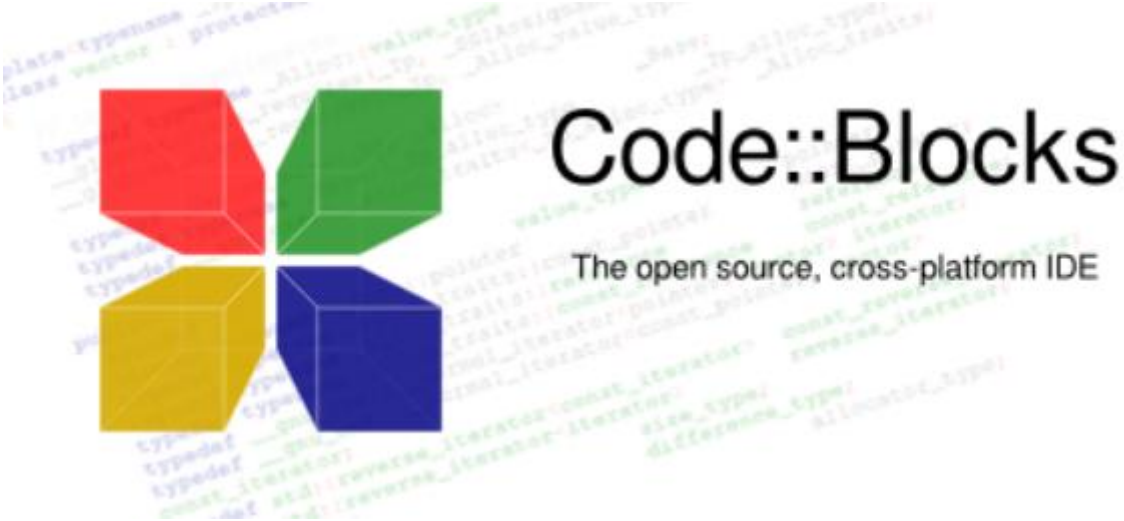


Code::Blocks

Podręcznik

Wersja 2.0.1 beta

Podziękowania dla zespołu Code::Blocks:

Anders F. Bj¨orklund (afb), Biplab Kumar Modak (biplab), Bartomiej Wiecki (byo), Paul A. Jimenez (ceniza), Koa Chong Gee (cyberkoa), Daniel Orb (daniel2000), Lieven de Cock (killerbot), Yiannis Mandravellos (mandrav), Mispunt (mispunt), Martin Halle (mortenmacfly), Jens Lody (jens), Jerome Antoine (dje), Damien Moore (dmoore), Pecan Heber (pecan), Ricardo Garcia (rickg22), Thomas Denk (thomasdenk), tiwag (tiwag), staht a01 (staht a01), oBFusCATed (oBFusCATed), BlueHazzard (BlueHazzard)

I wielu innych współtwórców...

Oryginalna instrukcja w języku angielskim i niemieckim (V1.x) autorstwa Mario Cupelli (mariocup) Tłumaczenie w języku francuskim (v1.x), poprawki i uzupełnienia do wersji 2 autorstwa Gerarda Duranda (gd on).

aktualizacja w październiku 2020 r.

Zawartość

1 Code::Blocks zarządzanie projektami 6
1.1 Widok projektu 7
1.2 Uwagi do projektów 8
1.3 Szablony projektów 8
1.4 Tworzenie projektów z kompilacji celów 8
1.5 Wirtualne cele 9
1.6 Kroki przed i po kompilacji 9
1.7 Dodawanie skryptów w Build Targets 10
1.8 Obszar roboczy i zależności projektu 10
1.9 Łączenie z plikami asemblera 11
1.10 Edytor i narzędzia 11
1.10.1 Kod domyślny 11
1.10.2 Skrót 12
1.10.3 Osobowości 12
1.10.4 Pliki konfiguracyjne 13
1.10.5 Nawigacja i wyszukiwanie 13
1.10.6 Widok symboli 15
1.10.7 Dołączanie zewnętrznych plików pomocy 17
1.10.8 Dołączanie narzędzi zewnętrznych 19
1.11 Wskazówki dotyczące pracy z Code::Blocks 19
1.11.1 Śledzenie modyfikacji 20
1.11.2 Wymiana danych z innymi aplikacjami 20
1.11.3 Konfiguracja zmiennych środowiskowych 21
1.11.4 Przełączanie między perspektywami 22
1.11.5 Przełączanie się między projektami 22
1.11.6 Rozszerzone ustawienia dla kompilatorów 23
1.11.7 Powiększanie w edytorze 23
1.11.8 Tryb owijania (Wrap Mode). 24
1.11.9 Wybór trybów w edytorze 24
1.11.10 Składanie kodu 24
1.11.11 Autouzupełnianie 25
1.11.12 Znajdź uszkodzone pliki 25
1.11.13 Dołączanie bibliotek 26
1.11.14 Kolejność łączenia obiektów 26
1.11.15 Autozapis 26
1.11.16 Ustawienia rozszerzeń plików 26
1.12 Code::Blocks w wierszu poleceń 27
1.13 Skróty 28
1.13.1 Wprowadzenie 28
1.13.2 Funkcje 28
1.13.3 Użytkowanie 28
1.13.4 Edytor 29
1.13.5 Pliki 31
1.13.6 Widok 31

1.13.7 Wyszukiwanie 31
1.13.8 Kompilacja 32
1.13.9 Debugowanie 32

2. Wtyczki 33
2.1 Ogólne 33
2.2 Styl 34
2.3 Automatyczne wersjonowanie (AutoVersioning). 35
2.3.1 Wprowadzenie 35
2.3.2 Funkcje 35
2.3.3 Użytkowanie 35
2.3.4 Okna dialogowe zakładek notatnika 36
2.3.5 Uwzględnianie w kodzie 40
2.3.6 Generator dziennika zmian 41
2.4 Śledzenie przeglądania (Browse Tracker). 42
2.5 Fragmenty kodu (CodeSnippets). 43
2.6 Doxyblocks 45
2.7 Wtyczka Modyfikacje Edytora (Editor Tweaks). 47
2.8 Menedżer plików (FileManager) i wtyczka PowerShell 48
2.9 Edytor szesnastkowy (HexEditor). 52
2.10 Wyszukiwanie przyrostowe (Incremental Search) 53
2.11 Wtyczka NassiShneiderman 55
2.11.1 Tworzenie diagramu 55
2.11.2 Edycja struktogramów 56
2.12 LibFinder 57
2.12.1 Wyszukiwanie bibliotek 57
2.12.2 Uwzględnianie bibliotek w projektach 59
2.12.3 Korzystanie z LibFindera i projektów wygenerowanych z kreatorów 59
2.13 Wtyczka Sprawdzanie pisowni (SpellChecker). 60
2.13.1 Wprowadzenie 60
2.13.2 Konfiguracja 60
2.13.3 Słowniki 61
2.13.4 Pliki tezaurusa 62
2.13.5 Bitmapy (flagi) 62
2.13.6 Style do sprawdzenia 62
2.14 Eksporter kodu źródłowego (Source Code Exporter) 63
2.15 Obsługa SVN 63
2.16 Lista rzeczy do zrobienia (ToDo List). 64
2.17 Narzędzia+ (Tools+). 65
2.17.1 Przykładowe narzędzia 69
2.18 Wyszukiwanie wątków (Thread Search) 72
2.18.1 Funkcje 72
2.18.2 Użytkowanie 72
2.18.3 Konfiguracja 73
2.18.4 Opcje (Options). 74
2.18.5 Opcje wyszukiwania wątków (Thread search options). 74
2.18.6 Układ (Layout) 75

2.18.7 Panel Zarządzanie (Panel Management) 75
2.18.8 Typ rejestratora (Logger Type). 75
2.18.9 Okno trybu rozdzielacza (Splitter Window Mode) 76
2.18.10 Sortuj wyniki wyszukiwania (Sort results by). 76
2.19 Statystyki kodu (Code statistics). 76
2.20 Profiler kodu (Code profiler). 76
2.21 Wtyczka do importowania projektów (ProjectsImporter). 76
2.22 Wyszukiwanie dostępnego kodu źródłowego 76
2.23 Wtyczka tabeli symboli (Symbol Table Plugin) 77
3 Rozszerzenie zmiennej 78
3.1 Składnia 78
3.2 Lista dostępnych funkcji wbudowanych 79
3.2.1 Code::Blocks obszar roboczy 79
3.2.2 Pliki i katalogi 79
3.2.3 Budowanie celów 80
3.2.4 Język i kodowanie 81
3.2.5 Godzina i data 81
3.2.6 Zależność od platformy 81
3.2.7 Rozszerzenia wiersza poleceń 81
3.2.8 Wartości losowe 81
3.2.9 Ścieżka standardowa 82
3.2.10 Wbudowane funkcje do konwersji ścieżek 82
3.2.11 Ocena warunkowa 82
3.3 Rozszerzanie skryptów 83
3.4 Makra poleceń 84
3.5 Kompilacja pojedynczego pliku 84
3.6 Połącz pliki obiektowe z plikiem wykonywalnym 84
3.7 Globalne zmienne kompilatora 84
3.7.1 Streszczenie 85
3.7.2 Nazwiska i członkowie 85
3.7.3 Ograniczenia 86
3.7.4 Używanie globalnych zmiennych kompilatora 87
3.7.5 Zestawy zmiennych 87
3.7.6 Mini-samouczek dotyczący członków niestandardowych 88

4 Praca z Code::Blocks 89
4.1 Proces budowania Code::Blocks 89
4.1.1 Kolejność budowania 89
4.1.2 Workspace (Obszar roboczy) 89
4.1.3 Projects (Projekty) 89
4.1.4 Budowanie celów 91
4.1.5 Faza przetwarzania wstępnego 91
4.1.6 Rzeczywiste wykonanie poleceń 91
4.1.7 Kroki przed kompilacją i po kompilacji 91
4.2 Tworzenie nowego projektu 92
4.2.1 Kreator projektu 93

4.2.2 Zmiana składu pliku 95
4.2.3 Modyfikowanie opcji kompilacji 98
4.3 Debugowanie za pomocą Code::Blocks 101
4.3.1 Zbuduj wersję debugowania swojego projektu 102
4.3.2 Dodaj obserwatorów 102
4.3.3 Dwukrotne kliknięcie w oknie Stos wywołań (Call stack). 104
4.3.4 Ustaw punkty przerwania 105
4.3.5 Uwagi 105
4.4 Skrypty debugera 107
4.4.1 Podstawowa zasada skryptu debuggera. 108
4.4.2 Funkcje skryptów 109
4.5 Code::Blocks i pliki Makefile 111
4.5.1 Artykuł Wiki 111
4.5.2 Uzupełnienia 115
4.6 Narzędzie Cbp2make 115
4.6.1 Informacje 116
4.6.2 Użycie 116
4.6.3 Konfiguracja 117
4.6.4 Składnia wiersza poleceń 117
4.7 Code::Blocks. Internacjonalizacja interfejsu 120

5 Instalacja i konfiguracja CodeBlocks z MinGW 123
5.1 Instalacja najnowszej oficjalnej wersji Code::Blocks w systemie Windows 123
5.2 Konfiguracja MinGW 124
5.2.1 Przegląd 124
5.2.2 Łańcuchy narzędzi kompilatora MinGW 124
5.2.3 Konfiguracja Code::Blocks 125
5.3 Nocna wersja Code::Blocks w systemie Windows 128

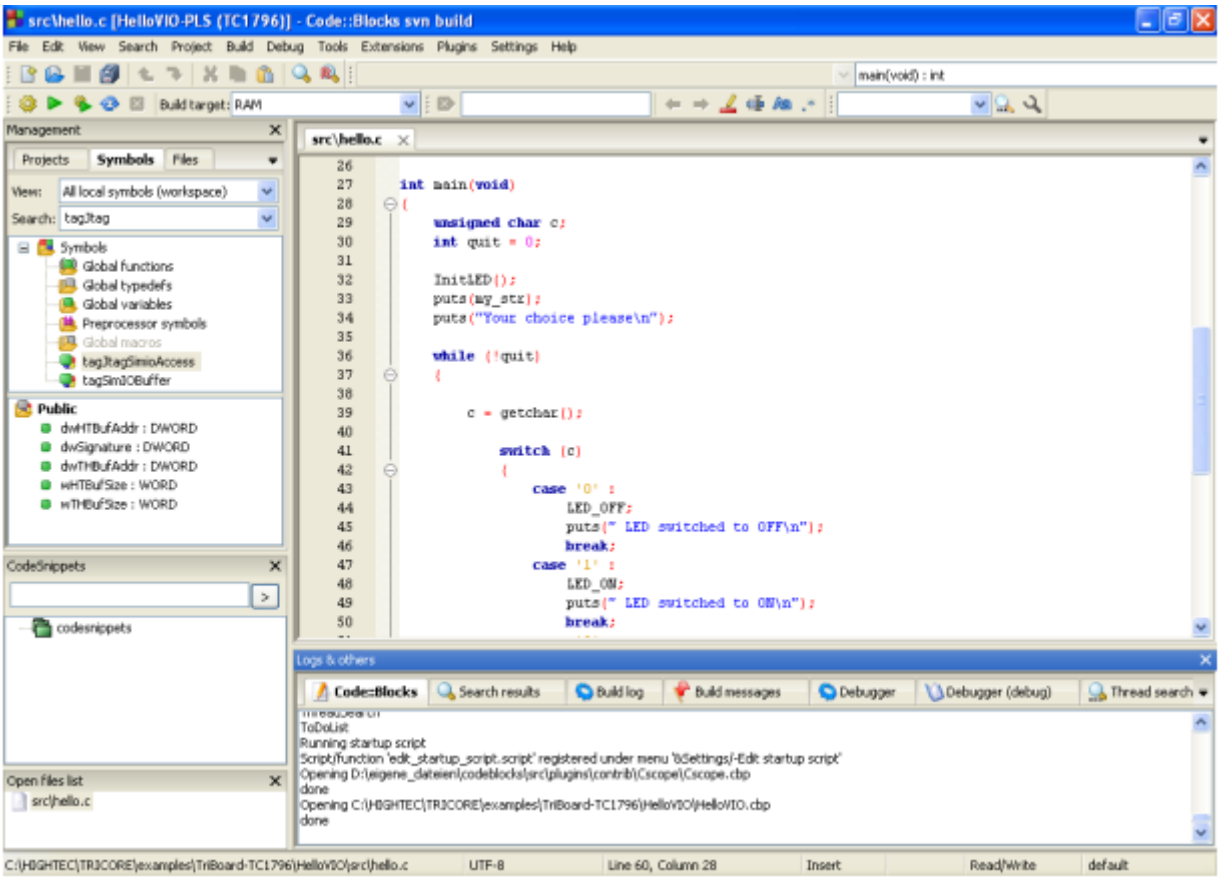
6 Budowanie CodeBlocks ze źródeł 131
6.1 Wprowadzenie 131
6.2 Windows lub Linux 131
6.2.1 Wstępny system kompilacji 133
6.2.2 System kontroli wersji 133
6.2.3 wxWidgets 134
6.2.4 Zip 136
6.2.5 Obszar roboczy 136
6.2.6 Budowanie Codeblocks 139
6.2.7 Generuj tylko wtyczki. 139

Katalog URL 140

1. Code::Blocks zarządzanie projektami

Instrukcje w kilku akapitach (na przykład rozdział 3 na stronie 78 lub rozdział 2 na strona 33) to oficjalna dokumentacja strony Code::Blocks Wiki (ostatecznie przejrzana) i poprawione) i dostępne tylko w języku angielskim. Niniejsza dokumentacja jest rozszerzeniem oryginalnej wersji 1.1, skompilowanej i/lub napisanej przez Mario Cupelli.

Poniższa ilustracja przedstawia projekt interfejsu użytkownika Code::Blocks.



Rysunek 1.1: IDE Code::Blocks

Management (Zarządzanie) To okno zawiera interfejs „Projects” (Projekty), który będzie dalej nazywany widokiem projektu. Ten widok pokazuje wszystkie projekty otwarte w Code::Blocks o określonym czasie. Zakładka „Symbols” (Symbole) w oknie Zarządzanie pokazuje symbole, zmienne itp.

