィンドウは返った後でも表示されたままです。

バッファ リスト関数

バッファ リストは、ファイル バッファ ハンドルの集まりです。Source Insight アプリケーションには、バッファ リストは 1 つしかありません。このリストには、開いているすべてのソース ファイルのファイル バッファ ハンドルが含まれます。バッファ リスト関数を使用して、すべてのバッファを列挙できます。

BufListCount ()

この関数は、バッファ リストのバッファ数を返します。BufListItem を使用して、特定のインデックス位置のバッファ ハンドルにアクセスすることができます。

BufListItem (index)

この関数は、指定されたインデックスのバッファ ハンドルを返します。バッファ リストのサイズは、BufListCount によって返されます。インデックス値は、ゼロから BufListCount で返される値よりも 1 つ少ない数までの値です。

次の例は、開いているバッファ ハンドルを列挙します。

```
cbuf = BufListCount()
ibuf = 0
while (ibuf < cbuf)
    {
    hbuf = BufListItem(ibuf)
    // … do something with buffer hbuf
    ibuf = ibuf + 1
    }
```

ファイル バッファ関数

ファイル バッファ関数は、ファイル バッファとファイル バッファ内のテキストの作成と操作に使用されます。ファイル バッファとは、ロードされたテキスト ファイルのイメージです。ファイル バッファは、ユーザーにより編集され、その後、[Save] コマンドでディスクに保存されます。

ファイル バッファ関数の多くは、バッファ ハンドル (hbuf) を使用します。これらのハンドルは、ファイル バッファを開くものです。hbuf は、通常小さな整数値です。hbuf 値が hNil (ゼロ) の場合はエラーを示します。

AppendBufLine (hbuf, s)

ファイル バッファ hbuf に新しいテキスト行 s を追加します。

ClearBuf (hbuf)

バッファ hbuf を空にして、行が含まれないようにします。

CloseBuf (hbuf)

ファイル バッファを閉じます。hbuf はバッファ ハンドルです。

CopyBufLine (hbuf, ln)

ファイル バッファ hbuf から ln 行をクリップボードにコピーします。

DelBufLine (hbuf, ln)

ファイル バッファ hbuf から ln 行を削除します。

GetBufHandle (filename)

filename という名前の開いているファイル バッファのハンドルを返します。この関数は、開いているすべてのファイル バッファを検索し、指定された filename パラメータに一致するものを検出します。GetBufHandle は、指定されたファイル名のバッファが見つからない場合、hNil を返します。

GetBufLine (hbuf, ln)

指定されたファイル バッファ hbuf の ln 行のテキストを返します。

GetBufLineCount (hbuf)

ファイル バッファのテキスト行数を返します。hbuf はファイル バッファ ハンドルです。

GetBufLineLength (hbuf, ln)

指定されたファイル バッファ hbuf の ln 行の文字数を返します。

GetBufLnCur (hbuf)

ファイル バッファ hbuf 内でユーザーにより選択された現在の行数を返します。指定されたファイル バッファがソース ファイル ウィンドウに表示されていない場合は、マクロ エラーが生成されます。

GetBufName (hbuf)

ファイル バッファに関連付けられているファイル名を返します。hbuf はファイル バッファ ハンドルです。

GetBufProps (hbuf)

Bufprop レコードを返します。このレコードには、指定されたバッファのプロパティが含まれています。350 ページの「Bufprop レコード」も参照してください。

GetBufSelText (hbuf)

ファイル バッファ hbuf で選択されている文字を文字列として返します。1 つのテキスト行の最大文字列を返します。選択された単語のテキストを取得するのに役立ちます。指定されたファイル バッファがソース ファイル ウィンドウに表示されていない場合は、マクロ エラーが生成されます。

GetCurrentBuf ()

現在のバッファへのハンドルを返します。現在のバッファは、最前面のソース ファイル ウィンドウに表示されているファイル バッファです。現在のバッファがない場合 (例 : 開いているソース ファイル ウィンドウがない)、hNil を返します。

InsBufLine (hbuf, ln, s)

ファイル バッファ hbuf の行番号 ln に新しいテキスト行 s を挿入します。

IsBufDirty (hbuf)

バッファがダーティな場合は、True を返します。ダーティなバッファとは、開かれた後や保存されて後で、編集されたバッファのことです。ダーティ バッファには、保存されていない変更が含まれています。

IsBufRW (hbuf)

指定されたバッファが読み取り / 書き込み可能な場合は、True を返します。この関数は、バッファが読み取り専用の場合、False を返します。

MakeBufClip (hbuf, fClip)

fClip が True の場合、ファイル バッファ hbuf は Clip バッファになります。以降、バッファは通常のクリップとして、クリップ ウィンドウに表示されます。

fClip が False の場合、バッファは通常の、非クリップ ファイル バッファになります。

クリップ バッファは、Source Insight が存在する場合は、Source Insight プログラム ディレクトリの Clips サブディレクトリに自動的に保存されます。

NewBuf (name)

新しい空のファイル バッファを作成し、ファイル バッファ (hbuf) にハンドルを返します。新しいバッファの名前は、name パラメータで指定されます。NewBuf は、エラーのためにバッファが作成できなかった場合、hNil を返します。

OpenBuf (filename)

filename という名前のファイルをファイル バッファに開き、ファイル バッファにハンドルを返します。OpenBuf は、ファイルを開くことができなかった場合に hNil を返します。

OpenMiscFile (filename)

filename という名前のファイルを開きます。開かれたファイルの種類に応じて、動作は異なります。たとえば、.CF3 拡張子のファイルが開かれた場合は、新しい設定ファイルをロードします。プロジェクト ファイル (.PR 拡張子) が開かれた場合は、プロジェクトを開きます。OpenMiscFile は、成功した場合は TRUE を失敗した場合は FALSE を返します。

PasteBufLine (hbuf, ln)

クリップボードの内容をファイル バッファ hbuf の ln 行の直前にコピーします。

PrintBuf (hbuf, fUseDialogBox)

指定されたファイル バッファをプリンタに出力します。fUseDialogBox が True の場合は、[Print] ダイアログ ボックスが最初に表示されます。そうでない場合は、ユーザーが操作しなくてもデフォルトのプリンタに出力されます。

PutBufLine (hbuf, ln, s)

ファイル バッファ hbuf の行番号 ln のテキスト行を文字列 s に置換します。

RenameBuf (hbuf, szNewName)

指定されたバッファの名前を szNewName に変更します。ディスク上のファイルの名前も変更されます。バッファが開いているプロジェクトのメンバの場合、プロジェクト内でファイル名が変更されます。

SaveBuf (hbuf)

ファイル バッファをディスクに保存します。hbuf はバッファ ハンドルです。

SaveBufAs (hbuf, filename)

ファイル バッファを異なるファイル名で保存します。hbuf はバッファ ハンドルです。filename は、新しいファイルの名前です。

SetBufDirty (hbuf, fDirty)

指定されたバッファのダーティ状態を fDirty に設定します。ダーティなバッファとは、開かれた後や保存されて後で、編集されたバッファのことです。ダーティ バッファには、保存されていない変更が含まれています。

ユーザーがダーティなバッファを閉じる際、Source Insight はファイルを保存しようとします。この関数を使用して、バッファのダーティ状態を解除し、保存を確認するメッセージが表示されないようにできます。

SetBufIns (hbuf, ln, ich)

カーソル位置の挿入ポイントをファイル バッファ hbuf の文字インデックス ich にある行番号 ln に設定します。指定されたファイル バッファがソース ファイル ウィンドウに表示されていない場合は、マクロ エラーが生成されます。

SetBufSelText (hbuf, s)

ファイル バッファ hbuf で現在選択されている文字を文字列 s に置換します。選択した単語のテキストを置換するのに役立ちます。指定されたファイル バッファがソース ファイル ウィンドウに表示されていない場合は、マクロ エラーが生成されます。

これは、バッファに新しいテキストを挿入する最も簡単な方法です。たとえば、このコードは「new text」を現在の挿入位置にある現在のバッファに挿入します。

```
hbuf = GetCurrentBuf()
SetBufSelText(hbuf, "new text")
```

SetCurrentBuf (hbuf)

アクティブなファイル バッファをハンドルが hbuf のバッファに設定します。現在のバッファは、最前面のソース ファイル ウィンドウに表示されているファイル バッファです。

環境関数とプロセス関数

環境関数とプロセス関数で、レジストリと環境値の取得と設定が行えます。また、内部コマンドと外部コマンドを実行することができます。

GetEnv (env_name)

env_name で指定された環境変数の値を返します。環境変数がない場合は、空の文字列を返します。

GetReg (reg_key_name)

reg_key_name という名前のレジストリ キーに関連付けられた値を返します。キー値は、HKEY_CURRENT_USER/Software/Source Dynamics/Source Insight/2.0 キー パス配下に格納されます。キーが存在しない場合は、空の文字列が返されます。

IsCmdEnabled (cmd_name)

cmd_name で指定されたコマンドが現在有効な場合は、TRUE を返します。プログラムの状態により Source Insight がコマンドを実行できない場合は、コマンドは有効になりません。たとえば、[Save] コマンドは、開いているソース ファイル ウィンドウがない場合は、有効になりません。

PutEnv (env_name, value)

env_name という名前の環境変数を value に設定します。

RunCmd (cmd_name)

cmd_name で指定されたコマンドを実行します。これにより、[Custom Commands] コマンドで定義されたカスタム コマンドなどの特殊なコマンドを実行することができます。

RunCmdLine (sCmdLine, sWorkingDirectory, fWait)

sCmdLine で指定されたコマンド行文字列を起動します。成功した場合は非ゼロを、エラーの場合はゼロを返します。

sWorkingDirectory が Nil でない場合、作業ディレクトリが使用されます。sWorkingDirectory が Nil の場合、現在のプロジェクトのホーム ディレクトリが使用されます。

fWait が非ゼロの場合、プロセスが終了するまで関数は値を返しません。そうでない場合は、すぐに値を返します。プロセスが別のウィンドウ アプリケーションの場合は、fWait の状態に関わらず、すぐに値を返します。

SetReg (reg_key_name, value)

reg_key_name という名前のレジストリ キーに関連付けられている値を設定します。キー値は、HKEY_CURRENT_USER/Software/Source Dynamics/Source Insight/2.0 キー パス下に格納されます。キーが存在しない場合は、作成されます。

SetReg 関数と GetReg 関数は、セッション間の独自の Source Insight 関連情報を格納する方法を提供します。

ShellExecute (sVerb, sFile, sExtraParams, sWorkingDirectory, windowstate)

指定されたファイルで "ShellExecute" 関数を実行します。これにより、Windows シェルに指定ファイルでの操作を実行するように指示することができます。

ShellExecute を使用する場合、特定の種類のファイルを処理するために登録されているアプリケーションを知っておく必要はありません。技術的なバックグラウンド情報については、Windows シェル API ドキュメントで「ShellExecute」関数を参照してください。

