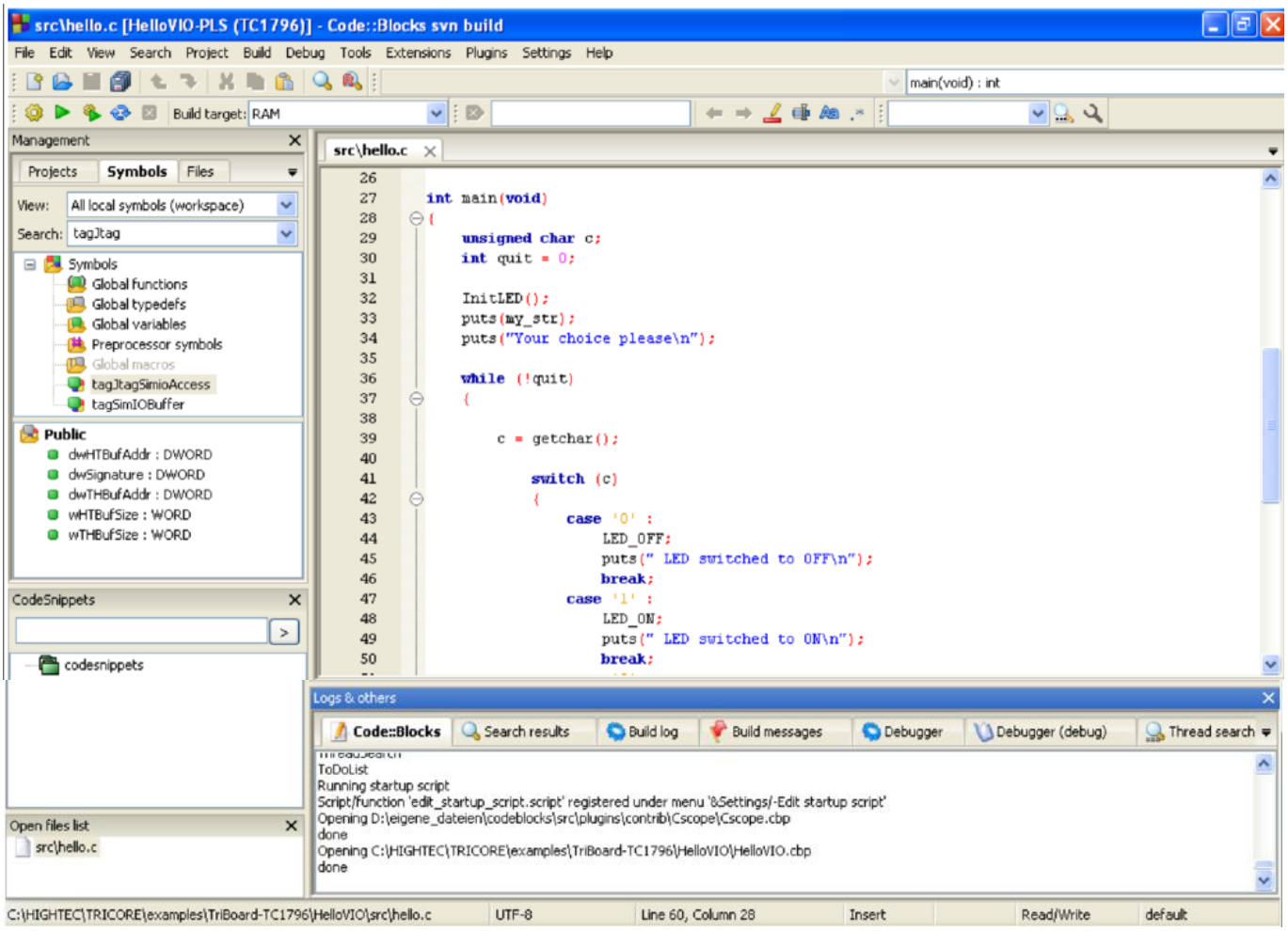
1. Zarządzanie projektami CodeBlocks

1.1. Widok projektu

W CodeBlocks źródła i ustawienia procesu kompilacji są przechowywane w pliku projektu <name>.cbp. Źródła C/C++ i odpowiadające im pliki nagłówkowe są typowymi składnikami projektu. Najprostszym sposobem utworzenia nowego projektu jest wykonanie polecenia „File” / „Project” i wybranie kreatora. Następnie możesz dodać pliki do projektu za pomocą menu kontekstowego „Add files” (Dodaj pliki) w oknie Management (Zarządzanie).

Instrukcje do rozdziału 3 i ?? są oficjalną dokumentacją witryny CodeBlocks Wiki i są dostępne tylko w języku angielskim.

Poniższa ilustracja przedstawia projekt interfejsu użytkownika CodeBlocks.



Rysunek 1.1: IDE CodeBlocks

Management (Zarządzanie)
To okno zawiera interfejs „Projects (Projekty)”, który w poniższym tekście będzie nazywany widokiem projektu. Ten widok pokazuje wszystkie projekty otwarte w CodeBlocks w określonym czasie. Karta „Symbols (Symbole)” okna Zarządzanie pokazuje symbole, zmienne itp.
Editor (Edytor)
Na powyższej ilustracji źródło o nazwie hello.c jest otwarte z podświetlaniem składni w edytorze.
Open files list (Lista otwartych plików)
pokazuje listę wszystkich plików otwartych w edytorze, w tym przykładzie: hello.c.

CodeSnippets (Fragmenty kodu)
można wyświetlić za pomocą menu „View (Widok)” / „CodeSnippets (Fragmenty kodu)”. Tutaj można zarządzać modułami tekstowymi, linkami do plików i linkami do adresów URL.
Logs & others (Logi i inne)
To okno służy do wyświetlania wyników wyszukiwania, komunikatów dziennika kompilatora itp.

Pasek stanu zawiera przegląd następujących ustawień:

- Ścieżka bezwzględna otwartego pliku w edytorze.
- Edytor używa domyślnego kodowania znaków systemu operacyjnego hosta. To ustawienie będzie wyświetlane jako **default** (domyślne).
- Numer wiersza i kolumny bieżącej pozycji kursora w edytorze.
- Skonfigurowany tryb klawiatury do wstawiania tekstu (Insert or Overwrite (Wstaw lub Nadpisz)).
- Aktualny stan pliku. Zmodyfikowany plik zostanie oznaczony jako Modified, w przeciwnym razie ten wpis będzie pusty.
- Uprawnienia pliku. Plik z ustawieniami tylko do odczytu będzie wyświetlany jako Read only na pasku stanu. W oknie „Open files list (Otwórz listę plików)” te pliki będą wyróżnione ikoną kłódki jako nakładką.

Uwaga:
W aktywnym edytorze użytkownik może wybrać właściwości menu kontekstowego. W pojawiającym się oknie dialogowym na karcie „General (Ogólne)” można wybrać opcję „File is read-only (Plik jest tylko do odczytu)”. Ta opcja spowoduje dostęp tylko do odczytu odpowiedniego pliku w CodeBlocks, ale oryginalne atrybuty odczytu i zapisu pliku w systemie plików nie zostaną zmodyfikowane.

Jeśli uruchomisz CodeBlocks z opcją wiersza poleceń `--personality=<profile>`, pasek stanu pokaże aktualnie używany profil, w przeciwnym razie zostanie wyświetlony domyślny. Ustawienia CodeBlocks są przechowywane w odpowiednim pliku konfiguracyjnym `<personality>.conf`.

CodeBlocks oferuje bardzo elastyczne i kompleksowe zarządzanie projektami. Poniższy tekst będzie dotyczył tylko niektórych funkcji zarządzania projektami.

1.1 Project View (Widok projektu)

W CodeBlocks źródła i ustawienia procesu kompilacji są przechowywane w pliku projektu `<name>.cbp`. Źródła C/C++ i odpowiadające im pliki nagłówkowe są typowymi składnikami projektu. Najprostszym sposobem utworzenia nowego projektu jest wykonanie polecenia „File (Plik)” / „Project (Projekt)” i wybranie kreatora. Następnie możesz dodać pliki do projektu za pomocą menu kontekstowego „Add files (Dodaj pliki)” w oknie Management (Zarządzanie).

1.2 Notes for Projects (Notatki dotyczące projektów)

W CodeBlocks można przechowywać tzw. notatki dla projektu. Notatki te powinny zawierać krótkie opisy lub wskazówki dotyczące odpowiedniego projektu. Wyświetlając te informacje podczas otwierania projektu, inni użytkownicy otrzymują szybki przegląd projektu. Wyświetlanie notatek można włączyć lub wyłączyć na karcie Notes (Notatki) w Properties (Właściwościach) projektu.

1.3 Project Templates (Szablony projektów)

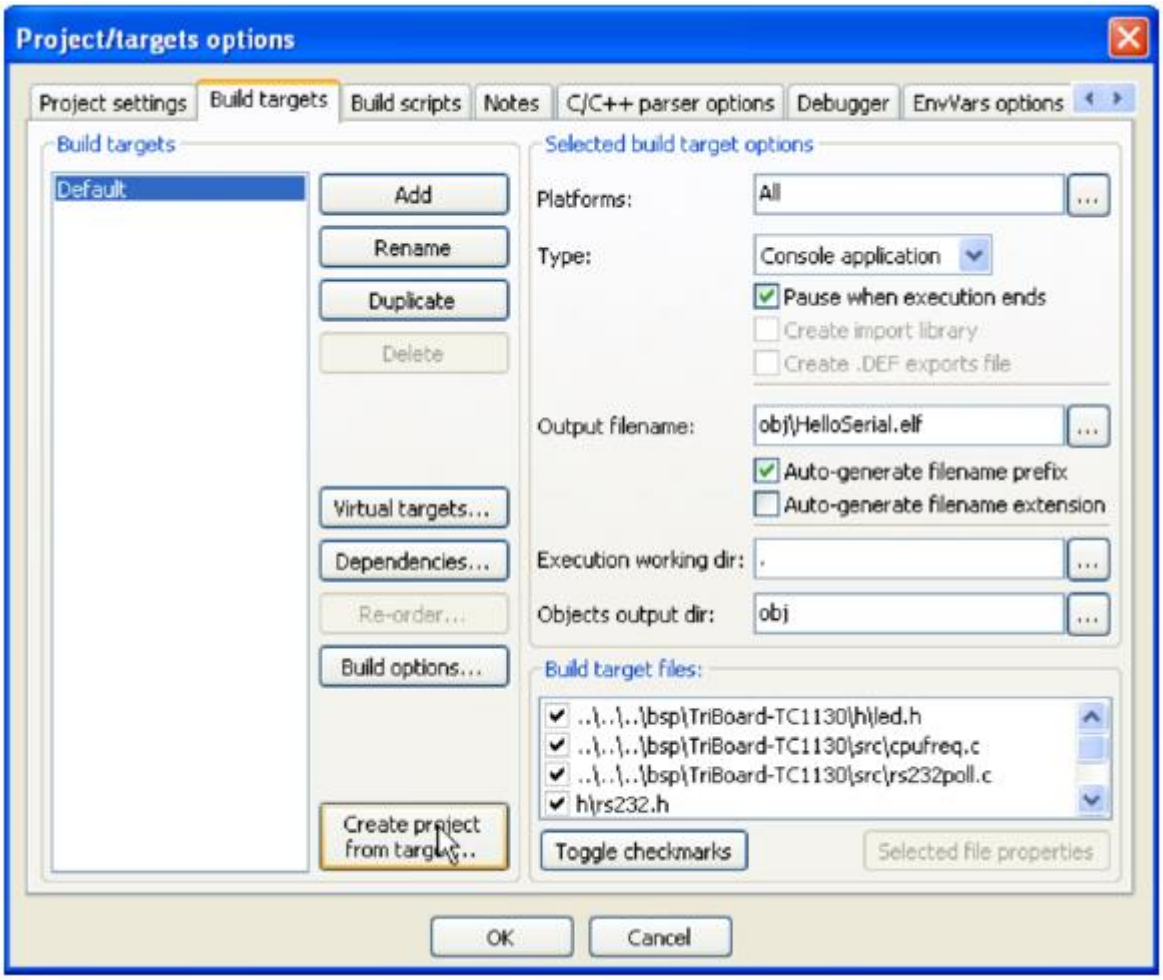
CodeBlocks jest dostarczany z różnymi szablonami projektów, które są wyświetlane podczas tworzenia nowego projektu. Jednak możliwe jest również przechowywanie niestandardowych szablonów w celu zebrania własnych specyfikacji dla przełączników kompilatora, optymalizacji, która ma być użyta, przełączników specyficznych dla maszyny itp. w szablonach. Szablony te będą przechowywane

w katalogu `Documents and Settings\<user>\Application Data\codeblocks\UserTemplates`. Jeśli szablony mają być otwarte dla wszystkich użytkowników, muszą zostać skopiowane do odpowiedniego katalogu instalacji CodeBlocks. Te szablony zostaną wyświetlone przy następnym uruchomieniu CodeBlocks w obszarze „New (Nowy)” / „Project (Projekt)” / „User templates (Szablony użytkownika)”.

Uwaga:
Dostępne szablony w Project Wizard (Kreatorze projektu) można edytować, wybierając je prawym przyciskiem myszy.

1.4 Create Projects from Build Targets (Tworzenie projektów z celów kompilacji)

W projektach konieczne jest posiadanie różnych wariantów projektu. Warianty są nazywane celami kompilacji. Różnią się one pod względem opcji kompilatora, informacji debugowania i/lub wyboru plików. Cel kompilacji może być również przekazany do osobnego projektu. Aby to zrobić, kliknij „Project (Projekt)” / „Properties (Właściwości)”, wybierz wariant z zakładki „Build Targets (Cele kompilacji)” i kliknij przycisk „Create project from target (Utwórz projekt z celu)” (patrz Rysunek 1.2).



Rysunek 1.2: Build Targets (Cele kompilacji)

1.5 Virtual Targets (Cele wirtualne)

Projekty mogą być dalej strukturyzowane w CodeBlocks przez tak zwane Virtual Targets. Często używana struktura projektu składa się z dwóch Build Targets, jednego 'Debug' Target, który zawiera informacje debugowania i jednego 'Release' Target bez tych informacji. Poprzez dodanie Virtual Targets poprzez 'Project'/'Properties (Właściwości)'/'Build Targets' poszczególne Build Targets mogą być łączone.

Na przykład Virtual Target 'All (Wszystkie)' może tworzyć Targets Debug i Release jednocześnie. Virtual Targets są wyświetlane na pasku symboli kompilatora pod Build Targets.

1.6 Pre- and Postbuild steps (Kroki przed i po kompilacji)

CodeBlocks umożliwia wykonywanie dodatkowych operacji przed lub po skompilowaniu projektu. Operacje te nazywane są Prebuilt lub Postbuilt Steps. Typowe kroki Postbuilt to:

- Tworzenie Intel Hexformat z gotowego obiektu
- Manipulowanie obiektami przez objcopy
- Generowanie plików zrzutu przez objdump

Przykład:

Tworzenie Disasembly (Demontaż) z obiektu w systemie Windows. Przekierowanie do pliku wymaga wywołania cmd z opcją /c.

```
cmd /c objdump -D name.elf > name.dis
```

Archiwizowanie projektu może być kolejnym przykładem kroku Postbuilt. W tym celu utwórz cel kompilacji „Archive (Archiwum)” i uwzględnij następującą instrukcję w kroku Postbuilt:

```
zip -j9 $(PROJECT_NAME)_$(TODAY).zip src h obj $(PROJECT_NAME).cbp
```

Za pomocą tego polecenia aktywny projekt i jego źródła, nagłówek i obiekty zostaną spakowane jako plik zip. W ten sposób zostaną wyodrębnione zmienne wbudowane $(PROJECT_NAME) i $(TODAY), nazwa projektu i bieżąca data (patrz sekcja 3.2). Po wykonaniu polecenia Target 'Archive' spakowany plik zostanie zapisany w katalogu projektu.

W katalogu `share/codeblocks/scripts` znajdziesz kilka przykładów skryptów. Możesz dodać skrypt za pomocą menu „Settings (Ustawienia)”/„Scripting (Tworzenie skryptów)” i zarejestrować się w menu. Jeśli wykonasz np. skrypt `make_dist` z menu, wszystkie pliki należące do projektu zostaną skompresowane w archiwum `<project>.tar.gz`.

1.7 Adding Scripts in Build Targets (Dodawanie skryptów w celach kompilacji)

CodeBlocks oferuje możliwość używania akcji menu w skryptach. Skrypt reprezentuje kolejny stopień swobody w kontrolowaniu generowania projektu.

Uwaga:
Skrypt można również uwzględnić w celu kompilacji.

1.8 Workspace and Project Dependencies (Zależności obszaru roboczego i projektu)

W CodeBlocks można otworzyć wiele projektów. Zapisując otwarte projekty za pomocą 'File (Plik)' / ' Save workspace (Zapisz obszar roboczy)' możesz zebrać je w jednym obszarze roboczym pod `<name>.workspace`. Jeśli otworzysz `<name>.workspace` podczas następnego uruchomienia w CodeBlocks, wszystkie projekty pojawią się ponownie.

Złożone systemy oprogramowania składają się z komponentów, które są zarządzane w różnych projektach CodeBlocks. Ponadto, przy generowaniu takich systemów oprogramowania, często występują zależności między tymi projektami.

Przykład:

Projekt A zawiera podstawowe funkcje, które są udostępniane innym projektom w formie biblioteki. Teraz, jeśli źródła tego projektu zostaną zmodyfikowane, biblioteka musi zostać przebudowana. Aby zachować spójność między projektem B, który używa funkcji, a projektem A, który implementuje funkcje, projekt B musi zależeć od projektu A. Niezbędne informacje o zależnościach projektów są przechowywane w odpowiednim obszarze roboczym, dzięki czemu każdy projekt można utworzyć osobno. Użycie zależności umożliwia również kontrolowanie kolejności, w jakiej będą generowane projekty. Zależności dla projektów można ustawić, wybierając menu „Project (Projekt)" / „Properties (Właściwości)", a następnie klikając przycisk „Project's dependencies (Zależności projektu)".

1.9 Including Assembler files (Dołączanie plików Assemblera)

W oknie Management zarządzania widoku projektu pliki asemblera są wyświetlane w kategorii źródeł ASM. Użytkownik może zmienić listę plików w kategoriach (patrz sekcja 1.1). Kliknięcie prawym przyciskiem myszy jednego z wymienionych plików asemblera spowoduje otwarcie menu kontekstowego. Wybierz „Properties (Właściwości)", aby otworzyć nowe okno. Teraz wybierz kartę „Build (Kompiluj)" i aktywuj dwa pola „Compile file (Kompiluj plik)" i „Link file (Połącz plik)". Następnie wybierz kartę „Advanced (Zaawansowane)" i wykonaj następujące kroki:

1. Ustaw „Compiler variable (Zmienną kompilatora)" na CC
2. Wybierz kompilator w „For this compiler (Dla tego kompilatora)"
3. Wybierz „Use custom command to build this file (Użyj polecenia niestandardowego, aby zbudować ten plik)"
4. W oknie enter (wprowadź):

```
$compiler $options $includes <asopts> -c $file -o $object
```

Zmienne CodeBlocks są oznaczone $ (patrz sekcja 3.4). Są one ustawiane automatycznie, więc wystarczy zastąpić opcję Assemblera <asopt> własnymi ustawieniami.

1.10 Editor and Tools (Edytor i narzędzia)

1.10.1 Default Code (Kod domyślny)

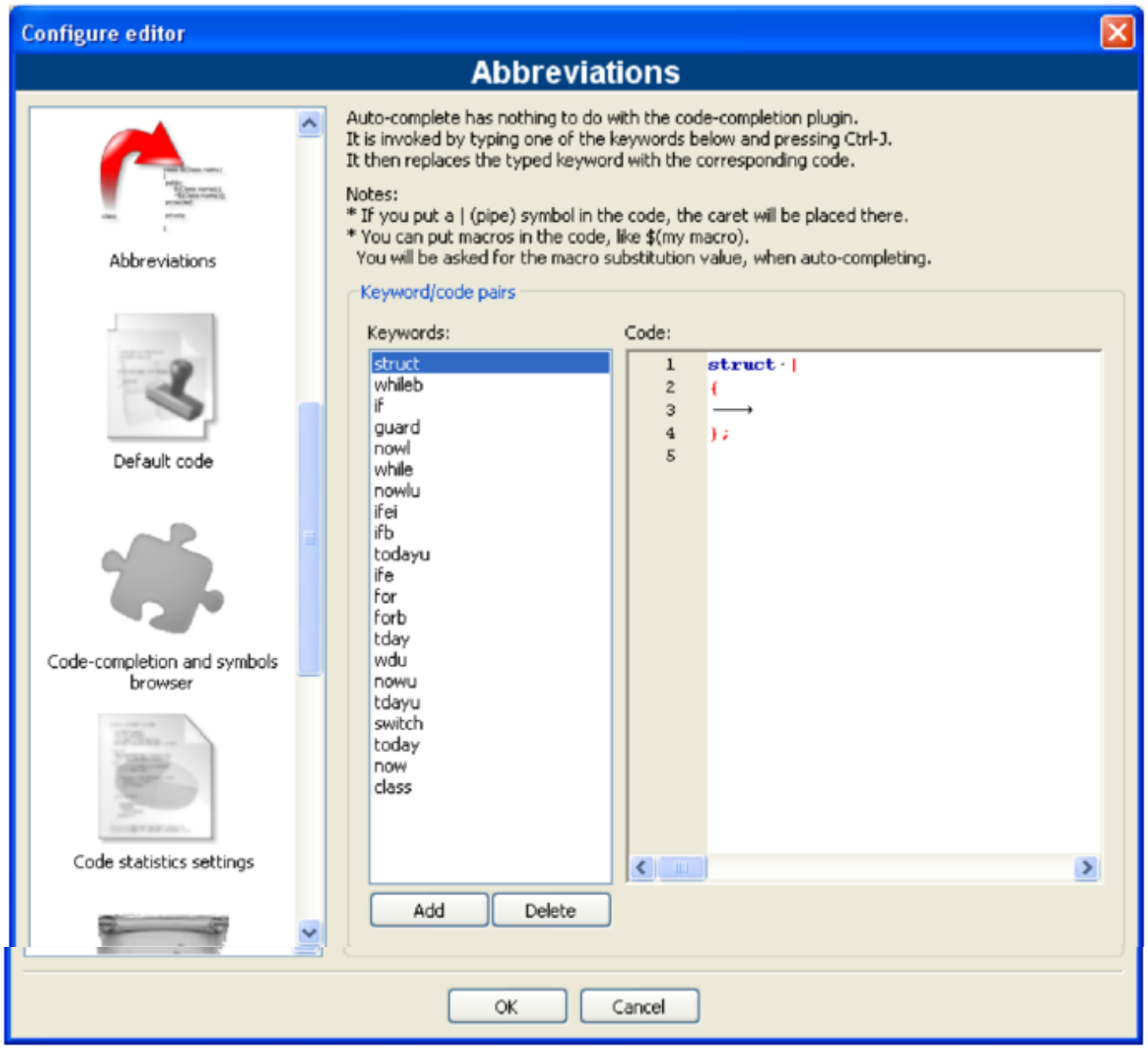
Zasady kodowania firmy wymagają, aby pliki źródłowe miały standardowy projekt. CodeBlocks umożliwia automatyczne dołączanie wstępnie zdefiniowanej zawartości na początku pliku podczas tworzenia nowych źródeł i nagłówków C/C++. Ta wstępnie zdefiniowana zawartość nazywana jest domyślnym kodem. To ustawienie można wybrać w „Stettings (Ustawienia)" / „Editor (Edytor)" Default Code. Jeśli utworzysz nowy plik, zostanie wykonana makroekspansja zmiennych, np. zdefiniowanych za pomocą menu „Settings" / „Global variables (Zmienne globalne)". Nowy plik można utworzyć za pomocą menu „File (Plik)" / „New (Nowy)" / „File".

Przykład:

```
/*************************************************************************
 * Project: $(proejct)
 * Function:
 *************************************************************************
 * $Author: mario $
 * $Name:  $
 *************************************************************************
 *
 * Copyright 2007 by company name
 *
 *************************************************************************/
```

1.10.2 Abbreviation (Skrót)

Wiele pisania można zaoszczędzić w CodeBlocks, definiując skróty. Można to zrobić, wybierając „Settings (Ustawienia)”/„Editor (Edytor)” i definiując skróty pod nazwą <nazwa>, którą można następnie wywołać za pomocą skrótu klawiaturowego Ctrl-J (patrz Rysunek 1.3).



Rysunek 1.3: Definiowanie skrótów

Parametryzacja jest możliwa również poprzez uwzględnienie w skrótach zmiennych $(NAME)

```
#ifndef $(Guard token)
 #define $(Guard token)
 #endif // $(Guard token)
```

Podczas wykonywania skrótu <nazwa> w tekście źródłowym i wykonywania kombinacji klawiszy Ctrl-J żądana jest i uwzględniana zawartość zmiennej.

1.10.3 Personalities (Osobowości)

Ustawienia CodeBlocks są zapisywane jako dane aplikacji w pliku o nazwie `<user>.conf` w katalogu codeblocks. Ten plik konfiguracyjny zawiera informacje takie jak ostatnio otwarte projekty, ustawienia edytora, wyświetlanie pasków symboli itp. Domyślnie „default (domyślna)" osobowość jest ustawiona tak, że konfiguracja jest przechowywana w pliku `default.conf`. Jeśli CodeBlocks zostanie wywołany z wiersza poleceń z parametrem `--personality=myuser`, ustawienia zostaną zapisane w pliku `myuser.conf`. Jeśli profil jeszcze nie istnieje, zostanie automatycznie utworzony. Ta procedura umożliwia utworzenie odpowiednich profili dla różnych etapów pracy. Jeśli uruchomisz CodeBlocks z wiersza poleceń z dodatkowym parametrem `--personality=ask`, zostanie wyświetlone pole wyboru dla wszystkich dostępnych profili.

Uwaga:
Nazwa bieżącego profilu/osobowości jest wyświetlana w prawym rogu paska stanu.

1.10.4 Configuration Files (Pliki konfiguracyjne)

Ustawienia CodeBlocks są przechowywane w profilu `default.conf` w katalogu `codeblocks` w Application Data. Podczas korzystania z personalities (patrz podsekcja 1.10.3) szczegóły konfiguracji zostaną zapisane w pliku `<personality>.conf`.

Narzędzie `cb_share_conf`, które można znaleźć w katalogu instalacyjnym CodeBlocks, służy do zarządzania tymi ustawieniami i ich przechowywania.

Jeśli chcesz zdefiniować standardowe ustawienia dla kilku użytkowników komputera, plik konfiguracyjny `default.conf` musi być zapisany w katalogu `\Documents and Settings\Default User\Application Data\codeblocks`. Podczas pierwszego uruchomienia CodeBlocks skopiuje ustawienia wstępne z „Default User (Domyślny użytkownik)" do danych aplikacji bieżących użytkowników.