Editor (Edytor) Na powyższej ilustracji źródło o nazwie hello.c jest otwierane z podświetleniem 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”. Tutaj możesz zarządzać modułami tekstowymi, linkami do plików i linkami do adresów URL.

Logs & others (Dzienniki i inne) . To okno służy do wyprowadzania wyników wyszukiwania, komunikatów dziennika kompilatora itp.

Pasek stanu zawiera przegląd następujących ustawień:

- Ścieżkę bezwzględną otwartego pliku w edytorze.
- Edytor używa domyślnego kodowania znaków systemu operacyjnego hosta. To ustawienie będzie wyświetlane z wartością default.
- Numer wiersza i kolumny aktualnej pozycji kursora w edytorze.
- Skonfigurowany tryb klawiatury do wstawiania tekstu („Insert or Overwrite” (Wstaw lub Zastąp)).
- Aktualny stan pliku. Zmodyfikowany plik zostanie oznaczony jako Zmodyfikowany, w przeciwnym razie to wpis jest pusty.
- Uprawnienie do pliku. Plik z ustawieniami tylko do odczytu będzie wyświetlał Read only (Tylko do odczytu) na pasku stanu. W oknie „Open files list” (Lista otwartych plików) pliki te zostaną wyróżnione kłódką jako nakładką ikony.

Uwaga:
W aktywnym edytorze użytkownik może wybrać właściwości menu kontekstowego.
W pojawiającym się oknie dialogowym w zakładce „General” (Ogólne) można wybrać opcję „File is readonly” (Plik jest tylko do odczytu). Ta opcja spowoduje dostęp tylko do odczytu odpowiedniego pliku w Code::Blocks, ale oryginalny odczyt i atrybuty zapisu pliku w systemie plików nie są modyfikowane.

- Jeśli uruchomisz Code::Blocks z opcją wiersza poleceń `--personality=<profile>` wtedy pasek stanu pokaże aktualnie używany profil, w przeciwnym razie będzie pokazane default . Ustawienia Code::Blocks są przechowywane w pliku odpowiedniej konfiguracji `<personality>.conf`.

Code::Blocks oferuje bardzo elastyczne i kompleksowe zarządzanie projektami. Poniższy tekst dotyczy tylko niektórych funkcji zarządzania projektem.

1.1 Widok projektu (Project View)

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

Code::Blocks zarządza plikami projektu w kategoriach zgodnie z ich rozszerzeniami. Oto wstępnie ustawione kategorie:

Źródła (Sources) obejmują pliki źródłowe z rozszerzeniami `*.c;*.cpp;`.

Źródła ASM (ASM Sources) zawierają pliki źródłowe z rozszerzeniami `*.s;*.S;*.ss;*.asm`.

Nagłówki (Headers) zawierają m.in. pliki z rozszerzeniem `*.h;`.

Zasoby (Resources) zawierają pliki opisów układów dla okien wxWidgets z rozszerzeniami `*.res;*.xrc;`. Te typy plików są wyświetlane w zakładce „Resources” (Zasoby) w oknie „Management” (Administrowanie, zarządzanie).

Ustawienia typów i kategorii plików można dostosowywać za pomocą menu kontekstowego „Project tree” (Drzewo projektu) → „Edit file types & categories” (Edytuj typy i kategorie plików). Tutaj możesz również zdefiniować niestandardowe kategorie dla własnych rozszerzeń plików. Na przykład, jeśli chcesz wyświetlić listę skryptów linkera z rozszerzeniem `*.ld` w kategorii o nazwie `Linkerscript`, musisz tylko utworzyć nową kategorię.

Uwaga:
Jeśli wyłączysz „Project tree” (Drzewo projektu) → „Categorize by file types” (Kategoryzuj według typów plików) w menu kontekstowym, wyświetlanie kategorii zostanie wyłączone, a pliki zostaną wyświetlone tak, jak są przechowywane w systemie plików.

1.2 Uwagi dotyczące projektów

W Code::Blocks można przechowywać tak zwane notatki dotyczące 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 w zakładce Notatki (Notes) we Właściwościach (Properties) projektu.

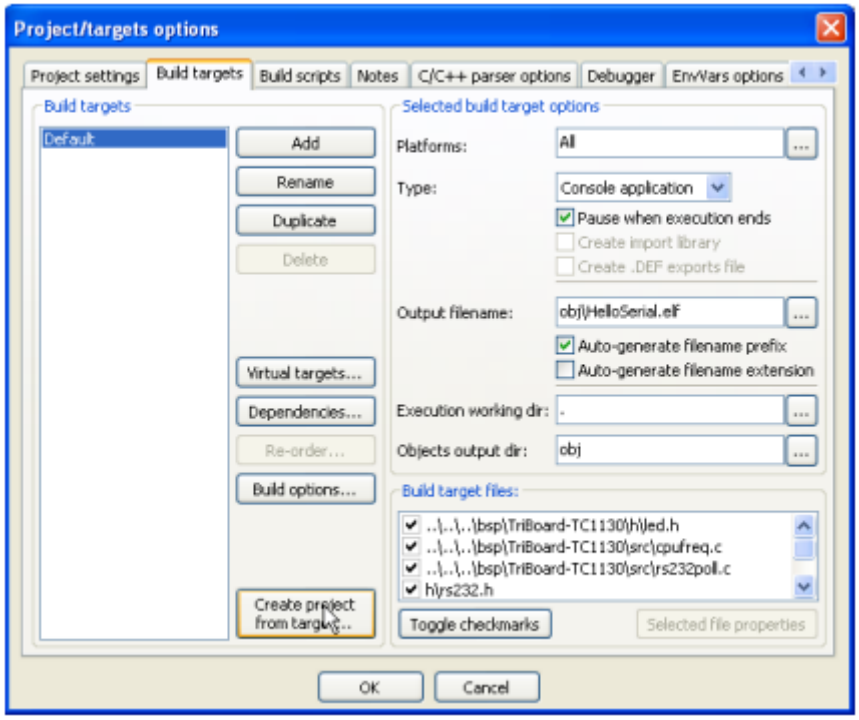
1.3 Szablony (Templates) projektów

Code::Blocks jest dostarczany z różnymi szablonami projektów, które są wyświetlane podczas tworzenia nowego projektu. Jednak możliwe jest również przechowywanie niestandardowych szablonów do zbierania 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` w Win 7 (lub równoważnej ścieżce w profilu użytkownika, dostosowanej do każdego systemu operacyjnego). Jeśli szablony mają być otwarte dla wszystkich użytkowników, należy je skopiować do odpowiedniego katalogu instalacji Code::Blocks. Szablony te zostaną wyświetlone przy następnym uruchomieniu Code::Blocks pod „New” (Nowy) → „Project” (Projekt) → „User Templates” (Szablony użytkownika).

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

1.4 Tworzenie projektów z kompilacji celów

W projektach konieczne jest posiadanie różnych wariantów projektu. Warianty są zwane Build Targets (Kompilacja Celów). Różnią się one opcjami kompilatora, informacjami debugowania i/lub wyborem plików. Cel kompilacji można również zlecić na zewnątrz osobnemu projektowi. Aby to zrobić, kliknij „Project” (Projekt) → „Properties” (Właściwości), wybierz wariant z zakładki „Build Targets” (Buduj cele) i kliknij przycisk „Create project from target” (Utwórz projekt z celu) (patrz Rysunek 1.2 na stronie 9).



Rysunek 1.2: Budowanie celów (Build Targets)

1.5 Wirtualne cele

Projekty mogą być dalej ustrukturyzowane w Code::Blocks za pomocą tak zwanych wirtualnych celów (Virtual Targets). Często używana struktura projektu składa się z dwóch celów kompilacji, jednego celu „Debug" (Debugowania), który zawiera informacje debugowania i jeden cel „Release" (Wydanie) bez tych informacji. Dodając Wirtualne Cele poprzez „Project" (Projekt) → „Properties" (Właściwości) → „Build Targets" (Cele Budowania) można łączyć poszczególne Cele Budowy. Na przykład wirtualny cel „All" (Wszystko) może jednocześnie tworzyć debugowanie i zwalnianie (uwolnienie) celów. Wirtualne cele są wyświetlane na pasku symboli kompilatora w obszarze Build Targets.

1.6 Kroki przed i po kompilacji

Code::Blocks umożliwia wykonanie dodatkowych operacji przed lub po kompilacji projektu. Te operacje są nazywane krokami prekompilowanymi (Prebuilt) lub postkompilowanymi (Postbuilt). Typowe kroki postkompilacyjne to:

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

Przykład

Tworzenie demontażu z obiektu w systemie Windows. Piping do pliku wymaga wywołania `cmd` z opcją `/c`.

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

Archiwizacja projektu może być kolejnym przykładem kroku pokompilacyjnego. W tym celu utwórz „Archive” (Archiwum) docelowej kompilacji i dołącz następującą instrukcję w kroku Po zbudowaniu (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 do pliku zip. W ten sposób wyodrębnione zostaną wbudowane zmienne $(PROJECT_NAME) i $(TODAY), nazwa projektu i bieżąca data (patrz sekcja 3.2 na stronie 79). Po wykonaniu celu „Archive” (Archiwum) spakowany plik zostanie zapisany w katalogu projektu.

W katalogu `share/codeblocks/scripts` znajdziesz kilka przykładów skryptów. Możesz dodać skrypt poprzez menu „Settings” (Ustawienia) → „Scripting” (Skrypty) i zarejestrować się w menu. Jeśli wykonasz m.in. skrypt `make dist` z menu to wszystkie pliki należące do projektu zostaną skompresowane w archiwum `<projekt>.tar.gz`

1.7 Dodawanie skryptów w Build Targets

Code::Blocks oferuje możliwość korzystania z akcji menu w skryptach. Skrypt reprezentuje kolejny stopień swobody w sterowaniu generowaniem twojego projektu.

Uwaga:
Skrypt można również dołączyć do celu kompilacji.

1.8 Obszar roboczy i zależności projektu

W Code::Blocks można otworzyć wiele projektów. Zapisując otwarte projekty poprzez „File” (Plik) → „Save workspace” (Zapisz obszar roboczy), możesz zebrać je w jednym obszarze roboczym pod `<nazwa>.workspace`. Jeśli otworzysz `<nazwa>.workspace` podczas następnego uruchomienia Code::Blocks, wszystkie projekty pojawią się ponownie.

Złożone systemy oprogramowania składają się z komponentów, które są zarządzane w różnych projektach Code::Blocks. Ponadto, wraz z generowaniem 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, to biblioteka musi zostać przebudowana. Aby zachować spójność między projektem B, który korzysta z funkcji, a projektem A, który implementuje funkcje, projekt B musi być zależny od projektu A. Niezbędne informacje o zależnościach projektów są przechowywane w odpowiednim obszarze roboczym, dzięki czemu można utworzyć każdy projekt osobno. Wykorzystanie zależności pozwala również kontrolować kolejność generowania projektów. Zależności dla projektów można ustawić wybierając z menu „Project” (Projekt) → „Properties” (Właściwości), a następnie klikając przycisk „Project’s dependencies” (Zależności projektu).

1.9 Łączenie z plikami asemblera

W oknie „Management” (Zarządzanie) w „Project View” (Widok Projektu), pliki asemblera są wyświetlane w kategorii „ASM Sources” (Źródła ASM). Użytkownik może zmienić listę plików w kategoriach (patrz rozdział 1.1 na stronie 7). Kliknięcie prawym przyciskiem myszy jednego z wymienionych plików asemblera otworzy menu kontekstowe. Wybierz „Properties” (Właściwości), aby otworzyć nowe okno. Teraz wybierz zakładkę „Build” (Buduj) i aktywuj dwa pola „Compile file” (Skompiluj plik) i „Link file” (Połącz plik). Następnie wybierz zakładkę „Advanced” (Zaawansowane) i wykonaj następujące czynności:

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

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

Zmienne Code::Blocks są oznaczone znakiem $ (patrz rozdział 3.4 na stronie 84). Są one ustawiane automatycznie, więc wystarczy zastąpić opcję Assemblera <asopt> własnymi ustawieniami.

1.10 Edytor i narzędzia

W tej sekcji opisano narzędzia w oknie edytora.

1.10.1 Kod domyślny

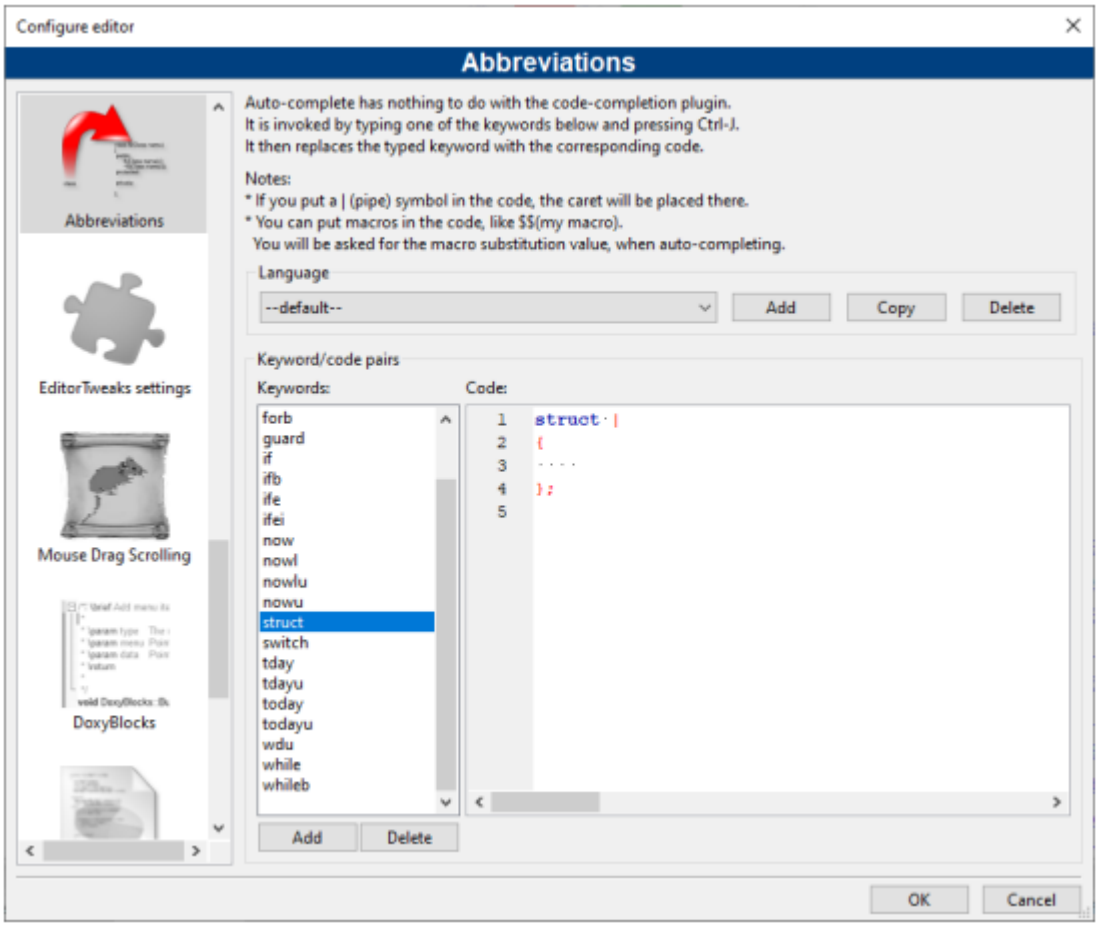
Zasady kodowania firmy wymagają, aby pliki źródłowe miały standardowy projekt. Code::Blocks umożliwia automatyczne dołączenie predefiniowanej zawartości na początku pliku podczas tworzenia nowych źródeł i nagłówków C/C++. Ta wstępnie zdefiniowana zawartość nazywana jest kodem domyślnym. To ustawienie można wybrać w „Settings” (Ustawienia) → „Editor” (Edytor) Domyślny kod. Jeśli tworzysz nowy plik, to makrorozwijanie zmiennych, np. zdefiniowana w menu „Settings” (Ustawienia) → „Global variables” (Zmienne globalne). Nowy plik można utworzyć za pomocą menu „File” (Plik) → „New” (Nowy) → „File” (Plik).