ShellExecute パラメータ

表 6.7:ShellExecute パラメータ

パラメータ	意味
sVerb	実行する動作を指定する単一の単語。値の例については下の表を参照してください。
sFile	filespec パラメータには有効なパスを指定します。スペースを含むパス名は二重引用符で囲んでください。実行ファイルの名前も指定できます。
sExtraParams	オプション パラメータ：最終的に実行されるアプリケーションに渡すパラメータ を指定します。フォーマットは起動する verb、および実行するアプリケーションによって決定されます。
sWorkingDir	コマンド実行時の作業ディレクトリ。空の場合は、プロジェクトのホーム ディレクトリが使用されます。
windowstate	開くウィンドウのサイズと状態を指定する整数。有効な値は 1 = 標準、2 = 最小化、3 = 最大化です。

sVerb 値

sVerb パラメータは、Shell Execute による動作を指定する単一の単語の文字列です。

表 6.8:sVerb パラメータの値

sVerb 値	意味
`edit`	ファイルのエディタを開きます。
`explore`	指定されたファイルを探索します。
`open`	指定されたファイルを開きます。実行形式ファイルまたはドキュメント ファイルを指定できます。フォルダも指定できます。
`print`	指定されたドキュメント ファイルを印刷します。filespec がドキュメント ファイルでない場合、関数は動作しません。
`properties`	ファイルやファイルのプロパティを表示します。
`find`	[スタート] - [検索] - [ファイルやフォルダ] を起動します。
"" (空の文字列)	このパラメータを ShellExecute に送ります。

例 :

Web サイトをブラウズする

```
ShellExecute("open", "http://www.somedomain.com", "", 1)
```

ドキュメント ファイルを開く

```
ShellExecute("open", "somefile.doc", "", 1)
```

Windows のドキュメント ファイル フォルダを表示する

```
ShellExecute("explore", "C :\Documents and Settings", "", 1)
```

Internet Explorer を起動する

```
ShellExecute("", "iexplore", "", 1)
```

Internet Explorer でファイルを表示する

```
ShellExecute("", "iexplore somefile.htm", "", 1)
```

カレント プロジェクト フォルダのファイルを検索する

```
ShellExecute("find",  filespec, "", 1)
```

ウィンドウ リスト関数

ウィンドウ リストとは、ソース ファイル ウィンドウ ハンドルの集まりです。Source Insight アプリケーションには、ウィンドウ リストは 1 つしかありません。このリストには、すべてのソース ウィンドウ ハンドルが含まれます。ウィンドウ リストを使用して、すべてのソース ファイル ウィンドウを列挙します。

WndListCount ()

この関数は、ウィンドウ リストのウィンドウ数を返します。WndListItem を使用して、リストの特定のインデックスのウィンドウ ハンドルにアクセスすることができます。

WndListItem (index)

この関数は、指定されたインデックスのウィンドウ ハンドルを返します。ウィンドウ リストのサイズは、WndListCount によって返されます。インデックス値は、ゼロから WndListCount で返される値よりも 1 つ少ない数までの値です。

次の例は、開いているウィンドウ ハンドルを列挙します。

```
cwnd = WndListCount()
iwnd = 0
while (iwnd < cwnd)
    {
    hwnd = WndListItem(iwnd)
    // … do something with window hwnd
    iwnd = iwnd + 1
    }
```

ウィンドウ関数

ウィンドウ関数で、ソース ファイル ウィンドウの操作を行うことができます。ファイル バッファがソース ウィンドウに表示されます。hwnd は、通常小さな整数値です。hNil の hwnd は、エラーを表示します。

ウィンドウ関数は、特殊なマクロレベルの hwnd パラメータを使用します。

関数は、ウィンドウ ハンドル (hwnd) パラメータを使用します。これらは、ソース ファイル ウィンドウを開くマクロレベルのハンドルです。hwnd マクロは Microsoft Windows API のウィンドウ ハンドル HWND と概念的には似ていますが、厳密には同じではありません。

各ソース ウィンドウには、選択された文字を説明する選択範囲があります。詳細は、373 ページの「GetWndSel (hwnd)」を参照してください。

hwnd パラメータは、システム レベル ウィンドウも表現することができます。

一部の関数では、ウィンドウ ハンドル (hwnd) は、アプリケーション ウィンドウなどのシステム レベル ウィンドウへのハンドルも表現することができます。システム レベル ウィンドウには、ファイル バッファや選択範囲は含まれません。GetApplicationWnd 関数は、Source Insight アプリケーション ウィンドウにハンドルを返します。

CloseWnd (hwnd)

ウィンドウ hwnd を閉じます。ウィンドウを閉じても、ウィンドウに表示されているファイル バッファは閉じられません。

GetApplicationWnd ()

Source Insight アプリケーション ウィンドウへのウィンドウ ハンドルを返します。

メモ： これは、システム レベル ウィンドウ ハンドルとは同じではなく、Source Insight のマクロレベル ハンドル 値です。

返されたハンドルは、GetWndDim や IsWndMax など、ファイル バッファや選択範囲を想定しない関数に渡すことができます。

GetCurrentWnd ()

アクティブな最前面のソース ファイル ウィンドウのハンドルを返すか、ウィンドウが開いていない場合は hNil を返します。

GetNextWnd (hwnd)

hwnd の後にある Z 順序での次のウィンドウのウィンドウ ハンドルを返します。通常は、アクティブだった前のウィンドウです。GetNextWnd は、ほかのウィンドウが開いていない場合は hNil を返します。

たとえば、hwnd が最前面のウィンドウの場合、GetNextWnd (hwnd) はその下の次のウィンドウを返します。SetCurrentWnd を使用して、最前面のアクティブなウィンドウを設定している場合は、後続の GetNextWnd への呼び出しが影響を受けることに注意してください。

GetWndBuf (hwnd)

hwnd ウィンドウに表示されているファイル バッファのハンドルを返します。

GetWndClientRect (hwnd)

Rect レコードを返します。指定されたウィンドウのクライアント領域を示す矩形が含まれています。座標は、ウィンドウのローカルの座標系で示されます。クライアント領域の矩形には、ウィンドウのフレームやその他の非クライアント領域は含まれません。353 ページの「Rect レコード」も参照してください。

GetWndDim (hwnd)

DIM レコードを返します。指定された hwnd ウィンドウのピクセル サイズを説明します。351 ページの「DIM レコード」も参照してください。

返される横サイズは、ウィンドウのテキスト領域のみの幅です。左余白やソース ウィンドウ付属のシンボル ウィンドウは含まれません。

GetWndHandle (hbuf)

hbuf によって指定されたファイル バッファを表示する最前面ウィンドウのウィンドウ ハンドルを返します。GetWndHandle は、ファイル バッファがウィンドウにない場合は hNil を返します。

ファイル バッファは、複数のウィンドウに表示されることもあるため、GetWndHandle は前面から背面へすべてのウィンドウを検索しま

す。そのため、指定されたファイル バッファがアクティブなウィンドウの現在のバッファの場合は、そのウィンドウのハンドルが常に返されます。

GetWndHorizScroll (hwnd)

hwnd ウィンドウの水平スクロールの状態を返します。水平スクロールの状態は、スクロールのピクセル数です。

GetWndLineCount (hwnd)

hwnd ウィンドウの垂直サイズを行で返します。これは、ウィンドウで潜在的に表示可能な最大行数です。ファイル バッファがウィンドウ全体を占めていない場合でも、GetWndLineCount は最大行数を返します。

GetWndLineWidth (hwnd, ln, cch)

指定されたウィンドウで指定されたテキスト行の幅を返します。

入力 :

パラメータ	説明
hwnd	ウィンドウ
ln	測定されるテキストを含む行番号。ln が範囲外の場合は、–1 が返されます。
cch	行で測定される文字数。cch が –1 に設定されている場合は、行全体の長さが測定されます。

この関数で指定されたウィンドウの文字幅を測定することができます。それぞれのウィンドウで使用されているフォントは、ファイルのドキュメントの種類で決定されるため、テキスト幅はウィンドウによって異なります。構文フォーマットもテキスト幅に影響します。

この関数は、特定の文字を表示するためにウィンドウを スクロールする ScrollWndHoriz とともに使用できます。

例

行 100 の行全体の幅を測定するには :

```
dim = GetWndLineWidth(hwnd, 100, -1)
Msg ("Line 100 is " # dim.cxp # " pixels wide.")
```

行 200 の最初の 3 文字の幅を測定するには :

```
Dim = GetWndLineWidth(hwnd, 200, 3)
```

GetWndParent (hwnd)

ウィンドウの親ウィンドウへのハンドルを返します。親がない場合は hNil を返します。

GetWndRect (hwnd)

指定されたウィンドウの画面の矩形座標を含む Rect レコードを返します。矩形には、ウィンドウ フレームと非クライアント領域が含まれます。353 ページの「Rect レコード」も参照してください。

GetWndSel (hwnd)

hwnd によって指定されたウィンドウの選択の状態を返します。選択の状態は、Selection レコードに返されます。353 ページの「Selection レコード」も参照してください。

GetWndSelIchFirst (hwnd)

hwnd ウィンドウの選択範囲にある最初の文字のインデックスを返します。

GetWndSelIchLim (hwnd)

hwnd ウィンドウの選択範囲にある最後の文字の 1 つ後のインデックスを返します。

GetWndSelLnFirst (hwnd)

hwnd ウィンドウの選択範囲の最初の行番号を返します。

GetWndSelLnLast (hwnd)

hwnd ウィンドウの選択範囲の最後の行番号を返します。

GetWndVertScroll (hwnd)

hwnd ウィンドウの垂直スクロールの状態を返します。垂直スクロールの状態とは、ウィンドウの上部に表示される行番号です。

IchFromXpos (hwnd, ln, xp)

指定されたウィンドウの行番号 (ln) にあるピクセルの x 位置 (xp) の文字インデックスを返します。文字インデックスは、指定された行のゼロベースの文字のインデックスです。この関数が呼ばれた時点で行が

実際に表示されている必要はありません。376 ページの「XposFromIch (hwnd, ln, ich)」も参照してください。

入力 :

パラメータ	説明
hwnd	ウィンドウ
ln	測定されるテキストを含む行番号。ln が範囲外の場合は、–1 が返されます。
xp	ウィンドウ全体の左端に対して相対的な x 位置。xp が行の幅を超える場合は、行の文字総数が返されます。

メモ : XposFromIch 関数を使用して、逆マッピングを実行することができます。

IsWndMax (hwnd)

hwnd ウィンドウが現在最大化されている場合は、TRUE を返します。

IsWndMin (hwnd)

hwnd ウィンドウが現在最小化されている場合は、TRUE を返します。

IsWndRestored (hwnd)

hwnd ウィンドウが現在最大化も最小化もされていない場合は、TRUE を返します。

MaximizeWnd (hwnd)

hwnd によって指定されたウィンドウを最大化 (または「拡大」) します。

MinimizeWnd (hwnd)

hwnd によって指定されたウィンドウを最小化 (または「アイコン化」) します。

NewWnd (hbuf)

新規ウィンドウを作成し、ウィンドウに hbuf ファイル バッファを表示します。NewWnd はウィンドウ ハンドルを返すか、またはエラーの場合は hNil を返します。