Aby utworzyć przenośną wersję CodeBlocks na pamięci USB, wykonaj następujące czynności. Skopiuj instalację CodeBlocks na pamięć USB i zapisz plik konfiguracyjny `default.conf` w tym katalogu. Konfiguracja zostanie użyta jako ustawienie globalne. Upewnij się, że plik jest zapisywalny, w przeciwnym razie zmiany konfiguracji nie mogą zostać zapisane.

1.10.5 Navigate and Search (Nawigacja i wyszukiwanie)

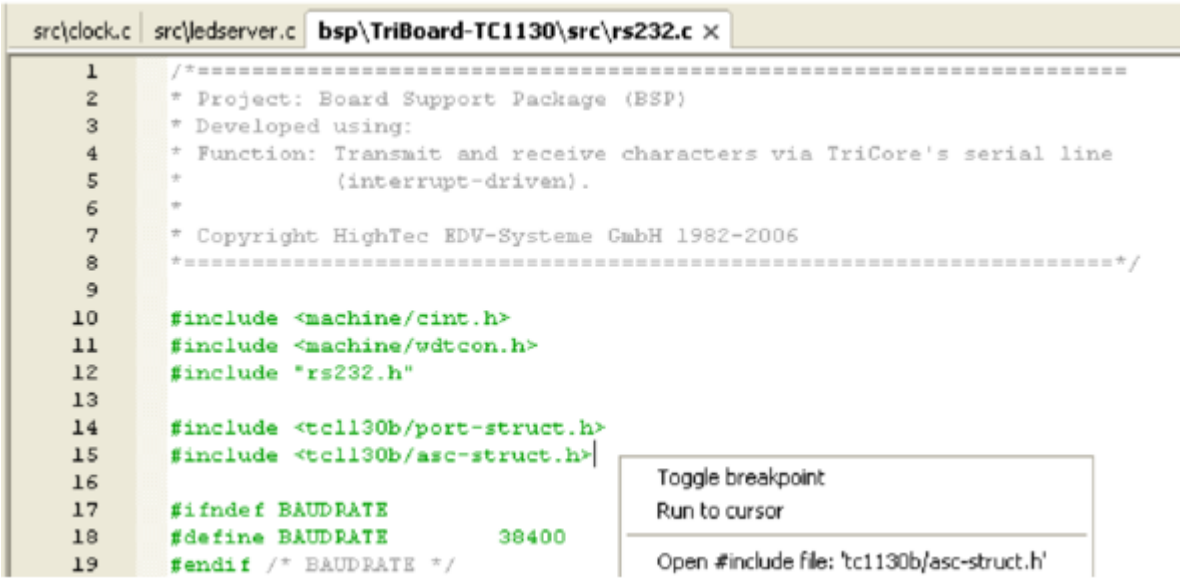
W CodeBlocks istnieją różne sposoby szybkiej nawigacji między plikami i funkcjami. Ustawianie zakładek to typowa procedura. Za pomocą skrótu Ctrl-B zakładka jest ustawiana lub usuwana w pliku źródłowym. Za pomocą Alt-PgUp możesz przejść do poprzedniej zakładki, a za pomocą Alt-PgDn możesz przejść do następnej zakładki.

Jeśli wybierzesz obszar roboczy lub projekt w obszarze roboczym w widoku projektu, będziesz mógł wyszukać plik w projekcie. Wystarczy wybrać „Znajdź plik" z menu kontekstowego, a następnie wpisać nazwę pliku, a plik zostanie wybrany. Jeśli naciśniesz Enter, ten plik zostanie otwarty w edytorze (patrz Rysunek 1.4).

Rysunek 1.4: Wyszukiwanie plików

W CodeBlocks możesz łatwo poruszać się pomiędzy plikami nagłówkowymi/źródłowymi, takimi jak:

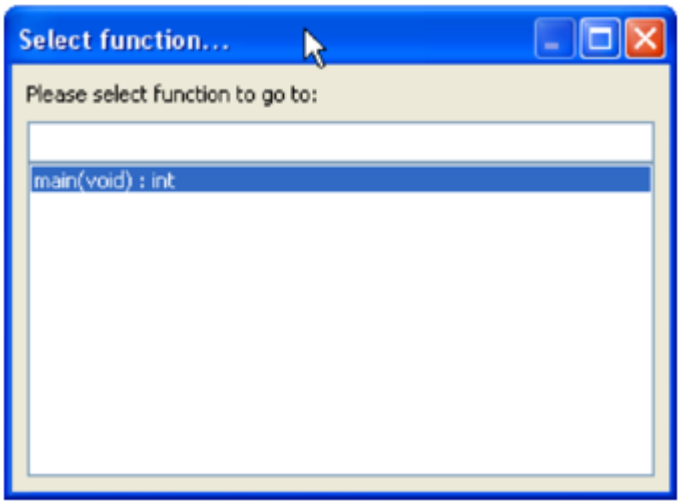
1. Ustaw kursor w miejscu, w którym znajduje się plik nagłówkowy, i otwórz ten plik za pomocą menu kontekstowego „open include file (otwórz plik dołączony)” (patrz Rysunek 1.5)
2. Zamień nagłówek i źródło za pomocą menu kontekstowego „Swap header/source (Zamień nagłówek/źródło)”
3. Wybierz np. definicję w edytorze i wybierz „Find declaration (Znajdź deklarację)” z menu kontekstowego, aby otworzyć plik z jego deklaracją.



Rysunek 1.5: Otwieranie pliku nagłówkowego

CodeBlocks oferuje kilka sposobów wyszukiwania w pliku lub katalogu. Okno dialogowe wyszukiwania otwiera się za pomocą „Search (Szukaj)”/„Find (Znajdź)” (Ctrl-F) lub „Find in Files (Znajdź w plikach)” (Ctrl-Shift-F).

Alt-G i Ctrl-Alt-G to kolejne przydatne funkcje. Dialog, który otwiera się po użyciu tego skrótu, pozwala wybrać pliki/funkcje, a następnie przejść do implementacji wybranej funkcji (patrz Rysunek 1.6) lub otworzyć wybrany plik w edytorze. Możesz użyć symboli wieloznacznych, takich jak * lub ? itp., aby przeprowadzić wyszukiwanie przyrostowe w oknie dialogowym.



Rysunek 1.6: Wyszukiwanie funkcji

Uwaga:
Za pomocą skrótu Ctrl-PgUp możesz przejść do poprzedniej funkcji, a za pomocą Ctrl-PgDn możesz przejść do następnej funkcji.

W edytorze możesz otworzyć nowe okno dialogowe Open Files (Otwórz pliki) za pomocą Ctrl-Tab i możesz przełączać się między wymienionymi wpisami. Jeśli naciśnięty jest klawisz Ctrl, plik można wybrać na różne sposoby:

1. Jeśli zaznaczysz wpis lewym przyciskiem myszy, wybrany plik zostanie otwarty.
2. Jeśli naciśniesz klawisz Tab, będziesz przełączać się między wymienionymi wpisami. Zwolnienie klawisza Crtl spowoduje otwarcie wybranego pliku.
3. Jeśli przesuniesz mysz nad wymienionymi wpisami, bieżący wybór zostanie podświetlony. Zwolnienie klawisza Crtl spowoduje otwarcie wybranego pliku.
4. Jeśli wskaźnik myszy znajduje się poza wyróżnionym wyborem, możesz użyć kółka myszy, aby przełączać się między wpisami. Zwolnienie klawisza Crtl spowoduje otwarcie wybranego pliku.

Częstą procedurą przy tworzeniu oprogramowania jest zmaganie się z zestawem funkcji, które są implementowane w różnych plikach. Wtyczka Browse Tracker pomoże Ci rozwiązać ten problem, pokazując kolejność, w jakiej pliki zostały wybrane. Następnie możesz wygodnie nawigować po wywołaniach funkcji (patrz sekcja 2.8).

Wyświetlanie numerów wierszy w CodeBlocks można aktywować za pomocą „Settings (Ustawień)”/„ General Settings (Ustawień ogólnych)” w polu „Show line numbers (Pokaż numery wierszy)”. Skrót Ctrl-G lub polecenie menu „Search (Szukaj)”/„Goto line (Przejdź do wiersza)” pomoże Ci przejść do żądanego wiersza.

Uwaga:
Jeśli przytrzymasz klawisz Ctrl, a następnie zaznaczysz tekst w edytorze CodeBlocks, możesz wykonać np. wyszukiwanie w Google za pomocą menu kontekstowego.

1.10.6 Symbol view (Widok symboli)

Okno CodeBlocks Management (Zarządzanie) oferuje widok drzewa dla symboli źródeł C/C++ do nawigacji za pomocą funkcji lub zmiennych. Jako zakres tego widoku możesz ustawić bieżący plik lub projekt albo cały obszar roboczy.

Uwaga:
Wpisanie terminu wyszukiwania lub nazw symboli w masce wprowadzania „Search (Szukaj)” w Symbol Browser (Przeglądarce symboli) powoduje wyświetlenie przefiltrowanego widoku symboli, jeśli wystąpiły jakieś trafienia.

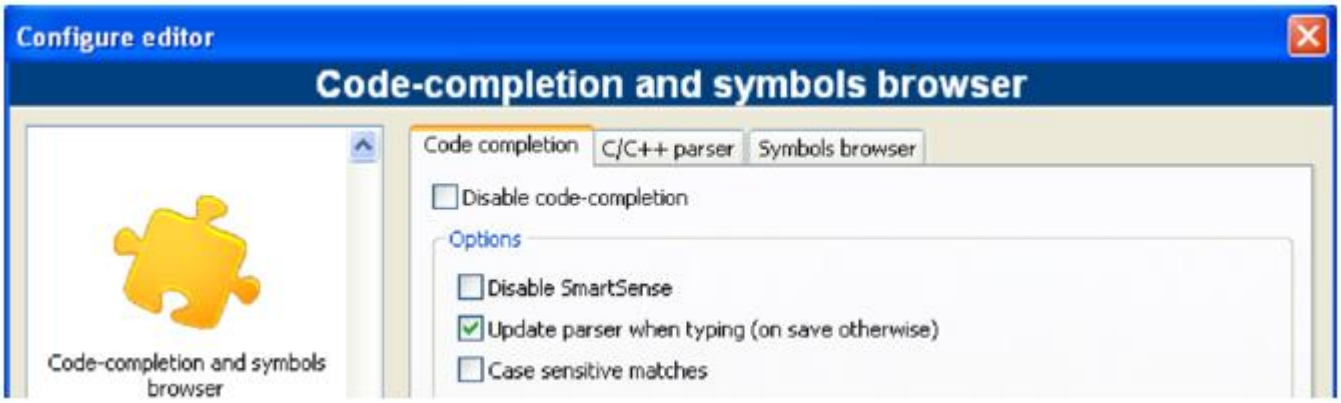
Istnieją następujące kategorie symboli:

Global functions (Funkcje globalne)
Wyświetla implementację funkcji globalnych.
Global typedefs (Globalne definicje typów)
Wyświetla użycie definicji typów (**typedef**).
Global variables (Zmienne globalne)
Wyświetla symbole zmiennych globalnych.
Preprocessor symbols (Symbole preprocesora)
Wyświetla dyrektywy preprocesora utworzone przez **#define**.
Global macros (Makra globalne)
Wyświetla makra dyrektyw preprocesora.

Rysunek 1.7: Widok symbolu

Struktury i klasy są wyświetlane w „bottom tree (dolnym drzewie)”, a kolejność sortowania można modyfikować za pomocą menu kontekstowego. Jeśli kategoria zostanie wybrana kliknięciem myszy, znalezione symbole zostaną wyświetlone w dolnej części okna (patrz Rysunek 1.7). Dwukrotne kliknięcie symbolu spowoduje otwarcie pliku, w którym symbol jest zdefiniowany lub zaimplementowana funkcja,

i przejście do odpowiedniego wiersza. Automatyczne odświeżanie przeglądarki symboli bez zapisywania pliku można aktywować za pomocą menu „Settings (Ustawienia)” / „Editor (Edytor)” / „Code Completion (Uzupełnianie kodu)” (patrz Rysunek 1.8). W przypadku projektów z wieloma symbolami wydajność w CodeBlocks będzie miała wpływ.



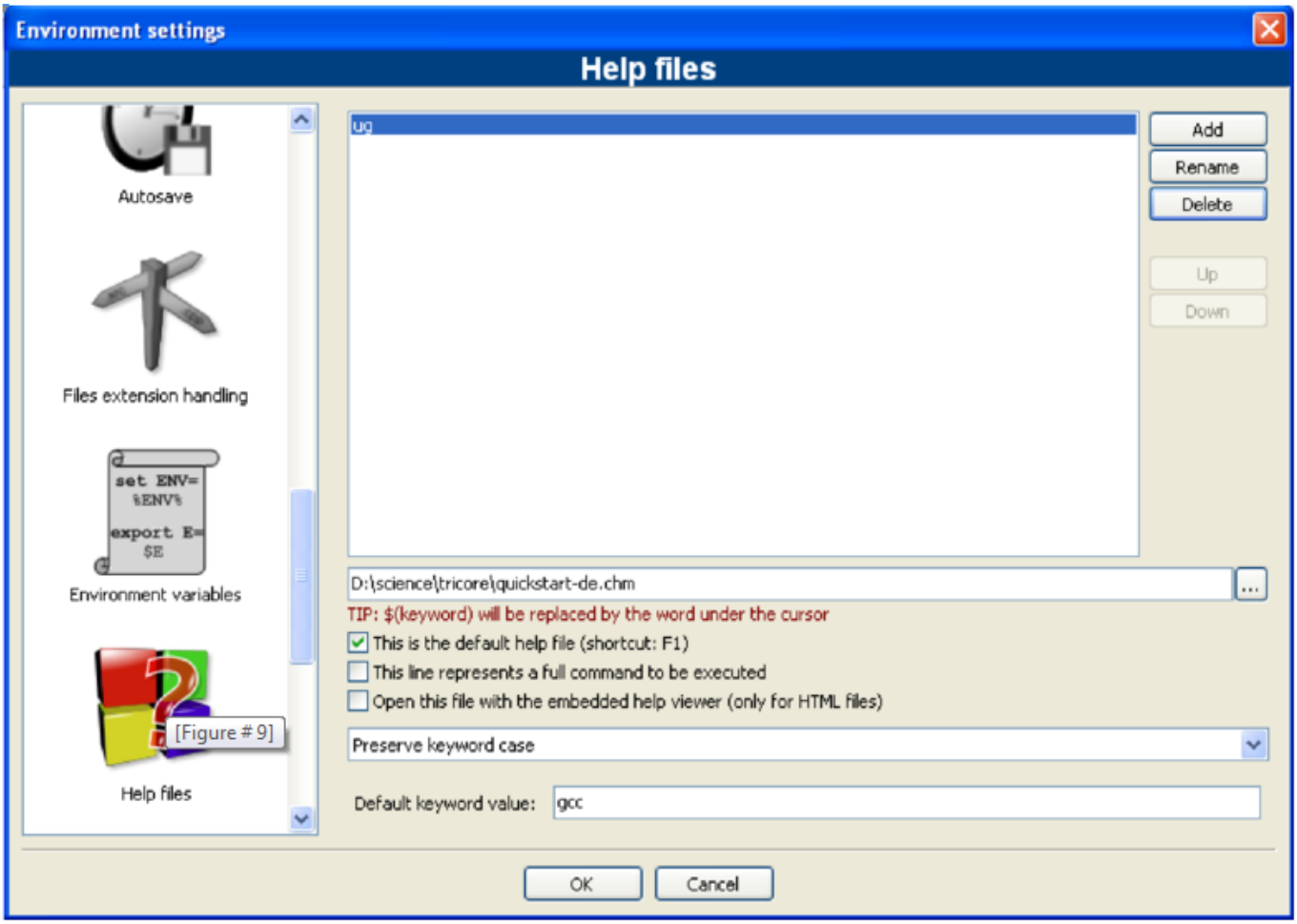
Rysunek 1.8: Włączanie analizy w czasie rzeczywistym

Uwaga:
W edytorze listę klas można wyświetlić za pomocą menu kontekstowego „Insert Class method declaration implementation (Wstaw implementację deklaracji metody klasy)” lub „All class methods without implementation (Wszystkie metody klasy bez implementacji)”

1.10.7 Including external help files (Dołączanie zewnętrznych plików pomocy)

Środowisko programistyczne CodeBlocks obsługuje dołączanie zewnętrznych plików pomocy za pomocą menu „Settings (Ustawienia)”/„ Environment (Środowisko)”. Dołącz wybrany podręcznik w formacie chm w „Help Files (Pliki pomocy)” i wybierz „this is the default help file (to jest domyślny plik pomocy)” (patrz Rysunek 1.9). Wpis $(słowo kluczowe) jest symbolem zastępczym dla wybranego elementu w edytorze. Teraz możesz wybrać funkcję w otwartym pliku źródłowym w CodeBlocks, klikając myszką, a odpowiednia dokumentacja pojawi się po naciśnięciu klawisza F1.

Jeśli dołączyłeś wiele plików pomocy, możesz zaznaczyć termin w edytorze i wybrać plik pomocy z menu kontekstowego „Locate in (Znajdź w)”, aby CodeBlocks mógł go przeszukać.

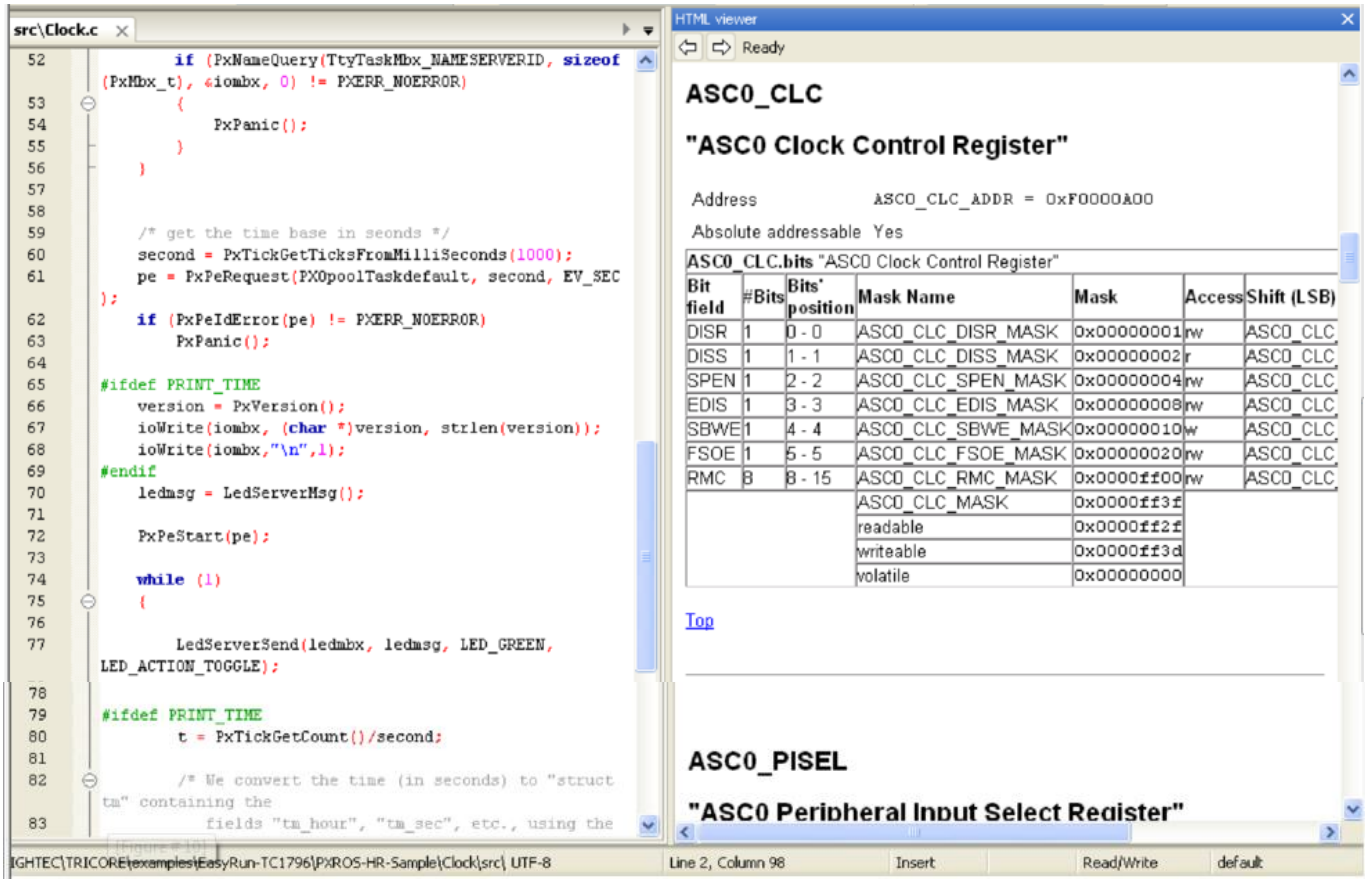


Rysunek 1.9: Ustawienia plików pomocy

W CodeBlocks możesz dodać nawet obsługę stron man. Wystarczy dodać wpis „man” i określić ścieżkę w następujący sposób:

```
man:/usr/share/man
```

CodeBlocks udostępnia „Embedded HTML Viewer (Wbudowana przeglądarka HTML)”, którego można użyć do wyświetlania prostego pliku html i znajdowania słów kluczowych w tym pliku. Wystarczy skonfigurować ścieżkę do pliku html, który powinien zostać przeanalizowany i zaznaczyć pole wyboru „Open this file with embedded help viewer (Otwórz ten plik za pomocą osadzonej przeglądarki pomocy)” za pomocą menu „Settings (Ustawienia)” / „Environment (Środowisko)” / „Help Files (Pliki pomocy)”.



Rysunek 1.10: Wbudowana przeglądarka HTML

Uwaga:
Jeśli wybierzesz plik HTML poprzez dwukrotne kliknięcie w eksploratorze plików (patrz sekcja 2.7), zostanie uruchomiony osadzony program do przeglądania plików HTML, o ile w programie do obsługi rozszerzeń plików nie zostanie utworzone żadne skojarzenie plików HTML.

1.10.8 Including external tools (Dołączanie narzędzi zewnętrznych)

Dołączanie narzędzi zewnętrznych jest możliwe w CodeBlocks poprzez „Tools (Narzędzia)” / „Configure Tools (Konfiguruj narzędzia)” / „Add (Dodaj)”. Wbudowane zmienne (patrz sekcja 3.2) są również dostępne dla parametrów narzędzi. Ponadto istnieje kilka rodzajów opcji uruchamiania aplikacji zewnętrznych. W zależności od opcji, aplikacje uruchamiane zewnętrznie są zatrzymywane po zamknięciu CodeBlocks. Jeśli aplikacje mają pozostać otwarte po zamknięciu CodeBlocks, należy ustawić opcję „Launch tool visible detached (Uruchom widoczne odłączone narzędzie)”.

1.11 Tips for working with CodeBlocks (Wskazówki dotyczące pracy z CodeBlocks)

W tym rozdziale przedstawimy kilka przydatnych ustawień w CodeBlocks.

1.11.1 Tracking of Modifications (Śledzenie modyfikacji)

CodeBlocks udostępnia funkcję śledzenia modyfikacji w pliku źródłowym i wyświetlania paska na marginesie dla zmian. Modyfikacje są oznaczone żółtym paskiem zmian, a modyfikacje, które zostały już zapisane, będą używać zielonego paska zmian (patrz Rysunek 1.11). Możesz poruszać się między swoimi zmianami za pomocą menu „Search (Szukaj)” / „Goto next changed line (Przejdź do następnego zmienionego wiersza)” lub „Search” / „Goto previous changed line (Przejdź do poprzedniego zmienionego wiersza)”. Ta sama funkcjonalność jest również dostępna za pomocą skrótów Ctrl-F3 i Ctrl-Shift-F3.

Rysunek 1.11: Śledzenie modyfikacji

Funkcję tę można włączyć lub wyłączyć za pomocą pola wyboru „Use Changebar (Użyj paska zmian)” w menu „Settings (Ustawienia)” / „Editor (Edytor)” / „Margins and caret (Marginesy i kursor)”.

Uwaga:
Jeśli zmodyfikowany plik zostanie zamknięty, historia zmian, taka jak cofanie/ponawianie i paski zmian, zostaną utracone. Za pomocą menu „Edit (Edycja)” / „Clear changes history (Wyczyść historię zmian)” lub odpowiedniego menu kontekstowego możesz wyczyścić historię zmian, nawet jeśli plik jest otwarty.

1.11.2 Data Exchange with other applications (Wymiana danych z innymi aplikacjami)

Dane mogą być wymieniane między CodeBlocks i innymi aplikacjami. Do tej komunikacji międzyprocesowej DDE (Dynamic Data Exchange) jest używane dla Windows i w różnych systemach operacyjnych jest to komunikacja oparta na TCP.

Za pomocą tego interfejsu można wysyłać do instancji CodeBlocks różne polecenia o następującej składni:

```
[<polecenie>("<parametr>")]
```

Polecenia te są obecnie dostępne:

Open (Otwórz)

Polecenie

```
[Open("d:\temp\test.txt")]
```

używa parametru, w naszym przypadku jest to plik określony za pomocą ścieżki bezwzględnej, i otwiera go w istniejącej instancji CodeBlocks lub uruchamia pierwszą instancję, jeśli jest to wymagane.