Przykład

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

1.10.2 Skrót

Dużo pisania można zaoszczędzić w Code::Blocks, definiując skrót. W tym celu wybierz „Settings” (Ustawienia) → „Editor” (Edytor) i zdefiniuj skróty pod nazwą <nazwa>, które następnie można wywołać skrótem klawiaturowym Ctrl-J (patrz Rysunek 1.3 na stronie 12).



Rysunek 1.3: Definiowanie skrótów

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

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

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

1.10.3 Osobowości

Ustawienia Code::Blocks 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 osobowość „default” jest ustawiona tak, że konfiguracja jest przechowywana w pliku default.conf. Jeśli Code::Blocks zostanie wywołane z wiersza poleceń z parametrem `--personality=myuser`, ustawienia zostaną zapisane w pliku myuser.conf. Jeśli profil jeszcze nie istnieje, zostanie utworzony automatycznie. Ta procedura umożliwia tworzenie odpowiednich profili dla różnych etapów pracy. Jeśli uruchomisz Code::Blocks z wiersza poleceń z dodatkowym parametrem `--personality=ask`, zostanie wyświetlone okno wyboru dla wszystkich dostępnych profili.

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

1.10.4 Pliki konfiguracyjne

Ustawienia Code::Blocks są przechowywane w profilu `default.conf` w katalogu `codeblocks` danych aplikacji. Podczas korzystania z osobowości (patrz podrozdział 1.10.3 na stronie 12), szczegóły konfiguracji zostaną zapisane w pliku `<personality>.conf`.

Do zarządzania i przechowywania tych ustawień służy narzędzie `cb_share_conf`, które można znaleźć w katalogu instalacyjnym Code::Blocks.

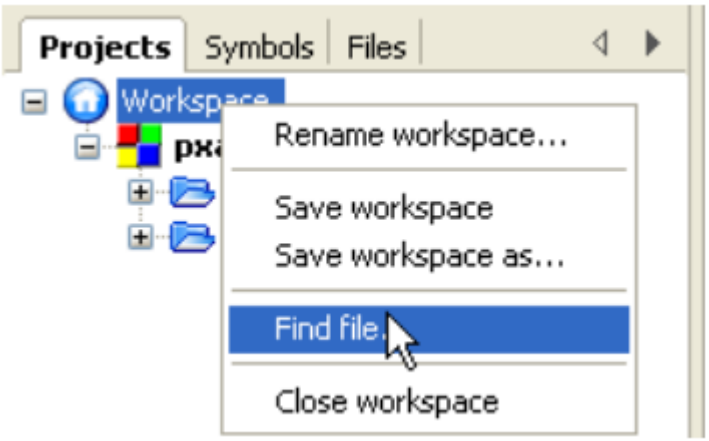
Jeśli chcesz zdefiniować standardowe ustawienia dla kilku użytkowników komputera, plik konfiguracyjny `default.conf` musi być przechowywany w katalogu `\Documents and Settings\Default User\Application Data\codeblocks`. Podczas pierwszego uruchomienia Code::Blocks skopiuje ustawienia wstępne z „Default User” (Domyślnego użytkownika) do danych aplikacji bieżących użytkowników.

Aby utworzyć przenośną wersję Code::Blocks na pamięci USB, wykonaj następujące czynności. Skopiuj instalację Code::Blocks na pamięć USB i zapisz plik konfiguracyjny `default.conf` w tym katalogu. Konfiguracja będzie używana jako ustawienie globalne. Upewnij się, że plik jest zapisywalny, w przeciwnym razie zmiany konfiguracji nie zostaną zapisane.

1.10.5 Nawigacja i wyszukiwanie

W Code::Blocks 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 przeskoczyć do poprzedniej zakładki, a za pomocą Alt-PgDn możesz przeskoczyć do następnej zakładki.

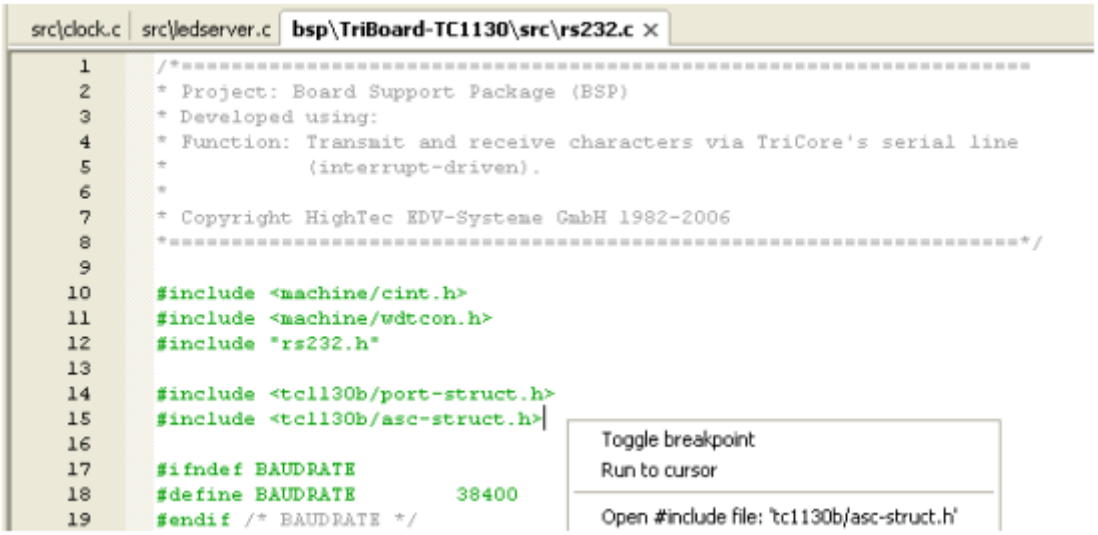
Jeśli wybierzesz obszar roboczy lub projekt w obszarze roboczym w widoku projektu, będziesz mógł wyszukać plik w projekcie. Po prostu wybierz „Find file” (Znajdź plik) z menu kontekstowego, a następnie wpisz nazwę pliku, a plik zostanie wybrany. Jeśli naciśniesz return, plik ten zostanie otwarty w edytorze (patrz Rysunek 1.4 na stronie 13).



Rysunek 1.4: Wyszukiwanie plików

W Code::Blocks możesz łatwo nawigować między plikami nagłówkowymi/źródłowymi, takimi jak:

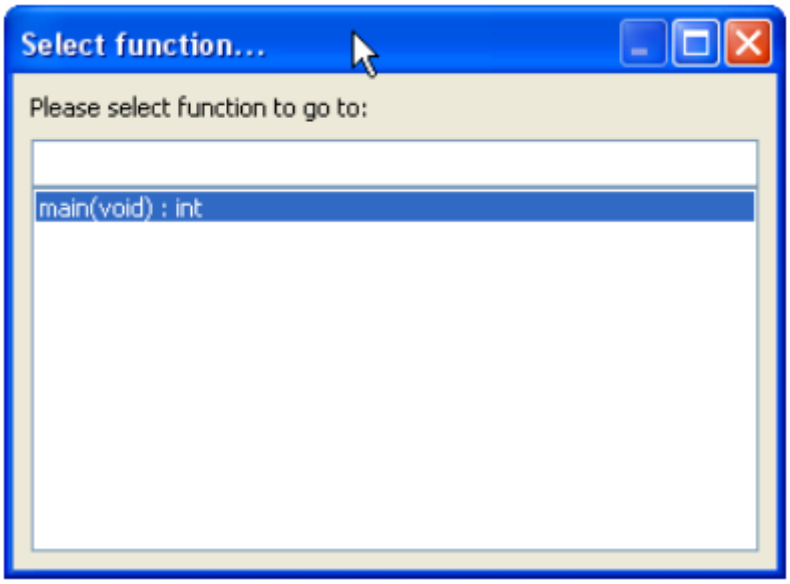
1. Ustaw kursor w miejscu, w którym znajduje się plik nagłówkowy i otwórz ten plik za pomocą menu kontekstowego „open include file” (otwórz plik dołączany) (patrz Rysunek 1.5 na stronie 14)
2. Przełączaj się między nagłówkiem a źródłem za pomocą menu kontekstowego „Swap header/source” (Zamień nagłówek/źródło)
3. Wybierz np. zdefiniować w edytorze i wybrać „Find declaration” (Znajdź deklarację) z kontekstu menu, aby otworzyć plik z jego deklaracją.



Rysunek 1.5: Otwieranie pliku nagłówkowego

Code::Blocks oferuje kilka sposobów wyszukiwania w pliku lub katalogu. Okno dialogowe wyszukiwania otwiera się przez „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. Okno dialogowe, które otworzy się po użyciu tego skrótu, pozwala wybrać pliki/funkcje, a następnie przeskakuje do realizacji wybranej funkcji (patrz Rysunek 1.6 na stronie 14) lub otwiera wybrany plik w edytorze. Możesz używać symboli wieloznacznych, takich jak * lub ? itp. dla wyszukiwania przyrostowego w oknie dialogowym.



Rysunek 1.6: Wyszukiwanie funkcji

Uwaga:
Za pomocą skrótu Ctrl-PgUp możesz przeskoczyć 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 wciśnięty jest klawisz Ctrl, plik można wybrać na różne sposoby:

1. Jeśli wybierzesz wpis lewym przyciskiem myszy, wybrany plik zostanie otwarty.
2. Jeśli naciśniesz klawisz Tab, będziesz przełączać się między wymienionymi wpisami. Zwolnienie klawisza Crtl otworzy wybrany plik.
3. Jeśli przesuniesz wskaźnik myszy na wymienione wpisy, podświetlony zostanie aktualny wybór. Zwolnienie klawisza Crtl otworzy wybrany plik.
4. Jeśli wskaźnik myszy znajduje się poza podświetlonym zaznaczeniem, możesz przełączać się między wpisami za pomocą kółka myszy. Zwolnienie klawisza Crtl otworzy wybrany plik.

Powszechną procedurą podczas tworzenia oprogramowania jest walka z zestawem funkcji zaimplementowanych w różnych plikach. Wtyczka Browse Tracker pomoże Ci rozwiązać ten problem, pokazując kolejność, w jakiej pliki zostały wybrane. Możesz wtedy wygodnie poruszać się po wywołaniach funkcji (patrz sekcja 2.4 na stronie 42).

Wyświetlanie numerów linii w Code::Blocks można aktywować poprzez „Settings” (Ustawienia) → „General Settings” (Ustawienia ogólne) w polu „Show line numbers” (Pokaż numery linii). Skrót Ctrl-G lub polecenie menu „Search” (Szukaj) → „Goto line” (Idź do wiersza) pomogą Ci przejść do żądanego wiersza.

Uwaga:
Jeśli przytrzymasz klawisz Ctrl, a następnie zaznaczysz tekst w edytorze Code::Blocks, możesz wykonać m.in. wyszukiwanie Google za pomocą menu kontekstowego.

1.10.6 Widok symboli

Okno Code::Blocks Management oferuje widok drzewa symboli źródeł C/C++ do nawigacji za pomocą funkcji lub zmiennych. Jako zakres tego widoku możesz ustawić bieżący plik lub projekt, lub cały obszar roboczy.

Uwaga:
Wprowadzenie wyszukiwanego terminu lub nazw symboli w masce wejściowej „Search” (Wyszukaj) przeglądarki symboli (Symbol Browser) powoduje wyświetlenie filtrowanego widoku symboli, jeśli wystąpiły jakiekolwiek trafienia.

Istnieją następujące kategorie symboli:

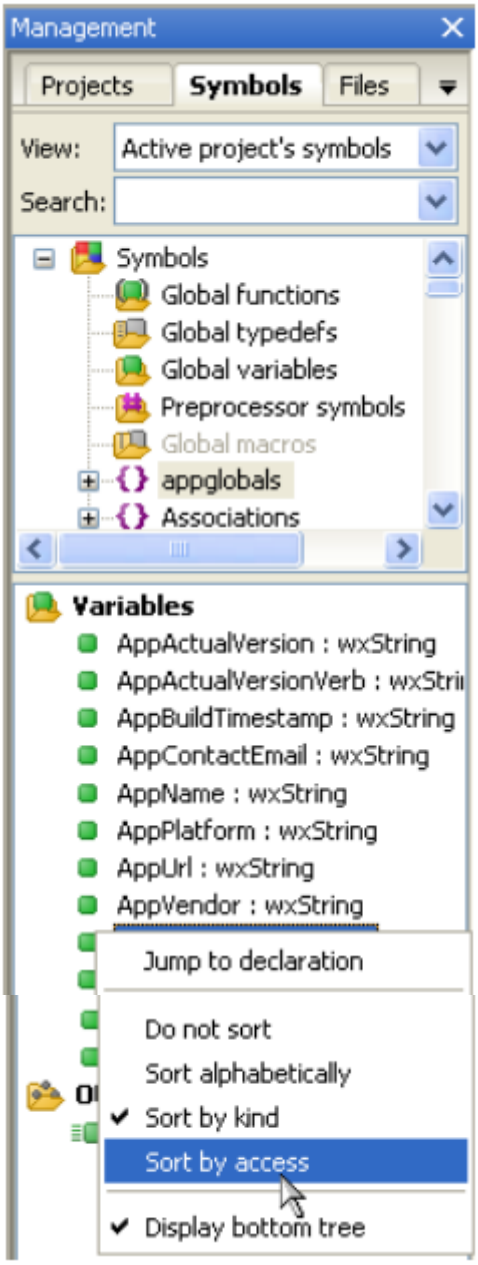
Global functions (Funkcje globalne) Wyświetla implementację funkcji globalnych.

Global typedefs (Globalne typedefs) Wyświetla użycie definicji **typedef.**

Global variables (Zmienne globalne) Wyświetla symbole zmiennych globalnych.

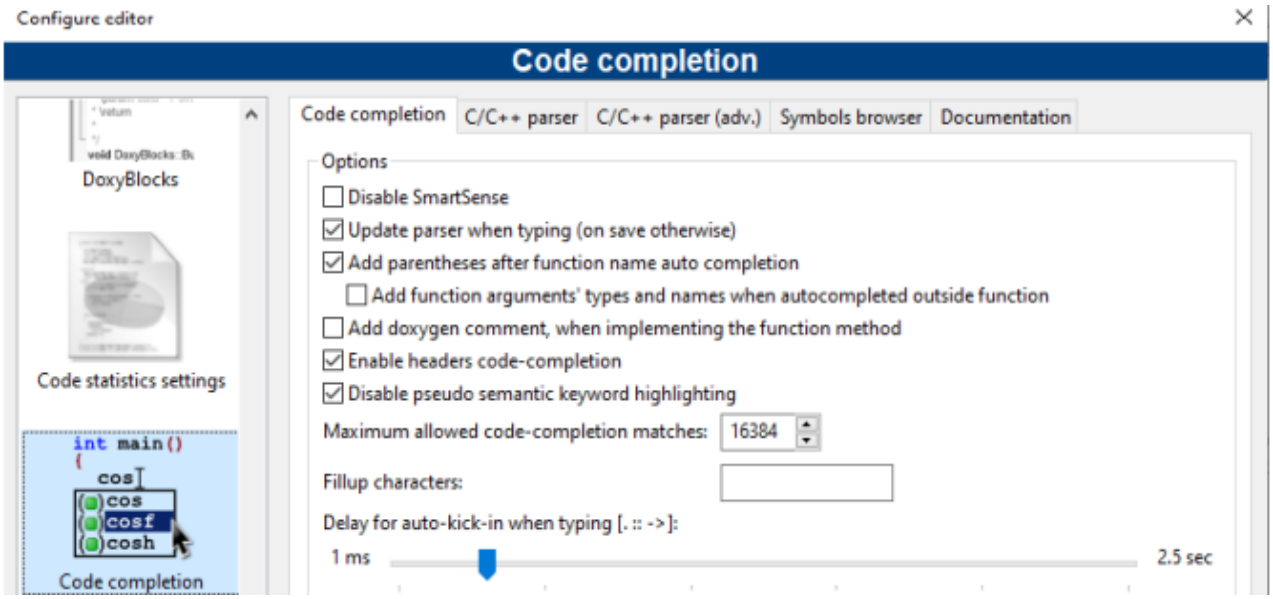
Preprocessor symbols (Symbole preprocesora) Wyświetla dyrektywy preprocesora utworzone przez **#define**.

Global macros (Makra globalne) Wyświetla listę makr dyrektyw preprocesora.