ScrollWndHoriz (hwnd, pixel_count)

hwnd ウィンドウを pixel_count で指定された分だけ水平にスクロールします。

pixel_count が 0 よりも少ない場合は、行の逆方向にスクロールします (画面の内容は、右にスクロールします)。

pixel_count が 0 よりも大きい場合は、行方向にスクロールします (画面の内容は、左にスクロールします)。

ScrollWndToLine (hwnd, ln)

hwnd ウィンドウをスクロールして、行番号 ln をウィンドウ上部に表示します。

ScrollWndVert (hwnd, line_count)

hwnd ウィンドウを line_count で指定された分だけ垂直にスクロールします。

line_count が 0 よりも少ない場合は、ファイルの逆方向にスクロールします (画面の内容は下にスクロールします)。

line_count が 0 よりも少ない場合は、ファイル方向にスクロールします (画面の内容は上にスクロールします)。

SetCurrentWnd (hwnd)

最前面のアクティブ ウィンドウを設定します。hwnd はウィンドウ ハンドルをアクティブにします。

SetWndRect (hwnd, left, top, right, bottom)

指定されたウィンドウの新しい位置を設定します。深度 (Z-order) には影響しません。指定された座標は、ウィンドウの親ウィンドウのローカルのピクセル座標系です。ウィンドウがアプリケーション ウィンドウの場合は、座標系はグローバル画面のピクセル座標系です。

SetWndSel (hwnd, selection_record)

hwnd で指定されたウィンドウの選択範囲の状態を selection_record で指定された Selectn レコードに設定します。373 ページの「GetWndSel (hwnd)」も参照してください。353 ページの「Selection レコード」を参照してください。

ToggleWndMax (hwnd)

最大化サイズとリストア サイズの hwnd ウィンドウを切り替えます。

XposFromIch (hwnd, ln, ich)

指定されたウィンドウの行番号 (ln) にある文字位置 (ich) のピクセルの x 位置番号を返します。x 位置はウィンドウ全体の左端に対して相対的です。この関数が呼ばれた時点で行が実際に表示されている必要はありません。373 ページの「IchFromXpos (hwnd, ln, xp)」も参照してください。

入力 :

パラメータ	説明
hwnd	ウィンドウ
ln	測定されるテキストを含む行番号。ln が範囲外の場合は、–1 が返されます。
ich	指定された行のゼロベースの文字のインデックスである文字インデックス。ich が行の文字数を超える場合は、行末の x 位置が返されます。

メモ : IchFromXpos 関数を使用して、逆マッピングを実行することができます。

Bookmark 関数

すべてのブックマークは、単一のブックマーク リストに保持されます。ブックマーク関数を使用して、すべてのブックマークを列挙し、ブックマークの追加と削除を行うことができます。ブックマーク リストは、ワークスペース ファイルに保持されます。

BookmarksAdd (name, filename, ln, ich)

新しいブックマークを追加します。新しいブックマーク名は name で指定されます。 ブックマーク位置は、filename ファイルの行番号 ln、文字インデックス ich です。

成功した場合は True、エラーの場合は False を返します。

BookmarksCount ()

この関数は、ブックマーク リストのブックマーク数を返します。BookmarksItem を使用して、リストの特定のインデックスのブックマークにアクセスすることができます。

BookmarksDelete (name)

name という名前のブック名を削除します。

BookmarksItem (index)

この関数は、指定されたインデックスのブックマークを返します。ブックマーク リストのサイズは BookmarksCount によって返されます。インデックス値は、ゼロから BookmarksCount で返される値よりも 1 つ少ない数までの値です。

この例では、すべてのブックマークを列挙します。

```
cmark = BookmarksCount()
imark = 0
while (imark < cmark)
    {
    bookmark = BookmarksItem(imark)
    // … do something with bookmark
    imark = imark + 1
    }
```

350 ページの「Bookmark レコード」も参照してください。

BookmarksLookupLine (filename, ln)

指定された位置のブックマークを検索します。ファイルは filename で、行番号は ln です。

Bookmark レコードを返すか、またはブックマークが見つからない場合は nil を返します。350 ページの「Bookmark レコード」も参照してください。

BookmarksLookupName (name)

name という名前のブックマークを検索します。

Bookmark レコードを返すか、またはブックマークが見つからない場合は nil を返します。350 ページの「Bookmark レコード」も参照してください。

シンボル リスト関数

シンボル リストはゼロベースでインデックスされた Symbol レコードの集まりです。354 ページの「Symbol レコード」も参照してください。

一部のシンボル アクセス関数はシンボル リスト ハンドルを返します。

シンボル リストは、割り当てリソースです。アクセスが終了したら SymListFree で解放する必要があります。Source Insight はマクロ終了時に自動で動的に割り当てられたリソースをクリーン アップします。ただし、使用されていないハンドラを解放せずに多くのハンドラを割り

当てた場合には、Source Insight でリソースを使い果たしてしまう可能性があります。

SymListCount ()

この関数は、シンボル リストのシンボル数を返します。SymListItem を使用して、リストの特定のインデックスのシンボル レコードにアクセスすることができます。354 ページの「Symbol レコード」も参照してください。

SymListFree (hsyml)

この関数は、指定されたシンボル リストの割り当てを解除します。

SymListInsert (hsyml, isym, symbolNew)

この関数は、シンボル レコードをシンボル リスト hsyml に挿入します。シンボルは、isymBefore の直前に挿入されます。isymBefore が –1 の場合、シンボルはリストの最後に追加されます。シンボル レコードは symbolNew に提供されます。354 ページの「Symbol レコード」も参照してください。

SymListItem (hsyml, isym)

この関数は、シンボル リスト hsyml のゼロベースのインデックス isym の Symbol レコードを返します。シンボル リストのサイズは SymListCount によって返されます。354 ページの「Symbol レコード」も参照してください。

インデックス値は、ゼロから SymListCount で返される値よりも 1 つ少ない数までの値です。

この例では、シンボル リストのすべてのシンボルを列挙します。

```
csym = SymListCount(hsyml)
isym = 0
while (isym < csym)
    {
    symbol = SymListItem(isym)
    // … do something with symbol
    Msg ("symbol name = " # symbol.name)
    isym = isym + 1
    }
```

SymListNew ()

新規の空のシンボル リストを割り当てます。新規のシンボル リスト ハンドルを返すか、またはエラーの場合は hNil を返します。シンボル リストでの作業が終了したら、SymListFree を呼び出す必要があります。

SymListRemove (hsyml, isym)

シンボル リスト hsyml から要素を削除します。isym にあるシンボル要素は削除されます。

シンボル関数

シンボル関数で、Source Insight のシンボル検索エンジンにアクセスすることができます。Source Insight は、プロジェクトのシンボル情報をシンボル データベースで管理します。これらのシンボル関数はシンボル データベースと Source Insight のビルトイン言語パーサを活用してソース ファイル中のシンボルを特定します。

Source Insight によるシンボル情報の管理方法および検索規則についての説明は、「プロジェクト」の章の「シンボルおよびプロジェクト」のセクションを参照してください。

Symbol レコード

Symbol レコードはシンボル宣言です。シンボルの場所と種類を指定します。プロジェクト内または開いているファイル バッファ内にあるシンボルを一意に指定するのに使用します。

Symbol レコードは、いくつかの関数によって返されます。また、いくつかの関数への入力として使用されます。354 ページの「Symbol レコード」も参照してください。

GetBufSymCount(hbuf)

hbuf バッファで宣言されているシンボル数を返します。シンボルが何も宣言されていない場合やファイルの処理を続けられない場合、またはファイルのドキュメントの種類が言語パーサを指定していない場合は、ゼロを返します。

GetBufSymLocation(hbuf, isym)

hbuf バッファの isym によってインデックスされたシンボルの Symbol レコードを返します。パースされたそれぞれのファイル バッファは定義されているシンボルのインデックスを管理します。インデックスはシンボル名で格納されます。Symbol インデックス値は、ゼロから GetBufSymCount で返される数から 1 を引いた数までの値です。この関数は、シンボル インデックス (isym) をシンボルの Symbol レコードにマップします。

354 ページの「Symbol レコード」も参照してください。

GetBufSymName(hbuf, isym)

hbuf バッファの isym によってインデックスされたシンボルの名前を返します。パースされたそれぞれのファイル バッファは定義されているシンボルのインデックスを管理します。インデックスはシンボル名で格納されます。Symbol インデックス値は、ゼロから GetBufSymCount で返される数から 1 を引いた数までの値です。この関数は、シンボル インデックス (isym) をシンボル名にマップします。

この例は、すべてのファイル バッファ シンボルを繰り返します。

```
isymMax = GetBufSymCount (hbuf)
isym = 0
while (isym < isymMax)
    {
    symname = GetBufSymName (hbuf, isym)
    ...
    isym = isym + 1
    }
```

GetCurSymbol ()

現在の選択範囲があるシンボルの名前を返します。現在の選択範囲はアクティブなウィンドウの選択範囲 (またはカーソル位置) です。GetCurSymbol は、シンボルが見つからない場合は空の文字列を返します。

GetSymbolLine (symbol_name)

symbol_name という名前のシンボルの行番号を返します。 複数のシンボルが同じ名前で定義されている場合にユーザーは適切なものを選択することができます。

GetSymbolLocation (symbol_name)

symbol_name で指定されているシンボル名の場所を返します。場所は Symbol レコードに返されます。シンボルが見つからない場合は空の文字列を返します。354 ページの「Symbol レコード」も参照してください。

この関数は、検索処理を Source Insight が Jump To Definition コマンドを使用したときにシンボルを検索するのと同じ方法で実行します。シンボルが現在のプロジェクトや開いているファイルで見つからない場合、すべてのプロジェクト シンボル パスにあるすべてのプロジェクトも検索されます。複数の宣言が symbol_name に対して見つかった場合は、ユーザーに複数の定義リストが呈示されます。

GetSymbolLocationEx を呼び出して、検索処理をより強力に制御でき、また、同じシンボル名で複数の定義を持つシンボルを特定できます。

この例では、シンボルの定義を検索し、ソース ファイルと行番号を表示します。

```
symbol = Ask("What symbol do you want to locate?")
loc = GetSymbolLocation(symbol)
if (loc == "")
    Msg (symbol # " was not found")
else
    Msg (symbol # " was found in " # loc.file #
            " at line " # loc.lnFirst)
```

ファイル名の特定

GetSymbolLocation では、ファイル名を検索することもできます。単純なファイル名を symbol_name パラメータとして指定することにより、GetSymbolLocation はプロジェクトまたはプロジェクト シンボル パスのファイルを検索し、完全修飾パスを location.file フィールドのファイルに返します。これは、単純なファイル名をフル パスに拡張するのに役立ちます。

例 :

```
loc = GetSymbolLocation("simple.c")
fullfilename = loc.file
// fullfilename could be something like "d:\proj\simple.c"
```