OpenLine

To polecenie otwiera plik pod podanym numerem wiersza w instancji CodeBlocks. Numer wiersza jest określony za pomocą `:line`.

```
[OpenLine("d:\temp\test.txt:10")]
```

Raise (Podnieś)

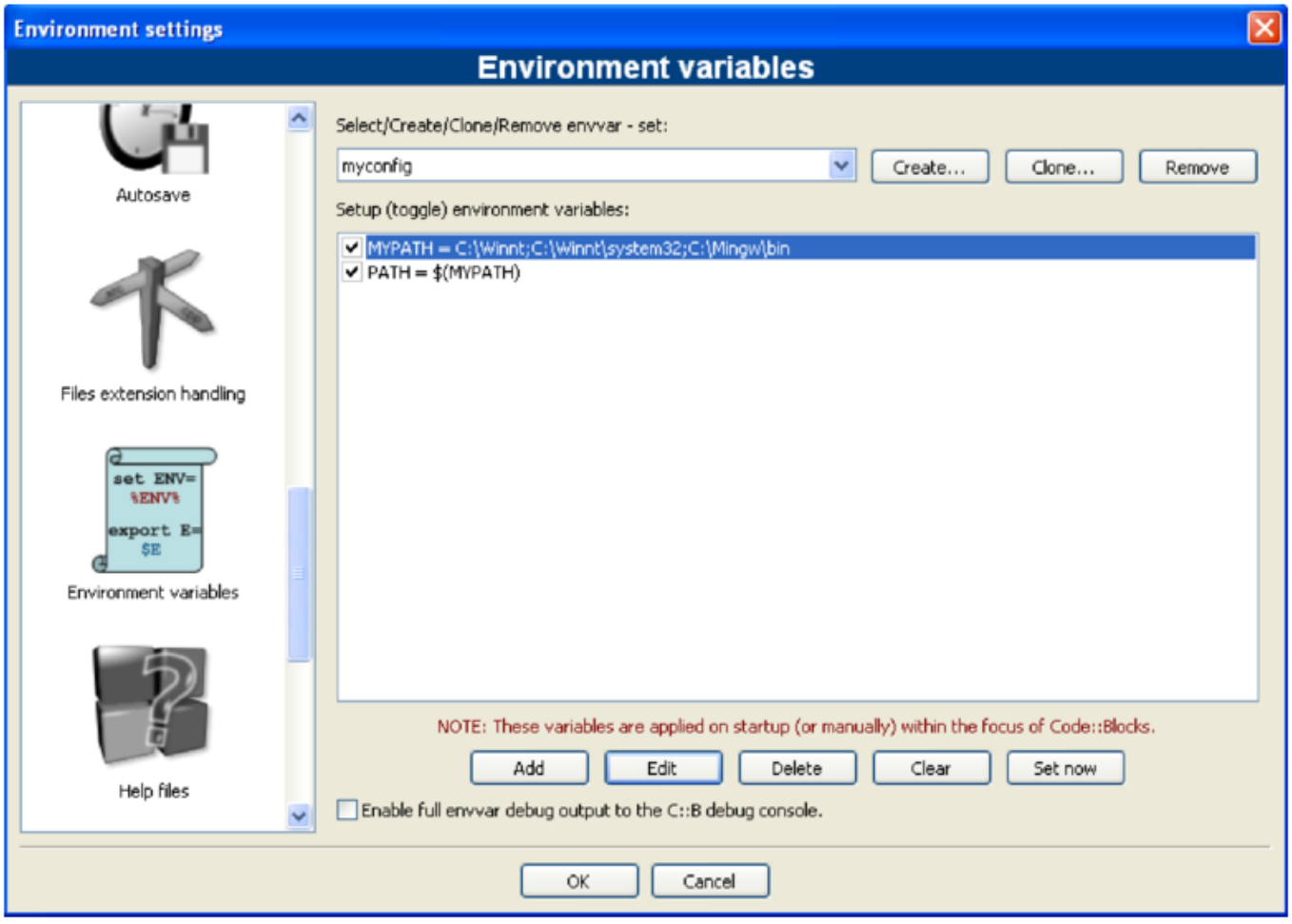
Ustaw fokus na instancję CodeBlocks. Nie można przekazać parametru.

1.11.3 Configuring environmental variables (Konfigurowanie zmiennych środowiskowych)

Konfiguracja systemu operacyjnego jest określana przez tzw. zmienne środowiskowe. Zmienna środowiskowa PATH na przykład zawiera ścieżkę do zainstalowanego kompilatora. System operacyjny będzie przetwarzał tę zmienną środowiskową od początku do końca, tj. wpisy na końcu będą przeszukiwane na końcu. Jeśli zainstalowane są różne wersje kompilatora lub innych aplikacji, mogą wystąpić następujące sytuacje:

- An incorrect version of a software is called (Nieprawidłowa wersja oprogramowania nazywana jest)
- Installed software packages call each other (Zainstalowane pakiety oprogramowania nazywają się wzajemnie)

Może się więc zdarzyć, że różne wersje kompilatorów lub innych narzędzi będą obowiązkowe dla różnych projektów. Jedną z możliwości w takim przypadku jest zmiana zmiennych środowiskowych w kontroli systemu dla każdego projektu. Jednak ta procedura jest podatna na błędy i nieelastyczna. W przypadku tego wymogu CodeBlocks oferuje eleganckie rozwiązanie. Można utworzyć różne konfiguracje zmiennych środowiskowych, które są używane tylko wewnętrznie w CodeBlocks. Ponadto można przełączać się między tymi konfiguracjami. Rysunek 1.12 przedstawia okno dialogowe, które można otworzyć za pomocą „Environment Varibales (Zmienne środowiskowe)" w „Settings (Ustawienia)" / „Environment (Środowisko)". Konfiguracja jest tworzona za pomocą przycisku „Create (Utwórz)".



Rysunek 1.12: Zmienne środowiskowe

Dostęp i zakres zmiennych środowiskowych utworzonych tutaj jest ograniczony do CodeBlocks. Możesz rozszerzyć te zmienne środowiskowe tak jak inne zmienne CodeBlocks za pomocą $(NAZWA).

Uwaga:
Konfigurację zmiennej środowiskowej dla każdego projektu można wybrać w menu kontekstowym „Properties (Właściwości)” na karcie „EnvVars options (Opcje EnvVars)”.

Przykład:

Możesz zapisać użyte środowisko w kroku postbuild (patrz sekcja 1.6) w pliku `<project>.env` i zarchiwizować je w swoim projekcie.

```
cmd /c echo \%PATH\% > project.env
```

lub pod Linuksem:

```
echo \$PATH > project.env
```

1.11.4 Switching between perspectives (Przełączanie między perspektywami)

W zależności od zadania, przydatne może być posiadanie różnych konfiguracji lub widoków w CodeBlocks i zapisywanie tych konfiguracji/widoków. Domyślnie ustawienia (np. pokazywanie/ukrywanie pasków symboli, układ itp.) są przechowywane w pliku konfiguracyjnym default.conf. Używając opcji wiersza poleceń --personality=ask podczas uruchamiania CodeBlocks, można wybrać różne ustawienia. Oprócz tego globalnego ustawienia, może wystąpić sytuacja, w której chcesz przełączać się między różnymi widokami okien i pasków symboli podczas sesji. Edycja plików i debugowanie projektów to dwa typowe przykłady takich sytuacji. CodeBlocks oferuje mechanizm przechowywania i wybierania różnych perspektyw, aby zapobiec częstemu otwieraniu i zamykaniu okien i pasków symboli ręcznie przez użytkownika. Aby zapisać perspektywę, wybierz menu „View (Widok)” / „Perspectives (Perspektywy)” / „Save current (Zapisz bieżącą)” i wprowadź nazwę w <nazwa>. Polecenie „Settings (Ustawienia)” / „Editor (Edytor)” / „Keyboard shortcuts (Skróty klawiaturowe)” / „View (Widok)” / „Perspectives (Perspektywy)” / „<nazwa>” umożliwia zdefiniowanie skrótu klawiaturowego dla tego procesu. Ten mechanizm umożliwia przełączanie się między różnymi widokami za pomocą klawiszy skrótu.

Uwaga:
Innym przykładem jest edycja pliku w trybie pełnego ekranu bez pasków symboli. Możesz utworzyć perspektywę, taką jak „Full (Pełna)” i przypisać do tego celu klawisz skrótu.

1.11.5 Switching between projects (Przełączanie się między projektami)

Jeśli kilka projektów lub plików jest otwartych w tym samym czasie, użytkownik potrzebuje sposobu na szybkie przełączanie się między projektami lub plikami. CodeBlocks ma wiele skrótów na takie sytuacje.

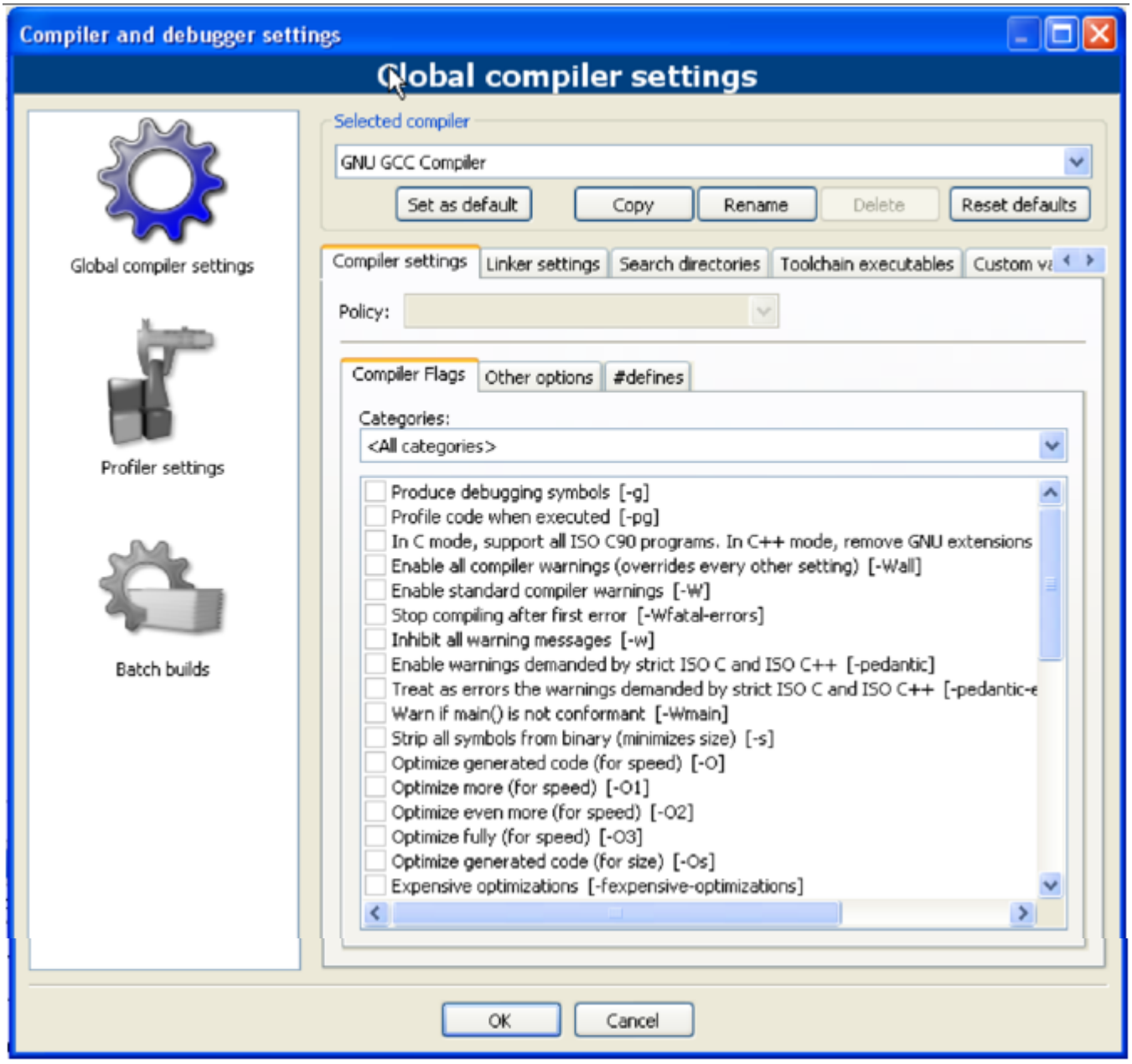
Alt-F5
Aktywuje poprzedni projekt z widoku projektu.

Alt-F6
Aktywuje następny projekt z widoku projektu.

F11
Przełącza w edytorze między plikiem źródłowym `<name>.cpp` a odpowiadającym mu plikiem nagłówkowym `<name>.h`

1.11.6 Extended settings for compilers (Rozszerzone ustawienia dla kompilatorów)

Podczas procesu kompilacji projektu komunikaty kompilatora są wyświetlane w oknie Messages (Wiadomości) na karcie Build Log (Dziennik kompilacji). Jeśli chcesz otrzymywać szczegółowe informacje, możesz rozszerzyć wyświetlanie. W tym celu kliknij „Settings (Ustawienia)” / „Compiler and Debugger (Kompilator i debuger)” i wybierz „Other Settings (Inne ustawienia)” w polu rozwijanym.



Rysunek 1.13: Ustawianie szczegółowych informacji

Upewnij się, że wybrano właściwy kompilator. Ustawienie „Full command line (Pełny wiersz poleceń)” w polu Compiler Logging wyprowadza kompletne informacje do Build Log. Ponadto dane wyjściowe można zapisać w pliku HTML. W tym celu wybierz opcję „Save build log to HTML file when finished (Zapisz dziennik kompilacji do pliku HTML po zakończeniu)”. Ponadto CodeBlocks oferuje pasek postępu dla procesu kompilacji w oknie Build Log, który można aktywować za pomocą ustawienia „Display build progress bar (Wyświetl pasek postępu kompilacji)”.

1.11.7 Zooming within the editor (Powiększanie w edytorze)

CodeBlocks oferuje bardzo wydajny edytor. Ten edytor pozwala zmienić rozmiar, w jakim wyświetlany jest otwarty tekst. Jeśli używasz myszy z kółkiem, wystarczy nacisnąć klawisz Ctrl i przewijać kółkiem myszy, aby powiększać i pomniejszać tekst.

Uwaga:
Skrótem Ctrl-Numepad-/ lub za pomocą menu „Edit (Edycja)" / „Special commands (Polecenia specjalne)" / „Zoom (Powiększenie)" / „Reset (Resetuj)" przywracany jest oryginalny rozmiar czcionki aktywnego pliku w edytorze.

1.11.8 Wrap Mode (Tryb zawijania)

Podczas edycji plików tekstowych, np. `*.txt`, w CodeBlocks, przydatne może być zawijanie tekstu, co oznacza, że długie wiersze będą wyświetlane w kilku wierszach na ekranie, aby można je było prawidłowo edytować. Funkcję „Word wrap (Zawijanie wierszy)" można aktywować za pomocą „Settings (Ustawienia)" / „Editor (Edytor)" / „Other Options (Inne opcje)" lub zaznaczając pole wyboru „Word wrap (Zawijanie wierszy)". Klawisze Home i End odpowiednio ustawiają kursor na początku lub na końcu zawiniętych wierszy. Podczas ustawiania „Ustawienia" / „Edytor" / „Inne opcje" i „Home key always move to caret to first column (Klawisz Home zawsze przesuwaj kursor do pierwszej kolumny)" kursor zostanie odpowiednio umieszczony na początku lub na końcu bieżącego wiersza, jeśli naciśnięto klawisze Home lub End. Jeśli pożądane jest umieszczenie kursora na początku pierwszego wiersza bieżącego akapitu, należy użyć kombinacji klawiszy „Alt-Home". To samo dotyczy analogicznie „Alt-End" w celu umieszczenia kursora na końcu ostatniego wiersza bieżącego akapitu.

1.11.9 Select modes in editor (Wybierz tryby w edytorze)

CodeBlocks obsługuje różne tryby zaznaczania i wklejania ciągów znaków.

1. Lewym przyciskiem myszy można wybrać tekst w aktywnym edytorze, a następnie zwolnić przycisk myszy. Za pomocą kółka myszy użytkownik może przewijać do pozycji. Jeśli naciśnie się środkowy przycisk myszy, zostanie wstawiony poprzednio zaznaczony tekst. Ta funkcja jest dostępna dla każdego pliku i można ją zobaczyć w schowku dla każdego pliku.

2. Naciśnięcie klawisza „ALT" aktywuje tzw. tryb zaznaczania bloku, a zaznaczenie prostokątne można podnieść lewym przyciskiem myszy. Jeśli klawisz Alt zostanie zwolniony, to zaznaczenie można skopiować lub wkleić. Ta funkcja jest pomocna, jeśli chcesz zaznaczyć niektóre kolumny, np. tablicy, a następnie skopiować i wkleić zawartość.

3. W menu „Settings (Ustawienia)"/„Editor (Edytor)"/„ Margins und Caret (Marginesy i kursor)" można aktywować tzw. „Virtual Spaces (Wirtualne przestrzenie)". Ta opcja umożliwia, aby zaznaczenie w trybie zaznaczania bloku mogło zaczynać się lub kończyć w pustym wierszu.

4. W menu „Settings (Ustawienia)"/„Editor (Edytor)"/„Margins and Caret (Marginesy i kursor)" można aktywować „Multiple Selection (Wielokrotne zaznaczenie)". Przytrzymując klawisz Ctrl, użytkownik może zaznaczać różne wiersze w aktywnym edytorze za pomocą lewego przycisku myszy. Zaznaczenia zostaną dołączone do schowka za pomocą skrótu Ctrl-C lub Ctrl-X. Ctrl-V wstawi zawartość w bieżącej pozycji kursora. Dodatkowa opcja o nazwie „Enable typing (and deleting) (Włącz pisanie (i usuwanie))" może zostać aktywowana dla wielu zaznaczeń. Ta funkcja jest przydatna, jeśli chcesz dodać dyrektywę preprocesora, taką jak **`#ifdef,`** w różnych wierszach źródłowych lub jeśli chcesz nadpisać lub zastąpić tekst w kilku pozycjach.

Uwaga:
Większość menedżerów okien Linux używa kombinacji klawiszy ALT-LeftClickDrag do przesuwania okna, więc aby blokowanie zaznaczania działało, najpierw musisz wyłączyć to zachowanie menedżera okien.

1.11.10 Code folding (Składanie kodu)

CodeBlocks obsługuje tzw. składanie kodu. Dzięki tej funkcji możesz składać np. funkcje w edytorze CodeBlocks. Punkt składania jest oznaczony symbolem minus na lewym marginesie widoku edytora. Na marginesie początek i koniec punktu składania jest widoczny jako pionowa linia. Jeśli klikniesz symbol minus lewym przyciskiem myszy, fragment kodu zostanie złożony lub rozwinięty. Poprzez menu „Edit (Edycja)" / „Folding (Składanie)" możesz wybrać składanie. W edytorze widzisz złożony kod jako ciągłą poziomą linię.

Uwaga:
Styl składania i limit głębokości składania można skonfigurować w menu „Settings (Ustawienia)"/„Editor (Edytor)"/„Folding (Składanie)".

CodeBlocks zapewnia funkcję składania również dla dyrektyw preprocesora. Aby włączyć tę funkcję, wybierz „Fold preprocessor commands (Polecenia preprocesora składania)" w menu „Settings (Ustawienia)"/„Editor (Edytor)" we wpisie składania.

Inną możliwością jest ustawienie zdefiniowanych przez użytkownika punktów składania. Początek punktu składania jest wprowadzany jako komentarz z otwierającym nawiasem, a koniec jest rynkiem z komentarzem z zamykającym nawiasem.

```
//{
kod ze zdefiniowanym przez użytkownika składaniem
//}
```

1.11.11 Auto complete (Automatyczne uzupełnianie)

Jeśli otworzysz projekt w CodeBlocks, „Search directories (katalogi wyszukiwania)" kompilatora i projektu, źródła i nagłówki projektu zostaną przeanalizowane. Ponadto przeanalizowane zostaną słowa kluczowe odpowiedniego pliku leksera. Informacje z analizy składniowej zostaną wykorzystane do funkcji automatycznego uzupełniania w CodeBlocks. Sprawdź ustawienia edytora, jeśli ta funkcja jest włączona. Funkcja automatycznego uzupełniania jest dostępna za pomocą skrótu Ctrl-Spacja. Poprzez menu „Settings (Ustawienia)" / „Editor (Edytor)" / „Syntax highlighting (Podświetlanie składni)" możesz dodać zdefiniowane przez użytkownika słowa kluczowe do swojego leksera.

1.11.12 Find broken files (Znajdź uszkodzone pliki)

Jeśli plik zostanie usunięty z dysku, ale nadal będzie zawarty w pliku projektu `<project>.cbp`, wówczas ten „broken file (uszkodzony plik)" zostanie wyświetlony jako uszkodzony symbol w widoku projektu. Zamiast usuwania plików należy użyć menu „Remove file from project (Usuń plik z projektu)".

1.11.13 Including libraries (Włączanie bibliotek)

W opcjach kompilacji projektu możesz dodać używane biblioteki za pomocą przycisku „Add (Dodaj)" w pozycji „Link librarys (Biblioteki linków)" w „Linker Settings (Ustawienia linkera)". Możesz przy tym użyć ścieżki bezwzględnej do biblioteki lub po prostu podać nazwę bez prefiksu lib i rozszerzenia pliku.

Przykład:
W przypadku biblioteki o nazwie `<path>\libs\lib<name>.a`, po prostu napisz `<nazwa>`. Linker z odpowiednimi ścieżkami wyszukiwania będzie wtedy poprawnie uwzględniał biblioteki.

Uwaga:
Inny sposób dołączenia bibliotek opisano w sekcji 2.10.

1.11.14 Object linking order (Kolejność łączenia obiektów)

Podczas kompilacji obiekty `nazwa.o` są tworzone ze źródeł `nazwa.c/cpp`. Następnie linker wiąże poszczególne obiekty do aplikacji `nazwa.exe` lub dla systemów wbudowanych `name.elf`.
W niektórych przypadkach może być pożądane wstępne zdefiniowanie kolejności, w jakiej obiekty będą łączone. W CodeBlocks można to osiągnąć, przypisując priorytety. W menu kontekstowym „Properties (Właściwości)" można zdefiniować priorytety pliku na karcie Build (Kompilacja). Niski priorytet spowoduje, że plik zostanie połączony wcześniej.

1.11.15 Autosave (Autozapis)

CodeBlocks oferuje sposoby automatycznego przechowywania projektów i plików źródłowych lub tworzenia kopii zapasowych. Funkcję tę można aktywować w menu „Settings (Ustawienia)" / „Environment (Środowisko)" / „Autosave (Autozapis)". W tym celu należy określić „Save to .save file (Zapisz do pliku .save)" jako metodę tworzenia kopii zapasowej.

1.11.16 Settings for file extensions (Ustawienia rozszerzeń plików)

W CodeBlocks możesz wybierać spośród kilku sposobów obsługi rozszerzeń plików. Okno dialogowe ustawień można otworzyć za pomocą „Settings (Ustawienia)"/„ Files extension handling (Obsługa rozszerzeń plików)". Możesz użyć aplikacji przypisanych przez system Windows do każdego rozszerzenia pliku (otworzyć je za pomocą powiązanej aplikacji) lub zmienić ustawienia dla każdego rozszerzenia w taki sposób, że zostanie uruchomiony program zdefiniowany przez użytkownika (uruchomienie programu zewnętrznego) lub plik zostanie otwarty w edytorze CodeBlocks (otworzenie go w edytorze Code::Blocks).

Uwaga:
Jeśli program zdefiniowany przez użytkownika jest przypisany do określonego rozszerzenia pliku, ustawienie „Disable Code::Blocks while the external program is running (Wyłącz Code::Blocks, gdy program zewnętrzny jest uruchomiony)" powinno zostać wyłączone, ponieważ w przeciwnym razie CodeBlocks zostanie zamknięty za każdym razem, gdy zostanie otwarty plik z tym rozszerzeniem.

1.12 CodeBlocks at the command line (CodeBlocks w wierszu poleceń)

IDE CodeBlocks można uruchomić z wiersza poleceń bez graficznego interfejsu. W takim przypadku dostępnych jest kilka przełączników do kontrolowania procesu kompilacji projektu. Ponieważ CodeBlocks jest skryptowalny, tworzenie plików wykonywalnych można zintegrować z własnymi procesami roboczymi.

```
codeblocks.exe /na /nd --no-splash-screen --built <name>.cbp --
target='Release'
<filename>
```

Określa nazwę pliku projektu `*.cbp` lub nazwę pliku obszaru roboczego `*.workspace`. Na przykład <filename> może być `project.cbp`. Umieść ten argument na końcu wiersza poleceń, tuż przed przekierowaniem wyjścia, jeśli takie istnieje.

```
--file=<filename>[:line]
```

Otwórz plik w Code::Blocks i opcjonalnie przejdź do określonego wiersza.

`/h, --help`
Wyświetla komunikat pomocy dotyczący argumentów wiersza poleceń.
`/na, --no-check-associations`
Nie wykonuj żadnych kontroli skojarzeń plików (tylko Windows).
`/nd, --no-dde`
Nie uruchamiaj serwera DDE (tylko Windows).
`/ni, --no-ipc`
Nie uruchamiaj serwera IPC (tylko Linux i Mac).
`/ns, --no-splash-screen`
Ukrywa ekran powitalny podczas ładowania aplikacji.
`/d, --debug-log`
Wyświetla dziennik debugowania aplikacji.
`--prefix=<str>`
Ustawia prefiks współdzielonego katalogu danych.
`/p, --personality=<str>, --profile=<str>`
Ustawia osobowość do użycia. Możesz użyć ask jako parametru, aby wyświetlić listę wszystkich dostępnych osobowości.
`--rebuild`
Wyczyść i zbuduj projekt lub obszar roboczy.
`--build`
Zbuduj projekt lub obszar roboczy.
`--target=<str>`
Ustawia cel kompilacji wsadowej. Na przykład --target='Release'.
`--no-batch-window-close`
Utrzymuje widoczne okno dziennika wsadowego po zakończeniu kompilacji wsadowej.
`--batch-build-notify`
Wyświetla komunikat po zakończeniu kompilacji wsadowej.
`--safe-mode`
Wszystkie wtyczki są wyłączane podczas uruchamiania.
`> <plik dziennika kompilacji>`
Umieszczony na ostatniej pozycji wiersza poleceń, może być użyty do przekierowania standardowego wyjścia do pliku dziennika. Nie jest to opcja bloku kodu jako taka, ale po prostu standardowe przekierowanie wyjścia powłoki DOS/*nix.