Rysunek 1.7: Widok symboli

Struktury i klasy są wyświetlane w „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 na stronie 16). Dwukrotne kliknięcie symbolu otworzy plik, w którym zdefiniowano symbol lub zaimplementowaną funkcję i przeskoczy do odpowiedniej linii. Automatyczne odświeżanie przeglądarki symboli bez zapisywania pliku można aktywować poprzez menu „Settings” (Ustawienia) → „Editor” (Edytor) → „Code Completion” (Uzupełnianie kodu) (patrz Rysunek 1.8 na stronie 17). W przypadku projektów z wieloma symbolami wpłynie to na wydajność w Code::Blocks.



Rysunek 1.8: Włącz parsowanie 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 klas bez implementacji).

1.10.7 Dołączanie zewnętrznych plików pomocy

Code::Blocks ma tylko swój własny plik pomocy: zwykle programiści potrzebują znacznie więcej pomocy i informacji o języku, bibliotekach, protokołach, formatach plików i tak dalej. Wtyczka pomocy udostępnia całą niezbędną dokumentację z samego Code::Blocks. Praktycznie każdy dokument można zintegrować z systemem pomocy Code::Blocks, ponieważ wtyczka pomocy ma możliwość uruchamiania zewnętrznych programów, jeśli to konieczne, w celu przeglądania dodanych dokumentów.

Po dodaniu nowego pliku pomocy lub dokumentu w menu „Help” (Pomoc) pojawi się nowy wpis, aby go otworzyć.

Środowisko programistyczne Code::Blocks obsługuje dołączanie zewnętrznych plików pomocy poprzez menu „Settings” (Ustawienia) → „Environment” (Środowisko). Dołącz wybraną instrukcję w formacie chm w „Help Files” (Plikach pomocy) wybierz „this is the default help file” (to jest domyślny plik pomocy) (patrz Rysunek 1.9 na stronie 18). Wpis $(keyword) jest symbolem zastępczym dla wybranego elementu w edytorze. Teraz możesz wybrać funkcję w otwartym pliku źródłowym w Code::Blocks, klikając myszą, a odpowiednia dokumentacja pojawi się po naciśnięciu F1.

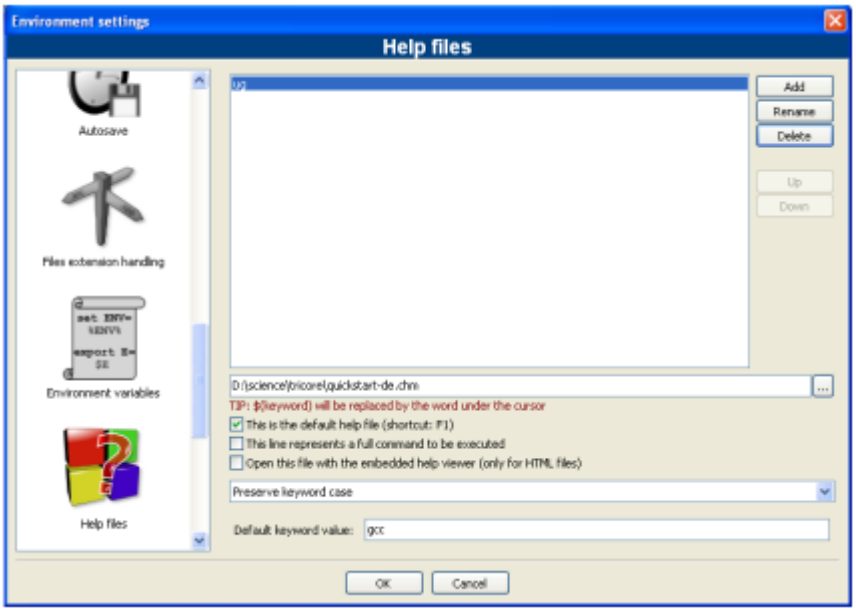
Jeśli dołączyłeś wiele plików pomocy, możesz wybrać termin w edytorze i wybrać plik pomocy z menu kontekstowego „Locate in” (Zlokalizuj) dla Code::Blocks do wyszukiwania.

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

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

W systemie Linux strony podręcznika są zwykle instalowane i tak, w systemie Windows można je pobrać np. stąd: http://www.win.tue.nl/~aeb/linux/man

Opcje pliku pomocy



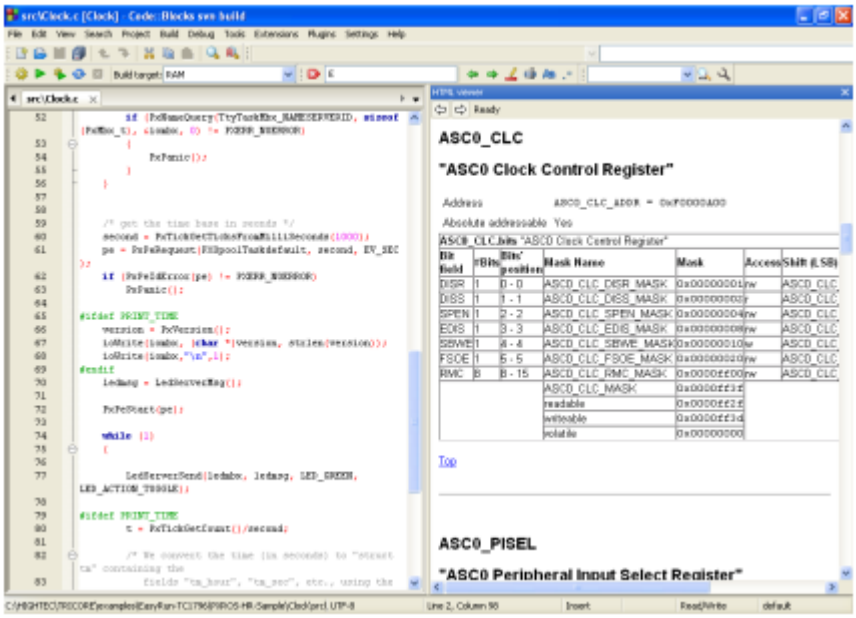
Rysunek 1.9: Ustawienia plików pomocy

- Możesz powiedzieć Code::Blocks, aby używał pliku jako domyślnego pliku pomocy, zaznaczając pole „This is the default help file” (To jest domyślny plik pomocy). W ten sposób ten plik będzie wyświetlany za każdym razem, gdy naciśniesz klawisz „F1”. Co więcej, jeśli wpiszesz słowo kluczowe $(keyword) jako domyślne słowo kluczowe (patrz poniżej), ten plik zostanie przeszukany pod kątem słów kluczowych (biorąc wybrane słowo lub słowo poniżej kursora w bieżącym pliku źródłowym) i zostanie otwarty z odpowiednim tematem, jeśli obecny.

- Możesz powiedzieć Code::Blocks, aby otworzył plik pomocy na wybrany temat, wpisując odpowiednie słowo kluczowe w polu tekstowym „Default keyword value” (Domyślna wartość słowa kluczowego). Jeśli plik pomocy jest domyślnym plikiem pomocy i użyjesz $(keyword) jako domyślnego słowa kluczowego, edytor użyje słowa pod kursorem (lub aktualnie wybranego) w aktualnie otwartym pliku jako słowa kluczowego, otwierając domyślny plik pomocy w odpowiednim temacie. Będzie to jednak prawdą tylko dla domyślnego pliku pomocy: żaden inny plik pomocy lub dokument nie będzie przeszukiwany w ten sposób. Na przykład, jeśli masz odnośnik do języka jako domyślny plik pomocy i dodasz plik pomocy biblioteki standardowej, będziesz miał objaśnione słowo kluczowe języka po naciśnięciu klawisza „F1”, ale nie będziesz miał wyjaśnionych w ten sposób funkcji bibliotecznych. I odwrotnie, ustawienie standardowego pliku pomocy biblioteki jako domyślnego da ci pomoc F1, ale stracisz tę funkcję dla słów kluczowych języka.

- Jeśli plik pomocy jest plikiem HTML, możesz polecić Code::Blocks otwarcie go za pomocą wbudowanej przeglądarki HTML, zaznaczając odpowiednią opcję.

Code::Blocks udostępnia „Wbudowaną przeglądarkę HTML”, której można użyć do wyświetlania prostego pliku html i znajdowania w nim słów kluczowych. Wystarczy skonfigurować ścieżkę do pliku html, który powinien zostać przeanalizowany, i zaznaczyć pole wyboru „Otwórz ten plik z wbudowaną przeglądarką pomocy” za pomocą menu „Settings” (Ustawienia) → „Environment” (Środowisko) → „Help Files” (Pliki pomocy).



Rysunek 1.10: Wbudowana przeglądarka HTML

Uwaga:
Jeśli wybierzesz plik html za pomocą dwukrotnego kliknięcia w eksploratorze plików (patrz sekcja 2.8 na stronie 48), wówczas zostanie uruchomiona osadzona przeglądarka html, o ile nie zostanie utworzone powiązanie z plikami html w programie obsługi rozszerzeń plików.

Pliki CHM

Pliki pomocy c++ chm można znaleźć w Internecie. Po prostu dodaj je za pomocą okna dialogowego.

W systemie Linux musisz zainstalować przeglądarkę chm, aby móc otwierać pliki chm. Istnieje kilka takich jak gnochm, kchmviewer, xchm i tak dalej.

1.10.8 Dołączanie narzędzi zewnętrznych

Dołączenie zewnętrznych narzędzi jest możliwe w Code::Blocks poprzez „Tools” (Narzędzia) → „Configure Tools” (Konfiguruj narzędzia) → „Add” (Dodaj). Do zmiennych wbudowanych (patrz rozdział 3.2 na stronie 79) można również uzyskać dostęp do parametrów narzędzia. Ponadto istnieje kilka rodzajów opcji uruchamiania do uruchamiania aplikacji zewnętrznych. W zależności od opcji, zewnętrznie uruchomione aplikacje są zatrzymywane po zamknięciu Code::Blocks. Jeśli aplikacje mają pozostać otwarte po wyjściu z Code::Blocks, musi być ustawiona opcja „Launch tool visible detached” (Uruchom narzędzie widoczne jako odłączone).

1.11 Wskazówki dotyczące pracy z Code::Blocks

W tym rozdziale przedstawimy kilka przydatnych ustawień w Code::Blocks.

1.11.1 Śledzenie modyfikacji

Code::Blocks udostępnia funkcję śledzenia modyfikacji w pliku źródłowym i wyświetlania paska na marginesie zmian. Modyfikacje są oznaczone żółtym paskiem zmian, a modyfikacje już zapisane będą używać zielonego paska zmian (patrz Rysunek 1.11 na stronie 20). Możesz poruszać się między zmianami za pomocą menu „Search” (Szukaj) → „’Goto next changed line” (Idź do następnej zmienionej linii) lub „Search” (Szukaj) → „Goto previous changed line” (Idź do poprzedniej zmienionej linii). 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 karetka).

Uwaga:
Jeśli zmodyfikowany plik zostanie zamknięty, historia zmian, taka jak cofnij / ponów i paski zmian, zostanie utracona. Za pomocą menu „Edit” (Edytuj) → „Clear changes history” (Wyczyść historię zmian) lub odpowiedniego menu kontekstowego możesz wyczyścić historię zmian, nawet jeśli plik jest otwarty.

1.11.2 Wymiana danych z innymi aplikacjami

Dane mogą być wymieniane między Code::Blocks i innymi aplikacjami. Dla tej komunikacji międzyprocesowej DDE (Dynamic Data Exchange (Dynamiczna wymiana danych)) jest używana dla okien, a w różnych systemach operacyjnych jest to komunikacja oparta na TCP.

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

```
[<polecenie>(”<parametr>” ) ]
```

Te polecenia są obecnie dostępne:

`Open` — polecenie
[Open (”d : \ temp\ test.txt”)]

używa parametru, w naszym przypadku jest to plik określony ścieżką bezwzględną i otwiera go w istniejącej instancji Code::Blocks lub uruchamia pierwszą instancję, jeśli jest to wymagane.

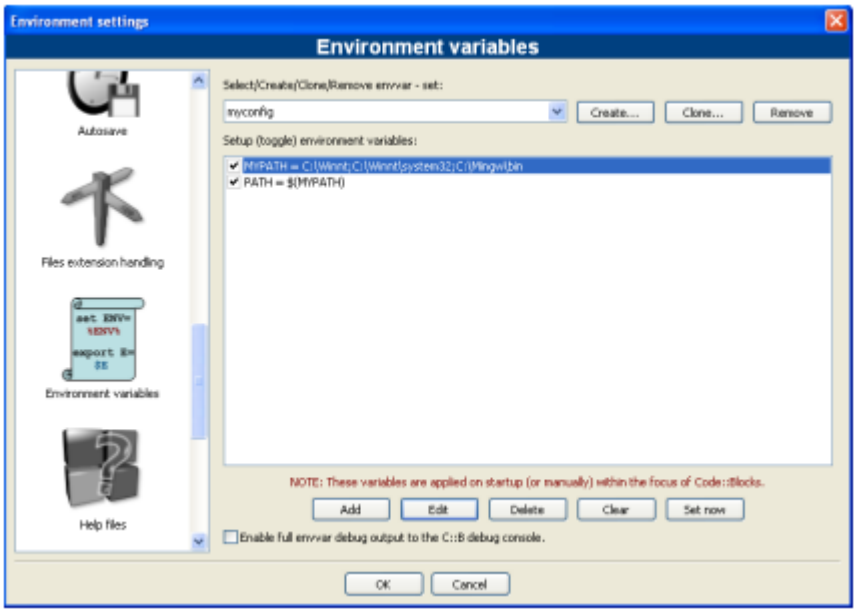
`OpenLine`	To polecenie otwiera plik o podanym numerze wiersza w instancji Code::Blocks. Numer linii jest określony przez : line. [OpenLine (”d : \ temp\ test.txt:10”)]
`Raise`	Ustaw fokus na instancję Code::Blocks. Nie wolno przekazywać parametru.

1.11.3 Konfiguracja zmiennych środowiskowych

Konfiguracja systemu operacyjnego jest określona przez tak zwane zmienne środowiskowe. Na przykład zmienna środowiskowa PATH zawiera ścieżkę do zainstalowanego kompilatora. System operacyjny przetworzy tę zmienną środowiskową od początku do końca, tj. wpisy na końcu zostaną przeszukane jako ostatnie. Jeśli zainstalowane są różne wersje kompilatora lub innych aplikacji, mogą wystąpić następujące sytuacje:

- Wywołała się nieprawidłowa wersja oprogramowania.
- Zainstalowane pakiety oprogramowania wywołują się wzajemnie.

Może się więc zdarzyć, że różne wersje kompilatorów lub innych narzędzi są obowiązkowe dla różnych projektów. Jedną z możliwości w takim przypadku jest zmiana zmiennych środowiskowych w sterowaniu systemem dla każdego projektu. Jednak ta procedura jest podatna na błędy i nie jest elastyczna. W tym celu Code::Blocks oferuje eleganckie rozwiązanie. Można tworzyć różne konfiguracje zmiennych środowiskowych, które są używane tylko wewnętrznie w Code::Blocks. Dodatkowo możesz przełączać się między tymi konfiguracjami. Rysunek 1.12 na stronie 21 pokazuje okno dialogowe, które można otworzyć poprzez „Zmienne środowiskowe” w „Settings” (Ustawienia) → „Environment” (Środowisko). Konfiguracja jest tworzona za pomocą przycisku „Create” (Utwórz).



Rysunek 1.12: Zmienne środowiskowe

Dostęp i zakres utworzonych tutaj zmiennych środowiskowych jest ograniczony do Code::Blocks. Możesz rozwinąć te zmienne środowiskowe tak jak inne zmienne Code::Blocks poprzez $(NAME).

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

Przykład

Możesz zapisać używane środowisko w kroku postbuild (zobacz rozdział 1.6 na stronie 9) w pliku `<project>.env` i zarchiwizować go w swoim projekcie.