GetSymbolLocationEx (symbol_name, output_buffer, fMatchCase, LocateFiles, fLocateSymbols)

symbol_name で指定されたシンボルに対するすべての宣言を見つけます。各宣言の位置は、Symbol レコードとしての output_buffer バッファに行として追加されます。 fMatchCase の場合は、シンボル名の大文字小文字が完全に一致しなければなりません。fLocateFiles の場合は、プロジェクトまたはプロジェクトのシンボル パスのファイル名が特定されます。ファイルの拡張子を指定する必要はありません。fLocateSymbols の場合、シンボル定義が特定されます。fLocateFiles も fLocateSymbols も True に設定されます。

354 ページの「Symbol レコード」も参照してください。

レコード変数を文字列とし表現できるため、Symbol レコードはバッファのテキスト行として記述されることがあります。Symbol レコードを出力バッファから読み取るには、GetBufLine 関数を使用して、行テキスト全体を返します。この場合は、Symbol レコードです。Symbol レコードの説明は GetSymbolLocation を参照してください。

GetSymbolLocationEx は一致した宣言の数を返すか、または何も見つからない場合はゼロを返します。

GetSymbolLocation 関数とは異なり、この関数は symbol_name と一致する複数の宣言を見つけます。この関数を使用して、出力バッファの各

行をスキャンすることにより、それぞれの場所を列挙することができます。

この例では、シンボルを検索し、見つかったそれぞれの宣言を列挙します。

```
symbol = Ask("What symbol do you want to locate?")
hbuf = NewBuf("output")
count = GetSymbolLocationEx(symbol, hbuf, 1, 1, 1)
ln = 0
while (ln < count)
    {
    loc = GetBufLine(hbuf, ln)
    msg (loc.file # " at line " # loc.lnFirst)
    ln = ln + 1
    }
CloseBuf(hbuf)
```

ファイル名の特定

GetSymbolLocationEx では、ファイル名を検索することもできます。単純なファイル名を symbol_name パラメータとして指定することにより、GetSymbolLocationEx はプロジェクトまたはプロジェクト シンボル パスのファイルを検索し、完全修飾パスを location.file フィールドのファイルに返します。GetSymbolLocation の例を参照してください。

fLocateFiles が TRUE で、シンボル名が拡張子なしで指定されている場合、GetSymbolLocationEx は、拡張子に関わらずこの基本名を持つファイルを特定します。たとえば、symbol_name に「dlg」を指定した場合、GetSymbolLocationEx は「dlg.c」(ファイル) と「dlg.h」(別のファイル) を返します。さらに、fLocateSymbols も TRUE に設定されている場合は、「DLG」(データ型) も返されます。

GetSymbolFromCursor (hbuf, ln, ich)

指定されたカーソル位置にあるシンボル名の Symbol レコードを返します。hbuf はバッファ ハンドルです。ln は行番号です。ich は、行上のゼロベースの文字インデックスです。

ジャンプする代わりに、Symbol レコードを返すことを除き、Jump To Definition コマンド同じように動作します。354 ページの「Symbol レコード」も参照してください。

GetSymbolLocationFromLn (hbuf, ln)

hbuf バッファ内の行番号 In にあるシンボルの Symbol レコードを返します。ln にあるシンボルは、宣言に指定された行番号 ln が含まれているシンボルです。354 ページの「Symbol レコード」も参照してください。

JumpToLocation (symbol_record)

symbol_record で指定された場所にジャンプします。Symbol レコードのファイルが開き、カーソルをそこで定義されているシンボルに移動します。これは、Jump To Definition コマンドと同じように動作します。

Symbol レコードは GetSymbolLocation 関数によって返されます。354 ページの「Symbol レコード」も参照してください。

JumpToSymbolDef (symbol_name)

symbol_name という名前のシンボルの定義にジャンプします。ファイルが開き、カーソルがそこで定義されているシンボルに移動します。これは、Jump To Definition コマンドと同じように動作します。

SymbolChildren (symbol)

指定されたシンボルの子を含む新しいシンボル リスト ハンドルを返します。シンボルの子は、シンボルのボディ内で宣言されたシンボルです。たとえば、クラスの子はクラス メンバです。

`symbol` には Symbol レコードが含まれます。354 ページの「Symbol レコード」も参照してください。

SymListFree を呼び出して SymbolChildren によって返されたシンボル リスト ハンドルを解放します。

Symbol List 関数を使用して、この関数によって返されたシンボル リストにアクセスできます。

例

この例では、シンボルの定義とその子を表示します。

```
symbolname = Ask("What symbol do you want to locate?")
symbol = GetSymbolLocation(symbolname)
if (symbol == nil)
    Msg (symbolname # " was not found")
else
    {
    hsyml = SymbolChildren(symbol)
    cchild = SymListCount(hsyml)
    ichild = 0
    while (ichild < cchild)
        {
        childsym = SymListItem(hsyml, ichild)
        Msg (childsym.symbol # " was found in "
            # childsym.file # " at line " # childsym.lnFirst)
        ichild = ichild + 1
        }
    SymListFree(hsyml)
    }
```

SymbolContainerName (symbol)

シンボル名のコンテナ コンポーネントを返します。

`symbol` には Symbol レコードが含まれます。354 ページの「Symbol レコード」も参照してください。

各シンボル名は、ドット (.) 文字で区切られたパスのコンポーネントに分類されます。たとえば、「myclass.member1」というシンボル名では、「member1」は「myclass」に含まれています。

SymbolDeclaredType (symbol)

symbol で指定された宣言の種類の Symbol レコードを返します。

`symbol` には Symbol レコードが含まれます。354 ページの「Symbol レコード」も参照してください。

SymbolLeafName (symbol)

「リーフ」を返すか、またはシンボル名の右端のコンポーネントを返します。

`symbol` には Symbol レコードが含まれます。354 ページの「Symbol レコード」も参照してください。

各シンボル名は、ドット (.) 文字で区切られたパスのコンポーネントに分類されます。たとえば、「myclass.member1」というシンボル名では、「member1」は「myclass」に含まれています。

SymbolParent (symbol)

指定されたシンボルの親の Symbol レコードを返します。シンボルの親は、そのシンボルを含むシンボルです。

symbol には Symbol レコードが含まれます。354 ページの「Symbol レコード」も参照してください。

SymbolRootContainer (symbol)

ルートまたはシンボル名の最も右側のコンポーネントを返します。

symbol には Symbol レコードが含まれます。354 ページの「Symbol レコード」も参照してください。

各シンボル名は、ドット (.) 文字で区切られたパスのコンポーネントに分類されます。たとえば、「myclass.member1」というシンボル名では、「member1」は「myclass」に含まれています。

SymbolStructureType (symbol)

指定されたシンボルの構造タイプの Symbol レコードを返します。構造タイプは、シンボルの struct タイプまたはクラス タイプで、typedefs によって間接的に参照される可能性があります。

symbol には Symbol レコードが含まれます。354 ページの「Symbol レコード」も参照してください。

関数の検索

関数は単語やパターンへの参照を検索します。

GetSourceLink (hbufSource, lnSource)

Link レコードのソース リンク先を返します。ソース バッファは hbufSource でソース行番号は lnSource です。指定された行にソース リンクが含まれていない場合は、空の文字列が返されます。351 ページの「Link レコード」も参照してください。

リンク先は、ある行番号のあるファイルの場所をポイントします。このソース リンク情報は、2 つの任意の場所にリンクします。たとえば、検索結果バッファには、検索パターンに一致するそれぞれの行のソース リンクが含まれています。

LoadSearchPattern(pattern, fMatchCase, fRegExp, fWholeWordsOnly)

[Search] コマンド、[Search Forward] コマンド、[Search Backward] コマンドで使用された検索パターンをロードします。

検索パターン文字列は pattern で指定されます。

fMatchCase の場合、検索では大文字と小文字が区別されます。

fRegExpr の場合、パターンには正規表現が含まれます。それ以外の場合は、パターンは単純な文字列です。

fWholeWordsOnly の場合は、完全に一致する単語だけが検索されます。

ReplaceInBuf(hbuf, oldPattern, newPattern, lnStart, lnLim, fMatchCase, fRegExp, fWholeWordsOnly, fConfirm)

指定されたバッファで検索処理と置換処理を実行します。

検索パターン文字列は、oldPattern で指定されます。

置換パターン文字列は、newPattern で指定されます。

行範囲は、lnSart から lnLim で指定されます。置換処理は、lnStart up から lnLim - 1 までの行で行われます。

fMatchCase の場合、検索では大文字と小文字が区別されます。

fRegExpr の場合、パターンには正規表現が含まれます。それ以外の場合は、パターンは単純な文字列です。

fWholeWordsOnly の場合は、完全に一致する単語だけが検索されます。

fConfirm の場合、置換を行う前に確認されます。

SearchForRefs (hbuf, word, fTouchFiles)

プロジェクト全体で単語中に含まれる単語文字列への参照を検索します。word を含むそれぞれの行がバッファ hbuf に追加されます。fTouchFiles が TRUE の場合、word を含む各ファイルでは、最終更新時のタイム スタンプが現在の時間に設定されます。

この関数は、[Lookup References] コマンドに似ています。word には複数の単語を含めることができますが、この関数は単一単語の方がより高速に動作します。

この例では、新規の検索ファイルを作成し、参照を検索します。

```
macro LookupRefs (symbol)
{
    hbuf = NewBuf("Results") // create output buffer
    if (hbuf == 0)
        stop
    SearchForRefs(hbuf, symbol, 0)
    SetCurrentBuf(hbuf) // put buffer in a window
}
```

SearchInBuf (hbuf, pattern, lnStart, ichStart, fMatchCase, fRegExp, fWholeWordsOnly)

hbuf バッファのパターンを検索します。検索は行番号 lnStart と文字インデックス ichFirst で開始します。SearchInBuf は、一致テキストの橋渡しをする Sel レコードを返します。何も見つからない場合は、空の文字列が返されます。Sel レコードの説明は GetWndSel を参照してください。

fMatchCase の場合、検索では大文字と小文字が区別されます。

fRegExpr の場合、パターンには正規表現が含まれます。それ以外の場合は、パターンは単純な文字列です。

fWholeWordsOnly の場合は、完全に一致する単語だけが検索されます。

SetSourceLink (hbufSource, lnSource, target_file, lnTarget)

新規のソース リンクを作成します。リンク ソース バッファは hbufSource です。リンク ソースの行番号は lnSource です。リンク ターゲット ファイルは target_file でパス文字列で指定されます。リンク ターゲットの行番号は lnTarget です。

成功した場合は True、そうでない場合は False を返します。Target_file は必ずしも存在する必要はありません。target_file が存在しないために処理が失敗することはありません。また、target_file が開いている必要もありません。

一貫した結果を得るため、target_file にはファイルの完全修飾パス名を含めます。ただし、この関数に簡単なファイル スペックを渡して、現在のプロジェクトとプロジェクト シンボル パスに含まれているファイルに基づいて target_file が拡張されます。

ソース バッファが閉じるか、またはソース行が削除されるとソース リンクは破棄されます。

プロジェクト関数

プロジェクト関数では、プロジェクトを開いたり閉じたり、またはプロジェクト情報を入手することが可能です。

AddConditionVariable(hprj, szName, szValue)