1.13 Shortcuts (Skróty)

1.13.1 Editor (Edytor)

Funkcja	Skrót klawiaturowy
Cofnij ostatnią czynność	Ctrl – Z
Powtórz ostatnią akcję	Ctrl – Shift – Z
Zamień nagłówek/źródło	F11
Skomentuj wyróżniony kod	Ctrl – Shift – C
Odkomentuj wyróżniony kod	Ctrl – Shift – X
Autouzupełnianie / Skróty	Ctrl – Space / Ctrl – J
Przełącz zakładkę	Ctrl – B
Przejdź do poprzedniej zakładki	Alt – PgUp
Przejdź do następnej zakładki	Alt – PgDown

To jest lista skrótów udostępnianych przez komponent edytora CodeBlocks. Skrótów tych nie można ponownie powiązać:

Funkcja	Skrót klawiaturowy
Utwórz lub usuń zakładkę	Ctrl – F2
Przejdź do następnej zakładki	F2
Wybierz następną zakładkę	Alt – F2
Znajdź wybór	Ctrl – F3
Znajdź zaznaczenie wstecz	Ctrl – Shift – F3
Znajdź pasujący preprocesor warunkowy, pomijając zagnieżdżone	Ctrl – K

1.13.2 Files (Pliki)

Funkcja	Skrót klawiaturowy
Nowy plik lub projekt	Ctrl – N
Otwórz istniejący plik lub projekt	Ctrl – O
Zapisz bieżący plik lub projekt	Ctrl – S
Zapisz wszystkie pliki	Ctrl – Shift – S
Zamknij bieżący plik	Ctrl – F4 / Ctrl – W
Zamknij wszystkie pliki	Ctrl – Shift – F4 / Ctrl – Shift – W

1.13.3 View (Widok)

Funkcja	Skrót klawiaturowy
Pokaż / ukryj panel wiadomości (Messages)	F2
Pokaż / ukryj panel zarządzania (Management)	Shift – F2
Aktywuj wcześniejszy (w drzewie projektu)	Alt – F5
Aktywuj następny (w drzewie projektu)	Alt – F6

1.13.4 Search (Szukaj)

Funkcja	Skrót klawiaturowy
Znajdź	Ctrl – F
Znajdź następny	F3
Znajdź poprzedni	Shift – F3
Znajdź w plikach	Ctrl – Shift – F
Zastąp	Ctrl – R
Zastąp w plikach	Ctrl – Shift – R
Przejdź do wiersza	Ctrl – G
Przejdź do następnej zmienionej linii	Ctrl – F3
Przejdź do poprzedniej zmienionej linii	Ctrl – Shift – F3
Przejdź do pliku	Alt – G
Przejdź do funkcji	Ctrl – Alt – G
Przejdź do poprzedniej funkcji	Ctrl – PgUp
Przejdź do następnej funkcji	Ctrl – PgDn
Przejdź do deklaracji	Ctrl – Shift –
Przejdź do implementacji	Ctrl –
Otwórz plik dołączony	Ctrl – Alt –

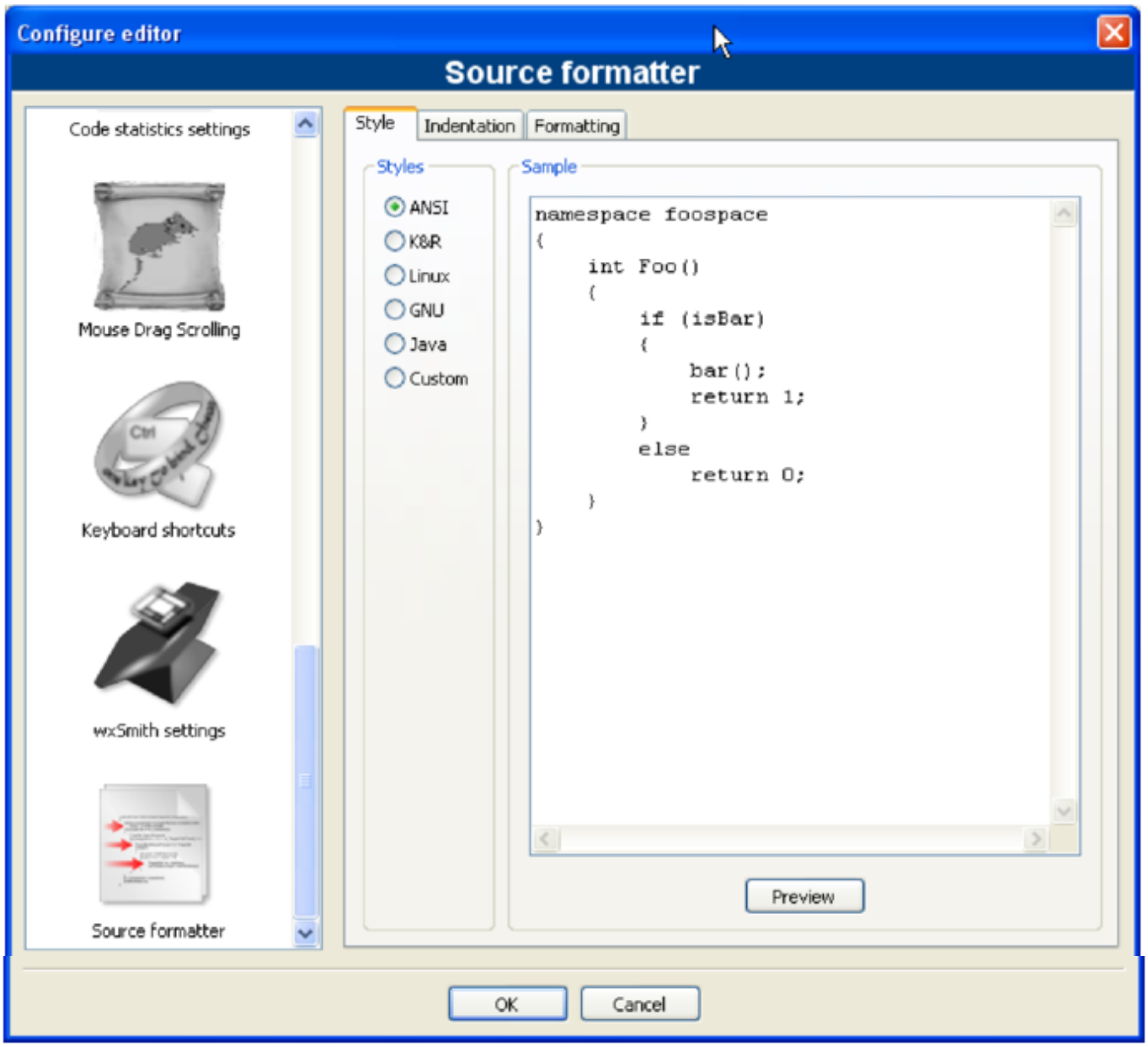
1.13.5 Build (Kompilacja)

Funkcja	Skrót klawiaturowy
Kompiluj (Build)	Ctrl – F9
Kompiluj bieżący plik	CTrl – Shift – F9
Uruchom (Run)	Ctrl – F10
Kompiluj i uruchom (Build and Run)	F9
Odbuduj (Rebuild)	Ctrl – F11

2 Plugins (Wtyczki)

2.1 Astyle (AStyl)

Artistic Style to wcięcie kodu źródłowego, formater kodu źródłowego i upiększacz kodu źródłowego dla języków programowania C, C++, C#. Można go używać do wybierania różnych stylów reguł kodowania w CodeBlocks.



Rysunek 2.1: Formatowanie kodu źródłowego

Podczas wcinania kodu źródłowego, my jako programiści mamy tendencję do używania zarówno spacji, jak i znaków tabulacji, aby utworzyć pożądane wcięcie. Co więcej, niektórzy edytorzy domyślnie wstawiają spacje zamiast tabulatorów po naciśnięciu klawisza tabulacji, a inni edytorzy mają możliwość upiększania wierszy poprzez automatyczne ustawianie odstępu przed kodem w wierszu, ewentualnie wstawiając spacje w kodzie, który do tej pory używał tylko tabulatorów do wcięć.

Ponieważ liczba znaków spacji wyświetlanych na ekranie dla każdego znaku tabulacji w kodzie źródłowym zmienia się między edytorami, jednym ze standardowych problemów, z jakimi spotykają się programiści podczas przechodzenia z jednego edytora do drugiego, jest to, że kod zawierający zarówno spacje, jak i tabulatory, który do tej pory był idealnie wcięty, nagle staje się bałaganem do oglądania

podczas zmiany edytora. Nawet jeśli jako programista dbasz o używanie TYLKO spacji lub tabulatorów, oglądanie cudzego kodu źródłowego nadal może być problematyczne.

Aby rozwiązać ten problem, stworzono Artistic Style — filtr napisany w języku C++, który automatycznie zmienia wcięcia i formatuje pliki źródłowe w językach C/C++/C#.

Uwaga:
Podczas kopiowania kodu, na przykład z Internetu lub podręcznika, kod ten zostanie automatycznie dostosowany do zasad kodowania w CodeBlocks.

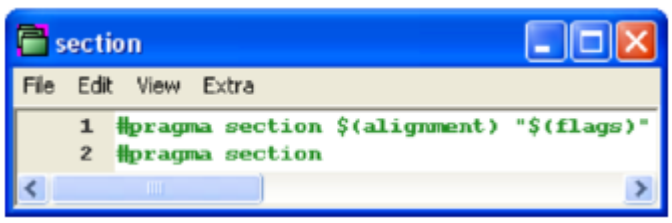
2.2 CodeSnippets (Fragmenty kodu)

Wtyczka CodeSnippets umożliwia strukturowanie modułów tekstowych i linków do plików według kategorii w widoku drzewa. Moduły służą do przechowywania często używanych plików i konstrukcji w modułach tekstowych i zarządzania nimi w centralnym miejscu. Wyobraź sobie następującą sytuację: Wiele często używanych plików źródłowych jest przechowywanych w różnych katalogach systemu plików. Okno CodeSnippets daje możliwość tworzenia kategorii, a pod kategoriami linków do wymaganych plików. Dzięki tym funkcjom możesz kontrolować dostęp do plików niezależnie od tego, gdzie są przechowywane w systemie plików, i możesz szybko nawigować między plikami bez konieczności przeszukiwania całego systemu.

Uwaga:
Możesz używać zmiennych CodeBlocks lub zmiennych środowiskowych w linkach do plików, np. $(VARNAME)/name.pdf, aby sparametryzować link w przeglądarce CodeSnippets.

Listę modułów tekstowych i linków można zapisać w oknie CodeSnippets, klikając prawym przyciskiem myszy i wybierając „Save Index (Zapisz indeks)” z menu kontekstowego. Plik `codesnippets.xml`, który zostanie utworzony za pomocą tej procedury, można znaleźć w podkatalogu `codeblocks` katalogu `Documents and Settings\Application data`. W systemie Linux te informacje są przechowywane w podkatalogu `.codeblocks` katalogu HOME. Pliki konfiguracyjne CodeBlocks zostaną załadowane podczas następnego uruchomienia. Jeśli chcesz zapisać zawartość CodeSnippets w innej lokalizacji, wybierz pozycję „Save Index As (Zapisz indeks jako)”. Aby załadować ten plik, wybierz opcję „Load Index File (Załaduj plik indeksu)” podczas następnego uruchomienia CodeBlocks lub uwzględnij katalog w menu kontekstowym „Settings (Ustawienia)” w obszarze „Snippet Folder (Folder fragmentów)”. Ustawienia są zapisywane w odpowiednim pliku `codesnippets.ini` w danych aplikacji.

Aby dodać kategorię, użyj menu „Add SubCategory (Dodaj podkategorię)”. Kategoria może zawierać fragmenty kodu (moduły tekstowe) lub łącza do plików. Moduł tekstowy jest tworzony za pomocą polecenia „Add Snippet (Dodaj fragment kodu)” w menu kontekstowym. Treść jest integrowana z modułem tekstowym jako „New snippet (Nowy fragment kodu)” poprzez wybranie fragmentu tekstu w edytorze CodeBlocks i przeciągnięcie go i upuszczenie na moduł, a następnie pojawia się okno dialogowe właściwości. Dwukrotne kliknięcie nowo dołączonego wpisu lub wybranie „Edit Text (Edytuj tekst)” spowoduje otwarcie edytora treści.



Rysunek 2.2: Edycja modułu tekstowego

Wyjście modułu tekstowego jest obsługiwane w CodeBlocks za pomocą polecenia menu kontekstowego „Apply (Zastosuj)” lub przez przeciąganie i upuszczanie do edytora. W systemie Windows zawartość Snippet można również przeciągać i upuszczać do innych aplikacji. W przeglądarce CodeSnippets można skopiować wybrany element za pomocą przeciągania i upuszczania do innej kategorii.

Poza tym moduły tekstowe mogą być parametryzowalne przez zmienne <name>, do których można uzyskać dostęp za pomocą $(name) (patrz Rysunek 2.2). Wartości zmiennych można pobrać w polu wprowadzania, jeśli moduł tekstowy jest wywoływany za pomocą polecenia menu kontekstowego „Apply”.

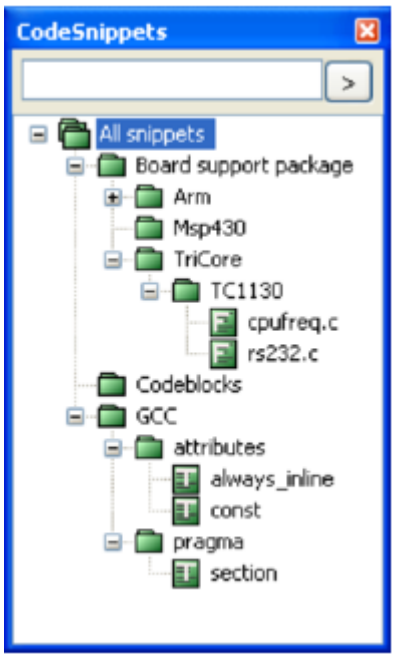
Oprócz modułów tekstowych, można również tworzyć linki do plików. Jeśli po utworzeniu modułu tekstowego klikniesz polecenie menu kontekstowego „Properties (Właściwości)”, możesz wybrać cel linku, klikając przycisk „Link target (Cel linku)”. Ta procedura automatycznie przekształci moduł tekstowy w link do pliku. W CodeSnippets wszystkie moduły tekstowe będą oznaczone symbolem T, linki do pliku symbolem F, a adresy URL symbolem U. Jeśli chcesz otworzyć wybrany plik (link) w widoku codesnippets, po prostu wybierz menu kontekstowe „Open File (Otwórz plik)” lub przytrzymaj klawisz „Alt” i kliknij dwukrotnie plik.

Uwaga:
Możesz dodać nawet adres URL (np. [Internet] http://www.codeblocks.org) w modułach tekstowych. Adres URL można otworzyć za pomocą menu kontekstowego „Open Url (Otwórz adres URL)” lub za pomocą funkcji przeciągnij i upuść w swojej ulubionej przeglądarce internetowej.

Dzięki temu ustawieniu, jeśli otworzysz link do pliku PDF z widoku codesnippets, przeglądarka PDF uruchomi się automatycznie. Ta metoda umożliwia użytkownikowi dostęp do plików rozproszonych w całej sieci, takich jak dane CAD, układy, dokumentacje itp., za pomocą typowych aplikacji, po prostu za pośrednictwem linku. Zawartość codesnippets jest przechowywana w pliku `codesnippets.xml`, konfiguracja jest przechowywana w pliku `codesnippets.ini` w katalogu `application data` (danych aplikacji). Ten plik ini będzie na przykład zawierał ścieżkę do pliku `codesnippets.xml`.

CodeBlocks obsługuje używanie różnych profili. Te profile nazywane są osobowościami (personalities). Uruchomienie CodeBlocks z opcją wiersza poleceń `--personality=<profile>` spowoduje utworzenie nowego profilu lub użycie istniejącego. Następnie ustawienia nie zostaną zapisane w pliku `default.conf`, ale w `<personality>.conf` w katalogu `application data` (danych aplikacji). Wtyczka Codesnippets zapisze swoje ustawienia w pliku. `<personality>.codesnippets.ini`. Teraz, jeśli załadujesz nową zawartość `<name.xml>` w ustawieniach Codesnippets za pomocą „Load Index File (Wczytaj plik indeksu)”, ta zawartość zostanie zapisana w odpowiednim pliku ini. Zaleta tej metody polega na tym, że w przypadku różnych profili można zarządzać różnymi konfiguracjami modułów tekstowych i linków.

Wtyczka oferuje dodatkową funkcję wyszukiwania do nawigacji między kategoriami i fragmentami kodu. Zakres wyszukiwania fragmentów kodu, kategorii lub fragmentów kodu i kategorii można dostosować. Po wprowadzeniu wymaganego wyrażenia wyszukiwania odpowiedni wpis jest automatycznie wybierany w widoku. Rysunek 2.3 pokazuje typowy wyświetlacz w oknie CodeSnippets.



Rysunek 2.3: Widok CodeSnippets (fragmentów kodu)

Uwaga:
W przypadku korzystania z obszernych modułów tekstowych, zawartość tych modułów powinna zostać zapisana w plikach za pomocą opcji „Convert to File Link” (Konwertuj na łącze do pliku), aby zmniejszyć zużycie pamięci w systemie. Jeśli usuniesz fragment kodu lub łącze do pliku, zostanie on przeniesiony do kategorii `.trash` (kosz); jeśli przytrzymasz klawisz Shift, element zostanie usunięty.

2.3 Incremental Search (Wyszukiwanie przyrostowe)

Aby skutecznie przeszukiwać otwarte pliki, CodeBlocks udostępnia tzw. wyszukiwanie przyrostowe. Tę metodę wyszukiwania inicjuje się dla otwartego pliku za pomocą menu „Search (Szukaj)” / „Incremental Search (Wyszukiwanie przyrostowe)” lub za pomocą skrótu klawiaturowego Ctrl-I. Następnie fokus jest automatycznie ustawiany na maskę wyszukiwania odpowiedniego paska narzędzi. Gdy tylko zaczniesz wprowadzać termin wyszukiwania, tło maski wyszukiwania zostanie dostosowane zgodnie z wystąpieniem terminu. Jeśli trafienie zostanie znalezione w aktywnym edytorze, odpowiednia pozycja w tekście zostanie oznaczona kolorem. Domyślnie bieżące trafienie zostanie podświetlone na zielono. To ustawienie można zmienić za pomocą „Settings (Ustawienia)” / „Editor (Edytor)” / „Incremental Search (Wyszukiwanie przyrostowe)” (patrz ??). Naciśnięcie klawisza Return powoduje przejście wyszukiwania do następnego wystąpienia ciągu wyszukiwania w pliku. Za pomocą Shift-Return można wybrać poprzednie wystąpienie. Ta funkcjonalność nie jest obsługiwana przez Scintilla, jeśli wyszukiwanie przyrostowe używa wyrażeń regularnych.

```
m_pToolbar->EnableTool(X

if (m_pControl != 0)
{
    m_SearchText=m_pText
    m_pToolbar->EnableTo
    m_pToolbar->EnableTo
    m_NewPos=m_pControl-
    m_OldPos=m_NewPos;
}
else
{
    m_pToolbar->EnableTo
    m_pToolbar->EnableTo
}
```

Jeśli szukany ciąg znaków nie zostanie odnaleziony w aktywnym pliku, fakt ten zostanie wyróżniony poprzez wyświetlenie tła maski wyszukiwania na czerwono.

ESC
Wyjdź z trybu wyszukiwania przyrostowego.
ALT-DELETE
Wyczyść dane wejściowe pola wyszukiwania przyrostowego.

Ikony na pasku narzędzi wyszukiwania przyrostowego mają następujące znaczenie:


Usuwanie tekstu w masce wyszukiwania paska narzędzi wyszukiwania przyrostowego.


Nawigowanie między wystąpieniami ciągu wyszukiwania.


Kliknięcie tego przycisku powoduje, że wszystkie wystąpienia ciągu wyszukiwania w edytorze są podświetlone kolorem, a nie tylko początkowe wystąpienie.


Aktywowanie tej opcji ogranicza wyszukiwanie do fragmentu tekstu zaznaczonego w edytorze.

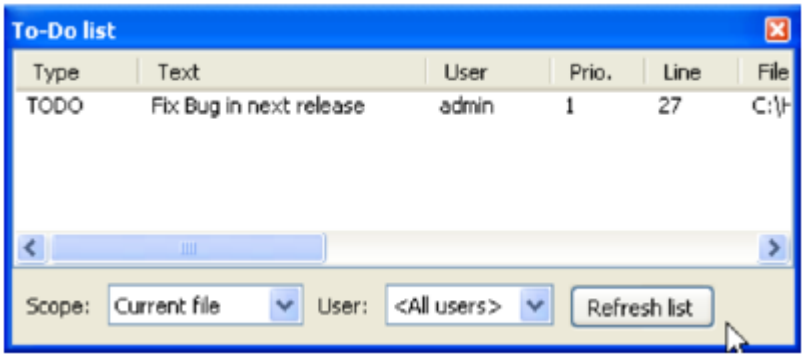

Ta opcja oznacza, że wykonywane jest wyszukiwanie z uwzględnieniem wielkości liter.


W polu wprowadzania wyszukiwania przyrostowego można używać wyrażeń regularnych.

Uwaga:
Standardowe ustawienia tego paska narzędzi można skonfigurować w „Settings (Ustawienia)” / „Editor (Edytor)” / „Incremental Search (Wyszukiwanie przyrostowe)”.

2.4 ToDo List (Lista zadań do wykonania)

W złożonych projektach oprogramowania, w których biorą udział różni użytkownicy, często wymagane jest wykonanie różnych zadań przez różnych użytkowników. W tym celu CodeBlocks oferuje listę zadań do wykonania. Listę tę można otworzyć za pomocą „View (Widok)” / „To-Do list (Lista zadań do wykonania)” i zawiera ona zadania do wykonania wraz z ich priorytetami, typami i odpowiedzialnymi użytkownikami. Listę można filtrować według zadań, użytkowników i/lub plików źródłowych. Sortowanie według kolumn można uzyskać, klikając podpis odpowiedniej kolumny.



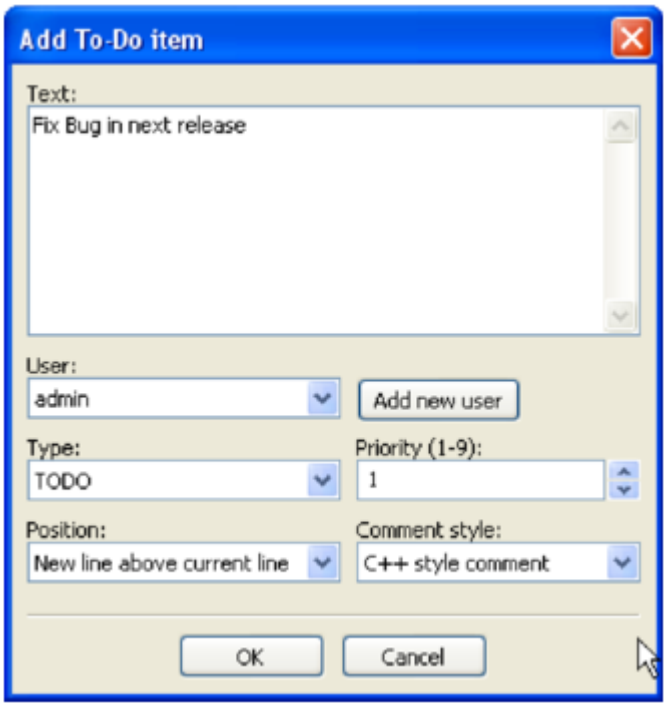
Rysunek 2.4: Wyświetlanie listy zadań do wykonania

Uwaga:
Listę zadań do wykonania można zadokować w konsoli wiadomości. Wybierz opcję „Include the To-Do list in the message pane (Dołącz listę zadań do wykonania do panelu wiadomości” w menu „Settings (Ustawienia)” / „Environment (Środowisko)”

Jeśli źródła są otwarte w CodeBlocks, Todo może zostać dodane do listy poprzez polecenie menu kontekstowego „Add To-Do item (Dodaj element To-Do)”. Komentarz zostanie dodany w wybranym wierszu kodu źródłowego.

```
// TODO (user#1#): dodaj nowe okno dialogowe dla następnego wydania
```

Podczas dodawania zadania do wykonania pojawi się okno dialogowe, w którym można wprowadzić następujące ustawienia (patrz rysunek 2.5).



Rysunek 2.5: Okno dialogowe dodawania zadania do wykonania

User (Użytkownik)

Nazwa użytkownika <user> w systemie operacyjnym. Zadania dla innych użytkowników można również tworzyć tutaj. W tym celu należy utworzyć odpowiednią nazwę użytkownika za pomocą opcji Add new (Dodaj nowy). Przypisanie zadania do wykonania jest następnie wykonywane poprzez wybór wpisów wymienionych dla użytkownika.

Uwaga:
Należy pamiętać, że Users (Użytkownicy) nie mają nic wspólnego z Personalities (Osobowościami) używanymi w CodeBlocks.

Type (Typ)
Domyślnie typ jest ustawiony na Todo.

Priority (Priorytet)
Ważność zadań można wyrazić za pomocą priorytetów (1–9) w CodeBlocks.

Position (Pozycja)
To ustawienie określa, czy komentarz ma być dołączony przed, po czy w dokładnej pozycji kursora.

Comment Style (Styl komentarza)
Wybór formatów komentarzy (np. doxygen).

2.5 Source Code Exporter (Eksporter kodu źródłowego)

Często zachodzi potrzeba przenoszenia kodu źródłowego do innych aplikacji lub do wiadomości e-mail. Jeśli tekst zostanie po prostu skopiowany, formatowanie zostanie utracone, co sprawi, że tekst będzie bardzo niejasny. Funkcja eksportu CodeBlocks służy jako środek zaradczy w takich sytuacjach. Wymagany format pliku eksportu można wybrać za pomocą „File (Plik)" / „Export (Eksport)". Następnie program przyjmie nazwę pliku i katalog docelowy z otwartego pliku źródłowego i zaproponuje je do zapisania pliku eksportu. Odpowiednie rozszerzenie pliku w każdym przypadku zostanie określone przez format eksportu. Dostępne są następujące formaty:

html
Format oparty na tekście, który można wyświetlić w przeglądarce internetowej lub w aplikacjach do przetwarzania tekstu.
rtf
Format Rich Text to format oparty na tekście, który można otworzyć w aplikacjach do przetwarzania tekstu, takich jak Word lub OpenOffice.
odt
Format Open Document Text to standardowy format, który został określony przez Sun i O'Reilly. Ten format może być przetwarzany przez Word, OpenOffice i inne aplikacje do przetwarzania tekstu.
pdf
Format Portable Document można otwierać w aplikacjach, takich jak Acrobat Reader.

2.6 Thread Search (Wyszukiwanie wątków)

Za pomocą menu „Search (Szukaj)"/„ Thread Search (Wyszukiwanie wątków)" odpowiednią wtyczkę można wyświetlić lub ukryć jako zakładkę w Konsoli wiadomości. W CodeBlocks można wyświetlić podgląd wystąpienia ciągu znaków w pliku, obszarze roboczym lub katalogu. W ten sposób lista wyników wyszukiwania zostanie wyświetlona po prawej stronie Konsoli ThreadSearch. Kliknięcie wpisu na liście powoduje wyświetlenie podglądu po lewej stronie. Dwukrotne kliknięcie na liście powoduje otwarcie wybranego pliku w edytorze CodeBlocks.

Uwaga:
Zakres rozszerzeń plików uwzględnianych w wyszukiwaniu jest wstępnie ustawiony i może wymagać dostosowania.