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

lub pod Linuksem

echo \$PATH > project.env

1.11.4 Przełączanie między perspektywami

W zależności od wykonywanego zadania, przydatne może być posiadanie różnych konfiguracji lub widoków w Code::Blocks i zapisanie tych konfiguracji/widoków. Domyślnie ustawienia (np. pokaż/ukryj paski symboli, układ itp.) są przechowywane w pliku konfiguracyjnym default.conf. Używając opcji wiersza poleceń `--personality = ask` podczas uruchamiania Code::Blocks, można wybrać różne ustawienia. Poza tym globalnym ustawieniem może wystąpić sytuacja, w której będziesz chciał przełączać się między różnymi widokami okien i pasków symboli podczas sesji. Edytowanie plików i debugowanie projektów to dwa typowe przykłady takich sytuacji. Code::Blocks oferuje mechanizm przechowywania i wybierania różnych perspektyw, aby uniemożliwić użytkownikowi częste otwieranie i zamykanie okien i pasków symboli. Do zapisać perspektywę, wybierz menu „View” (Widok) → „Perspectives” (Perspektywy) → „Save current” (Zapisz bieżącą) i wprowadź nazwę w `<name>`. 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 skrótów klawiszowych.

Uwaga:
Innym przykładem jest edycja pliku w trybie pełnoekranowym bez pasków symboli. Możesz utworzyć perspektywę, taką jak „Full” (Pełna) i przypisać w tym celu skrót klawiszowy.

1.11.5 Przełączanie między projektami

Jeśli jednocześnie otwieranych jest kilka projektów lub plików, użytkownik potrzebuje sposobu na szybkie przełączanie się między projektami lub plikami. Code::Blocks ma kilka skrótów do takich sytuacji.

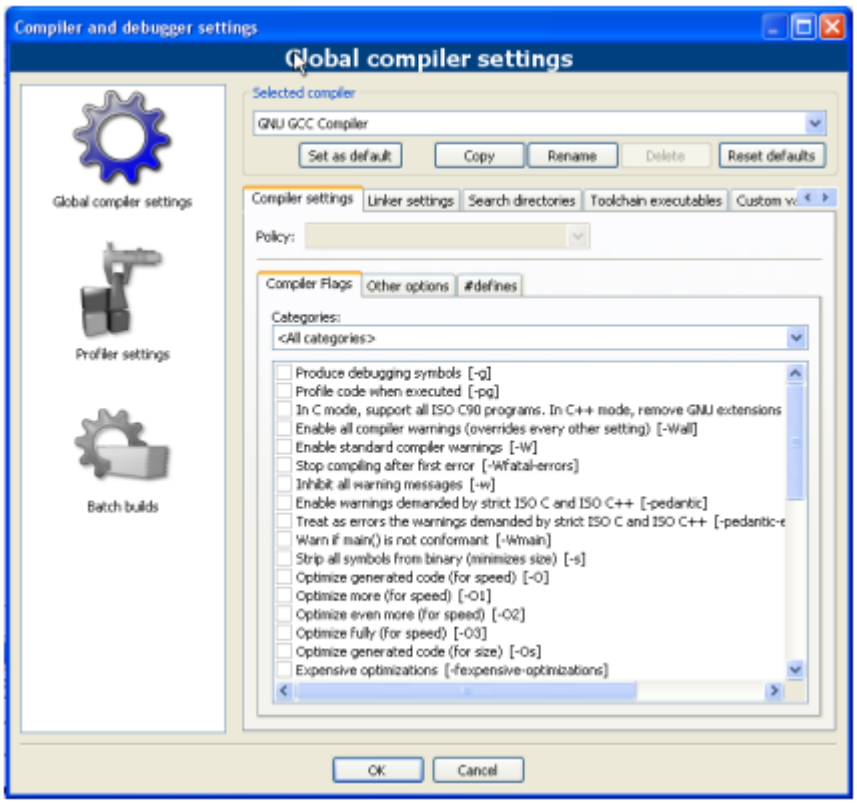
Alt-F5 Aktywuje poprzedni projekt z widoku projektu.

Alt-F6 Aktywuje następny projekt z widoku projektu.

F11 Przełącza w edytorze między plikiem źródłowym <nazwa>.cpp a odpowiednim plikiem nagłówkowym <nazwa>.h

1.11.6 Rozszerzone ustawienia dla kompilatorów

Podczas procesu budowania projektu komunikaty kompilatora są wyświetlane w oknie Messages na karcie Build Log. Jeśli chcesz otrzymywać szczegółowe informacje, wyświetlacz można rozszerzyć. 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

Zadbaj o to, aby został wybrany poprawny kompilator. Ustawienie „Full command line” (Pełny wiersz poleceń) w polu Compiler Logging (Rejestrowanie kompilatora) wyświetla pełne informacje w dzienniku kompilacji. Ponadto dane wyjściowe można rejestrować w pliku HTML. W tym celu wybierz „Save build log to HTML file when finished” (Po zakończeniu zapisz dziennik kompilacji do pliku HTML). Co więcej, Code::Blocks oferuje pasek postępu procesu kompilacji w oknie Build Log (Dziennik kompilacji), który można aktywować za pomocą ustawienia „Display build progress bar” (Wyświetl pasek postępu kompilacji).

1.11.7 Powiększanie w edytorze

Code::Blocks oferuje bardzo wydajny edytor. Ten edytor umożliwia zmianę rozmiaru, w jakim wyświetlany jest otwarty tekst. Jeśli używasz myszy z kółkiem, wystarczy nacisnąć klawisz Ctrl i przewijać kółkiem myszy, aby powiększyć lub pomniejszyć tekst.

Uwaga:
Za pomocą skrótu Ctrl-Numepad-/ lub menu „Edit” (Edycja) → „Special commands” (Polecenia specjalne) → „Zoom” → „Reset” (Resetuj) przywracany jest oryginalny rozmiar czcionki aktywnego pliku w edytorze.

1.11.8 Tryb owijania (Wrap Mode)

Podczas edycji plików tekstowych, m.in. *.txt, wewnątrz Code::Blocks, przydatne może być zawijanie tekstu, co oznacza, że długie linie będą wyświetlane w kilku liniach na ekranie, aby można je było odpowiednio edytować. Funkcję „Word wrap” (Zawijanie słów) można aktywować poprzez „Settings” (Ustawienia) → „Editor” (Edytor) → „Other Options” (Inne opcje) lub zaznaczając pole wyboru „Word wrap” (Zawijanie słów). Klawisze Home i End ustawiają kursor odpowiednio na początku lub końcu zawiniętych linii. Podczas ustawiania „Settings” (Ustawienia) → „Editor” (Edytor) → „Other Options” (Inne opcje) i „Home key always move to caret to first column” (Klawisz Home zawsze przesuń do karetki do pierwszej kolumny), kursor zostanie umieszczony odpowiednio na początku lub na końcu bieżącego wiersza, jeśli klawisze Home lub End są wcisnięte. Jeśli pożądane jest umieszczenie kursora na początku pierwszego wiersza bieżącego akapitu, należy użyć kombinacji klawiszy „Alt-Home”. To samo dotyczy „Alt-End” w celu umieszczenia kursora na końcu ostatniego wiersza bieżącego akapitu.

1.11.9 Wybierz tryby w edytorze

Code::Blocks obsługuje różne tryby wybierania lub wklejania ciągów:

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

2. Naciśnięcie klawisza „ALT” aktywuje tak zwany tryb wyboru bloku, a zaznaczenie prostokąta można podnieść lewym przyciskiem myszy. Jeśli klawisz Alt zostanie zwolniony, ten wybór można skopiować lub wkleić. Ta funkcja jest przydatna, jeśli chcesz wybrać niektóre kolumny, np. tablicy oraz skopiować i wkleić zawartość.

3. W menu „Settings” (Ustawienia) → „Editor” (Edytor) → „Margins and Caret” (Marginesy i karetka) można aktywować tzw. „Virtual Spaces” (Przestrzenie wirtualne). Ta opcja umożliwia, że zaznaczenie w trybie wyboru bloku może zaczynać się lub kończyć w pustym wierszu.

4. W menu „Settings” (Ustawienia) → „Editor” (Edytor) → „Margins and Caret” (Marginesy i karetka) można aktywować „Multiple Selection” (Wybór wielokrotny). Przytrzymując klawisz Ctrl, użytkownik może wybierać różne linie w aktywnym edytorze za pomocą lewego przycisku myszy. Wybory zostaną dołączone do schowka za pomocą skrótu Ctrl-C lub Ctrl-X. Ctrl-V wstawi zawartość w bieżącej pozycji kursora. Dodatkową opcję o nazwie „Enable typing (and deleting)” (Włącz wpisywanie (i usuwanie)) można aktywować dla wielu wyborów. 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 Linuksa używa ALT-LeftClickDrag do przenoszenia okna, więc będziesz musiał najpierw wyłączyć to zachowanie menedżera okien, aby wybór blokowy działał.

1.11.10 Składanie kodu

Code::Blocks obsługuje tzw. składanie kodu. Dzięki tej funkcji możesz złożyć m.in. Funkcje w edytorze Code::Blocks. Punkt składania jest oznaczony symbolem minusa na lewym marginesie widoku edytora.

Na marginesie widoczny jest początek i koniec punktu składania w postaci linii pionowej. Jeśli klikniesz symbol minusa lewym przyciskiem myszy, fragment kodu zostanie zwinięty lub rozwinięty. Za pomocą menu „Edit” (Edycja) → „Folding” (Składanie) można wybrać składanie. W edytorze widzisz złożony kod jako ciągłą linię poziomą.

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

Code::Blocks udostępnia funkcję składania również dla dyrektyw preprocesora. Aby włączyć tę funkcję, wybierz „Fold preprocessor commands” (Składanie poleceń preprocesora) w menu „Settings” (Ustawienia) → „Editor” (Edytor) we wpisie składania.

Inną możliwością jest ustawienie zdefiniowanych przez użytkownika punktów składania. Początek punktu składania wpisuje się jako komentarz z nawiasem otwierającym, a koniec to rynek z komentarzem z nawiasem zamykającym.

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

1.11.11 Autouzupełnianie

Jeśli otworzysz projekt w Code::Blocks 'Search directorys' twojego kompilatora i projektu, źródła i nagłówki twojego projektu są analizowane. Ponadto słowa kluczowe odpowiedniego pliku lexer są analizowane. Informacje o analizie są używane do funkcji autouzupełniania w Code::Blocks. Sprawdź ustawienia edytora, jeśli ta funkcja jest włączona. Automatyczne uzupełnianie jest dostępne za pomocą skrótu Ctrl-Spacja. Za pomocą menu „Settings” (Ustawienia) → „Editor” (Edytor) → „Syntax highlighting” (Podświetlanie składni) możesz dodać do swojego leksera słowa kluczowe zdefiniowane przez użytkownika.

1.11.12 Znajdź uszkodzone pliki

Jeśli plik zostanie usunięty z dysku, ale nadal znajduje się w pliku projektu <projekt>.cbp, wtedy ten „uszkodzony plik” zostanie wyświetlony jako uszkodzony symbol w widoku projektu. Zamiast usuwania plików należy użyć menu „Remove file from project” (Usuń plik z projektu).

W dużych projektach z dużą ilością podkatalogów poszukiwanie uszkodzonych plików może być czasochłonne. Code::Blocks oferuje wraz z wtyczką ThreadSearch (zobacz sekcję 2.18 na stronie 72) proste rozwiązanie tego problemu. Jeśli wprowadzisz wyrażenie wyszukiwania w ThreadSearch i wybierzesz opcję „Project files” (Pliki projektu) lub „Workspace files” (Pliki obszaru roboczego), ThreadSearch przeanalizuje wszystkie pliki zawarte w projekcie lub obszarze roboczym. Jeśli zostanie znaleziony uszkodzony plik, ThreadSerch wyświetli błąd z brakującym plikiem.

1.11.13 Dołączanie bibliotek

W opcjach kompilacji projektu możesz dodać używane biblioteki za pomocą przycisku „Add” (Dodaj) w pozycji „Link libraries” (Połącz biblioteki) w „Linker Settings” (Ustawienia linkera). W ten sposób możesz użyć bezwzględnej ścieżki do biblioteki lub po prostu podać nazwę bez przedrostka `lib` i rozszerzenia pliku.

Przykład

Dla biblioteki o nazwie `<ścieżka>\libs\lib<nazwa>.a`, po prostu napisz `<nazwa>`. Linker z odpowiednimi ścieżkami wyszukiwania będzie wtedy poprawnie zawierał biblioteki.

Uwaga:
Inny sposób dołączania bibliotek opisano w sekcji 2.12 na stronie 57.

1.11.14 Kolejność łączenia obiektów

Podczas kompilacji obiekty `nazwa.o` są tworzone ze źródeł `nazwa.c/cpp`. Linker następnie wiąże poszczególne obiekty z nazwą aplikacji `nazwa.exe` lub nazwą systemu wbudowanego `system.elf`. W niektórych przypadkach może być pożądane wstępne zdefiniowanie kolejności, w jakiej obiekty zostaną połączone. W Code::Blocks można to osiągnąć poprzez przypisanie priorytetów. W menu kontekstowym „Properties” (Właściwości) możesz zdefiniować priorytety pliku w zakładce „Build” (Buduj). Niski priorytet spowoduje wcześniejsze połączenie pliku.

1.11.15 Autozapis

Code::Blocks oferuje sposoby automatycznego przechowywania projektów i plików źródłowych lub tworzenia kopii zapasowych. Funkcję tę można aktywować w menu „Settings” (Ustawienia) → „Environment” (Środowisko) → „Autosave” (Autozapis) . Czyniąc to, jako metodę tworzenia kopii zapasowej należy określić „Save to .save file” (Zapisz w pliku .save).

1.11.16 Ustawienia rozszerzeń plików

W Code::Blocks możesz wybierać spośród kilku sposobów traktowania rozszerzeń plików. Okno dialogowe ustawień można otworzyć przez „Settings” (Ustawienia) → „Files extension handling” (Obsługa rozszerzeń plików). Możesz użyć aplikacji przypisanych przez system Windows dla każdego rozszerzenia pliku (otworzyć go za pomocą powiązanej aplikacji) lub zmienić ustawienie dla każdego rozszerzenia w taki sposób, aby uruchomił się program zdefiniowany przez użytkownika (uruchomienie programu zewnętrznego) lub plik zostanie otwarty w edytorze Code::Blocks (otwórz 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 podczas działania programu zewnętrznego) powinno zostać dezaktywowane, ponieważ w przeciwnym razie Code::Blocks zostanie zamknięty za każdym razem, gdy plik z tym rozszerzeniem zostanie otwarty.

1.12 Code::Blocks w wierszu poleceń

IDE Code::Blocks można uruchomić z wiersza poleceń bez interfejsu graficznego.
W takim przypadku dostępnych jest kilka przełączników do kontrolowania procesu budowania projektu. Ponieważ Code::Blocks jest w ten sposób skryptowalny, tworzenie plików wykonywalnych może być zintegrowane we własne procesy robocze.

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

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

`--file=<nazwa pliku>[:wiersz]`
Otwiera plik w Code::Blocks i opcjonalnie przeskakuje do określonego wiersza.

`/h, --help` Wyświetla komunikat pomocy dotyczący argumentów wiersza poleceń.

`/na, --no-check-associations`
Nie sprawdzaj 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`
Wyświetl dziennik debugowania aplikacji.

`--prefix=<str>` Ustawia prefiks współdzielonego katalogu danych.

`/p, --personality=<str>, --profile=<str>`
Ustawia używaną osobowość. Możesz użyć ask jako parametru, aby wyświetlić wszystkie dostępne osobowości.

`--rebuild` Oczyść i zbuduj projekt lub obszar roboczy.

`--build` Zbuduj projekt lub obszar roboczy.

`--target=<str>` Ustawia cel dla kompilacji wsadowej. Na przykład `--target='Release'`.

`--no-batch-window-close`
Utrzymuje okno dziennika wsadowego widoczne po zakończeniu kompilacji wsadowej.
`--batch-build-notify`
Wyświetla komunikat po zakończeniu kompilacji wsadowej.

`--safe-mode` Wszystkie wtyczki są wyłączone podczas uruchamiania.

> <build log file> 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 Skróty

Ta sekcja opisuje skróty, które są lub mogą być używane w Code::Blocks.

1.13.1 Wprowadzenie

Ta wtyczka może być używana do powiązania jednego lub więcej skrótów klawiszowych z elementami menu.

Nawet jeśli IDE, takie jak Code::Blocks, jest obsługiwane głównie za pomocą myszy, skróty klawiaturowe są jednak bardzo pomocnym sposobem na przyspieszenie i uproszczenie procesów pracy. W poniższej tabeli zebraliśmy niektóre z dostępnych skrótów klawiaturowych.