コードの解析中に #if などの条件文を評価するのに使用される条件付き解析変数を追加します。

Hprj はプロジェクトへのハンドルです。hprj が hNil の場合、新しい変数がグローバル条件リストに追加されます。

変数名は szName で指定され、値は szValue で指定されます。

グローバル リストとプロジェクト固有リストの 2 つの条件リストがあります。プロジェクトを開くと、2 つのリストがマージされ、グローバル リストのエントリは、プロジェクト固有のリストで上書きされます。

388 ページ の「DeleteConditionVariable」も参照してください。86 ページの「条件付き解析」も参照してください。

AddFileToProj(hprj, filename)

指定された filename を hprj プロジェクトに追加します。

CloseProj (hprj)

hprj プロジェクトを終了します。

DeleteConditionVariable(hprj, szName)

コードの解析中に #if などの条件文を評価するのに使用される条件付き解析変数を削除します。

Hprj はプロジェクトへのハンドルです。hprj が hNil の場合、変数がグローバル条件リストから削除されます。

変数名は、szName で指定されます。

グローバル リストとプロジェクト固有リストの 2 つの条件リストがあります。プロジェクトを開くと、2 つのリストがマージされ、グローバル リストのエントリは、プロジェクト固有のリストで上書きされます。

388 ページ の「AddConditionVariable」も参照してください。86 ページの「条件付き解析」も参照してください。

DeleteProj (proj_name)

proj_name で指定されているプロジェクトを削除します。そのプロジェクトが現在開いている場合は、最初にプロジェクトを閉じるかどうかを確認するメッセージが表示されます。ユーザーがプロジェクトを閉じない場合は、プロジェクトは削除されません。

EmptyProj ()

プロジェクトからすべてのファイルを削除することにより、プロジェクトを空にします。実際のファイル自体には影響はありません。成功した場合は True、エラーの場合は False を返します。

GetCurrentProj ()

現在開いているプロジェクトのハンドル (hprj) を返します。Source Insight では、ユーザーは一度に 1 つのプロジェクトを開くことができますが、マクロ言語からは複数のプロジェクトを開くことができます。

GetProjDir (hprj)

hprj プロジェクトのソース ディレクトリ パスを返します。

GetProjFileCount (hprj)

hprj プロジェクトに追加されたファイル数を返します。

GetProjFileName (hprj, ifile)

hprj プロジェクトのインデックス ファイルに関連付けられているプロジェクト ファイルの名前を返します。

各プロジェクトには、ファイル名でソートされたプロジェクト ファイルのインデックスがあります。GetProjFileName は、インデックスをファイル名にマップします。ファイル インデックス値は、ゼロから GetProjFileCount で返される数までの値です。

次の例は、すべてのプロジェクト ファイルを統合します。

```
ifileMax = GetProjFileCount (hprj)
ifile = 0
while (ifile < ifileMax)
    {
    filename = GetProjFileName (hprj, ifile)
    ..
    ifile = ifile + 1
    }
```

GetProjName (hprj)

hprj プロジェクトの名前を返します。名前には、プロジェクト ファイルのフルパスが含まれています。

GetProjSymCount (hprj)

hprj プロジェクトのシンボル数を返します。

GetProjSymLocation (hprj, isym)

hprj プロジェクトのインデックス isym に関連付けられたシンボルの Symbol レコードにあるシンボルの位置情報を返します。354 ページの「Symbol レコード」も参照してください。

各プロジェクトには、シンボル名でソートされたシンボルのインデックスがあります。GetProjSymLocation はインデックスをシンボルの Symbol レコードにマップします。シンボル インデックス値は、ゼロから GetProjSymCount で返される数までの値です。

JumpToLocation を呼び出して、GetProjSymLocation により返される Symbol レコードに移動することができます。

Symbol レコードについての詳細は、GetSymbolLocation を参照してください。

GetProjSymName (hprj, isym)

プロジェクトのインデックス isym に関連付けられているシンボルの名前を返します。

各プロジェクトには、シンボル名でソートされたシンボルのインデックスがあります。GetProjSymName は、インデックスをシンボル名にマップします。Symbol インデックス値は、ゼロから GetProjSymCount で返される数から 1 を引いた数までの値です。

この例は、すべてのプロジェクト シンボルを統合します。

```
isymMax = GetProjSymCount (hprj)
isym = 0
while (isym < isymMax)
    {
    symname = GetProjSymName (hprj, isym)
    ..
    isym = isym + 1
    }
```

NewProj (proj_name)

新規プロジェクトを作成し、プロジェクト ハンドル (hprj) を返します。またはエラーの場合は、hNil を返します。

OpenProj (proj_name)

proj_name という名前のプロジェクトを開き、プロジェクト ハンドル (hprj) を返します。またはエラーの場合は hNil を返します。

RemoveFileFromProj(hprj, filename)

hprj プロジェクトから指定された filename を削除します。ディスク上のファイルは変更されたり、削除されません。

SyncProj (hprj)

hprj プロジェクトを同期します。プロジェクトのすべてのファイルが、外部の変更に対してチェックされ、変更されたファイルに対して Source Insight のシンボル データベースがインクリメンタルに更新されます。

SyncProjEx(hprj, fAddNewFiles, fForceAll, fSupressWarnings)

hprj プロジェクトを同期します。プロジェクトのすべてのファイルが、外部の変更に対してチェックされ、変更されたファイルに対して Source Insight のシンボル データベースがインクリメンタルに更新されます。

fAddNewFiles が True の場合は、新しいファイルが自動でプロジェクトに追加されます。[Document Options] コマンドで定義されているドキュメント タイプに一致するファイル名のみが追加されます。

fForceAll の場合、プロジェクトにある各ファイルはタイム スタンプに関わらず、再同期されます。

fSuppressWarnings の場合、ファイルを開く際にエラーが発生しても Source Insight は警告メッセージを発行しません。

その他のマクロ関数

これらの関数は、ほかのカテゴリーには属しませんが、役立つ関数です。

DumpMacroState (hbufOutput)

この関数は、hbufOutput バッファに実行中のマクロの現在の状態を説明するテキストを追加します。マクロの状態は、すべての変数の値、実行スタックから成ります。この関数は、マクロをデバッグする際に役立ちます。

GetProgramEnvironmentInfo ()

Source Insight が実行されている環境についての情報が含まれている ProgEnvInfo レコードを返します。351 ページの「ProgEnvInfo レコード」も参照してください。

GetProgramInfo ()

Source Insight の情報が含まれている ProgInfo 構造体を返します。352 ページの「ProgInfo レコード」も参照してください。

マクロについてのその他の情報

デバッグ

Source Insight にはマクロのデバッガーは含まれていません。しかし、マクロは解釈されるため、"Msg" 関数をコード中の重要なポイントで使用して、文字列と変数値を出力し、簡単に何が行われているかを知ることができます。360 ページの「Msg (s)」も参照してください。

現在のカーソル位置でマクロ文の実行を開始するには、[Run Macro] を使用します。実行を開始する行に挿入ポイントを配置し、[Run Macro] コマンドを実行するだけです。

DumpMacroState 関数を使用して、実行マクロの実行スタックと変数状態をダンプすることができます。391 ページの「DumpMacroState (hbufOutput)」も参照してください。

持続性

グローバル変数は、実行間で保持されますが、セッション間では保持されません。ローカル変数は、実行間でもセッション間でも保持されません。ただし、ファイルに値を格納するか、またはレジストリ キーの書き込み / 読み取りを行うことにより値を保持することができます。

非自己書き換えマクロ

実行中にマクロが自身を書き換えないようにしてください。Source Insight は、マクロが実行中のマクロを含むファイルを編集しようとすると異常終了します。

サンプル マクロ

マクロ ファイルは Source Insight の「utils.em」に含まれています。このファイルには、いくつかの役立つ関数が含まれていて、サンプルとして参考にできます。

イベント ハンドラ

イベント ハンドラは、特定のイベントが発生すると呼び出される、Source Insight のマクロ言語で記述された関数です。詳細は、395 ページの第 7 章「マクロ イベント ハンドラ」を参照してください。

第 7 章　マクロ イベント ハンドラ

この章では、Source Insight のマクロ言語で記述されたイベント ハンドラ関数について説明します。イベント ハンドラは、特定のイベントが発生すると呼び出される関数です。この章は、マクロ言語の規則と構文について理解していることを前提にしています。

マクロ イベント ハンドラ

イベント ハンドラは、特定のイベントが発生すると呼び出される、Source Insight のマクロ言語で記述された関数です。'macro' キーワードを使用して関数を定義する代わりに、'event' キーワードを使用します。例 :

```
event FileOpen(sFile)
{
}
```

イベント ハンドラ関数は値を返しません。イベント ハンドラ関数を使用してイベントを終了させたり、プログラムに値を返すことはできません。イベント関数パラメータは 正確に入力してください。

イベント ハンドラ名

イベント ハンドラの名前と関数パラメータは事前に定義されています。イベント ハンドラの関数名とパラメータは正確に記述してください。

Source Insight でサポートされているイベントは次のとおりです。

アプリケーション イベント

```
event AppStart()
event AppShutdown()
event AppCommand(sCommand)
```

ドキュメント イベント

```
event DocumentNew(sFile)
event DocumentOpen(sFile)
event DocumentClose(sFile)
event DocumentSave(sFile)
event DocumentSaveComplete(sFile)
event DocumentChanged(sFile)
event DocumentSelectionChanged(sFile)
```

プロジェクト イベント

```
event ProjectOpen(sProject)
event ProjectClose(sProject)
```

ステータス バー イベント

```
event StatusbarUpdate(sMessage)
```

イベント ハンドラの使用

イベント ハンドラにはさまざまな用途があります。次にいくつかの例を示します。

- アクティビティのモニタと記録。たとえば、ログ ファイルに動作を記録し、後でその情報を使用してプロジェクトの編集された部分を解析できます。
- Source Insight 内の動作と他のプログラムやファイルの同期。
- Source Insight の動作の変更。
- ファイル保存前の後処理。
- 新しいファイルの前処理。

イベント ハンドラの Source Insight への追加

イベント ハンドラは、マクロ ソース ファイルに保存されます。つまり、拡張子は .em です。同じファイルにイベント関数とマクロ関数を記述することができます。イベント ハンドラを記述したら、カレント プロジェクトに追加します。プロジェクトに関係なくイベントを処理する場合は、ベース プロジェクトに追加します。Source Insight は指定されたイベント ハンドラが見つからない場合、無視します。Source Insight はプロジェト、プロジェクト シンボル パス、およびベース プロジェクトを検索します。

イベント ハンドラ ファイルをプロジェクトに追加します。

.em ファイルは、必ずプロジェクトに追加してください。追加しないと、Source Insight はファイル内のイベント ハンドラを呼び出しません。これは、イベント ハンドラが記述されている .em ファイルを開いた際に、誤ってイベント ハンドラが実行されないようにするためです。