2.6.1 Features (Funkcje)

- Wtyczka ThreadSearch oferuje następujące funkcje:
- Wielowątkowe „Wyszukiwanie w plikach"
- Wewnętrzny edytor tylko do odczytu, aby wyświetlić podgląd wyników
- Plik otwarty w notatniku edytora
- Menu kontekstowe „Find occurrences (Znajdź wystąpienia)", aby rozpocząć wyszukiwanie w plikach ze słowem pod kursorem

Rysunek 2.6: Panel wyszukiwania wątków

2.6.2 Usage (Stosowanie)

1. Skonfiguruj swoje preferencje wyszukiwania (patrz Rysunek 2.7)
   Po zainstalowaniu wtyczki dostępne są 4 sposoby przeprowadzenia wyszukiwania:

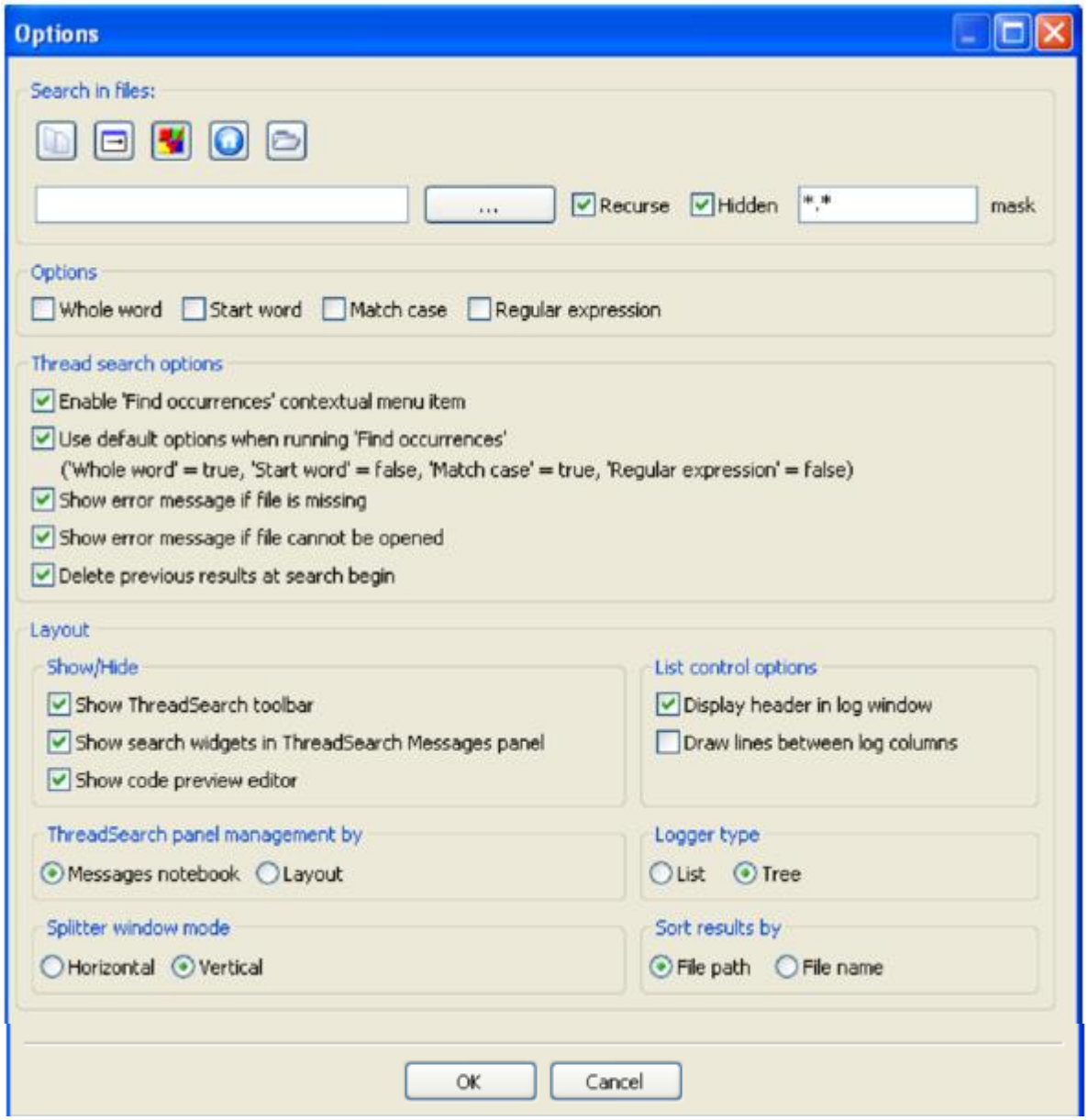
   a. Wpisz/wybierz słowo w polu wyszukiwania i naciśnij Enter lub kliknij Search (Szukaj) w panelu wyszukiwania wątków w notatniku Messages (Wiadomości).
   b. Wpisz/wybierz słowo w polu wyszukiwania na pasku narzędzi i naciśnij Enter lub kliknij przycisk Search (Szukaj).
   c. Kliknij prawym przyciskiem myszy dowolne „słowo” w aktywnym edytorze i kliknij „Find occurrences (Znajdź wystąpienia)”.
   d. Kliknij Search (Szukaj) / Thread search (Wyszukiwanie wątków), aby znaleźć bieżące słowo w aktywnym edytorze.

Uwaga:
Elementy 1, 2 i 3 mogą być niedostępne w zależności od bieżącej konfiguracji.

2. Kliknij ponownie przycisk wyszukiwania, aby anulować bieżące wyszukiwanie.
3. Pojedyncze kliknięcie elementu wyniku wyświetla go w edytorze podglądu w odpowiednim miejscu.
4. Podwójne kliknięcie elementu wyniku otwiera lub ustawia edytor w notatniku edytora w odpowiednim miejscu.

2.6.3 Configuration (Konfiguracja)

Aby uzyskać dostęp do panelu konfiguracji wtyczki ThreadSearch kliknij (patrz rysunek 2.7):



Rysunek 2.7: Konfiguracja wyszukiwania wątków

1. Przycisk Options (Opcje) w panelu Thread search (wyszukiwania wątków) w notatniku Messages (Wiadomości).
2. Przycisk Options (Opcje) na pasku narzędzi ThreadSearch (wyszukiwania wątków).
3. Element menu Settings (Ustawienia) / Environment (Środowisko), a następnie element ThreadSearch (wyszukiwania wątków) w kolumnach po lewej stronie.

Uwaga:
Elementy 1, 2 i 3 mogą być niedostępne w zależności od bieżącej konfiguracji.

Wyszukiwanie częściowo definiuje zbiór plików, które zostaną przeanalizowane:

- Pola wyboru Project (Projekt) i Workspace (Obszar roboczy) wykluczają się wzajemnie.
- Ścieżkę katalogu można edytować lub ustawić za pomocą przycisku Select (Wybierz).
- Maska to zestaw specyfikacji pliku oddzielonych znakiem '`;`'. Na przykład: `*.cpp;*.c;*.h`.

2.6.4 Options (Opcje)

Whole word (Całe słowo)
Jeśli zaznaczone, wiersz pasuje do wyrażenia wyszukiwania, jeśli wyrażenie wyszukiwania nie zawiera znaku alfanumerycznego +'_' przed i po.

Start word (Słowo początkowe)
Jeśli zaznaczone, wiersz pasuje do wyrażenia wyszukiwania, jeśli wyrażenie wyszukiwania znajduje się na początku słowa, tj. nie zawiera znaku alfanumerycznego +'_' przed wyrażeniem wyszukiwania.

Match case (Dopasuj wielkość liter)
Jeśli zaznaczone, wyszukiwanie uwzględnia wielkość liter.

Regular expression (Wyrażenie regularne)
Wyrażenie wyszukiwania jest wyrażeniem regularnym.

Uwaga:
Jeśli chcesz wyszukiwać wyrażenia regularne, takie jak `n`, musisz ustawić opcję „Use Advanced RegEx searches (Użyj zaawansowanego wyszukiwania RegEx)" w menu „Settings (Ustawienia)" / „Editor (Edytor)" / „General Settings (Ustawienia ogólne)".

2.6.5 Thread search options (Opcje wyszukiwania wątków)

Enable 'Find occurrences contextual menu item' (Włącz element menu kontekstowego „Znajdź wystąpienia")
Jeśli zaznaczone, wpis Znajdź wystąpienia „Focused word (Skupione słowo)" zostanie dodany do menu kontekstowego edytora.

Use default options when running 'Find occurrences' (Użyj domyślnych opcji podczas uruchamiania „Znajdź wystąpienia")
Jeśli zaznaczone, zestaw domyślnych opcji zostanie zastosowany do wyszukiwań uruchamianych za pomocą elementu menu kontekstowego „Find occurrences (Znajdź wystąpienia)". Domyślnie włączone są opcje „Whole word (Całe słowo)" i „Match case (Dopasuj wielkość liter)".

Delete previous results at search begin (Usuń poprzednie wyniki na początku wyszukiwania)
Jeśli ThreadSearch jest skonfigurowany z „Tree View (Widokiem drzewa)", wyniki wyszukiwania zostaną wyświetlone hierarchicznie,

- pierwszy węzeł zawiera termin wyszukiwania
- powyżej wymienione są pliki zawierające termin wyszukiwania
- na tej liście wyświetlany jest numer wiersza i odpowiadająca mu zawartość wystąpienia

Jeśli będziesz wyszukiwać różne terminy, lista będzie niejasna, dlatego poprzednie wyniki wyszukiwania można wyczyścić na początku wyszukiwania, korzystając z tej opcji.

Uwaga:
Na liście wystąpień pojedyncze elementy lub wszystkie elementy można usunąć za pomocą menu kontekstowego „Delete item (Usuń element)" lub „Delete all items (Usuń wszystkie elementy)".

2.6.6 Layout (Układ)

Display header in log window (Wyświetl nagłówek w oknie dziennika)
jeśli zaznaczone, nagłówek jest wyświetlany w kontrolce listy wyników.

Uwaga:
Jeśli ta opcja nie jest zaznaczona, nie można zmieniać rozmiaru kolumn, ale oszczędza się miejsce.

Draw lines between columns (Rysuj linie między kolumnami)
Rysuje linie między kolumnami w trybie listy.
Show ThreadSearch toolbar (Pokaż pasek narzędzi ThreadSearch)
Wyświetl pasek narzędzi wtyczki Thread Search.
Show search widgets in ThreadSearch Messages panel (Pokaż widżety wyszukiwania w panelu Wiadomości ThreadSearch)
Jeśli zaznaczone, wyświetlane są tylko kontrolka listy wyników i edytor podglądu. Wszystkie inne widżety wyszukiwania są ukryte (oszczędzają miejsce).
Show code preview editor (Pokaż edytor podglądu kodu)
Podgląd kodu można ukryć za pomocą tego pola wyboru lub za pomocą dwukrotnego kliknięcia na środkowej krawędzi okna podziału. Tutaj można go ponownie wyświetlić.

2.6.7 Panel Management (Zarządzanie panelem)

Możesz wybrać różne tryby zarządzania oknem ThreadSearch. Przy ustawieniu „Message Notebook (Notatnik wiadomości)" okno ThreadSearch będzie dokowalnym oknem w panelu wiadomości. Jeśli wybierzesz ustawienie „Layout (Układ)", będziesz mógł oddokować okno z panelu wiadomości i umieścić je w innym miejscu.

2.6.8 Logger Type (Typ rejestratora)

Widok wyników wyszukiwania można wyświetlić na różne sposoby. Ustawienie „List (Lista)" wyświetla wszystkie wystąpienia jako listę. Inny tryb „Tree (Drzewo)" zbiera wszystkie wystąpienia w pliku jako węzeł.

2.6.9 Splitter Window Mode (Tryb okna rozdzielacza)

Użytkownik może skonfigurować poziomy lub pionowy podział okna podglądu i okna wyjściowego wyników wyszukiwania.

2.6.10 Sort Search Results (Sortowanie wyników wyszukiwania)

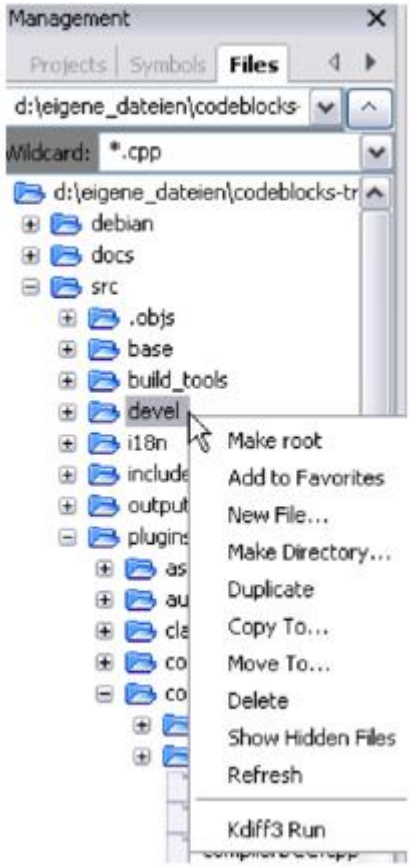
Widok wyników wyszukiwania można sortować według ścieżki lub nazwy pliku.

2.7 FileManager and PowerShell Plugin (Menedżer plików i wtyczka PowerShell)

Eksplorator plików Rysunek 2.8 jest zawarty we wtyczce FileManager i można go znaleźć w zakładce „Files (Pliki)". Skład Eksploratora plików pokazano na Rysunku 2.8.

Na górze znajdziesz pole do wpisania ścieżki. Klikając przycisk na końcu tego pola, pole rozwijane wyświetli historię poprzednich wpisów, po których można poruszać się za pomocą paska przewijania. Klawisz strzałki w górę po prawej stronie pola przesuwa strukturę katalogów o jeden katalog w górę.

W polu „Wildcard" możesz wprowadzić termin filtru dla wyświetlania plików. Pozostawienie pola pustego lub wprowadzenie * spowoduje wyświetlenie wszystkich plików. Wprowadzenie na przykład `*.c;*.h` spowoduje wyświetlenie wyłącznie źródeł C i plików nagłówkowych. Otwarcie pola rozwijanego ponownie wyświetli historię ostatnich wpisów.



Rysunek 2.8: Menedżer plików

Naciśnięcie klawisza Shift i kliknięcie powoduje wybranie grupy plików lub katalogów, naciśnięcie klawisza Ctrl i kliknięcie powoduje wybranie wielu oddzielnych plików lub katalogów.

Następujące operacje można uruchomić za pomocą menu kontekstowego, jeśli w Eksploratorze plików wybrano jeden lub wiele katalogów:

Make Root (Uczyń korzeń)
: określa bieżący katalog jako katalog główny.

Add to Favorites (Dodaj do ulubionych)
: ustawia znacznik dla katalogu i zapisuje go jako ulubiony. Ta funkcja umożliwia szybkie poruszanie się między często używanymi katalogami, również na różnych dyskach sieciowych.

New File (Nowy plik)
: tworzy nowy plik w wybranym katalogu.

New Directory (Nowy katalog)
: tworzy nowy podkatalog w wybranym katalogu.

Jeśli w Eksploratorze plików zaznaczono jeden lub więcej plików lub katalogów, można uruchomić następujące operacje za pomocą menu kontekstowego:

Duplicate (Duplikuj)
: kopiuje plik/katalog i zmienia jego nazwę.

Copy To (Kopiuj do)
: otwiera okno dialogowe umożliwiające wprowadzenie katalogu docelowego, w którym ma zostać zapisany skopiowany plik/katalog.

Move To (Przenieś do)
: przenosi zaznaczenie do lokalizacji docelowej.

Delete (Usuń)
usuwa wybrane pliki/katalogi.
Show Hidden Files (Pokaż ukryte pliki)
aktywuje/dezaktywuje wyświetlanie ukrytych plików systemowych. Po aktywacji ta pozycja menu jest zaznaczona.
Refresh (Odśwież)
aktualizuje wyświetlanie drzewa katalogów.

Jeśli w Eksploratorze plików zaznaczono jeden lub więcej plików, można uruchomić następujące operacje za pomocą menu kontekstowego:

Open in CB Editor (Otwórz w edytorze CB)
otwiera wybrany plik w edytorze CodeBlocks.
Rename (Zmień nazwę)
zmienia nazwę wybranego pliku.
Add to active project (Dodaj do aktywnego projektu)
dodaje plik(i) do aktywnego projektu.

Uwaga:
Do plików/katalogów wybranych w Eksploratorze plików można uzyskać dostęp za pomocą wtyczki PowerShell za pomocą zmiennej `mpaths`.

Funkcje zdefiniowane przez użytkownika można określić za pomocą polecenia menu „Settings (Ustawienia)" / „Environment (Środowisko)" / „PowerShell". W masce PowerShell nowa funkcja, której można nadać losową nazwę, jest tworzona za pomocą przycisku „New (Nowy". W polu „ShellCommand Executable" podany jest program wykonywalny, a w polu na dole okna można przekazać programowi dodatkowe parametry. Kliknięcie funkcji w menu kontekstowym lub menu PowerShell powoduje uruchomienie funkcji, która następnie przetworzy wybrane pliki/katalogi. Dane wyjściowe są przekierowywane do oddzielnego okna powłoki.

Na przykład wpis menu w „PowerShell" / „SVN" i w menu kontekstowym jest tworzony dla „SVN". `$file` w tym kontekście oznacza plik wybrany w Eksploratorze plików, `$mpath` wybrane pliki lub katalogi (patrz sekcja 3.2).

```
Add;$interpreter add $mpaths;;;
```

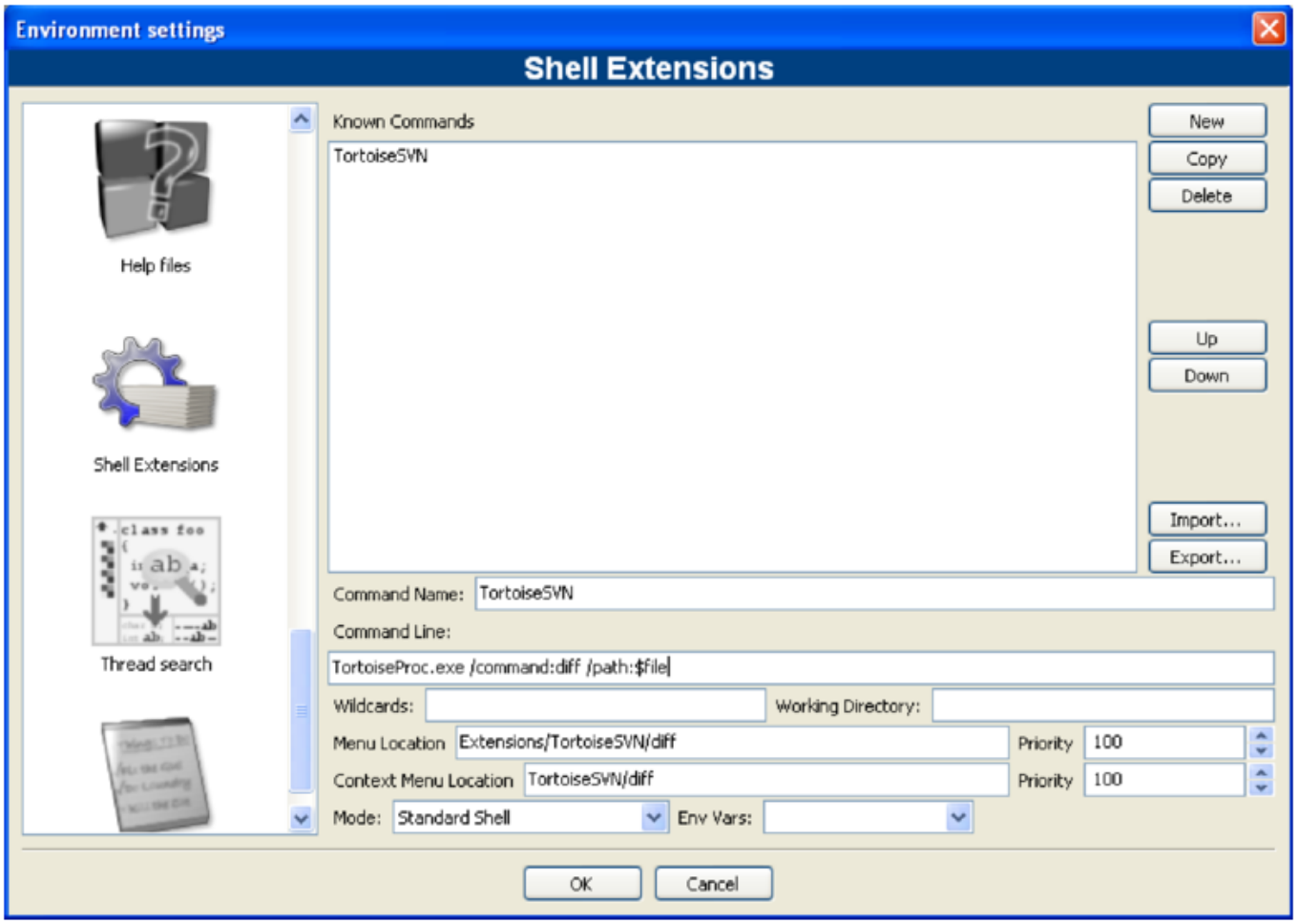
Ta i każda kolejna komenda utworzy podmenu, w tym przypadku nazwane 'Extensions (Rozszerzenia)' / 'SVN' / 'Add (Dodaj)'. Menu kontekstowe zostanie odpowiednio rozszerzone. Kliknięcie polecenia w menu kontekstowym spowoduje, że polecenie SVN `add` przetworzy wybrane pliki / katalogi.

TortoiseSVN to szeroko rozpowszechniony program SVN z integracją w eksploratorze. Program TortoiseProc.exe TortoiseSVN można uruchomić w wierszu poleceń i wyświetla on okno dialogowe w celu zebrania danych wejściowych użytkownika. Dzięki temu polecenia, które są dostępne jako menu kontekstowe w eksploratorze, można wykonywać również w wierszu poleceń. Dlatego można go również zintegrować jako rozszerzenie powłoki w CodeBlocks. Na przykład polecenie:

```
TortoiseProc.exe /command:diff /path:$file
```

porówna wybrany plik w eksploratorze plików CodeBlocks z bazą SVN. Zobacz Rysunek 2.9, jak zintegrować to polecenie.

Uwaga:
W przypadku plików znajdujących się pod kontrolą SVN eksplorator plików wyświetla ikony nakładek, jeśli są one aktywowane za pomocą menu „View (Widok)" / „SVN Decorators (Dekoratory SVN)".



Rysunek 2.9: Dodawanie rozszerzenia powłoki do menu kontekstowego

Przykład:
Możesz użyć eksploratora plików, aby porównać pliki lub katalogi. Wykonaj następujące kroki:

1. Dodaj nazwę za pomocą menu „Settings (Ustawienia)” / „Environment (Środowisko)” / „PowerShell”. Jest to wyświetlane jako wpis w menu interpretera i menu kontekstowym.
2. Wybierz ścieżkę bezwzględną pliku wykonywalnego Diff (np. `kdiff3`). Dostęp do programu uzyskuje się za pomocą zmiennej `$interpreter`.
3. Dodaj parametry interpretera

```
Diff;$interpreter $mpaths;;;
```

To polecenie zostanie wykonane przy użyciu wybranych plików lub katalogów jako parametru. Dostęp do wyboru odbywa się za pośrednictwem zmiennej `$mpaths`. Jest to łatwy sposób na różnicowanie plików lub katalogów.

Uwaga:
Wtyczka obsługuje używanie zmiennych CodeBlocks w rozszerzeniu powłoki.

$interpreter
Nazwij ten plik wykonywalny.
$fname
Nazwa pliku bez rozszerzenia.
$fext
Rozszerzenie wybranego pliku.

$file
Nazwa pliku.
$relfile
Nazwa pliku bez informacji o ścieżce.
$dir
Nazwa wybranego katalogu.
$reldir
Nazwa katalogu bez informacji o ścieżce.
$path
Ścieżka bezwzględna.
$relpath
Ścieżka względna pliku lub katalogu.
$mpaths
Lista aktualnie wybranych plików lub katalogów.
$inputstr{<msg>}
Ciąg wprowadzany w oknie komunikatu.
$parentdir
Katalog nadrzędny (../).

Uwaga:
Wpisy rozszerzeń powłoki są również dostępne jako menu kontekstowe w edytorze CodeBlocks.

2.8 Browse Tracker (Przeglądacz Abstraktu)

Browse Tracker to wtyczka, która pomaga w nawigacji między ostatnio otwartymi plikami w CodeBlocks. Lista ostatnio otwieranych plików jest zapisywana w historii. Za pomocą menu „View (Widok)” / „Browse Tracker” / „Clear All (Wyczyść wszystko)” historia jest czyszczona.

Za pomocą okna „Browsed Tabs (Przeglądane karty)” możesz poruszać się pomiędzy elementami ostatnio otwartych plików, korzystając z wpisu menu „View (Widok)” / „Browse Tracker” / „Backward Ed (Edycja wsteczna) / Forward Ed (Edycja do przodu)” lub skrótu Alt-Lewo/Alt-Prawo. Menu Browse Tracker jest również dostępne jako menu kontekstowe. Znaczniki są zapisywane w pliku układu `<projectName>.bmarks`

Częstą procedurą przy tworzeniu oprogramowania jest zmaganie się z zestawem funkcji, które są implementowane w różnych plikach. Wtyczka BrowseTracks pomoże Ci rozwiązać ten problem, pokazując kolejność, w jakiej pliki zostały wybrane. Następnie możesz wygodnie nawigować po wywołaniach funkcji.

Wtyczka umożliwia nawet przeglądanie znaczników w każdym pliku w edytorze CodeBlocks. Pozycja kursora jest zapamiętywana dla każdego pliku. Możesz ustawić te znaczniki za pomocą pozycji menu „View (Widok)” / „Browse Tracker” / „Set BrowseMarks (Ustaw znaczniki przeglądania)” lub zaznaczając linię lewym przyciskiem myszy. Znacznik z † jest wyświetlany na lewym marginesie. Za pomocą menu „View (Widok)” / „Browse Tracker” / „Prev Mark (Poprzedni znacznik) / Next Mark (Następny znacznik)” lub skrótu Alt-up / Alt-down możesz poruszać się po znacznikach w pliku. Jeśli chcesz poruszać się w pliku między znacznikami posortowanymi według numerów linii, po prostu wybierz menu „View (Widok)” / „Browse Tracker” / „Sort BrowseMark (Sortuj znaczniki przeglądania)”.

Za pomocą 'Clear BrowseMark (Wyczyść znacznik przeglądania)' znacznik w wybranym wierszu jest usuwany. Jeśli znacznik jest ustawiony dla wiersza, przytrzymanie lewego przycisku myszy przez 1/4 sekundy i naciśnięcie klawisza Ctrl spowoduje usunięcie znacznika dla tego wiersza. Poprzez menu 'Clear All BrowseMarks (Wyczyść wszystkie znaczniki przeglądania' lub za pomocą Ctrl-lewego kliknięcia na dowolnym nieoznaczonym wierszu spowoduje zresetowanie znaczników w pliku.