1.13.2 Funkcje

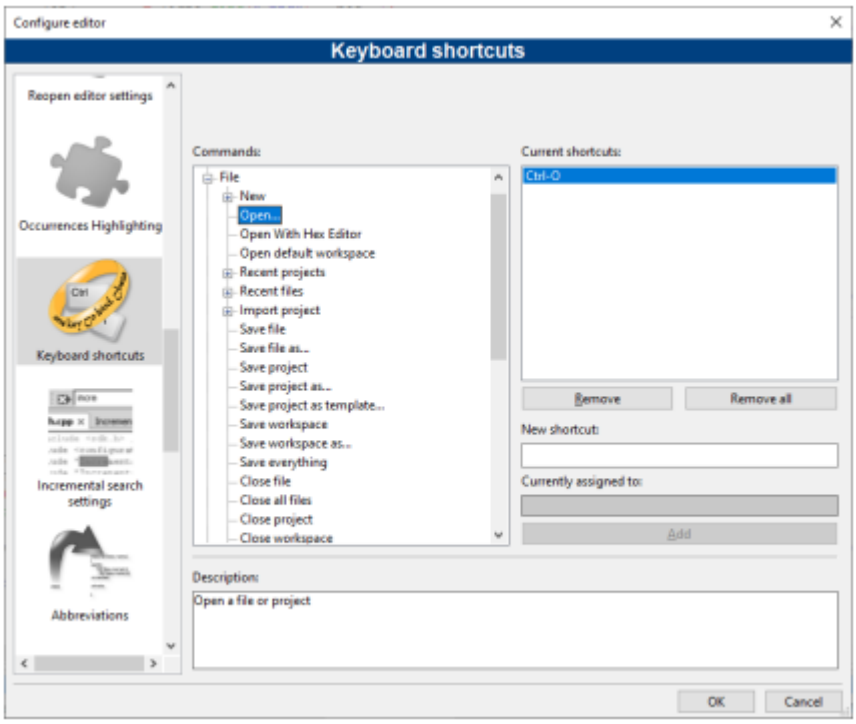
Zawiera panel konfiguracyjny i kompletny system do przeglądania/usuwania/dodawania/edycji skrótów poleceń.

Obsługuje wiele profili kluczy i obecny jest kompletny system ładowania/zapisywania.

Pozwala użytkownikom dostosować dowolne polecenie menu i zdefiniować wiele skrótów klawiszowych do każdego polecenia.

1.13.3 Użycie

Strona konfiguracji wtyczki jest dostępna poprzez „Settings” (Ustawienia) → „Editor” (Edytor) menu, a następnie wybierz sekcję Keyboard Shortcuts (Skróty klawiaturowe).



Rysunek 1.14: Okno dialogowe konfiguracji skrótów klawiaturowych

Wybranie polecenia z drzewa Polecenia pokazuje po prawej stronie bieżące skróty do tego polecenia. Na obrazku zaznaczona jest opcja Open... (Otwórz...) i wyświetlany jest domyślny skrót Ctrl-O.

Aby dodać nowy skrót dla wybranego polecenia, wykonaj następujące kroki:

1. Umieść fokus na polu tekstowym w obszarze New Shortcut (Nowy skrót) i naciśnij klawisze, na przykład F3 lub Ctrl-A.

2. Zaznacz „Currently assigned to" (Aktualnie przypisane do), jeśli inne polecenie ma już ten skrót, zobaczysz tam jego nazwę. Jeśli tekst mówi „None" (Brak), to jest to bezpieczne.

3. Naciśnij „Add" (Dodaj), aby dodać skrót do listy.

4. Naciśnij OK w oknie dialogowym, aby zapisać zmiany i powrócić do edytora.

1.13.4 Edytor

Funkcja	Skrót klawiaturowy
Cofnij ostatnią czynność	Ctrl+Z
Ponów ostatnią czynność	Ctrl+ Shift+Z
Wytnij zaznaczony tekst	Ctrl+X
Skopiuj zaznaczony tekst	Ctrl+C
Wklej tekst ze schowka	Ctrl+V
Zaznacz cały tekst	Ctrl+A
Zamień nagłówek/źródło	F11
Kod podświetlony w komentarzu	Ctrl+Shift+C
Odkomentuj podświetlony kod	Ctrl+Shift+X
Zduplikowana karetka linii jest włączona	Ctrl+D
Auto+uzupełnianie / Skróty	Ctrl+Space/Ctrl+J
Pokaż wskazówkę dotyczącą połączenia	Ctrl+Shift+Space
Zamień linię karetki jest włączona z linią nad nią	Ctrl+T
Przełącz zakładkę	Ctrl+B
Idź do poprzedniej zakładki	Alt+PgUp
Idź do następnej zakładki	Alt+PgDown
Przełącz bieżące składanie bloków	F12
Przełącz wszystkie foldery	Shift+F12

To jest lista skrótów udostępnianych przez komponent edytora Code::Blocks. Tych skrótów nie można odbić / zmienić.

Funkcja	Skrót klawiaturowy
Powiększ rozmiar tekstu.	Ctrl+Keypad ”+”
Zmniejsz rozmiar tekstu.	Ctrl+Keypad ”-”
Przywróć normalny rozmiar tekstu.	Ctrl+Keypad ”/”
Przeglądaj najnowsze pliki.	Ctrl+Tab
Wcięcie bloku.	Tab
Wgnieciony blok.	Shift+Tab
Usuń na początek słowa.	Ctrl+BackSpace
Usuń do końca słowa.	Ctrl+Delete
Usuń na początek wiersza.	Ctrl+Shift+BackSpace
Usuń do końca wiersza.	Ctrl+Shift+Delete
Przejdź do początku dokumentu.	Ctrl+Home
Rozszerz zaznaczenie do początku dokumentu.	Ctrl+Shift+Home
Przejdź na początek linii wyświetlacza.	Alt+Home
Rozszerz zaznaczenie do początku linii wyświetlacza.	Alt+Shift+Home
Przejdź na koniec dokumentu.	Ctrl+End
Rozszerz zaznaczenie do końca dokumentu.	Ctrl+Shift+End
Przejdź do końca linii wyświetlacza.	Alt+End
Rozszerz zaznaczenie do końca wyświetlanej linii.	Alt+Shift+End
Rozwiń lub zawęź punkt zagięcia.	Ctrl+Keypad ”*”
Utwórz lub usuń zakładkę.	Ctrl+F2
Przejdź do następnej zakładki.	F2
Wybierz do następnej zakładki.	Alt+F2
Znajdź wybór.	Ctrl+F3
Znajdź zaznaczenie od tyłu.	Ctrl+Shift+F3
Przewiń do góry.	Ctrl+Up
Przewiń w dół.	Ctrl+Down
Linia cięcia.	Ctrl+L
Kopiowanie linii.	Ctrl+Shift+T
Usuń wiersz.	Ctrl+Shift+L
Transpozycja linii z poprzednią.	Ctrl+T
Duplikat wiersza.	Ctrl+D
Znajdź pasujące warunki warunkowe preprocesora, z pominięciem zagnieżdżonych.	Ctrl+K
Zaznacz, aby dopasować warunek warunkowy preprocesora.	Ctrl+Shift+K
Znajdź pasujące warunki warunkowe preprocesora wstecz, z pominięciem zagnieżdżonych.	Ctrl+J
Wybierz, aby dopasować warunki preprocesora wstecz.	Ctrl+Shift+J
Poprzedni akapit. Shift rozszerza zaznaczenie.	Ctrl+[
Następny akapit. Shift rozszerza zaznaczenie.	Ctrl+]
Poprzednie słowo. Shift rozszerza zaznaczenie.	Ctrl+Left
Następne słowo. Shift rozszerza zaznaczenie.	Ctrl+Right
Poprzednia część słowa. Shift rozszerza zaznaczenie.	Ctrl+/
Następna część słowa. Shift rozszerza zaznaczenie.	Ctrl+\

1.13.5 Pliki

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

To jest lista skrótów udostępnianych przez komponent edytora Code::Blocks. Tych skrótów nie można zmienić.

Funkcja	Skrót klawiaturowy
Aktywuj następny otwarty plik.	Ctrl+Tab
Aktywuj poprzedni otwarty plik	Ctrl+Shift+Tab

1.13.6 Widok

Funkcja	Skrót klawiaturowy
Pokaż/ukryj okienko wiadomości Messages	F2
Pokaż/ukryj panel zarządzania Management	Shift+F2
Przenieś projekt w górę (w drzewie projektów)	Ctrl+Shift+Up
Przenieś projekt w dół (w drzewie projektów)	Ctrl+Shift+Down
Aktywuj wcześniej (w drzewie projektu)	Alt+F5
Aktywuj dalej (w drzewie projektu)	Alt+F6
Powiększ / pomniejsz	Ctrl+obróć kółko myszy
Edytor ostrości Focus	CTRL+Alt+E

1.13.7 Wyszukiwanie

Funkcja	Skrót klawiaturowy
Znajdź	Ctrl+F
Znajdź następny	F3
Znajdź poprzedni	Shift+F3
Znajdź w plikach	Crtl+Shift+F
Zamień	Ctrl+R
Zamień w plikach	Ctrl+Shift+R
Przejdź do linii	Ctrl+G
Przejdź do następnej zmienionej linii	Ctrl+F3
Przejdź do poprzedniej zmienionej linii	Ctrl+Shift+F3
Idź 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 wdrożenia	Ctrl+.
Otwórz dołącz plik	Ctrl+Alt+.

1.13.8 Kompilacja

Funkcja	Skrót klawiaturowy
Kompiluj	Ctrl+F9
Skompiluj bieżący plik	Ctrl+Shift+F9
Uruchom	Ctrl+F10
Skompiluj i uruchom	F9
Skompiluj na nowo	Ctrl+F11

1.13.9 Debugowanie

Funkcja	Skrót klawiaturowy
Debuguj	F8
Kontynuuj debugowanie	Ctrl+F7
Przejdź przez blok kodu	F7
Wejdź do bloku kodu	Shift+F7
Wyjdź z bloku kodu	Ctrl+Shift+F7
Przełącz punkt przerwania	F5
Uruchom do kursora	F4
Poprzedni błąd	Alt+F1

2 Wtyczki

Większość wtyczek opisanych w tym rozdziale znajduje się również na Wiki. Teksty i ryciny zostały skopiowane z Wiki, ale przystosowane do umieszczenia w dokumentach lateksowych (Miktex 2.9).

2.1 Ogólne

Funkcje Code::Blocks można rozszerzyć za pomocą wtyczek. Generalnie istnieją trzy rodzaje wtyczek:

Wtyczki rdzenia: opracowane i utrzymywane przez zespół rdzenia C::B.

Wtyczki Contrib: opracowane i utrzymywane przez społeczność i okazały się bardzo cenne. Są więc zintegrowane z C::B SVN.

Wtyczki innych firm: opracowane i utrzymywane przez społeczność, ale nie (jeszcze?) w repozytorium C::B. Te wtyczki często mają własne repozytorium lub są publikowane (w tym kod źródłowy) na forach.

Jeśli szukasz wtyczek:

1. Zajrzyj do oficjalnej wersji. Zauważ, że instalator/menedżer pakietów może wymagać włączenia niektórych wtyczek. Więc PRZECZYTAJ uważnie.

2. Przeszukaj fora w poszukiwaniu ogłoszeń, zwłaszcza fora na http://forums.codeblocks.org/index.php/board,14.0.html.

3. Na Wiki mogą znajdować się informacje dotyczące innych wtyczek na tej stronie i tutaj : http://Wiki.codeblocks.org/index.php/Announcement_for_plugins/patches.

W przypadku użytkowników systemu Windows domyślne zachowanie bieżącego instalatora **nie** powoduje instalowania wtyczek contrib. Musisz ręcznie zaznaczyć pole wyboru „contrib plugin” (wtyczka contrib), gdy zostaniesz poproszony o zainstalowanie wybranych komponentów. Nie ma możliwości późniejszego zainstalowania ich ręcznie.

Jeśli tworzysz wtyczki: Z pewnością możesz pracować z wtyczkami, jak chcesz, ale oto kilka sugestii:

Ogłoś je na forum rozwoju wtyczek - w tym (wstępny) kod źródłowy.

LUB

Skonfiguruj własną stronę internetową (lub użyj platformy do udostępniania plików) i opublikuj link do sources/binaries/svn na forum rozwoju wtyczek na forach.

LUB

Skonfiguruj repozytorium, prawdopodobnie w BerliOS lub SourceForge, umieść link do sources/binaries/svn na forum rozwoju wtyczek na forach. **Uwaga**: jest to bardzo wygodne, ponieważ załączniki na naszym forum mogą być od czasu do czasu usuwane. Dlatego publikowanie kodu źródłowego na forach nie jest bezpieczne.

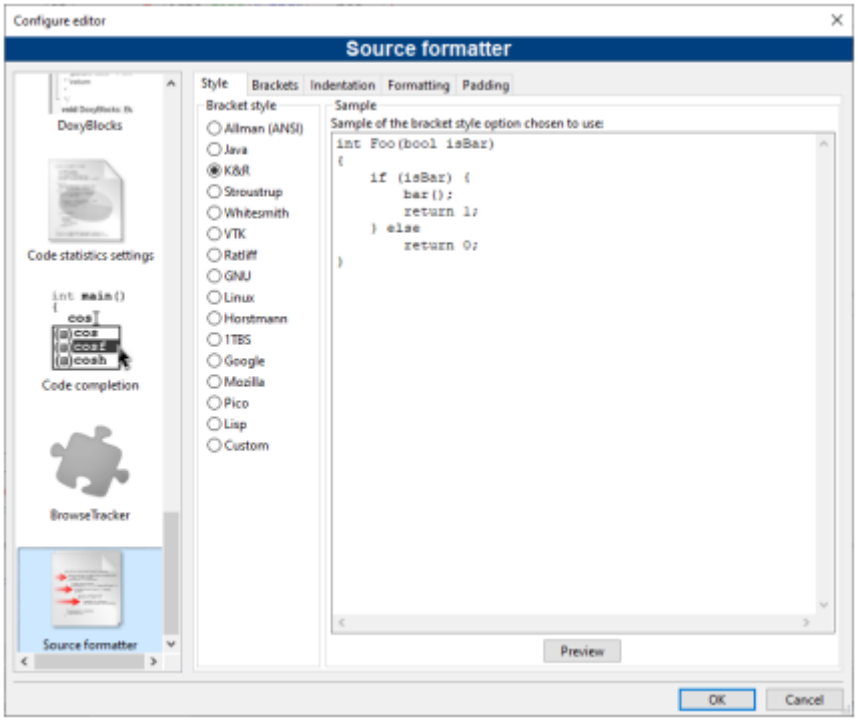
POTEM

Wpisz opis wtyczek na tej stronie.

Ogłoś wtyczkę przy użyciu tego szablonu na
http://Wiki.codeblocks.org/index.php/Template_for_plugin_announcement

2.2 Styl (Astyle)

Styl artystyczny to wcinacz kodu źródłowego, program do formatowania kodu źródłowego i moduł upiększający kod źródłowy dla języków programowania C, C++, C#. Może być używany do wybierania różnych stylów zasad kodowania w ramach Code::Blocks.



Rysunek 2.1: Formatowanie kodu źródłowego

Podczas tworzenia wcięć w kodzie źródłowym, jako programiści, mamy tendencję do używania zarówno spacji, jak i tabulacji, aby utworzyć pożądane wcięcie. Co więcej, niektórzy edytorzy domyślnie wstawiają spacje zamiast tabulatorów po naciśnięciu klawisza tabulacji, a inni edytorzy mają możliwość upiększania linii poprzez automatyczne ustawianie białej spacji przed kodem w linii, prawdopodobnie 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 pomiędzy edytorami, jednym ze standardowych problemów, z jakimi borykają się programiści podczas przechodzenia z jednego edytora do drugiego, jest ten kod zawierający spacje i tabulatory, który do tej pory był doskonale wcięty, nagle staje się bałaganem, na który trzeba patrzeć, przechodząc do innego edytora. Nawet jeśli jako programista starasz się używać TYLKO spacji lub tabulatorów, przeglądanie kodu źródłowego innych osób nadal może być problematyczne.