イベント ハンドラの有効化

[Preferences] でイベント ハンドラを有効にしてください。

イベント ハンドラは使用する前に有効にしなければなりません。セキュリティ上の理由から、デフォルトでは無効にされています。イベント ハンドラを有効にするには、**[Options] > [Preferences]** を選択して **[General]** タブをクリックします。次に、[Enable event handlers] チェック ボックスをオンにします。いったんイベント ハンドラを有効にすると、設定が保存されるため、再度有効にする必要はありません。

また、メニューに割り当てたり、キーストロークを割り当てることができる [Enable Event Handlers] ユーザーレベル コマンドもあります。

メモ： セキュリティ上の理由から、[Enable Event Handlers] コマンドをマクロから実行することはできません。

イベント ハンドラ ファイルの編集

Source Insight は、変更され、保存されていないファイルのイベント ハンドラを無視します。そのため、イベント ハンドラ ソース ファイルを編集している場合、編集中のイベント ハンドラは実行されません。編集が完了したら、ファイルを保存してください。ファイルを保存すると、Source Insight はイベント ハンドラを実行するようになります。

イベント ハンドラのエラー

1 つのイベント ハンドラで構文エラーまたはランタイム エラーが発生すると、すべてのイベント ハンドラが残りの Source Insight のセッショ

ンで無効にされ、マクロ エラー メッセージが表示されます。再度イベント ハンドラを有効にするには、Source Insight を再起動します。

同期イベントと 非同期イベント

一部のイベント ハンドラは、イベントが発生するとすぐに呼び出されます。これらは「同期」イベントと呼ばれます。たとえば、DocumentNew はユーザーが新規ドキュメントを作成するとすぐに呼び出されます。

しかし、一部のイベント ハンドラは、イベントが発生した後 (通常、短いアイドル時間の後) に呼び出されます。これらは「非同期」イベントと呼ばれます。これらのイベント ハンドラは、イベント発生と同時にユーザーによって記述されたマクロを呼び出した場合、Source Insight が不安定になるため、非同期になっています。

その他のヒント

イベント ハンドラは、1 つのファイルにすべてを保存するか、または「event-something.em」のような名前のいくつかのファイルに保存してください。これにより、これらのファイルをプロジェクトから削除し、効率的にイベント ハンドラを無効にすることができます。

グローバル変数は、カウンタの追加、およびイベント間の状態の管理に便利です。

アプリケーション イベント

アプリケーション イベントは、Source Insight アプリケーション全体に適用されます。

event AppStart()

Source Insight アプリケーションをロードし、初期化した後に呼び出されます。カレント プロジェクトとワークスペース セッションは既にロードされています。

event AppShutdown()

Source Insight アプリケーションを終了する直前に呼び出されます。

event AppCommand(sCommand)

指定されたユーザーレベル コマンドを実行した直後に呼び出されます。

ドキュメント イベント

ドキュメント イベントは、ファイル バッファを開いた時、閉じた時、保存した時、および変更した時に適用されます。

event DocumentNew(sFile)

指定されたファイル バッファが作成された直後に呼び出されます。

event DocumentOpen(sFile)

ファイル バッファが開かれた直後に呼び出されます。

event DocumentClose(sFile)

ファイル バッファが閉じられた直後に呼び出されます。

event DocumentSave(sFile)

ファイル バッファが保存される*直前*に呼び出されます。保存される直前にファイル バッファを編集できます。保存された*後*に編集する場合は、DocumentSaveComplete イベントを使用します。

event DocumentSaveComplete(sFile)

ファイル バッファが保存された*直後*に呼び出されます。保存される*前*に編集する場合は、DocumentSave イベントを使用します。

event DocumentChanged(sFile)

ファイル バッファがユーザーによって編集された場合に呼び出されます。このイベントは非同期で処理されます。つまり、ユーザーが入力している時に呼び出されるわけではありません。一定時間後に呼び出されます。この関数を使用して、このイベント ハンドラ内のファイルを編集できます。非同期で呼び出されるため、sFile ファイルは開かれていない可能性があることに注意してください。

event DocumentSelectionChanged(sFile)

ユーザーがカレントファイルでテキストを選択した場合やカーソルを移動した場合に呼び出されます。このイベントは非同期で処理されます。つまり、ユーザーがカーソルを移動している時に呼び出されるわけではありません。一定時間後に呼び出されます。非同期で呼び出されるため、sFile ファイルは開かれていない可能性があることに注意してください。

プロジェクト イベント

プロジェクト イベントは、Source Insight プロジェクトを開いた時、または閉じた時に適用されます。

event ProjectOpen(sProject)

プロジェクトが開かれた後に呼び出されます。

event ProjectClose(sProject)

プロジェクトが閉じられる前に呼び出されます。

ステータス バー イベント

ステータス バー イベントは、ステータス バー テキストが変更された時に発生します。

event StatusbarUpdate(sMessage)

ステータス バーの内容が変更された場合に呼び出されます。このイベントは非同期で処理されます。つまり、ステータス バーを変更している時に呼び出されるわけではありません。一定時間後に呼び出されます。この関数を使用して、このイベント ハンドラ内のファイルを編集できます。

第 8 章 付録：以前のバージョンからのアップグレード

この付録では、Source Insight バージョン 2.0、2.1、3.0、および 3.1 からのアップグレードについて説明します。

バージョン 3.1 またはバージョン 3.0 からのアップグレード

バージョン 3.0 または 3.1 からアップグレードする場合は、このセクションをお読みください。また、この章を読むと、Source Insight バージョン 3.5 について詳しく知ることもできます。バージョン 3.5 には、重要な変更が含まれています。

ユーザーごとのデータ フォルダ

ユーザーごとのデータは、My Documents\Source Insight フォルダに保存されます。

バージョン 3.5 では、ユーザーごとのデータの場所が変更されています。ユーザーごとのデータは、My Documents フォルダの Source Insight

サブフォルダ内に保存されます。Source Insight フォルダ内には、次のサブフォルダがあります。

- **Projects フォルダ** - プロジェクト データ ファイルのデフォルトの場所です。デフォルトでは、各プロジェクトは Projects フォルダ内の別々のサブフォルダに保存されます。
- **Settings フォルダ** - 設定ファイルのためのフォルダです。以前の設定ファイルは、ここにコピーされます、
- **Backup フォルダ** - ファイルを保存する際に作成されるバックアップ ソース ファイルのためのフォルダです。
- **Projects\NetFramework フォルダ** - NetFramework プロジェクトのためのフォルダです (インストールされている場合)。このプロジェクトには、Source Insight が使用する .Net Framework クラス ライブラリ シンボルが格納されています。
- **Projects\Base フォルダ** - ベース プロジェクトのためのフォルダです。ユーザーごとに別々のベース プロジェクトがあります。

My Documents 以下にユーザー データを保存することで、データの安全性と機密性が確保され、データへの書き込み権が保証されます。

ユーザーごとのプロジェクト リスト

[Open Project] コマンドで表示されるように、ユーザーごとに別々のプロジェクト リストがあります。プロジェクト リスト ファイルは、ユーザーの Projects フォルダに保存されています。**[Open Project]** ダイアログ ボックスで **[Browse]** ボタンをクリックし、リストにないプロジェクトを検索することもできます。アクセス権がある限り、プロジェクトを開くことができます。開いたプロジェクトは、プロジェクト リストに追加されます。

プロジェクト ファイルの保存

各プロジェクトのプロジェクト設定は、次の 2 つのフォルダを指定します。

- **プロジェクト データ ディレクトリ** - Source Insight がプロジェクト データ ファイルを保管する場所です。たとえば、.pr ファイルはここに格納されます。デフォルトでは、新規プロジェクトを作成する場合、Source Insight は Projects フォルダにプロジェクト データ ディレクトリを作成します。
- **プロジェクト ソース ディレクトリ** - プロジェクト ソース ファイルが位置するメインの場所です 。以前のバージョンの Source Insight では、*プロジェクト ルート ディレクトリ*と呼ばれていました。

これら 2 つの別のフォルダを使用することで、Source Insight のデータをソース ファイルと別の場所に格納できます。さらに、Source Insight のプロジェクト ファイルは、常に個々のユーザー データ エリアに保存されるため、同じマシンの他のユーザーからアクセスできません。両方のフォルダで同じ場所を使用することは可能です。

プロジェクト ソース ディレクトリの場所を編集するには、**[Project Settings]** コマンドを使用します。

カスタム コマンド ディレクトリの置換

カスタム コマンドでは、次のメタ文字がこれらのディレクトリに置換されます。

- `%j` - プロジェクト ソース ディレクトリ
- `%J` - プロジェクト データ ディレクトリ

.NET Framework のサポート

.NET Framework クラス ライブラリ シンボルは Source Insight が作成した **NetFramework** プロジェクトに格納されます。Source Insight は、ユーザーのプロジェクト フォルダの NetFramework フォルダにプロジェクトを保存します。

Source Insight は、.NET Framework クラス ライブラリのシンボルを宣言する「ソース ファイル」のセットもインストールします。これらのソースは、Source Insight プログラム フォルダの NetFramework フォルダに保存されます。マシンごとに 1 つのコピーがあります。これらの「ソース ファイル」は C# 構文のマシン生成ファイルです。しかし、これらのソース ファイルは厳密な C# 互換ではありません。ファイルの内容は、Source Insight の新しいバージョンでは変更されることがあります。

Source Insight で NetFramework プロジェクトを作成するには、**[Setup Common Projects]** コマンドを使用するか、**[Preferences: Symbol Lookups]** ダイアログ ボックスで **[Create Common Projects]** ボタンをクリックします。

バージョン 2 からのアップグレード

バージョン 2.0 または 2.1 からアップグレードする場合は、このセクションをお読みください。

バージョン 3 のインストール

マシンにバージョン 2.x をインストールしている場合、バージョン 2.x. と異なるディレクトリにバージョン 3.5 をインストールしてください。

以前のプロジェクトを開く

Source Insight 3.5 は、バージョン 2.x で作成されたプロジェクト ファイルを読み取ることができますが、バージョン 3.5 の形式に変換する必要があります。Source Insight バージョン 2.x を続けて使用する場合は、バージョン 3.5 用にプロジェクトを作成し直すことをお勧めします。

以前のプロジェクトの検索

バージョン 3.5 をバージョン 2.x と異なるディレクトリにインストールする場合、バージョン 3.5 に以前のプロジェクトの記録はありません。ただし、以前のプロジェクトを開くことは可能です。

バージョン 2.x のプロジェクトを開くには、[Project] > [Open Project] コマンドを使用します。ダイアログ ボックスで [Browse] ボタンをクリックし、以前の .PR ファイルを参照します。以前の .PR ファイルを選択し、[OK] をクリックしてから [Open] をクリックします。Source Insight は、プロジェクトをバージョン 3.5 の形式に変換します。

以前のカスタマイズのロード

設定ファイルは、カスタマイズを保存するために使用されます。バージョン 3.5 では、設定ファイルのフォーマットが変更されていますが、Source Insight は 2.x 形式のファイルも読み取ることができます。また、ファイルの拡張子も .CF から .CF3 に変更されています。