Ustawienia wtyczki można skonfigurować w menu „Settings (Ustawienia)” / „Editor (Edytor)” / „Browse Tracker”:

Mark Style (Styl znacznika)
Znaczniki przeglądania są domyślnie wyświetlane jako † na marginesie. Przy ustawieniu „Book_Marks (Zakładki książkowe)” będą wyświetlane jak zakładki jako niebieska strzałka na marginesie. Po ukryciu wyświetlanie znaczników przeglądania jest wyłączone.
Toggle Browse Mark key (Przełącz klawisz znacznika przeglądania)
Znaczniki można ustawiać lub usuwać, klikając lewym przyciskiem myszy lub klikając, przytrzymując klawisz Crtl.
Toggle Delay (Przełącz opóźnienie)
Czas przytrzymania lewego przycisku myszy, aby przejść do trybu znacznika przeglądania.
Clear All BrowseMarks (Wyczyść wszystkie znaczniki przeglądania)
przytrzymując klawisz Ctrl, klikając po prostu lub dwukrotnie lewym przyciskiem myszy.

Konfiguracja wtyczki jest przechowywana w katalogu danych aplikacji w pliku `default.conf`. Jeśli używasz funkcji osobowości CodeBlocks, konfiguracja jest odczytywana z pliku `<personality>.conf`.

2.9 SVN Support (Wsparcie SVN)

Obsługa systemu kontroli wersji SVN jest zawarta w wtyczce CodeBlocks TortoiseSVN. Poprzez menu „TortoiseSVN” / „Plugin settings (Ustawienia wtyczki)” możesz skonfigurować dostępne polecenia svn w zakładce „Integration (Integracja)”.

Menu integration (Integracja menu)
Dodaj wpis TortoiseSVN z różnymi ustawieniami na pasku menu.
Project manager (Menedżer projektu)
Aktywuj polecenia TortoiseSVN w menu kontekstowym menedżera projektu.
Editor (Edytor)
Aktywuj polecenia TortoiseSVN w menu kontekstowym edytora.

W ustawieniach wtyczki możesz skonfigurować, które polecenia svn są dostępne za pośrednictwem menu lub menu kontekstowego. Integracja kart zapewnia wpisy „Edit main menu (Edytuj menu główne)” i „Edit popup menu (Edytuj menu podręczne)”, aby skonfigurować te polecenia.

Uwaga:
Eksplorator plików w CodeBlocks używa różnych nakładek ikon do wskazywania statusu svn. Polecenia TortoiseSVN (Żółw SVN) są tutaj zawarte w menu kontekstowym.

2.10 LibFinder (Bibliotekarz)

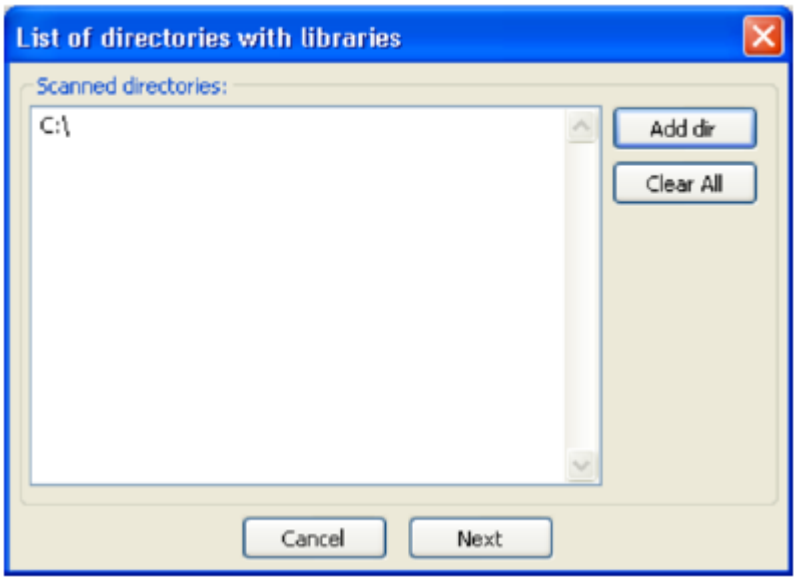
Jeśli chcesz użyć niektórych bibliotek w swojej aplikacji, musisz skonfigurować swój projekt, aby ich używał. Taki proces konfiguracji może być trudny i irytujący, ponieważ każda biblioteka może używać schematu opcji niestandardowych. Innym problemem jest to, że konfiguracja różni się na platformach, co skutkuje niezgodnością między projektami Unix i Windows.

LibFinder oferuje dwie główne funkcjonalności:

- Wyszukiwanie bibliotek zainstalowanych w systemie
- Dołączanie biblioteki do projektu za pomocą kilku kliknięć myszką, co czyni projekt niezależnym od platformy

2.10.1 Searching for libraries (Wyszukiwanie bibliotek)

Wyszukiwanie bibliotek jest dostępne w menu „Plugins (Wtyczki)”/„ Library finder 9Wyszukiwarka bibliotek)”. Jego celem jest wykrywanie bibliotek zainstalowanych w systemie i przechowywanie wyników w bazie danych LibFindera (należy pamiętać, że wyniki te nie są zapisywane w plikach projektu CodeBlocks). Wyszukiwanie rozpoczyna się od dialogu, w którym można podać zestaw katalogów z zainstalowanymi bibliotekami. LibFinder przeskanuje je rekurencyjnie, więc jeśli nie jesteś pewien, możesz wybrać kilka ogólnych katalogów. Możesz nawet wprowadzić całe dyski — w takim przypadku proces wyszukiwania zajmie więcej czasu, ale może wykryć więcej bibliotek (patrz Rysunek 2.10).

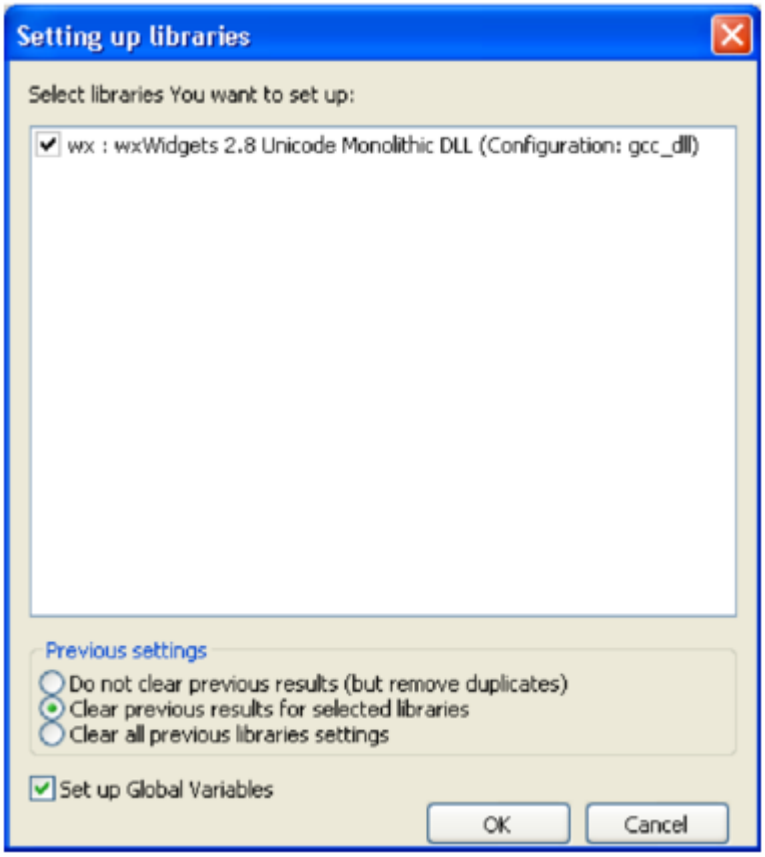


Rysunek 2.10: Lista katalogów

Gdy LibFinder skanuje biblioteki, używa specjalnych reguł, aby wykryć obecność biblioteki. Każdy zestaw reguł znajduje się w pliku xml. Obecnie LibFinder może wyszukiwać wxWidgets 2.6/2.8, CodeBlocks SDK i GLFW - lista zostanie rozszerzona w przyszłości.

Uwaga:
Aby uzyskać więcej szczegółów na temat dodawania obsługi bibliotek do LibFinder, przeczytaj `src/plugins/contrib/lib_finder/lib_finder/readme.txt` w źródłach CodeBlocks.

Po zakończeniu skanowania LibFinder wyświetli wyniki (patrz rysunek 2.11).



Rysunek 2.11: Wyniki wyszukiwania

Na liście zaznaczasz biblioteki, które powinny być przechowywane w bazie danych LibFinder. Należy pamiętać, że każda biblioteka może mieć więcej niż jedną prawidłową konfigurację, a ustawienia dodane wcześniej są bardziej prawdopodobne do użycia podczas budowania projektów.

Poniżej listy możesz wybrać, co zrobić z wynikami poprzednich skanów:

Do not clear previous results (Nie czyść poprzednich wyników)
: Ta opcja działa jak aktualizacja istniejących wyników — dodaje nowe i aktualizuje te, które już istnieją. Ta opcja nie jest zalecana.

Second option (Clear previous results for selected libraries) (Druga opcja (Wyczyść poprzednie wyniki dla wybranych bibliotek))
: wyczyści wszystkie wyniki dla bibliotek, które zostały wybrane przed dodaniem nowych wyników. Jest to zalecana opcja.

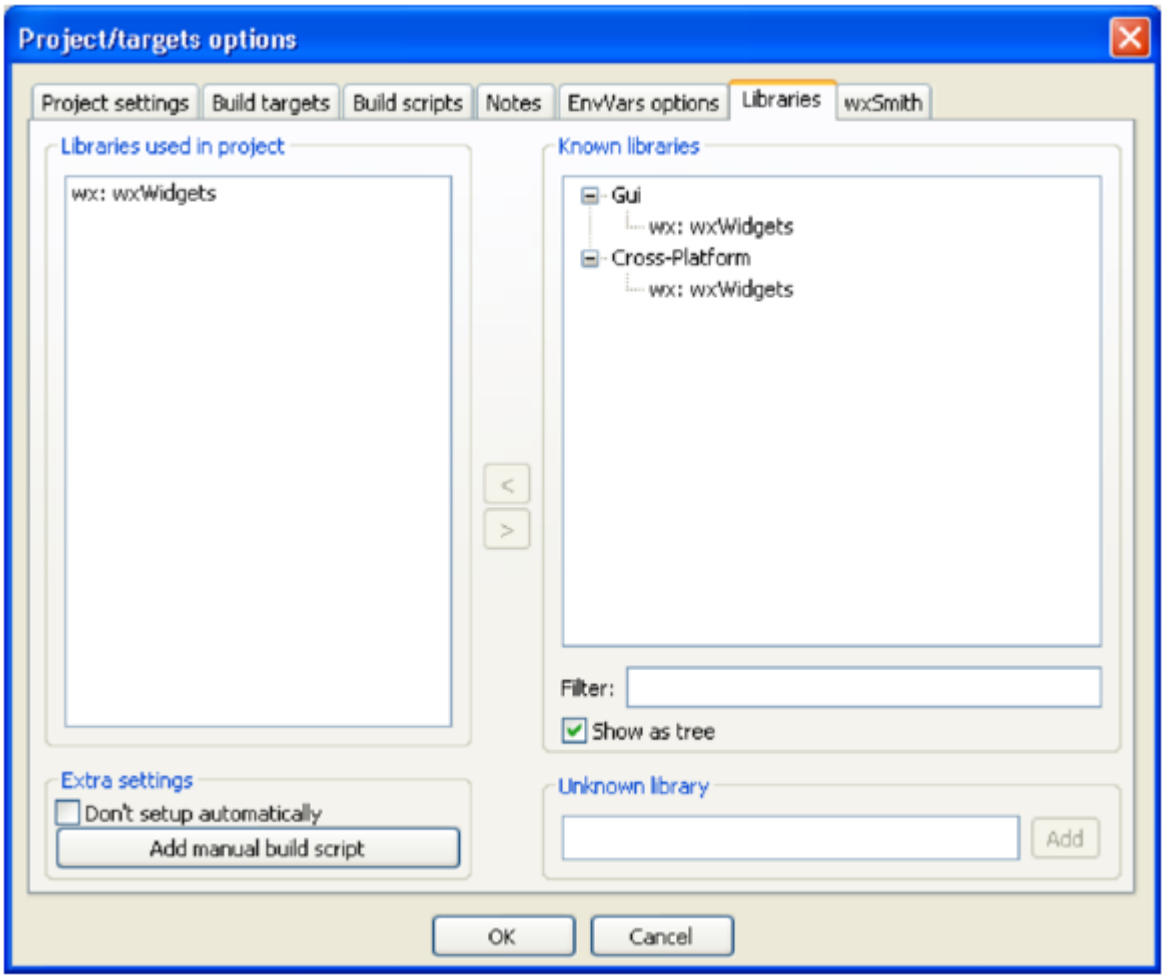
Clear all previous library settings (Wyczyść wszystkie poprzednie ustawienia biblioteki)
: po wybraniu tej opcji baza danych LibFinder zostanie wyczyszczona przed dodaniem nowych wyników. Jest to przydatne, gdy chcesz wyczyścić nieprawidłową bazę danych LibFinder.

Inną opcją w tym oknie dialogowym jest „Set up Global Variables (Konfigurowanie zmiennych globalnych)”. Po zaznaczeniu tej opcji LibFinder spróbuje automatycznie skonfigurować zmienne globalne, które są również używane do pomocy w obsłudze bibliotek.

Jeśli masz zainstalowany pkg-config w swoim systemie (jest instalowany automatycznie w większości wersji Linuksa), LibFinder udostępni również biblioteki z tego narzędzia. Nie ma potrzeby wykonywania skanowania dla nich — są one automatycznie ładowane podczas uruchamiania CodeBlocks.

2.10.2 Including libraries in projects (Dodawanie bibliotek do projektów)

LibFinder (Bibliotekarz) dodaje dodatkową zakładkę w Project Properties 'Libraries' (Właściwości projektu ‘Biblioteki’) - ta zakładka pokazuje biblioteki używane w projekcie i biblioteki znane w LibFinder. Aby dodać bibliotekę do projektu, wybierz ją w prawym panelu i kliknij przycisk <. Aby usunąć bibliotekę z projektu, wybierz ją w lewym panelu i kliknij przycisk > (patrz Rysunek 2.12).



Rysunek 2.12: Konfiguracja projektu

Możesz filtrować biblioteki znane LibFinderowi, podając filtr wyszukiwania. Pole wyboru „Show as Tree (Pokaż jako drzewo)” umożliwia przełączanie między widokiem skategoryzowanym i nieskategoryzowanym.

Jeśli chcesz dodać bibliotekę, która nie jest dostępna w bazie danych LibFinder, możesz użyć pola „Unknown Library (Nieznana biblioteka)”. Należy pamiętać, że należy wprowadzić krótki kod biblioteki (który zwykle odpowiada nazwie zmiennej globalnej) lub nazwę biblioteki w pkg-config. Listę sugerowanych krótkich kodów można znaleźć w [Internet] Global Variables (Zmienne globalne). Użycie tej opcji jest zalecane tylko podczas przygotowywania projektu do zbudowania na innych maszynach, na których taka biblioteka istnieje i jest prawidłowo wykrywana przez LibFinder. Można uzyskać dostęp do zmiennej globalnej w CodeBlocks, takiej jak:

```
$(#GLOBAL_VAR_NAME.include)
```

Zaznaczenie opcji „Don’t setup automatically” (Nie instaluj automatycznie) powiadomi LibFinder, że nie powinien dodawać bibliotek automatycznie podczas kompilacji tego projektu. W takim przypadku LibFinder może zostać wywołany ze skryptu kompilacji. Przykład takiego skryptu jest generowany i dodawany do projektu poprzez naciśnięcie „Add manual build script” (Dodaj ręczny skrypt kompilacji).

2.10.3 Using LibFinder and projects generated from wizards (Korzystanie z LibFinder i projektów generowanych przez kreatory)

Kreatory stworzą projekty, które nie używają LibFindera. Aby zintegrować je z tą wtyczką, musisz ręcznie zaktualizować opcje kompilacji projektu. Można to łatwo osiągnąć, usuwając wszystkie ustawienia specyficzne dla biblioteki i dodając bibliotekę za pomocą zakładki „Libraries (Biblioteki)” we właściwościach projektu.

Taki projekt staje się wieloplatformowy. Dopóki używane biblioteki są zdefiniowane w bazie danych LibFindera, opcje kompilacji projektu będą automatycznie aktualizowane, aby pasowały do ustawień biblioteki specyficznych dla platformy.

2.11 AutoVersioning (Automatyczne wersjonowanie)

Wtyczka do kontroli wersji aplikacji, która zwiększa wersję i numer kompilacji aplikacji za każdym razem, gdy zostanie wprowadzona zmiana i przechowuje ją w pliku `version.h` z łatwymi w użyciu deklaracjami zmiennych. Posiada również funkcję zatwierdzania zmian w stylu SVN, edytor schematów wersji, generator dziennika zmian i wiele więcej †.

2.11.1 Introduction (Wprowadzenie)

Pomysł na wtyczkę AutoVersioning zrodził się podczas opracowywania oprogramowania pre-alpha, które wymagało informacji o wersji i statusie. Byłem zbyt zajęty kodowaniem, nie miałem czasu na utrzymanie numeru wersji, więc postanowiłem opracować wtyczkę, która wykona zadanie z jak najmniejszą interwencją.

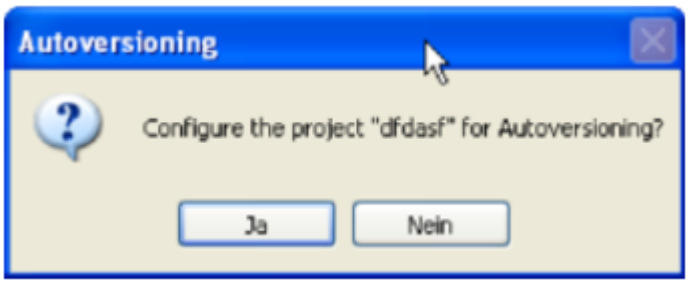
2.11.2 Features (Funkcje)

Poniżej znajduje się podsumowanie funkcji, jakie obejmuje ta wtyczka:

- Obsługuje języki C i C++.
- Generuje i automatycznie zwiększa zmienne wersji.
- Edytor statusu oprogramowania.
- Zintegrowany edytor schematów do zmiany zachowania automatycznego zwiększania wartości wersji.
- Deklaracje daty jako miesiąc, dzień i rok.
- Wersja w stylu Ubuntu.
- Sprawdzanie rewizji SVN.
- Generator dziennika zmian.
- Działa w systemach Windows i Linux.

2.11.3 Usage (Użytkowanie)

Wystarczy przejść do menu „Project” / „Autoversioning”. Pojawi się okno pop-up takie jak to:



Rysunek 2.13: Konfigurowanie projektu pod kątem autowersji

Po kliknięciu „Yes (tak)” w polu wiadomości „ask to configure (prośba o skonfigurowanie)” otworzy się główne okno dialogowe konfiguracji automatycznego wersjonowania, w którym możesz skonfigurować informacje o wersji swojego projektu.

Po skonfigurowaniu projektu do automatycznego wersjonowania ustawienia wprowadzone w oknie dialogowym konfiguracji zostaną zapisane w pliku projektu, a plik `version.h` zostanie utworzony. Na razie za każdym razem, gdy klikniesz menu „Project (Projekt)”/„Autoversioning (Autowersjonowanie)”, pojawi się okno dialogowe konfiguracji, w którym możesz edytować wersję projektu i ustawienia związane z wersjonowaniem, chyba że nie zapiszesz nowych zmian wprowadzonych przez wtyczkę do pliku projektu.

2.11.4 Dialog notebook tabs (Karty notatnika dialogowego)

2.11.4.1 Version Values (Wartości wersji)

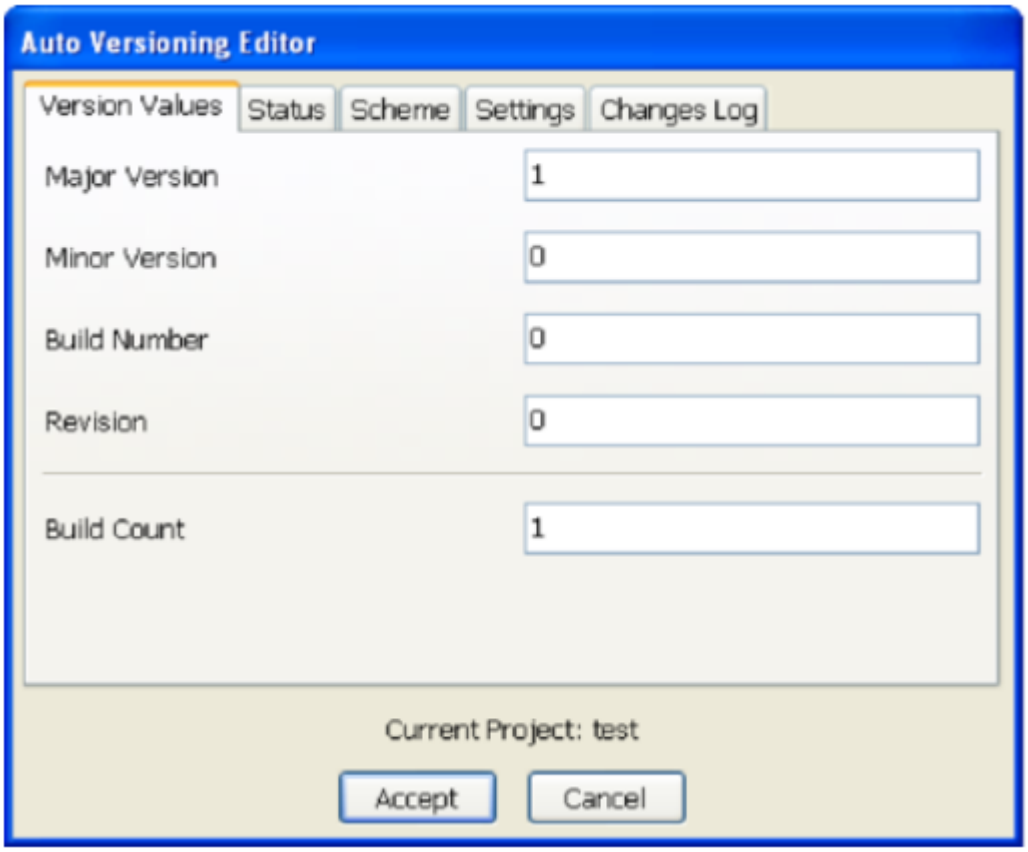
Tutaj wystarczy wprowadzić odpowiednie wartości wersji lub pozwolić wtyczce automatycznego zarządzania wersjami je zwiększyć (patrz rysunek 2.14).

Major
: Zwiększa się o 1, gdy wersja drugorzędna osiągnie swoje maksimum

Minor
: Zwiększa się o 1, gdy numer kompilacji przekroczy barierę czasu kompilacji, wartość jest resetowana do 0, gdy osiągnie maksymalną wartość.

Build Number (Numer kompilacji)
: (również równoważny z wydaniem) — zwiększa się o 1 za każdym razem, gdy zwiększa się numer rewizji.

Revision
: Zwiększa się losowo, gdy projekt został zmodyfikowany, a następnie skompilowany.



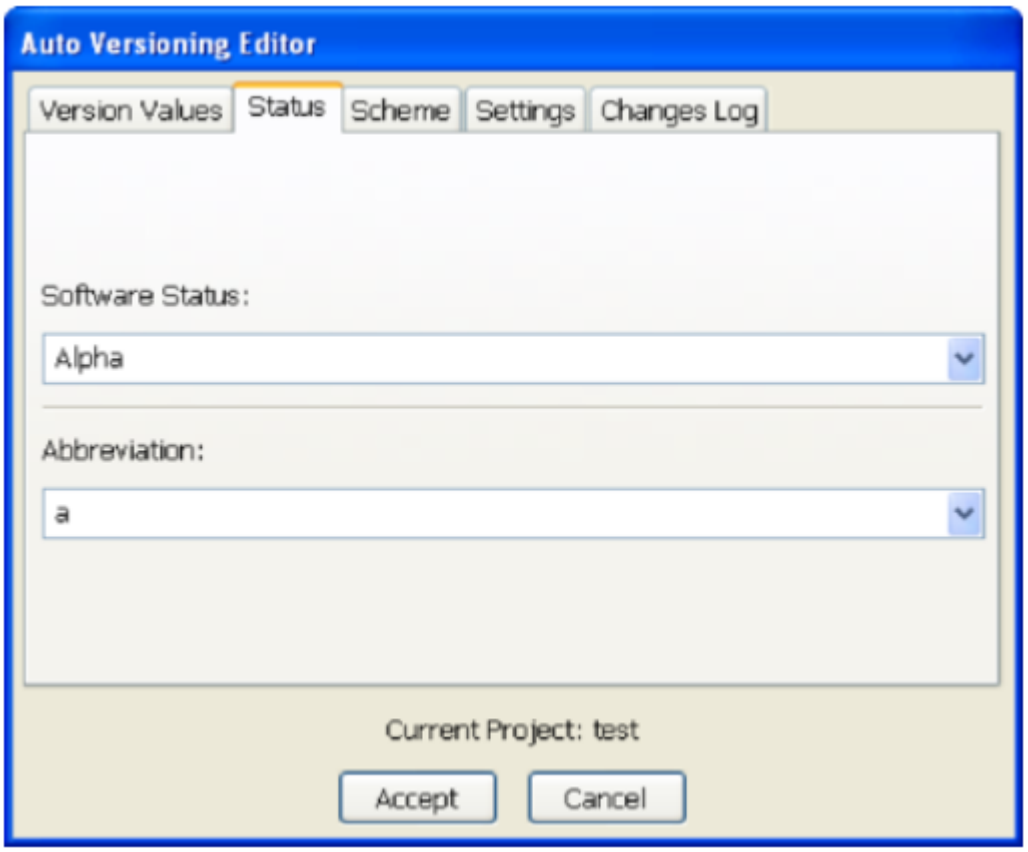
Rysunek 2.14: Ustaw wartości wersji

2.11.4.2 **Status** (Stan)

Niektóre pola umożliwiają śledzenie statusu oprogramowania za pomocą listy predefiniowanych wartości, co ułatwia pracę (patrz rysunek 2.15).