Aby rozwiązać ten problem, stworzono Artistic Style (Styl Artystyczny) - filtr napisany w C++, który automatycznie zmienia wcięcia i formatuje pliki źródłowe C/C++/C#.

Uwaga:
Podczas kopiowania kodu, na przykład z internetu lub instrukcji, kod ten zostanie automatycznie dostosowany do zasad kodowania w Code::Blocks.

2.3 Automatyczne wersjonowanie (AutoVersioning)

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

2.3.1 Wprowadzenie

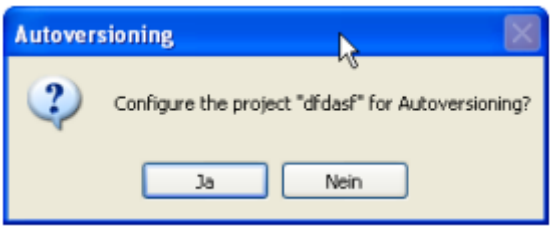
Pomysł na wtyczkę AutoVersioning powstał podczas opracowywania oprogramowania w wersji pre-alpha, które wymagało informacji o wersji i statusie. Byłem zajęty kodowaniem, nie mając czasu na utrzymanie numeru wersji, po prostu postanowiłem opracować wtyczkę, która mogłaby wykonać zadanie przy możliwie najmniejszej interwencji.

2.3.2 Funkcje

- Oto podsumowana lista funkcji, które obejmuje wtyczka:
- Obsługuje C i C++.
- Generuje i automatycznie inkrementuje zmienne wersji.
- Edytor statusu oprogramowania.
- Zintegrowany edytor schematów do zmiany zachowania automatycznej inkrementacji wartości wersji,
- Deklaracje dat jako miesiąc, dzień i rok.
- Wersja w stylu Ubuntu.
- Sprawdzenie wersji SVN.
- Zmieniony generator dziennika.
- Działa w systemach Windows i Linux.

2.3.3 Użytkowanie

Wystarczy przejść do menu „Project” (Projekt) → „AutoVersioning” (Autowersjonowanie). Pojawi się wyskakujące okienko, takie jak to:



Rysunek 2.2: Skonfiguruj projekt pod kątem autowersjonowania

Po naciśnięciu przycisku „Yes” w polu komunikatu z prośbą o skonfigurowanie otworzy się główne okno dialogowe konfiguracji automatycznego wersjonowania, umożliwiające skonfigurowanie informacji o wersji projektu.

Po skonfigurowaniu projektu do automatycznego wersjonowania ustawienia wprowadzone w oknie dialogowym konfiguracji zostaną zapisane w pliku projektu i zostanie utworzony plik version.h. Na razie za każdym razem, gdy klikniesz menu „Project” (Projekt) → „Autowersioning” (Autowersjonowanie), pojawi się okno dialogowe konfiguracji umożliwiające edycję wersji projektu i ustawień związanych z wersjonowaniem, chyba że nie zapiszesz nowych zmian wprowadzonych przez wtyczkę do projektu plik.

2.3.4 Okna dialogowe zakładek notatnika

Version Values (Wartości wersji)
Tutaj wystarczy wpisać odpowiednie wartości wersji lub pozwolić wtyczce automatycznego wersjonowania na ich zwiększenie (patrz Rysunek 2.3 na stronie 37).

Major (Większe) Przyrosty o 1, gdy wersja podrzędna osiągnie maksimum.

Minor (Mniejsze) Zwiększa się o 1, gdy numer kompilacji przekroczy barierę czasów kompilacji, wartość jest resetowana do 0, gdy osiągnie swoją maksymalną wartość.

Build Number (Numer kompilacji) (również odpowiednik wydania) — zwiększa się o 1 za każdym razem, gdy numer wersji jest zwiększany.

Revision (Rewizja) Zwiększa się losowo, gdy projekt został zmodyfikowany, a następnie skompilowany.

Status

Niektóre pola do śledzenia stanu oprogramowania z listą wstępnie zdefiniowanych wartości dla wygody (patrz Rysunek 2.4 na stronie 37).

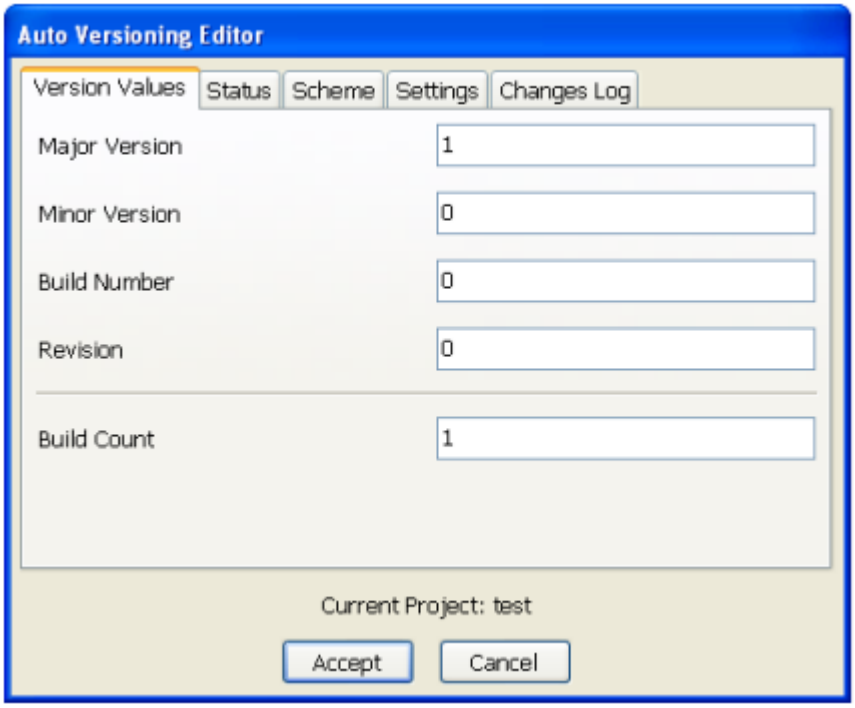
Software Status (Stan oprogramowania) Typowym przykładem powinna być wersja 1.0 Alpha

Abbreviation (Skrót) Taki sam jak status oprogramowania, ale taki jak: v1.0a

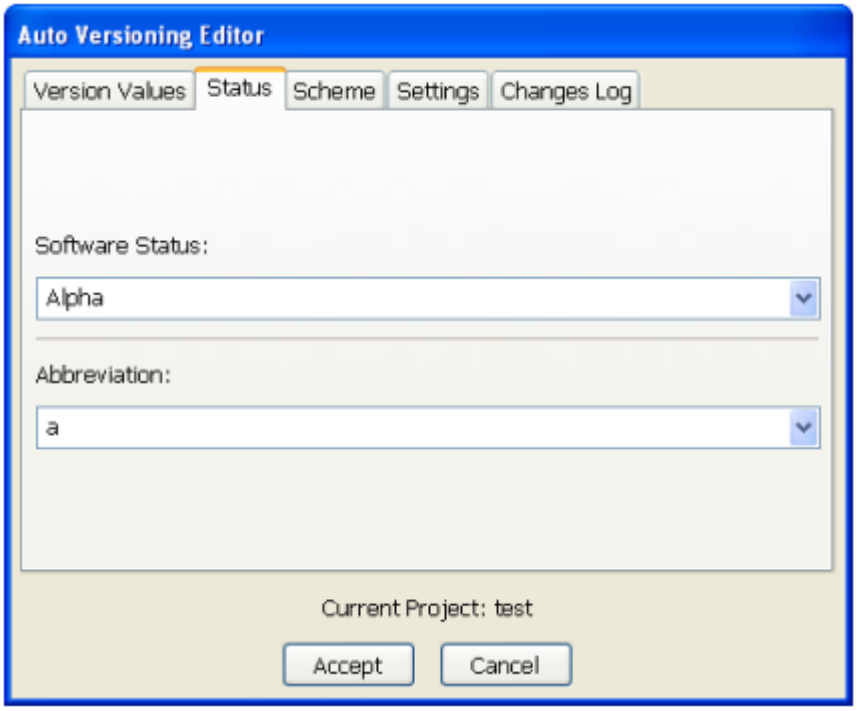
Scheme (Schemat)

Umożliwia edycję sposobu, w jaki wtyczka będzie zwiększać wartości wersji (patrz Rysunek 2.5 na stronie 38).

Minor maximum Maksymalna liczba, jaką może osiągnąć wartość Minor, po osiągnięciu tej wartości Major jest zwiększana o 1, a przy następnej kompilacji projektu Minor jest ustawiany na 0.



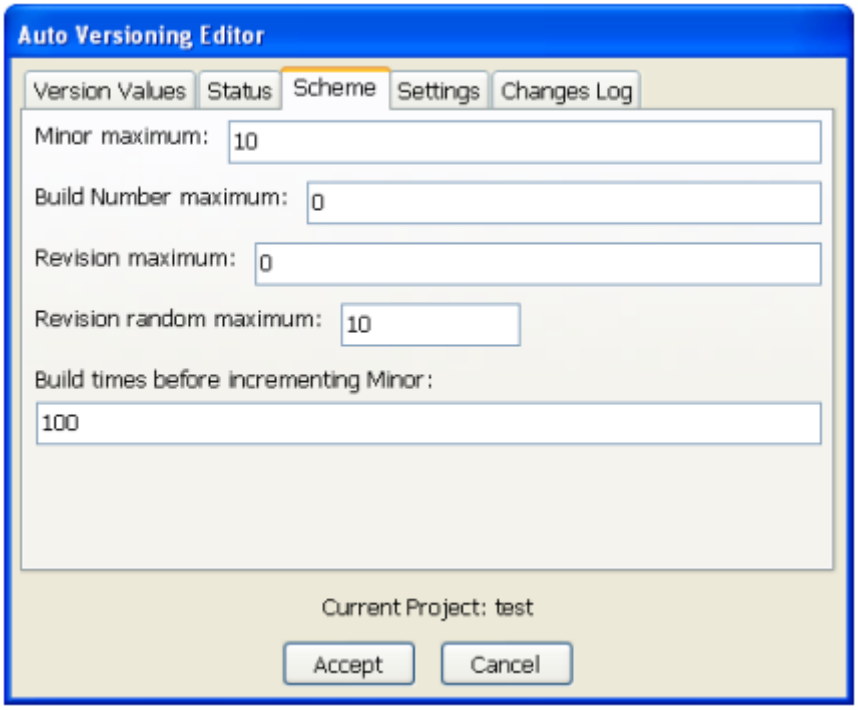
Rysunek 2.3: Ustaw wartości wersji



Rysunek 2.4: Ustaw status autowersjonowania

Build Number maximum (Maksymalna liczba kompilacji). Gdy wartość zostanie osiągnięta, następnym razem, gdy projekt jest kompilowany, jest ustawiane na 0. Wstaw 0 dla liczby nieograniczonej.

Revision maximum (Maksymalna wersja) Taka sama jak maksymalna liczba kompilacji. Wstaw 0 dla nieograniczonego.



Rysunek 2.5: Schemat autowersjonowania

Revision random maximum (Losowe maksimum wersji) Wersja zwiększa się o losowe liczby, o których decydujesz, jeśli wstawisz tutaj 1, wersja oczywiście zwiększy się o 1.

Build times before incrementing Minor (Czasy kompilacji przed inkrementacją Minor) Po pomyślnych zmianach w kodzie i kompilacji, historia kompilacji będzie inkrementowana, a gdy osiągnie tę wartość, Minor będzie inkrementowany.

Settings (Ustawienia)

Tutaj możesz ustawić niektóre ustawienia zachowania automatycznego wersjonowania (patrz Rysunek 2.6 na stronie 39).

Autoincrement Major and Minor (Autoincrementacja Major i Minor) Pozwala wtyczce zwiększać te wartości przy użyciu schematu. Jeśli nie jest zaznaczony, zwiększany będzie tylko numer kompilacji i wersja.

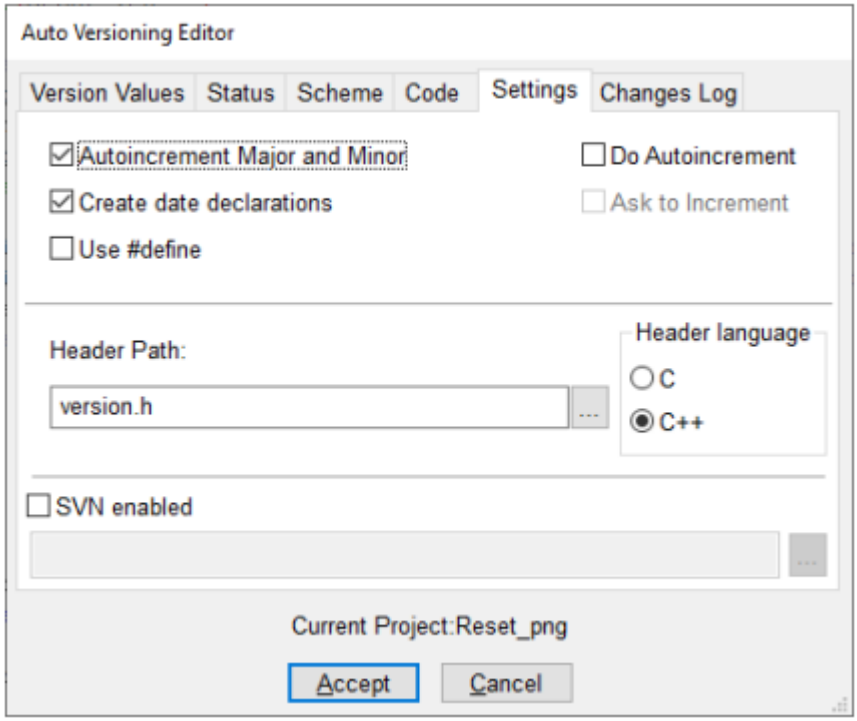
Create date declarations (Utwórz deklaracje dat) Utwórz wpisy w pliku `version.h` z datami i wersją w stylu ubuntu.

Do Auto Increment To mówi wtyczce, aby automatycznie zwiększała zmiany po dokonaniu modyfikacji, ta inkrementacja nastąpi przed kompilacją.

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

Ask to increment (Pytaj o inkrementację) Jeśli zaznaczone, Wykonaj automatyczną incrementację, przed kompilacją (jeśli wprowadzono zmiany) zostaniesz poproszony o zwiększenie wartości wersji.

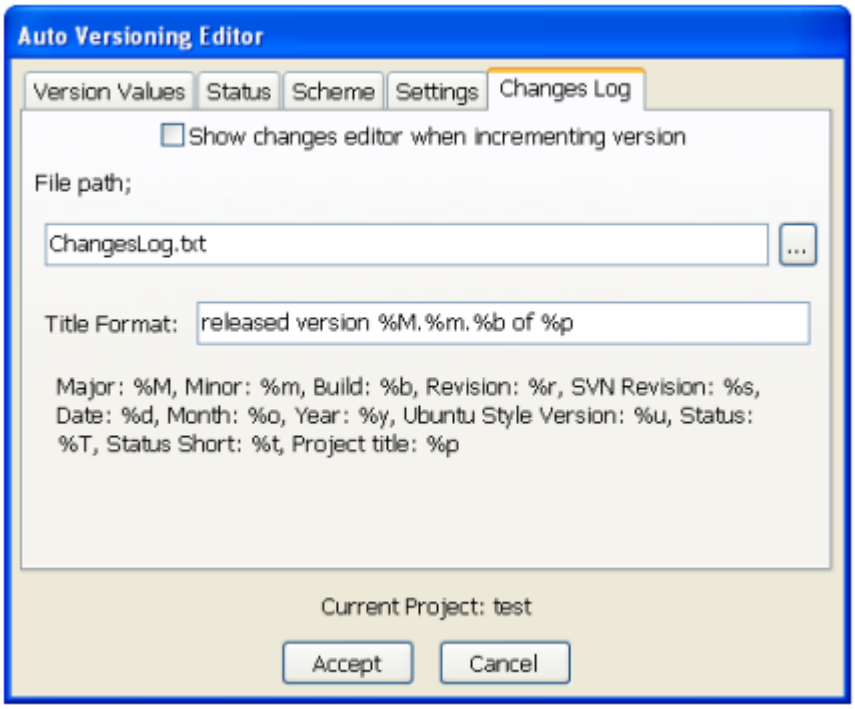
svn enabled Szukaj wersji svn i daty w bieżącym folderze i generuje poprawne wpisy w `version.h`



Rysunek 2.6: Ustawienia autowersjonowania

Changes Log

Pozwala to na wprowadzenie każdej zmiany dokonanej w projekcie w celu wygenerowania pliku ChangesLog.txt (patrz Rysunek 2.7 na stronie 39).