以前の設定ファイルをロードするには、[Options] > [Load Configuration] コマンドを使用します。[Load] ボタンをクリックし、Source Insight バージョン 2.x のインストール ディレクトリを参照します。拡張子が .CF のファイルが表示されます。バージョン 2.x のメイン グローバル設定ファイルは global.cf です。このファイルを選択し、[OK] をクリックします。以前の設定ファイルがロードされ、自動的に新しいグローバル設定ファイル global.cf3 に保存されます。

バージョン 3 とバージョン 2 の併用

たとえば、バージョン 3.5 と 2.x を同じマシンで使用できます。それぞれ別のレジストリ設定を使用するため、併用しても問題ありません。しかし、下記のガイドラインに従ってください。

- マシンにバージョン 2.x をインストールしている場合、バージョン 2.x. と違うディレクトリにバージョン 3.5 をインストールしてください。
- バージョン 2.x とバージョン 3.5 のインスタンスを同時に実行しないでください。
- プロジェクト ファイルは互換性がありますが、バージョン 3.5 用に別のプロジェクトを作成することをお勧めします。
- バージョン 2.x のプロジェクトを開くには、[Open Project] ダイアログ ボックスで [Browse] ボタンをクリックし、以前の .PR ファイルを指定する必要があります。新しいディレクトリにバージョン 3.5 をインストールした場合、バージョン 3.5 のディレクトリに以前のプロジェクトは含まれません。
- [Options] > [Load Configuration] コマンドで以前の設定ファイルを開くことができます。2.x のディレクトリで以前の *.CF ファイルを選択してください。新しい設定ファイルの拡張子は .CF3 に変更されていることに注意してください。

バージョン 3 の新機能

Source Insight は、バージョン 2.x から大幅に変更されています。主な変更点を次に要約します。

言語機能の改良

バージョン 3.0 では、構文解析やシンボル参照など、言語固有の機能が大幅に変更されています。いくつかの変更点を次に示します。

- クラスと構造体のメンバを追跡し、コンパイルせずに動的にクラス継承を解読する状況依存シンボル参照エンジン。バージョン 2.1 では、フィールド メンバや継承は追跡できませんでした。
- C++ と Java の入れ子のクラスと構造体のサポート。
- C++ のクラス テンプレートと関数テンプレートのサポート。
- C++ の名前空間のサポート。
- 補完された C 構造体とユニオン (つまり、インライン構造) のサポート。
- C++ のテンプレート クラスとテンプレート関数のサポート。
- 匿名の構造体とユニオンのサポート。
- Source Insight のパーサーのアップストリームでソース コード トークンを置換するため、さまざまなキーワードやプリプロセッサを処理することができるユーザー定義可能なトークン置換マクロ。
- #ifdef 分岐の有効 / 無効をマークするためのユーザー定義可能なコンパイル時定数。
- コード メトリクス。
- 新しいビルトイン言語の解析と表示のサポート。
  - Perl および PerlScript
  - Visual Basic および VBScript
  - JavaScript および JScript
  - 埋め込みスクリプトを含む HTML、ASP、および JSP
  - C#
- ユーザー定義のカスタム言語のサポート。

参照機能と解析機能の改良

Source Insight は、プログラムのシンボル情報へのアクセスに優れています。

- 作業中、動的にコール ツリー、クラス ツリー、および参照ツリーを更新して表示する新しいリレーション ウィンドウ。
- より優れたファイル リスト機能とシンボル リスト機能を備えたプロジェクト ウィンドウ。
- シンボル シラブル インデックス。シンボル名の部分文字列がインデックス付けされているため、シンボル名の最初の部分が分からなくても、名前の一部を入力するだけで済みます。
- 非常に大規模なプロジェクトにも迅速に対応するように最適化されたシンボル データベース。
- [Smart Reference Matching] オプションを使用し、適切なシンボルへの参照のみを表示するように改良された [Lookup References] コマンド。コメントや非アクティブ コードをスキップ / 認識することもできます。
- キーワード検索機能を備えた新しい [Search Project] コマンド。インターネット検索に似ており、プロジェクト内の単語を検索します。

編集機能と表示機能の改良

バージョン 3.0 では、編集機能と表示機能が大幅に変更されています。

- ユーザー定義スタイルのリッチ テキスト書式にも対応し、大幅に改良された表示機能を提供する構文フォーマット。Source Insight は、プロジェクトの単語情報に基づいてスタイルを自動的に適用します。構文フォーマットには次のものが含まれます。
- 関数、クラス、変数などの宣言スタイル。
- いくつかのリッチ「コメント」スタイル。
- ローカル、パラメータ、グローバル、マクロ、関数など、異なるタイプのシンボルへの参照スタイル。
- 自動注釈と特殊な構文修飾。
- シンボル名入力時の自動補完 。
- インクリメンタル検索。
- すべての適切なコンテキストのシンボル名を変更する新しい状況依存 [Smart Rename] コマンド。ローカル変数の名前を素早く変更できます。
- ファイル保存時の操作履歴と変更履歴の保存、および 2 段階の改訂マークの表示。
- 行番号の表示。
- 改ページとページ番号の表示。
- 右余白の表示。
- ユーザー定義の [Work] メニュー。
- 保存操作中のファイル制御の改良。
- その他の UI の改良。次のものが含まれます。
- ウィンドウの結合の改善。
- すべてのウィンドウ用のカスタマイズ可能なフォントと色。
- ダイアログ ボックスの位置の記録。
- 複数選択リスト。
- ツールバーの改良。

新しいコマンド

新しいコマンドのリスト

Source Insight バージョン 2.x からバージョン 3.5 で新しく追加されたコマンドを次の表に要約します。

表 8.1: バージョン 2.x 以降新しく追加されたコマンド

コマンド	説明
Activate Relation Window	リレーション ウィンドウを開いて選択します。
Add and Remove Project Files	カレント プロジェクトのファイルを追加 / 削除します。以前の [Add Files] コマンドと [Remove Files] コマンドの代わりです。
Advanced Options	さまざまな内部キャッシュを有効 / 無効にします。トラブルシューティング用に提供されています。
Build Project	カスタム ツール コマンド : プロジェクトをビルドします。
Check In	カスタム ソース管理 : カレント ファイルをチェックインします。
Check Out	カスタム ソース管理 : カレント ファイルをチェックアウトします。
Checkpoint	カレント ファイルをディスクに保存し、変更履歴を消去します。
Checkpoint All	開いているすべてのファイルをディスクに保存して、変更履歴を消去します。
Clean Build	カスタム ツール コマンド : プロジェクト全体をビルドします。
Clear Highlights	すべてのソース ウィンドウで単語のハイライト表示をクリアします。ハイライト表示を行うには、[Highlight Word] コマンドを使用します。
Color Options	ユーザ インターフェイス項目の色を指定します。
Compile File	カスタム ツール コマンド : カレント ファイルをコンパイルします。
Drag Line Down	選択されているテキストを 1 行下に移動します。
Drag Line Down More	選択されているテキストを数行下に移動します。
Drag Line Up	選択されているテキストを 1 行上に移動します。
Drag Line Up More	選択されているテキストを数行上に移動します。
Edit Condition	選択されている解析条件の値を編集します。
Expand Special	ツリー リスト内で使用 : 選択された項目を指定されたツリー レベルまで展開します。
General Options	全般的な設定を指定します。

表 8.1: バージョン 2.x 以降新しく追加されたコマンド

コマンド	説明
Go To Next Change/Go To Previous Change	カーソルを次 / 前の変更したブロックに移動します。つまり、次 / 前の変更マークに移動します。
Highlight Word	すべてのソース ウィンドウでカーソル位置の単語のハイライト表示を切り替えます。紙面で蛍光ペンを使用するのに似ています。
HTML Help	HTML ヘルプ ファイルで現在選択している単語をキーワード検索します。
Incremental Search	入力に従って、パターンをインクリメンタルに検索します。
Incremental Search Backward	入力に従って、パターンを逆方向にインクリメンタルに検索します。
Insert ASCII	ASCII 値で文字を挿入します。
Jump To Base Type	選択されているシンボルのベース構造タイプにジャンプします。
Jump To Caller	選択されている関数を呼び出す関数にジャンプします。
Jump To Prototype	選択されている関数の関数プロトタイプにジャンプします。
Keyword List	現在の言語の構文フォーマットに使用されるキーワード リストを編集します。
Language Properties	カスタム言語プロパティを編集します。
Line Numbers	行番号の表示を切り替えます。
Lock Relation Window	リレーション ウィンドウのロックを切り替えます。ロックしてもコンテンツは変更されません。
Lowercase	選択されているテキストを小文字に変換します。
New Relation Window	新規リレーション ウィンドウを作成します。リレーション ウィンドウはいくつでも作成できます。各ウィンドウに、独自のオプションを設定できます。
Next Relation Window View	リレーション ウィンドウのアウトライン ビューとグラフビューを交互に表示します。
Preferences	各種ユーザー オプションを指定します。このダイアログ ボックスにはプロパティ シートが 1 つあり、いくつかのタブに、表示、ファイル、構文フォーマットなど、さまざまなオプションが含まれています。
Project Document Types	プロジェクト ウィンドウでドキュメント タイプごとにプロジェクト ファイルを表示します。
Project File Browser	プロジェクト ウィンドウでファイル ブラウザを表示します。
Project File List	プロジェクト ウィンドウに、すべてのプロジェクト ファイルを表示します。
Project Symbol Classes	プロジェクト ウィンドウに、クラスごとにプロジェクト シンボルを表示します。

表 8.1: バージョン 2.x 以降新しく追加されたコマンド

コマンド	説明
Project Symbol List	プロジェクト ウィンドウに、すべてのプロジェクト シンボルを表示します。
Project Window Properties	プロジェクト ウィンドウのプロパティを表示します。
Recent Files	サブメニューに最近開いたファイルの名前が含まれています。
Refresh Relation Window	現在選択されているシンボルのリレーションでリレーション ウィンドウを更新します。
Relation Graph Properties	リレーション ウィンドウの Graph プロパティを表示します。
Relation Window	リレーション ウィンドウのオンとオフを切り替えます。
Relation Window Properties	リレーション ウィンドウのプロパティを表示します。
Reload File	ディスクからカレント ファイルをリロードします。保存した後の変更はすべて消去されます。
Run Project	カスタム ツール コマンド : プロジェクトの実行ファイルを実行します。
Save A Copy	カレント ファイルを新しいファイルに保存します。カレント ファイルは上書きまたは変更されません。
Search Project	すべてのプロジェクト ファイルでテキストまたはキーワードを検索します。
Searching Options…	検索結果を制御するオプションを指定します。
Setup Common Projects…	コモン外部プロジェクトを作成します。
Setup HTML Help	ディスク上の HTML Help ファイルを検索します。
Show Relations	選択されているシンボルの情報を表示するようにリレーション ウィンドウを更新します。
Simple Tab	タブを挿入します。Smart Tab モードは無効になります。
Smart Beginning of Line	状況依存の Beginning of Line コマンド。
Smart End of Line	状況依存の End of Line コマンド。
Smart Tab command	さまざまな位置で使用され、カーソルを次の「フィールド」に移動します。フィールドは、「カレント コンテキスト内の位置」として定義されます。
Source Dynamics on the Web	インターネット ブラウザで Source Dynamics の Web サイトを表示します。
Special Edit	サブメニューに特殊な編集コマンドが含まれています。
Style Properties	表示スタイルのフォーマット プロパティを設定します。
Symbol Lookup Options	シンボル定義の検索オプションを設定します。
Symbol Window Properties	各ソース ウィンドウの左側にシンボル ウィンドウのプロパティを表示します。