Software Status (Status oprogramowania)
: Typowy przykład powinien brzmieć v1.0 Alpha

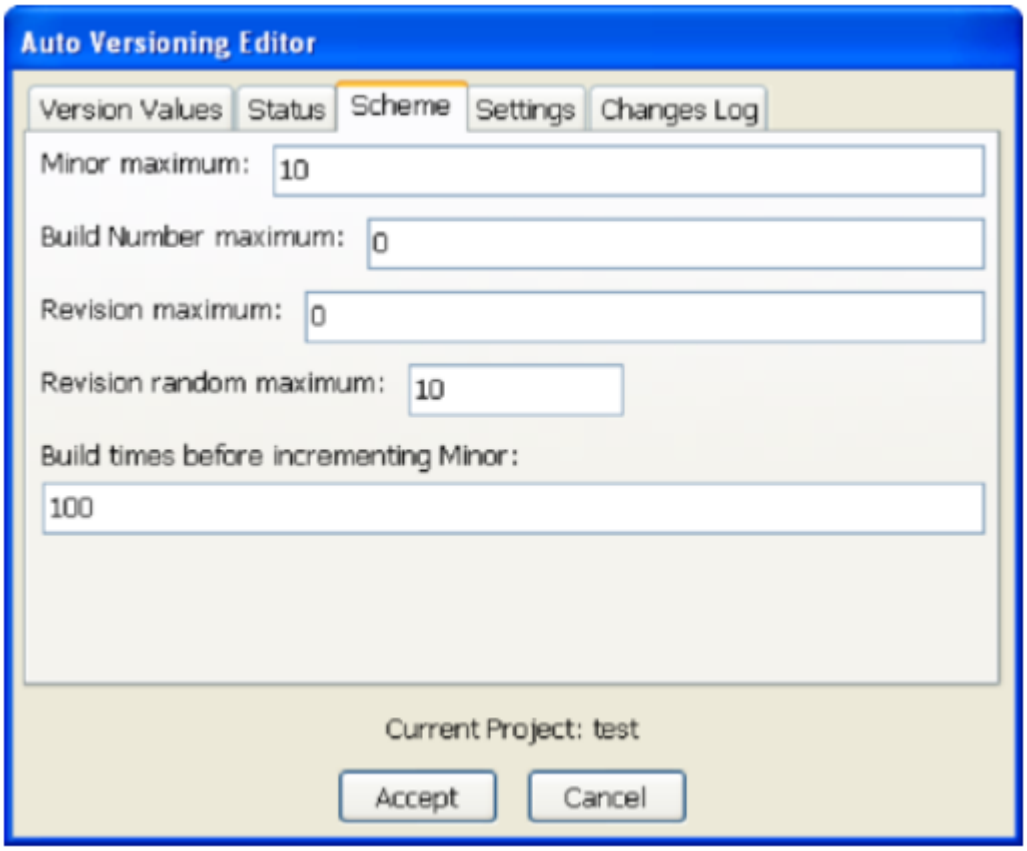
Abbreviation (Skrót)
: Tak samo jak status oprogramowania, ale tak: v1.0a



Rysunek 2.15: Ustaw status autowersji

2.11.4.3 **Scheme** (Schemat)

Umożliwia edycję sposobu, w jaki wtyczka będzie zwiększać wartości wersji (patrz rysunek 2.16).



Rysunek 2.16: Schemat autowersji

Minor maksimum
Maksymalna liczba, jaką może osiągnąć wartość Minor, po osiągnięciu tej wartości Major jest zwiększany o 1, a przy następnym kompilowaniu projektu Minor jest ustawiany na 0.

Build Number maximum (Maksymalny numer kompilacji)
Po osiągnięciu wartości, przy następnym kompilowaniu projektu jest ustawiany na 0. Wpisz 0, aby uzyskać nieograniczoną wartość.

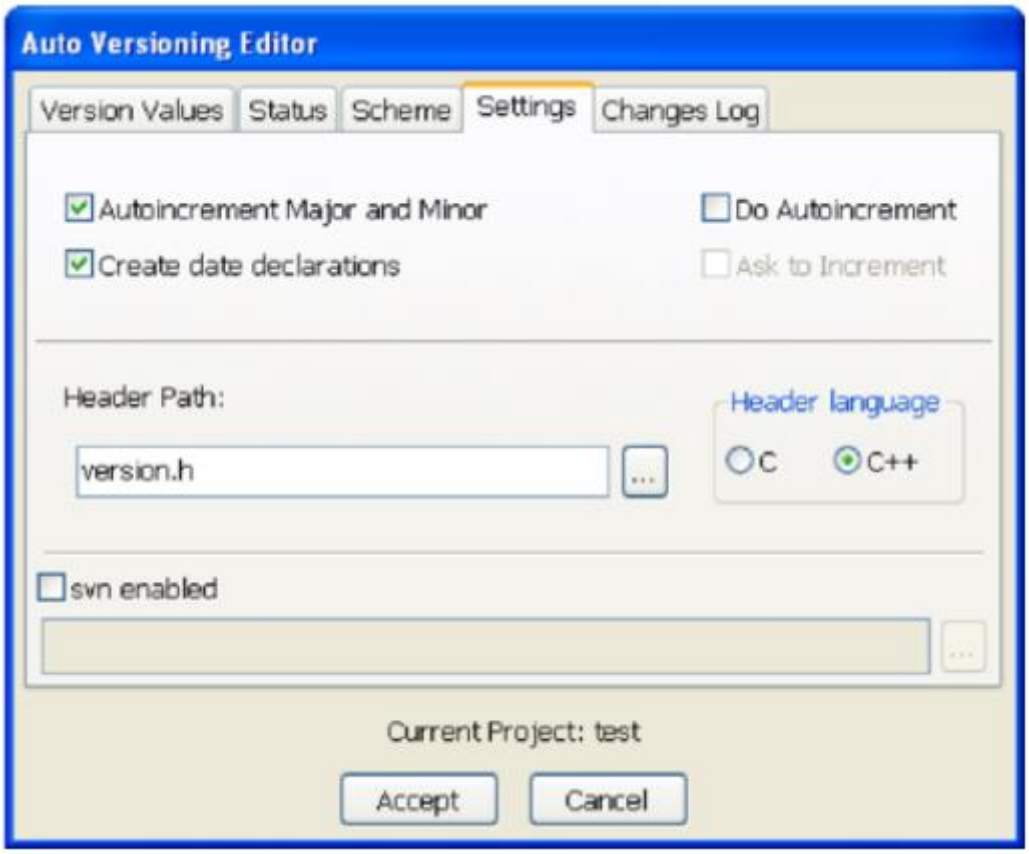
Revision maximum (Maksymalna rewizja)
Tak samo jak maksymalny numer kompilacji. Wpisz 0, aby uzyskać nieograniczoną wartość.

Revision random maximum (Losowa maksymalna rewizja)
Rewizja zwiększa się o losowe liczby, które wybierzesz, jeśli wpiszesz tutaj 1, rewizja oczywiście zwiększy się o 1.

Build times before incrementing Minor (Czasy kompilacji przed zwiększeniem Minor)
Po pomyślnych zmianach kodu i kompilacji historia kompilacji zostanie zwiększona, a po osiągnięciu tej wartości Minor zostanie zwiększony.

2.11.4.4 Settings (Ustawienia)

Tutaj możesz skonfigurować niektóre ustawienia funkcji automatycznego zarządzania wersjami (patrz rysunek 2.17).



Rysunek 2.17: Ustawienia autowersji

Autoincrement Major and Minor (Autoinkrementacja Major i Minor)
Pozwala wtyczce zwiększać te wartości za pomocą schematu. Jeśli nie jest zaznaczone, zwiększane będą tylko numer kompilacji i rewizja.

Create date declarations (Utwórz deklaracje daty)
Utwórz wpisy w pliku `version.h` z datami i wersją w stylu Ubuntu.

Do Auto Increment (Wykonaj Auto Inkrementację)
To polecenie mówi wtyczce, aby automatycznie zwiększała zmiany, gdy zostanie wprowadzona modyfikacja. Zwiększenie nastąpi przed kompilacją.

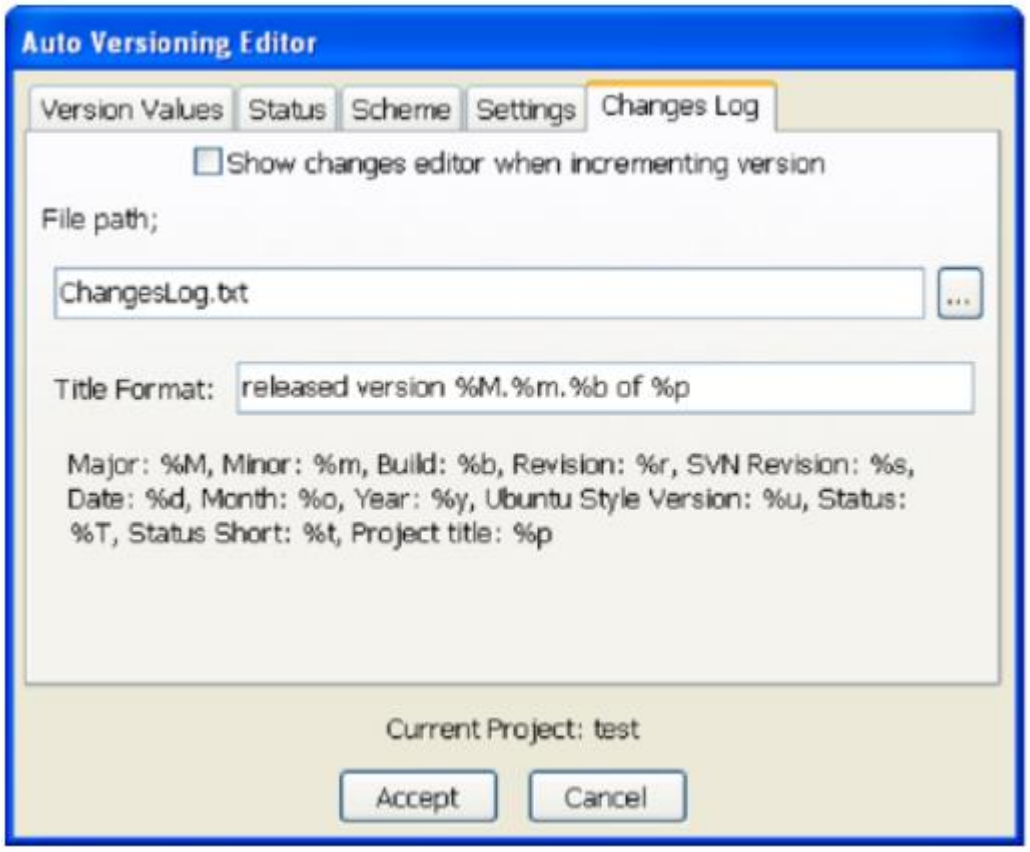
Header language (Język nagłówka)
Wybierz język wyjściowy `version.h`

Ask to increment (Pytaj o zwiększenie)
Jeśli zaznaczono, Wykonaj automatyczne zwiększanie, przed kompilacją (jeśli zostały wprowadzone zmiany) zostaniesz poproszony o zwiększenie wartości wersji.

svn enabled (SVN włączony)
Wyszukaj wersję i datę SVN w bieżącym folderze i wygeneruj prawidłowe wpisy w `version.h`

2.11.4.5 Changes Log (Dziennik zmian)

Umożliwia to wprowadzenie wszystkich zmian wprowadzonych w projekcie w celu wygenerowania pliku `ChangesLog.txt` (patrz rysunek 2.18).



Rysunek 2.18: Dziennik zmian w autowersjonowaniu

Show changes editor when incrementing version (Pokaż edytor zmian podczas zwiększania wersji)
Wyświetli edytor dziennika zmian podczas zwiększania wersji.
Title Format (Format tytułu)
Tytuł, który można sformatować, z listą wstępnie zdefiniowanych wartości.

2.11.5 Including in your code (Dołączanie do kodu)

Aby użyć zmiennych wygenerowanych przez wtyczkę, po prostu `#include <version.h>`.
Przykładowy kod wyglądałby tak:

```
#include <iostream>
#include "version.h"

void main(){
	std::cout<<AutoVersion::Major<<endl;
}
```

2.11.5.1 Output of version.h (Wyjście version.h)

Wygenerowany plik nagłówkowy. Oto przykładowa zawartość pliku w trybie c++:

```
 #ifndef VERSION_H
  #define VERSION_H
```

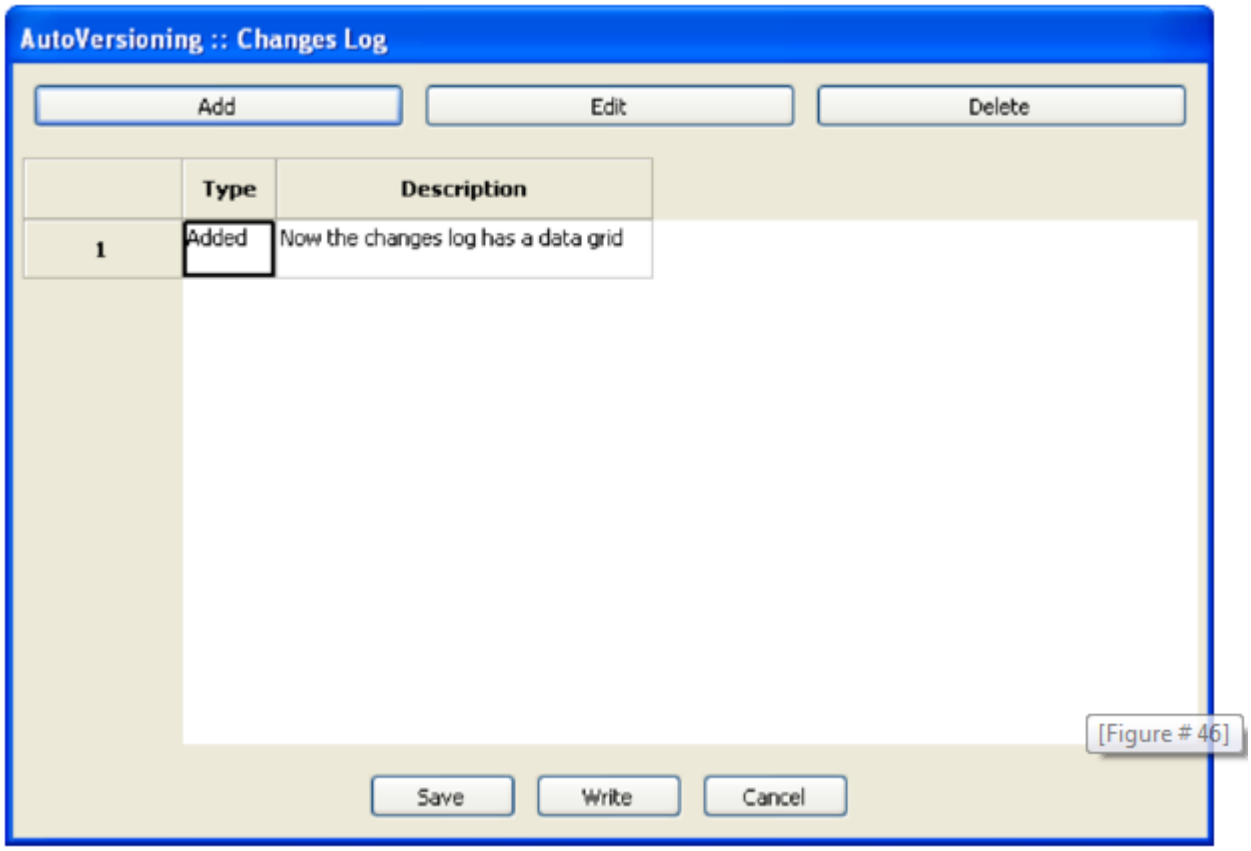
```
 namespace AutoVersion{

  // Typy wersji daty
  static const char DATE[] = "15";
  static const char MONTH[] = "09";
  static const char YEAR[] = "2007";
  static const double UBUNTU_VERSION_STYLE = 7.09;

  // Status oprogramowania
  static const char STATUS[] = "Pre-alpha";
  static const char STATUS_SHORT[] = "pa";

  // Wersja standardowa Typ
  static const long MAJOR = 0;
  static const long MINOR = 10;
  static const long BUILD = 1086;
  static const long REVISION = 6349;

  // Różne wersje typów
 static const long BUILDS_COUNT = 1984;
  #define RC_FILEVERSION 0,10,1086,6349
  #define RC_FILEVERSION_STRING "0, 10, 1086, 6349\0"
  static const char FULLVERSION_STRING[] = "0.10.1086.6349";

 }
 #endif //VERSION_h
```

Tryb C jest taki sam jak w C++, ale bez przestrzeni nazw:

```
#ifndef VERSION_H
 #define VERSION_H

  // Typy wersji daty
  static const char DATE[] = "15";
  static const char MONTH[] = "09";
  static const char YEAR[] = "2007";
  static const double UBUNTU_VERSION_STYLE = 7.09;

  // Status oprogramowania
  static const char STATUS[] = "Pre-alpha";
  static const char STATUS_SHORT[] = "pa";

  // Wersja standardowa Typ
  static const long MAJOR = 0;
  static const long MINOR = 10;
  static const long BUILD = 1086;
  static const long REVISION = 6349;

  // Różne wersje typów
  static const long BUILDS_COUNT = 1984;
  #define RC_FILEVERSION 0,10,1086,6349
  #define RC_FILEVERSION_STRING "0, 10, 1086, 6349\0"
  static const char FULLVERSION_STRING[] = "0.10.1086.6349";

 #endif //VERSION_h
```

2.11.6 Change log generator (Generator dziennika zmian)

To okno dialogowe jest dostępne z menu „Project (Projekt)”/„ Changes Log (Dziennik zmian)”. Również jeśli zaznaczono opcję „Show changes editor” (Pokaż edytor zmian) podczas zwiększania wersji w ustawieniach dziennika zmian, okno zostanie otwarte, aby umożliwić wprowadzenie listy zmian po modyfikacji źródeł projektu lub zdarzeniu zwiększania (patrz Rysunek 2.19).



Rysunek 2.19: Zmiany w projekcie

2.11.6.1 **Buttons Summary** (Podsumowanie przycisków)

Add (Dodaj)
: Dołącza wiersz do siatki danych

Edit (Edytuj)
: Umożliwia modyfikację wybranej komórki

Delete (Usuń)
: Usuwa bieżący wiersz z siatki danych

Save (Zapisz)
: Przechowuje w pliku tymczasowym (`changes.tmp`) rzeczywiste dane w celu późniejszego przetworzenia w pliku dziennika zmian

Write (Zapisz)
: Przetwarza dane siatki danych w pliku dziennika zmian

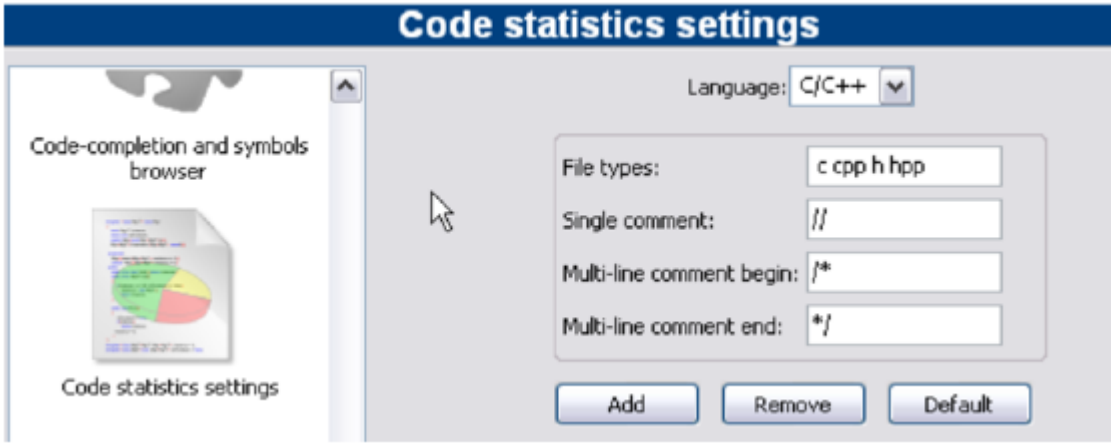
Cancel (Anuluj)
: Po prostu zamyka okno dialogowe bez podejmowania żadnych działań

Oto przykład danych wyjściowych wygenerowanych przez wtyczkę do pliku `ChangesLog.txt`:

```
03 September 2007
    released version 0.7.34 of AutoVersioning-Linux (wydana wersja)
```

```
      Change log:
         -Fixed: pointer declaration (deklaracja wskaźnika)
         -Bug: blah blah (ble ble)

  02 September 2007
     released version 0.7.32 of AutoVersioning-Linux

      Change log:
         -Documented some areas of the code (Udokumentowano niektóre obszary
                                              kodu)
         -Reorganized the code for readability (Zreorganizowano kod w celu
                                                zwiększenia czytelności)

  01 September 2007
     released version 0.7.30 of AutoVersioning-Linux

      Change log:
         -Edited the change log window (Edytowano okno dziennika zmian)
         -
If the change log windows is leave blank no changes.txt is modified
```
(Jeśli okno dziennika zmian jest puste, plik changes.txt nie zostanie zmodyfikowany)

2.12 Code statistics (Statystyki kodu)



Rysunek 2.20: Konfiguracja statystyk kodu

Na podstawie wpisów w masce konfiguracji ta prosta wtyczka wykrywa proporcje kodu, komentarzy i pustych linii dla projektu. Ocena jest wywoływana za pomocą polecenia menu „Plugins (Wtyczki)” / „Code statistics (Statystyki kodu)”.

2.13 Searching Available Source Code (Przeszukiwanie dostępnego kodu źródłowego)

Ta wtyczka umożliwia wybranie terminu w edytorze i wyszukanie go za pomocą menu kontekstowego „Search at Koders (Szukaj w Koders)” w bazie danych [?]. Dialog oferuje dodatkowe możliwości filtrowania według języków programowania i licencji.

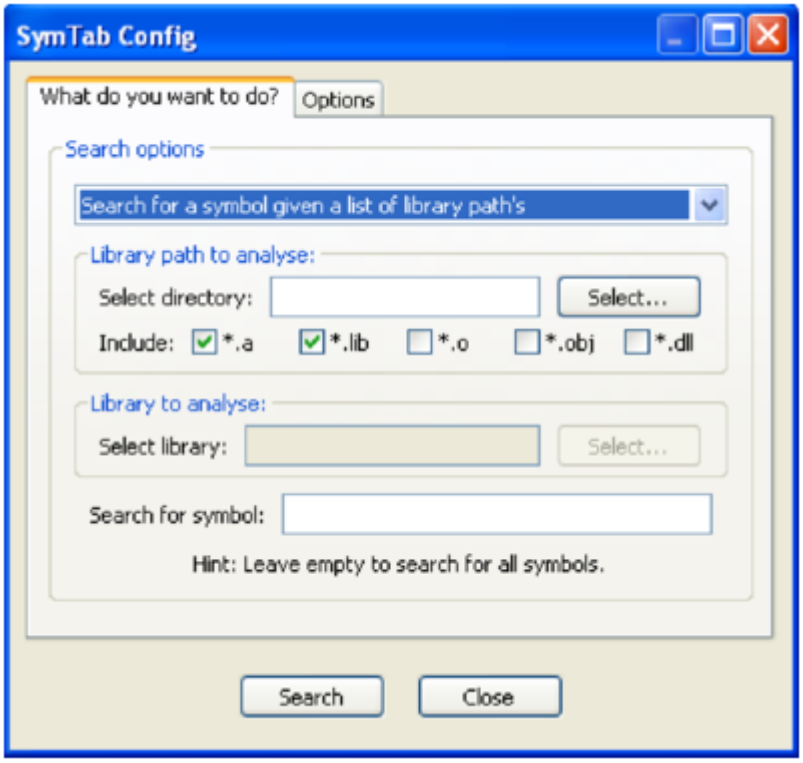
To wyszukiwanie w bazie danych pomoże Ci znaleźć kod źródłowy pochodzący z innych światowych projektów uniwersytetów, konsorcjów i organizacji, takich jak Apache, Mozilla, Novell Forge, SourceForge i wielu innych, który można ponownie wykorzystać bez konieczności ponownego wyważania otwartych drzwi za każdym razem. Prosimy o przestrzeganie licencji kodu źródłowego w każdym indywidualnym przypadku.

2.14 Code profiler (Profiler kodu)

Prosty graficzny interfejs do GNU GProf Profiler.

2.15 Symbol Table Plugin (Wtyczka tabeli symboli)

Ta wtyczka umożliwia wyszukiwanie symboli w obiektach i bibliotekach. Opcje i ścieżka dla programu wiersza poleceń `nm` są zdefiniowane w zakładce „Options” (Opcje).



Rysunek 2.21: Konfigurowanie tabeli symboli

Kliknięcie „Search (Szukaj)” powoduje wyszukiwanie, wyniki programu NM są wyświetlane w oddzielnym oknie o nazwie „SymTabs Result (Wynik SymTabs)”. Nazwy obiektów lub bibliotek zawierających symbol są wymienione pod tytułem „NM’s Output (Wyjście NM)”.

3 Variable Expansion (Zmienna ekspansja)

CodeBlocks rozróżnia kilka typów zmiennych. Typy te służą do konfigurowania środowiska do tworzenia programu, a jednocześnie do poprawy łatwości obsługi i przenośności. Dostęp do zmiennych CodeBlocks jest uzyskiwany za pośrednictwem `$<name>`.

Envrionment Variable (Zmienne środowiskowe)
są ustawiane podczas uruchamiania CodeBlocks. Mogą modyfikować zmienne środowiskowe systemu, takie jak PATH. Może to być przydatne w przypadkach, gdy zdefiniowane środowisko jest niezbędne do tworzenia projektów. Ustawienia zmiennych środowiskowych w CodeBlocks są dokonywane w 'Settings (Ustawienia)' /'Environment (Środowisko)' /'Environment Variables (Zmienne środowiskowe)' .

Builtin Variables (Wbudowane zmienne)
są wstępnie zdefiniowane w CodeBlocks i można uzyskać do nich dostęp za pomocą ich nazw (szczegóły w sekcji 3.2).

Command Macros (Makra poleceń)
Ten typ zmiennych służy do kontrolowania procesu kompilacji. Aby uzyskać więcej informacji, zapoznaj się z sekcją 3.4.

Custom Variables (Zmienne niestandardowe)
to zmienne zdefiniowane przez użytkownika, które można określić w opcjach kompilacji projektu. Tutaj możesz na przykład zdefiniować swoją pochodną jako zmienną `MCU` i przypisać jej odpowiednią wartość. Następnie ustaw opcję kompilatora `-mcpu=$(MCU)`, a CodeBlocks automatycznie zastąpi zawartość. Za pomocą tej metody ustawienia projektu można dalej parametryzować.

Global Variables (Zmienne globalne)
są głównie używane do tworzenia CodeBlocks ze źródeł lub opracowań aplikacji wxWidgets. Te zmienne mają bardzo szczególne znaczenie. W przeciwieństwie do wszystkich innych, jeśli skonfigurujesz takie zmienne i udostępnisz plik swojego projektu innym, którzy *nie* skonfigurowali tego GV CodeBlocks poprosi użytkownika o skonfigurowanie zmiennej. To bardzo prosty sposób, aby upewnić się, że „inny programista" wie, co łatwo skonfigurować. CodeBlocks poprosi o wszystkie ścieżki, które są zazwyczaj niezbędne.

3.1 Syntax (Składnia)

CodeBlocks traktuje następujące funkcjonalnie identyczne sekwencje znaków wewnątrz kroków przed kompilacją, po kompilacji i kompilacji jako zmienne:

- $VARIABLE
- $(VARIABLE)
- ${VARIABLE}
- %VARIABLE%

Nazwy zmiennych muszą składać się ze znaków alfanumerycznych i nie są rozróżniane wielkością liter. Zmienne zaczynające się od pojedynczego znaku hash (#) są interpretowane jako globalne zmienne użytkownika (szczegóły w sekcji 3.7). Nazwy wymienione poniżej są interpretowane jako typy wbudowane.

Zmienne, które nie są ani globalnymi zmiennymi użytkownika, ani typami wbudowanymi, zostaną zastąpione wartością podaną w pliku projektu lub zmienną środowiskową, jeśli ta ostatnia się nie powiedzie.

Uwaga:
Definicje na poziomie celu mają pierwszeństwo przed definicjami na poziomie projektu.

3.2 List of available built-ins (Lista dostępnych wbudowanych elementów)

Zmienne wymienione tutaj są wbudowanymi zmiennymi CodeBlocks. Nie można ich używać w plikach źródłowych.

3.2.1 CodeBlocks workspace (Obszar roboczy CodeBlocks)

$(WORKSPACE_FILENAME), $(WORKSPACE_FILE_NAME), $(WORKSPACEFILE), $(WORKSPACEFILENAME)
Nazwa pliku bieżącego projektu obszaru roboczego (.workspace).

$(WORKSPACENAME), $(WORKSPACE_NAME)
Nazwa obszaru roboczego wyświetlana na karcie Projekty panelu Zarządzanie.

$(WORKSPACE_DIR), $(WORKSPACE_DIRECTORY), $(WORKSPACEDIR), $(WORKSPACEDIRECTORY)
Lokalizacja katalogu obszaru roboczego.

3.2.2 Files and directories (Pliki i katalogi)

$(PROJECT_FILENAME), $(PROJECT_FILE_NAME), $(PROJECT_FILE), $(PROJECTFILE)
Nazwa pliku aktualnie kompilowanego projektu.
$(PROJECT_NAME)
Nazwa aktualnie kompilowanego projektu.
$(PROJECT_DIR), $(PROJECTDIR), $(PROJECT_DIRECTORY)
Wspólny katalog najwyższego poziomu aktualnie kompilowanego projektu.
$(ACTIVE_EDITOR_FILENAME)
Nazwa pliku otwartego w aktualnie aktywnym edytorze.
$(ACTIVE_EDITOR_LINE)
Zwróć bieżący wiersz w aktywnym edytorze.
$(ACTIVE_EDITOR_COLUMN
Zwróć kolumnę bieżącego wiersza w aktywnym edytorze.
$(ACTIVE_EDITOR_DIRNAME)
katalog zawierający aktualnie aktywny plik (względem wspólnej ścieżki najwyższego poziomu).
$(ACTIVE_EDITOR_STEM)
Nazwa bazowa (bez rozszerzenia) aktualnie aktywnego pliku.
$(ACTIVE_EDITOR_EXT)
Rozszerzenie aktualnie aktywnego pliku.
$(ALL_PROJECT_FILES)
Ciąg zawierający nazwy wszystkich plików w bieżącym projekcie.
$(MAKEFILE)
Nazwa pliku makefile.
$(CODEBLOCKS), $(APP_PATH), $(APPPATH), $(APP-PATH)
Ścieżka do aktualnie uruchomionej instancji CodeBlocks.
$(DATAPATH), $(DATA_PATH), $(DATA-PATH)
Katalog „współdzielony” aktualnie uruchomionej instancji CodeBlocks.
$(PLUGINS)
Katalog `plugins` (wtyczek) aktualnie uruchomionej instancji CodeBlocks.
$(TARGET_COMPILER_DIR)
Katalog instalacyjny kompilatora, tzw. ścieżka główna.

3.2.3 Build targets (Cele kompilacji)

$(FOOBAR_OUTPUT_FILE)
Plik wyjściowy określonego celu.
$(FOOBAR_OUTPUT_DIR)
Katalog wyjściowy określonego celu.
$(FOOBAR_OUTPUT_BASENAME)
Podstawowa nazwa pliku wyjściowego (bez ścieżki, bez rozszerzenia) określonego celu.
$(TARGET_OUTPUT_DIR)
Katalog wyjściowy bieżącego celu.
$(TARGET_OBJECT_DIR)
Katalog obiektów bieżącego celu.
$(TARGET_NAME)
Nazwa bieżącego celu.
$(TARGET_OUTPUT_FILE)
Plik wyjściowy bieżącego celu.
$(TARGET_OUTPUT_BASENAME)
Podstawowa nazwa pliku wyjściowego (bez ścieżki, bez rozszerzenia) bieżącego celu.
$(TARGET_CC), $(TARGET_CPP), $(TARGET_LD), $(TARGET_LIB)
Plik wykonywalny narzędzia do kompilacji (kompilator, linker itp.) bieżącego celu.

3.2.4 Language and encoding (Język i kodowanie)

$(LANGUAGE)
Język systemowy w języku potocznym.
$(ENCODING)
Kodowanie znaków w języku potocznym.

3.2.5 Time and date (Czas i data)

$(TDAY)
Bieżąca data w formacie YYYYMMDD (na przykład 20051228)
$(TODAY)
Bieżąca data w formacie YYYY-MM-DD (na przykład 2005-12-28)
$(NOW)
Znacznik czasu w formacie YYYY-MM-DD-hh.mm (na przykład 2005-12-28-07.15)
$(NOW_L)
] Znacznik czasu w formacie YYYY-MM-DD-hh.mm.ss (na przykład 2005-12-28-07.15.45)
$(WEEKDAY)
Dzień tygodnia w języku potocznym (na przykład „Wednesday (Środa)”)
$(TDAY_UTC), $(TODAY_UTC), $(NOW_UTC), $(NOW_L_UTC), $(WEEKDAY_UTC)
Są one identyczne z poprzednimi typami, ale są wyrażone względem UTC.
$(DAYCOUNT)
Liczba dni, które upłynęły od dowolnie wybranego dnia zerowego (1 stycznia 2009). Przydatne jako ostatni składnik numeru wersji/kompilacji.

3.2.6 Random values (Wartości losowe)

$(COIN)
Ta zmienna rzuca wirtualną monetą (raz na wywołanie) i zwraca 0 lub 1.
$(RANDOM)
16-bitowa dodatnia liczba losowa (0-65535)

3.2.7 Operating System Commands (Polecenia systemu operacyjnego)

Zmienne są podstawiane za pomocą polecenia systemu operacyjnego:

$(CMD_CP)
Polecenie kopiowania plików.
$(CMD_RM)
Polecenie usuwania plików.
$(CMD_MV)
Polecenie przenoszenia plików.
$(CMD_MKDIR)
Polecenie tworzenia katalogu.
$(CMD_RMDIR)
Polecenie usuwania katalogu.

3.2.8 Conditional Evaluation (Ocena warunkowa)

```
$if(condition){true clause}{false clause}
```

Ocena warunkowa zakończy się stwierdzeniem prawdy, jeśli:

- warunek to niepusta sekwencja znaków inna niż 0 lub fałsz
- warunek to niepusta zmienna, która nie jest rozwiązywana na 0 lub fałsz
- warunek to zmienna, która jest oceniana jako prawda (domyślna z poprzedniego warunku)

Ocena warunkowa zakończy się klauzulą fałszywą, jeżeli:

- warunek jest pusty
- warunek jest równy 0 lub fałsz
- warunek jest zmienną, która jest pusta lub ma wartość 0 lub fałsz

Uwaga:
Należy pamiętać, że ani warianty składni zmiennej %if(...), ani $(if)(...) nie są obsługiwane w przypadku tej konstrukcji.

Przykład:
Na przykład, jeśli używasz kilku platform i chcesz ustawić różne parametry w zależności od systemu operacyjnego. W poniższym kodzie polecenia skryptu [[ ]] są oceniane, a `<command>` zostanie wykonane. Może to być przydatne w kroku po kompilacji.

```
 [[ if (PLATFORM ==  PLATFORM_MSW) { print (_T("cmd /c")); } else { pr
int (_T("sh ")); } ]] <command>
```

3.3 Script expansion (Rozszerzenie skryptu)

Aby uzyskać maksymalną elastyczność, możesz osadzać skrypty za pomocą operatora [[ ]] jako szczególnego przypadku rozszerzenia zmiennej. Osadzone skrypty mają dostęp do wszystkich standardowych funkcjonalności dostępnych dla skryptów i działają w zasadzie jak backticki bash (z wyjątkiem dostępu do przestrzeni nazw CodeBlocks). W związku z tym skrypty nie są ograniczone do generowania wyników tekstowych, ale mogą również manipulować stanem CodeBlocks (projekty, cele itp.).

Uwaga:
Manipulowanie stanem CodeBlocks powinno być implementowane raczej za pomocą skryptu pre-build niż skryptu

Przykład z odwrotnymi znakami:

```
objdump -D `find . -name *.elf` > name.dis
```

Wyrażenie w znakach odwrotnego apostrofu zwraca listę wszystkich plików wykonywalnych `*.elf` w dowolnych podkatalogach. Wynik tego wyrażenia może być użyty bezpośrednio przez `objdump`. Na koniec dane wyjściowe są przesyłane do pliku o nazwie `name.dis`. W ten sposób procesy mogą być automatyzowane w prosty sposób, bez konieczności programowania pętli.

Przykład użycia skryptu:

Tekst skryptu jest zastępowany przez dowolny wynik wygenerowany przez skrypt lub odrzucany w przypadku błędu składni.

Ponieważ ocena warunkowa jest uruchamiana przed rozwinięciem skryptów, ocena warunkowa może być używana do funkcji preprocesora. Wbudowane zmienne (i zmienne użytkownika) są rozwijane po skryptach, więc możliwe jest odwoływanie się do zmiennych w wynikach skryptu.

```
[[ print(GetProjectManager().GetActiveProject().GetTitle()); ]]
```

wstawia tytuł aktywnego projektu do wiersza poleceń.

3.4 Command Macros (Makra poleceń)

$compiler
Dostęp do nazwy pliku wykonywalnego kompilatora.
$linker
Dostęp do nazwy pliku wykonywalnego linkera.
$options
Flagi kompilatora
$link_options
Flagi linkera
$includes
Ścieżki dołączania kompilatora
$c
Ścieżki dołączania linkera
$libs
Biblioteki linkera
$file
Plik źródłowy (pełna nazwa)
$file_dir
Katalog pliku źródłowego bez nazwy pliku i rozszerzenia nazwy pliku.
$file_name
Nazwa pliku źródłowego bez informacji o ścieżce i rozszerzenia nazwy pliku.
$exe_dir
Katalog pliku wykonywalnego bez nazwy pliku i rozszerzenia nazwy pliku.
$exe_name
Nazwa pliku wykonywalnego bez ścieżki i rozszerzenia nazwy pliku.
$exe_ext
Rozszerzenie nazwy pliku wykonywalnego bez ścieżki i nazwy pliku.

$object
Plik obiektu
$exe_output
Plik wyjściowy pliku wykonywalnego
$objects_output_dir
Katalog wyjściowy obiektu

3.5 Compile single file (Kompilacja pojedynczego pliku)

```
$compiler $options $includes -c $file -o $object
```

3.6 Link object files to executable (Połącz pliki obiektów z plikami wykonywalnymi)

```
  $linker $libdirs -
o $exe_output $link_objects $link_resobjects $link_options $libs
```

3.7 Global compiler variables (Globalne zmienne kompilatora)

3.8 Synopsis (Streszczenie)

Praca jako programista nad projektem, który opiera się na bibliotekach stron trzecich, wiąże się z wieloma niepotrzebnymi, powtarzalnymi zadaniami, takimi jak ustawianie zmiennych kompilacji zgodnie z układem lokalnego systemu plików. W przypadku plików projektu należy uważać, aby nie zatwierdzić przypadkowo lokalnie zmodyfikowanej kopii. Jeśli nie zwróci się uwagi, może się to łatwo zdarzyć, na przykład po zmianie flagi kompilacji w celu utworzenia kompilacji wydania.

Koncepcja globalnych zmiennych kompilatora to unikalne nowe rozwiązanie dla CodeBlocks, które rozwiązuje ten problem. Globalne zmienne kompilatora umożliwiają jednokrotne skonfigurowanie projektu, przy czym dowolna liczba programistów korzystających z dowolnej liczby różnych układów systemu plików może kompilować i rozwijać ten projekt. Żadne informacje o lokalnym układzie nie muszą być nigdy zmieniane więcej niż raz.

3.9 Names and Members (Nazwy i członkowie)

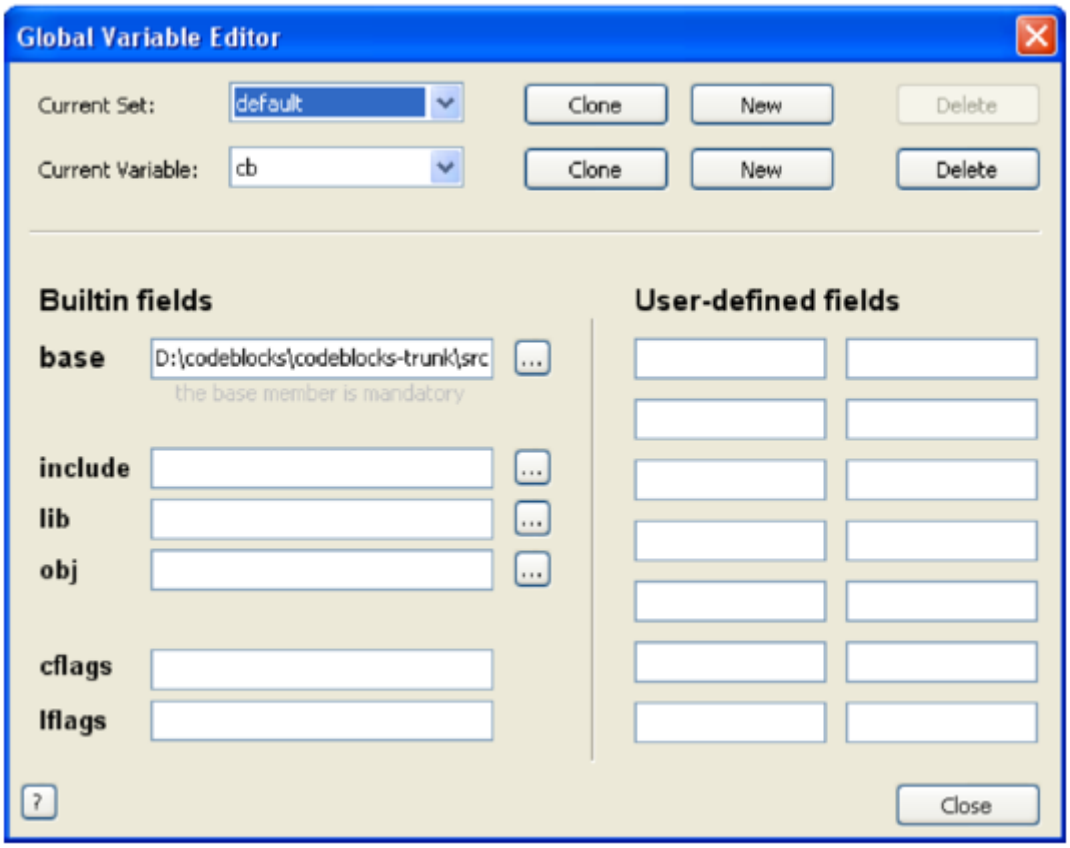
Globalne zmienne kompilatora w CodeBlocks są odróżniane od zmiennych dla każdego projektu przez wiodący znak hash. Globalne zmienne kompilatora są ustrukturyzowane; każda zmienna składa się z nazwy i opcjonalnego członka. Nazwy są swobodnie definiowalne, podczas gdy niektóre z członków są wbudowane w IDE. Chociaż w zasadzie możesz wybrać dowolną nazwę zmiennej, zaleca się wybranie znanego identyfikatora dla typowych pakietów. W ten sposób ilość informacji, które użytkownik musi podać, jest zminimalizowana. Zespół CodeBlocks udostępnia listę zalecanych zmiennych dla znanych pakietów.

Element base rozwiązuje się do tej samej wartości, której używa nazwa zmiennej bez członka (aliasu).

Elementy `include` i `lib` są domyślnie aliasami odpowiednio dla `base/include` i `base/lib`. Jednak użytkownik może je ponownie zdefiniować, jeśli wymagana jest inna konfiguracja.

Ogólnie zaleca się używanie składni `$(#variable.include)` zamiast `$(#variable)/include`, ponieważ zapewnia ona dodatkową elastyczność i jest dokładnie taka sama pod względem funkcjonalności (szczegóły można znaleźć w podsekcji 3.12.1 i na Rysunku 3.1).

Elementy `cflags` i `lflags` są domyślnie puste i można ich używać, aby zapewnić możliwość przekazywania tego samego spójnego zestawu flag kompilatora/linkera do wszystkich kompilacji na jednej maszynie. CodeBlocks umożliwia definiowanie niestandardowych elementów zmiennych oprócz wbudowanych.



Rysunek 3.1: Środowisko zmiennej globalnej

3.10 Constraints (Ograniczenia)

- Nazwy zmiennych kompilatora globalnego i zbioru nie mogą być puste, nie mogą zawierać spacji, muszą zaczynać się od litery i muszą składać się ze znaków alfanumerycznych. Litery cyrylicy lub chińskie nie są znakami alfanumerycznymi. Jeśli CodeBlocks otrzyma nieprawidłowe sekwencje znaków jako nazwy, może je zastąpić bez pytania.

- Każda zmienna wymaga zdefiniowania swojej bazy. Wszystko inne jest opcjonalne, ale baza jest absolutnie obowiązkowa. Jeśli nie zdefiniujesz bazy zmiennej, nie zostanie ona zapisana (bez względu na to, jakie inne pola zdefiniowałeś).

- Nie można zdefiniować niestandardowego elementu, który ma taką samą nazwę jak wbudowany element. Obecnie niestandardowy element nadpisuje wbudowany element, ale ogólnie rzecz biorąc, zachowanie w tym przypadku jest niezdefiniowane.

- Wartości zmiennych i elementów mogą zawierać dowolne sekwencje znaków, podlegające następującym trzem ograniczeniom:
    - Nie można definiować zmiennej za pomocą wartości, która odwołuje się do tej samej zmiennej lub któregokolwiek z jej członków
    - Nie można definiować członka za pomocą wartości, która odwołuje się do tego samego członka
    - Nie można definiować członka lub zmiennej za pomocą wartości, która odwołuje się do tej samej zmiennej lub członka poprzez zależność cykliczną.

CodeBlocks wykryje najbardziej oczywiste przypadki definicji rekurencyjnych (które mogą się zdarzyć przypadkowo), ale nie przeprowadzi dogłębnej analizy każdego możliwego nadużycia. Jeśli wpiszesz crap, to crap otrzymasz; teraz jesteś ostrzegany.

Przykłady

Definiowanie `wx.include` jako `$(#wx)/include` jest zbędne, ale całkowicie legalne. Definiowanie `wx.include` jako `$(#wx.include)` jest nielegalne i zostanie wykryte przez CodeBlocks. Definiowanie `wx.include` jako `$(#cb.lib)`, które z kolei jest zdefiniowane jako `$(#wx.include)`, spowoduje utworzenie nieskończonej pętli.

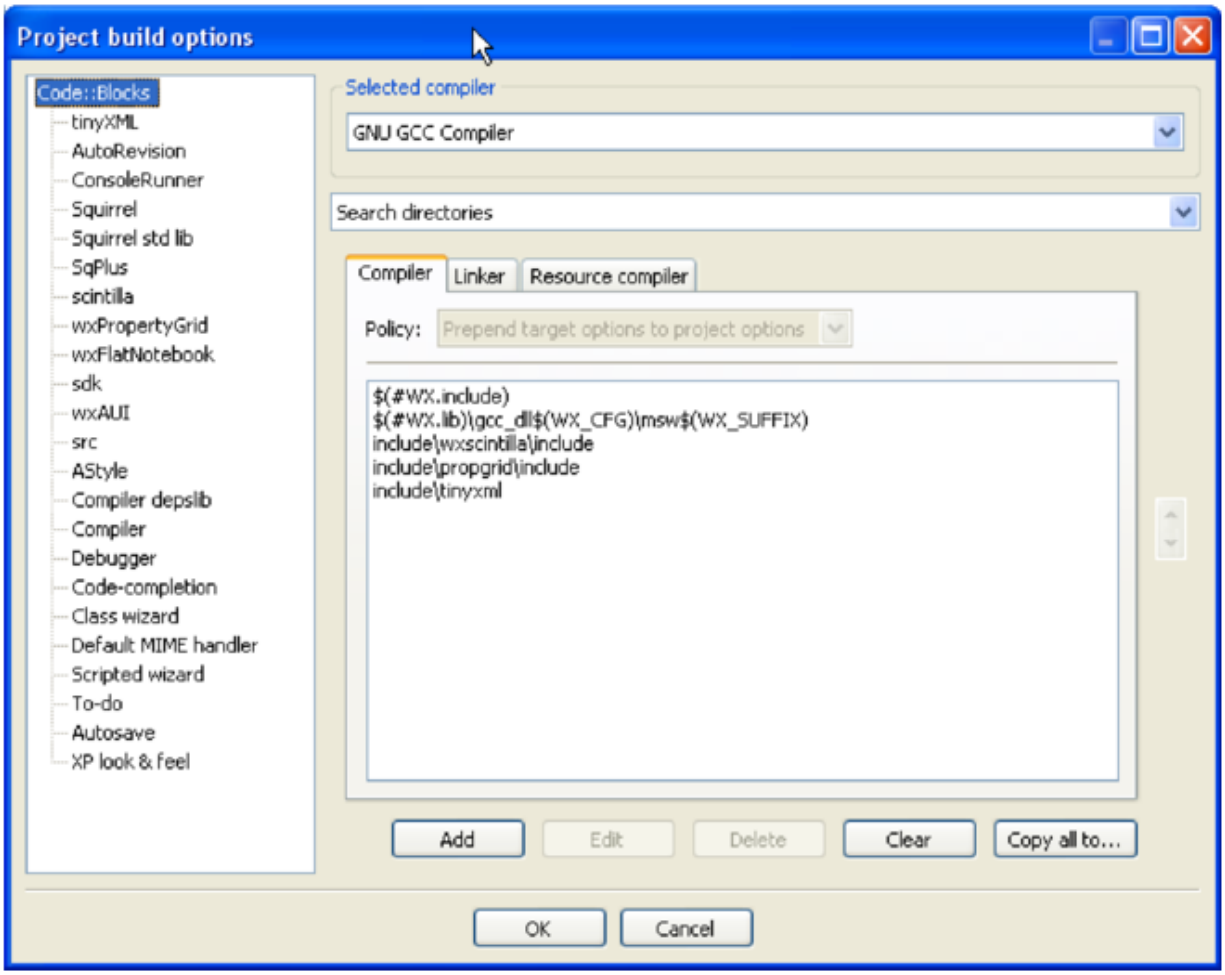
3.11 Using Global Compiler Variables (Korzystanie ze zmiennych globalnych kompilatora)

Wszystko, co musisz zrobić, aby używać globalnych zmiennych kompilatora, to umieścić je w swoim projekcie! Tak, to takie proste.

Gdy IDE wykryje obecność nieznanej zmiennej globalnej, poprosi Cię o podanie jej wartości. Wartość zostanie zapisana w Twoich ustawieniach, więc nigdy nie będziesz musiał wprowadzać informacji dwa razy.

Jeśli będziesz musiał zmodyfikować lub usunąć zmienną w późniejszym czasie, możesz to zrobić z menu ustawień.

Przykład



Rysunek 3.2: Zmienne globalne

Powyższy obraz pokazuje zarówno zmienne per-project, jak i globalne. WX_SUFFIX jest zdefiniowany w projekcie, ale WX jest globalną zmienną użytkownika.

3.12 Variable Sets (Zestawy zmiennych)

Czasami chcesz użyć różnych wersji tej samej biblioteki lub rozwijasz dwie gałęzie tego samego programu. Chociaż możliwe jest dogadanie się z globalną zmienną kompilatora, może to stać się żmudne. W takim celu CodeBlocks obsługuje zestawy zmiennych. Zestaw zmiennych to niezależna kolekcja zmiennych zidentyfikowana przez nazwę (nazwy zestawów mają takie same ograniczenia jak nazwy zmiennych).

Jeśli chcesz przełączyć się na inny zestaw zmiennych, po prostu wybierz inny zestaw z menu. Różne zestawy nie muszą mieć takich samych zmiennych, a identyczne zmienne w różnych zestawach nie muszą mieć takich samych wartości ani nawet takich samych niestandardowych elementów członkowskich.

Inną pozytywną rzeczą w zestawach jest to, że jeśli masz kilkanaście zmiennych i chcesz mieć nowy zestaw, w którym jedna z tych zmiennych wskazuje inną lokalizację, nie musisz ponownie wprowadzać wszystkich danych. Możesz po prostu utworzyć klon bieżącego zestawu, który następnie zduplikuje wszystkie zmienne.

Usunięcie zestawu usuwa również wszystkie zmienne w tym zestawie (ale nie w innym zestawie). Domyślny `default` zestaw jest zawsze obecny i nie można go usunąć.

3.12.1 Custom Members Mini-Tutorial (Mini-samouczek dotyczący niestandardowych członków)

Jak wspomniano powyżej, pisanie `$(#var.include)` i `$(#var)/include` jest domyślnie dokładnie tym samym. Więc dlaczego chciałbyś pisać coś tak nieintuicyjnego jak `$(#var.include)`?

Weźmy na przykład standardową instalację Boost w systemie Windows. Generalnie można by się spodziewać, że fikcyjny pakiet ACME będzie miał swoje pliki include w ACME/include, a swoje biblioteki w ACME/lib. Opcjonalnie może umieścić swoje nagłówki w kolejnym podfolderze o nazwie acme. Tak więc po dodaniu poprawnych ścieżek do opcji kompilatora i linkera, można by się spodziewać **#include** `<acme/acme.h>` i linkowania do `libacme.a` (lub czegoś innego).