表 8.1: バージョン 2.x 以降新しく追加されたコマンド

コマンド	説明
Sync File to Source Control Project	カスタム ソース管理 : 選択されているファイルの最新版を取得します。
Sync to Source Control Project	カスタム ソース管理 : すべてのプロジェクト ファイルの最新版を取得します。
構文フォーマット	ソース ファイルを表示するための構文フォーマット オプションを指定します。
Toggle Case	選択されているテキストの大文字 / 小文字を切り替えます。
Touch All Files in Relation	リレーション ウィンドウで参照されているすべてのファイルのタイムスタンプを更新します。
Typing Options	入力オプションと編集オプションを指定します。
Undo Check Out	カスタム ソース管理 : カレント ファイルのチェックアウトを元に戻します。
Uppercase	選択されているテキストを大文字に変換します。
Window List	ソース ウィンドウのリストを管理します。

以前のバージョンとのファイル形式の互換性

バージョン 3.5 は、バージョン 2.x のプロジェクト ファイルと他のいくつかのファイルを開くことができますが、すべての情報がコピーさ

れるわけではありません。また、一部のインデックスを作成し直す必要があります。

バージョン 2.x のファイル	バージョン 3.5 での対応
プロジェクト ファイル – *.PR	バージョン 3.5 は、プロジェクトファイルを開き、バージョン 3.5 の形式に変換します。シンボル データベースのインデックスを作成し直す必要があるため、時間がかかることがあります。完了すると、シンボルの検索を行うことができます。ただし、バックグランドまたは [Project] > [Synchronize Files] コマンドを使用して、後でプロジェクト全体を解析し直す必要があります。この変換により、ソース ファイルを一から追加し直す必要がないという利点が得られます。 バージョン 2.1 は、バージョン 3.5 形式のプロジェクトを開くことができますが、2.1 形式にダウンコンバートし、再度シンボル データベースのインデックスを作成し直す必要があります。 v 2.x と v 3.5 の間で頻繁に切り替える場合は、それぞれの形式のプロジェクト ファイルを管理することをお勧めします。 v 3.5 のプロジェクトの方が関連ファイルが多く、一部の拡張子が変更されていることに注意してください。各プロジェクトには次のファイル拡張子が含まれています : .PR、.PS、.PO、.PFI、.PRI、.IMB、.IMD、.IAB、.IAD
設定ファイル - *.CF	バージョン 3.5 は、バージョン 2.x の設定ファイルを開くことができますが、バージョン 3.5 のファイル形式には下位互換性がありません。v 2.x と v 3.5 を併用できるように、デフォルトの拡張子が .CF から .CF3 へ変更されています。v 3.5 をインストールしてから、[Options] > [Load Configuration] コマンドを使用して、.CF ファイル (通常は、Global.CF と呼ばれ、v 2.x のプログラム ディレクトリにある) を検索できます。 v 2.x の設定ファイルに保存されている表示オプションは変換されませんが、キー割り当てやメニュー割り当てなど、他のカスタマイズは変換されます。 v 2.x では、終了時に確認ダイアログが表示されましたが、v 3.5 では、設定の変更が自動的に保存されることに注意してください。

バージョン 2.x のファイル	バージョン 3.5 での対応
ワークスペース ファイル - *.WK	バージョン 3.5 は、バージョン 2.x のワークスペース ファイルを読み取ることができません。以前の 2.x のセッションは、v 3.5 で開く際にリストアされます。
リカバリ ファイル - *.RCV	バージョン 3.5 は、バージョン 2.x のリカバリ ファイルを読み取ることができません。クラッシュしたバージョン 2.x のセッションを修復する場合、最初にバージョン 2.x でリカバリを実行する必要があります。
プロジェクト リスト ファイル – Projects.DB	バージョン 2.x と重複しないように、プロジェクト リスト ファイル名が Projects.DB3 に変更されています。
sihook プログラム コンポーネント	sihook.exe プログラムは変更されています。また、重複しないように sihook3.exe に名前が変更されています。

第 9 章 License Agreement

SOURCE DYNAMICS
SOURCE INSIGHT VERSION 3.x END-USER LICENSE AGREEMENT

IMPORTANT-READ CAREFULLY: This Source Insight End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and Source Dynamics, Inc. for the Source Dynamics SOFTWARE identified above, which includes the User Manual, any associated SOFTWARE components, any media, any printed materials other than the User Manual, and any "online" or electronic documentation ("SOFTWARE").By installing, copying, or otherwise using the SOFTWARE, you agree to be bound by the terms of this EULA.If you do not agree to the terms of this EULA, do not install or use the SOFTWARE.If the SOFTWARE was mailed to you, return the media envelope, UNOPENED, along with the rest of the package to the location where you obtained it within 30 days from purchase.

1.The SOFTWARE is licensed, not sold.

2.GRANT OF LICENSE.

(a)Evaluation Copy.You may use the SOFTWARE without charge on an evaluation basis for thirty (30) days from the day that you install the SOFTWARE.You must pay the license fee and register your copy to continue to use the SOFTWARE after the thirty (30) days.If you continue to use the SOFTWARE after the thirty (30) days without paying the license fee you will be using the SOFTWARE on an unlicensed basis.

(b)Redistribution of Evaluation Copy.If you are using SOFTWARE on an evaluation basis you may make copies of the evaluation SOFTWARE as you wish; give exact copies of the original evaluation SOFTWARE to anyone; and distribute the evaluation SOFTWARE in its unmodified form via electronic means (Internet, BBS's, Shareware distribution libraries, CD-ROMs, etc.).You may not charge any fee for the copy or use of the evaluation SOFTWARE itself, but you may charge a distribution fee that is reasonably related to any cost you incur distributing the evaluation SOFTWARE (e.g. packaging).You must not represent in any way that you are selling the SOFTWARE itself.Your distribution of the evaluation SOFTWARE will not entitle you to any compensation from Source Dynamics.You must distribute a copy of this EULA with any copy of the SOFTWARE and anyone to whom you distribute the SOFTWARE is subject to this EULA.

(c)Registered Copy.After you have purchased the license for SOFTWARE, and have received the serial number enabling the registered copy, you are licensed to copy the SOFTWARE only into the memory of the number of computers corresponding to the number of licenses purchased.The primary user of the computer on which each licensed copy of the SOFTWARE is installed may make a second copy for his or her exclusive use on a portable computer.Under no other circumstances may the SOFTWARE be operated at the same time on more than the number of computers for which you have paid a separate license fee.You may not duplicate the SOFTWARE in whole or in part, except that you may make one copy of the SOFTWARE for backup or archival purposes.You may terminate this license at any time by destroying the original and all copies of the SOFTWARE in whatever form.You may permanently transfer all of your rights under this EULA provided you transfer all copies of the SOFTWARE (including copies of all prior versions if the SOFTWARE is an upgrade) and retain none, and the recipient agrees to the terms of this EULA.

3.RESTRICTIONS.You may not reverse engineer, de-compile, or disassemble the SOFTWARE, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.You may not rent, lease, or lend the SOFTWARE.You may permanently transfer all of your rights under this EULA, provided the recipient agrees to the terms of this EULA.You may not publish or publicly distribute any serial numbers, access codes, unlock-codes, passwords, or other end-user-specific registration information that would allow a third party to activate the SOFTWARE without a valid license.

4.SUPPORT SERVICES.Source Dynamics may provide you with support services related to the SOFTWARE.Use of Support Services is governed by the Source Dynamics polices and programs described in the user manual, in online documentation, and/or other Source Dynamics-provided materials, as they may be modified from time to time.Any supplemental SOFTWARE code provided to you as part of the Support Services shall be considered part of the SOFTWARE and subject to the terms and conditions of this EULA.

5.TERMINATION.Without prejudice to any other rights, Source Dynamics may terminate this EULA if you fail to comply with the terms and conditions of this EULA.In such event, you must destroy all copies of the SOFTWARE.

6.COPYRIGHT.The SOFTWARE is protected by United States copyright law and international treaty provisions.You acknowledge that no title to the intellectual property in the SOFTWARE is transferred to you.You further acknowledge that title and full ownership rights to the SOFTWARE will remain the exclusive property of Source Dynamics and you will not acquire any rights to the SOFTWARE except as expressly set forth in this license.You agree that any copies of the SOFTWARE will contain the same proprietary notices which appear on and in the SOFTWARE.

7.EXPORT RESTRICTIONS.You agree that you will not export or re-export the SOFTWARE to any country, person, entity, or end user subject to U.S.A.export restrictions.Restricted countries currently include, but are not necessarily limited to Cuba, Iran, Iraq, Libya, North Korea, Sudan, and Syria.You warrant and represent that neither the U.S.A.Bureau of Export Administration nor any other federal agency has suspended, revoked or denied your export privileges.

8.LIMITED WARRANTY.Source Dynamics, Inc. warrants that the Software will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of your receipt of the Software.Any implied warranties on the Software are limited to 90 days.Some states do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.SOURCE DYNAMICS, INC. DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, WITH RESPECT TO THE SOFTWARE AND THE ACCOMPANYING WRITTEN MATERIALS.This limited warranty gives you specific legal rights.You may have others, which vary from state to state.

9.LIMITATION OF LIABILITY.IN NO EVENT SHALL SOURCE DYNAMICS OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES OF ANY KIND ARISING OUT OF THE DELIVERY, PERFORMANCE, OR USE OF THE SOFTWARE, EVEN IF SOURCE DYNAMICS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.IN ANY EVENT, SOURCE DYNAMICS'S LIABILITY FOR ANY CLAIM, WHETHER IN CONTRACT, TORT, OR ANY OTHER THEORY OF LIABILITY WILL NOT EXCEED THE GREATER OF U.S.$1.00 OR LICENSE FEE PAID BY YOU.

10.U.S.GOVERNMENT RESTRICTED RIGHTS.The SOFTWARE is provided with RESTRICTED RIGHTS.Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph ©(1)(ii) of The Rights in Technical Data and Computer SOFTWARE clause of DFARS 252.227-7013 or subparagraphs ©(i) and (2) of the Commercial Computer SOFTWARE-Restricted Rights at 48 CFR 52.227-19, as applicable.Manufacturer is Source Dynamics, Inc., 22525 SE 64th Place, Suite 260, Issaquah, WA 98027, USA.

11.MISCELLANEOUS.If you acquired the SOFTWARE in the United States, this EULA is governed by the laws of the state of Washington.If you acquired the SOFTWARE outside of the United States, then local laws may apply.

If you have any questions concerning this EULA or wish to contact Source Dynamics for any reason, please write: Source Dynamics, Inc., 22524 SE 64th Place, Suite 260, Issaquah, WA 98027, USA; or call (425) 557-3630; or send electronic mail to support@sourceinsight.com.