Rysunek 2.7: Dziennik zmian autowersjonowania

Show changes editor when incrementing version (Pokaż edytor zmian podczas zwiększania wersji)
Pojawi się edytor dziennika zmian podczas zwiększania wersji.

Title Format (Format tytułu) Tytuł z możliwością formatowania z listą wstępnie zdefiniowanych wartości.

Aby użyć zmiennych generowanych przez wtyczkę wystarczy #include <version.h>. Przykładowy kod wyglądałby następująco:

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

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

Wyjście version.h

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

```
#ifndef  VERSION_H
#define  VERSION_H

namespace AutoVersion {

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

        // Stan oprogramowania
        static const char STATUS[ ] = ”Pre-alpha”;
        static const char STATUS_SHORT[ ] = ”pa”;

        // Standardowy typ wersji
        static const long MAJOR = 0;
        static const long MINOR = 10;
        static const long BUILD = 1086;
        static const long REVISION = 6349;

        // Różne typy wersji
        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
```

W trybie C jest taki sam jak C++, ale bez przestrzeni nazw:

```
#ifndef  VERSION_H
#define  VERSION_H

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

        // Stan oprogramowania
        static const char STATUS[ ] = ”Pre-alpha”;
        static const char STATUS_SHORT[ ] = ”pa”;

        // Standardowy typ wersji
        static const long MAJOR = 0;
        static const long MINOR = 10;
        static const long BUILD = 1086;
        static const long REVISION = 6349;

        // Różne typy wersji
        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
```

2.3.6 Generator dziennika zmian

To okno dialogowe jest dostępne z menu „Project” (Projekt) → „Changes Log” (Dziennik zmian). Również, jeśli zaznaczono opcję „Show changes editor when incrementing version on the changes log settings” (Pokaż edytor zmian podczas inkrementacji wersji w ustawieniach dziennika zmian), otworzy się okno umożliwiające wprowadzenie listy zmian po modyfikacji źródeł projektu lub zdarzeniu inkrementacji (patrz Rysunek 2.8 na stronie 42).

Podsumowanie przycisków

Add (Dodaj) Dołącza wiersz do siatki danych

Edit (Edytuj) Umożliwia modyfikację wybranej komórki

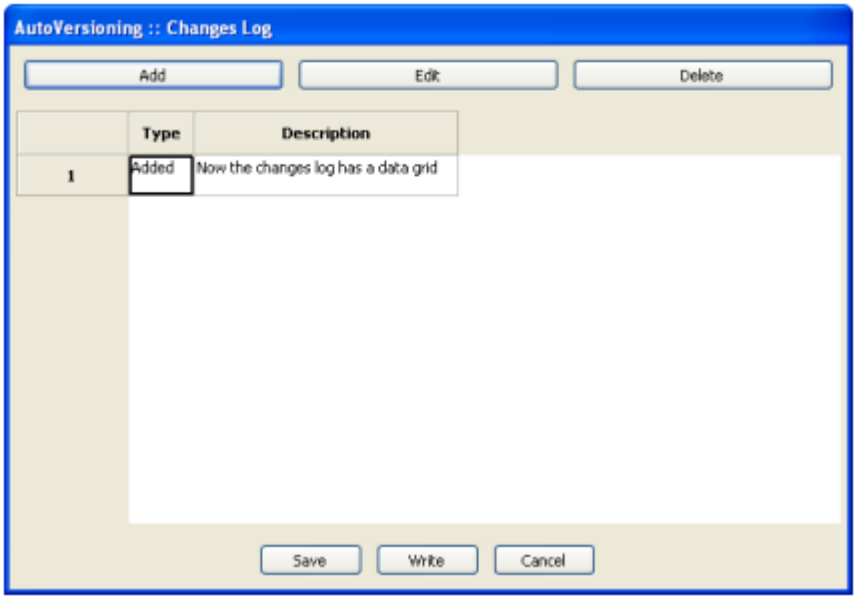
Delete (Usuń) Usuwa bieżący wiersz z siatki danych

Save (Zapisz) Zapisuje do pliku tymczasowego (changes.tmp) rzeczywiste dane do późniejszego przetworzenia w pliku dziennika zmian

Write (Zapis) Przetwarza dane siatki danych do pliku dziennika zmian

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

Oto przykład wyjścia wygenerowanego przez wtyczkę do pliku `ChangesLog.txt`:
03 września 2007



Rysunek 2.8: Zmiany w projekcie

released version 0.7.34 of AutoVersioning-Linux

Change log:
-Fixed: pointer declaration
-Bug: blah blah

02 September 2007
released version 0.7.32 of AutoVersioning-Linux

Change log:
-Documented some areas of the code
-Reorganized the code for readability

01 September 2007
released version 0.7.32 of AutoVersioning-Linux

Change log:
-Edited the change log window
-If the change log windows is leave blank no changes.txt is modified

2.4 Śledzenie przeglądania (Browse Tracker)

Browse Tracker to wtyczka, która pomaga nawigować między ostatnio otwieranymi plikami w Code::Blocks. Lista ostatnich plików jest zapisywana w historii. Za pomocą menu „View” (Widok) → „Browse Tracker” (Przeglądania Naganiacz) → „Clear All” (Wyczyść wszystko) historia jest wyczyszczana.

W oknie „Browsed Tabs” (Przeglądane karty) można nawigować między elementami ostatnio otwieranych plików, korzystając z pozycji menu „View” (Widok) → „Browse Tracker” (Przeglądania Naganiacz) → „Backward Ed/Forward Ed” (Wstecz Ed/Do przodu Ed) lub skrótu Alt-Left/Alt-Right. Menu Browse Tracker jest również dostępne jako menu kontekstowe. Znaczniki są zapisywane w pliku układu `<nazwaprojektu>.bmarks`

Powszechną procedurą podczas tworzenia oprogramowania jest walka z zestawem funkcji zaimplementowanych w różnych plikach. Wtyczka BrowseTracks pomoże Ci rozwiązać ten problem, pokazując kolejność, w jakiej pliki zostały wybrane. Następnie możesz wygodnie poruszać się po wywołaniach funkcji.

Wtyczka umożliwia nawet przeglądanie znaczników w każdym pliku w edytorze Code::Blocks. Pozycja kursora jest zapamiętywana dla każdego pliku. Możesz ustawić te znaczniki za pomocą pozycji menu „View" (Widok) → „Browse Tracker" (Naganiacz Przeglądania) → „Set BrowseMarks" (Ustaw BrowseMarks) lub zaznaczając linię lewym przyciskiem myszy. Znacznik z . . . jest pokazany na lewym marginesie. Za pomocą menu „View" (Widok) → „Browse Tracker" → „Prev Mark/Next Mark" (Poprzedni znak/Następny znak) lub skrót Alt-up/Alt-down możesz poruszać się po znacznikach w pliku. Jeśli chcesz nawigować w pliku pomiędzy znacznikami posortowanymi według numerów linii, wybierz menu „View" (Widok) → „Browse Tracker" → „Sort BrowseMark" (Sortuj BrowseMark).

Dzięki „Clear BrowseMark" znacznik w wybranej linii jest usuwany. Jeśli znacznik jest ustawiony dla linii, przytrzymanie lewego przycisku myszy przez 1/4 sekundy podczas wciskania klawisza Ctrl spowoduje usunięcie znacznika tej linii. Za pomocą menu „Clear All BrowseMarks" (Wyczyść wszystkie BrowseMarks) lub przytrzymując klawisz Ctrl lewym przyciskiem myszy na dowolnej nieoznaczonej linii, zresetujesz znaczniki w pliku.

Ustawienia wtyczki można skonfigurować w menu „Settings" (Ustawienia) → „Editor" (Edytor) → „Browse Tracker".

Mark Style (Oznacz styl). Browse Marks (Przeglądaj Znaki) są domyślnie wyświetlane jako . . . w granicach marginesu. Przy ustawieniu „Book Marks" (Zakładki książek) będą wyświetlane jako zakładki jako niebieska strzałka na marginesie. Przy ukrytym wyświetlaczu, Browse Marks (Przeglądaj znaczniki) jest wyłączone.

Toggle Browse Mark key (Przełącz klawisz Przeglądaj Znak). Znaczniki można ustawiać lub usuwać, klikając lewym przyciskiem myszy lub klikając i przytrzymując klawisz crtl.

Toggle Delay (Przełącz opóźnienie). Czas przytrzymania lewego przycisku myszy, aby przejść do trybu przeglądania znaczników.

Clear All BrowseMarks (Wyczyść wszystkie BrowseMarks) przytrzymując klawisz Ctrl i przez pojedyncze lub podwójne kliknięcie lewym przyciskiem myszy.

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

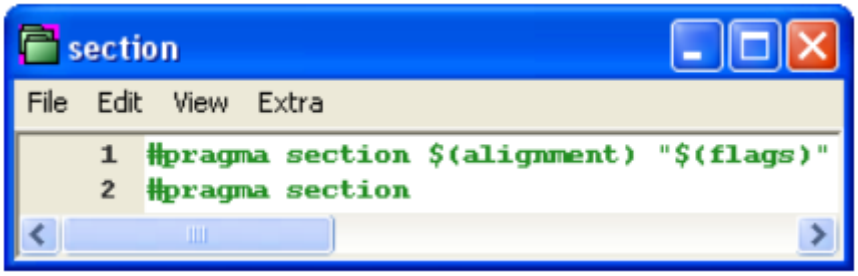
2.5 Fragmenty kodu (CodeSnippets)

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

Uwaga:
Możesz użyć zmiennych Code::Blocks lub zmiennych środowiskowych w linkach do plików, np. `$(VARNAME)/name.pdf`, aby sparametryzować łącze 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 `codenippets.xml`, który zostanie utworzony przez tę procedurę, można znaleźć w podkatalogu `codeblocks` w katalogu Documents and Settings\Application data. W systemie Linux te informacje są przechowywane w podkatalogu `.codeblocks` w katalogu HOME. Pliki konfiguracyjne Code::Blocks zostaną załadowane podczas następnego uruchomienia. Jeśli chcesz zapisać zawartość CodeSnippets w innej lokalizacji, wybierz wpis „Save Index As” (Zapisz indeks jako). Aby załadować ten plik, wybierz „Load Index File” (Załaduj plik indeksu) podczas następnego uruchamiania Code::Blocks lub dodaj katalog do menu kontekstowego „Settings” (Ustawienia) w „Snippet Folder” (Folderze fragmentów). Ustawienia są zapisywane w odpowiednim pliku `codenippets.ini` w danych aplikacji.

Aby dodać kategorię, użyj menu „Add SubCategory” (Dodaj podkategorię). Kategoria może zawierać fragmenty (moduły tekstowe) lub łącza do plików. Moduł tekstowy jest tworzony za pomocą polecenia „Add Snippet” (Dodaj fragment) w menu kontekstowym. Treść jest zintegrowana z modułem tekstowym jako „New snippet” (Nowy fragment) poprzez wybranie fragmentu tekstu w edytorze Code::Blocks oraz przeciągnięcie go i upuszczenie do modułu, aby wyświetlić okno dialogowe właściwości. Dwukrotne kliknięcie nowo dodanego wpisu lub wybranie „Edit Text” (Edytuj tekst) otworzy edytor treści.



Rysunek 2.9: Edycja modułu tekstowego

Wyjście modułu tekstowego jest obsługiwane w Code::Blocks za pomocą polecenia menu kontekstowego „Apply” (Zastosuj) lub przez przeciągnięcie i upuszczenie do edytora. W systemie Windows zawartość fragmentu można również przeciągać i upuszczać do innych aplikacji. W przeglądarce CodeSnippets możesz skopiować wybrany element za pomocą przeciągania i upuszczania do innej kategorii.

Poza tym moduły tekstowe mogą być parametryzowane przez zmienne `<name>`, do których można uzyskać dostęp poprzez `$(name)` (patrz Rysunek 2.9 na stronie 44). Wartości zmiennych można pobrać w polu wprowadzania, jeśli moduł tekstowy zostanie wywołany za pomocą polecenia menu kontekstowego „Apply” (Zastosuj).

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 przekonwertuje moduł tekstowy do łącza 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 kodów, wystarczy wybrać menu kontekstowe „Open File” (Otwórz plik) lub przytrzymać klawisz „Alt” i dwukrotnie kliknąć plik.

Uwaga:
Możesz dodać parzysty adres URL (np. http://www.codeblocks.org) w modułach tekstowych. Adres URL można otworzyć za pomocą menu kontekstowego „Open Url” (Otwórz adres URL) lub przeciągnij i upuść do ulubionej przeglądarki internetowej.

Przy tym ustawieniu, jeśli otworzysz link do pliku pdf z widoku kodów, przeglądarka pdf zostanie uruchomiona automatycznie. Ta metoda umożliwia użytkownikowi dostęp do plików rozsianych po całej sieci, takich jak dane cad, układy, dokumentacje itp., za pomocą typowych aplikacji, po prostu przez łącze. Zawartość codenippets jest przechowywana w pliku `codenippets.xml`, konfiguracja jest przechowywana w pliku `codenippets.ini` w katalogu danych aplikacji (`application data`). Ten plik ini będzie na przykład zawierał ścieżkę do pliku `codenippets.xml`.

Code::Blocks obsługuje korzystanie z różnych profili. Profile te nazywane są osobowościami. Uruchomienie Code::Blocks z opcją wiersza poleceń `--personality=<profil>` utworzy nowy lub użyje istniejącego profilu. Wtedy ustawienia nie zostaną zapisane w pliku `default.conf`, ale w `<personality>.conf` w katalogu danych aplikacji. Wtyczka Codesnippets będzie następnie przechowywać swoje ustawienia w określonym pliku o nazwie `<personality>.codesnippets.ini`. Teraz, jeśli załadujesz nową zawartość `<name.xml>` w ustawienia Codesnippets poprzez „Load Index File”, ta zawartość zostanie zapisana w odpowiednim pliku ini. Zaletą tej metody jest to, że w przypadku różnych profili można zarządzać różnymi konfiguracjami modułów tekstowych i linków.

Wtyczka oferuje dodatkową funkcję wyszukiwania do poruszania się między kategoriami i Snippets. Można dostosować zakres wyszukiwania Snippets (Fragmentów), kategorii lub Snippets i kategorii. Po wprowadzeniu wymaganego wyrażenia wyszukiwania, odpowiedni wpis jest automatycznie wybierany w widoku. Rysunek 2.10 na stronie 46 przedstawia typowy ekran w oknie CodeSnippets

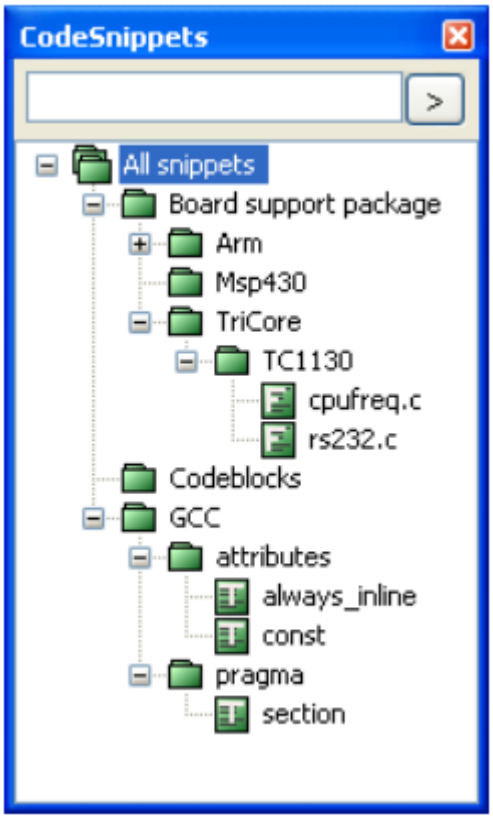
Uwaga:
W przypadku korzystania z obszernych modułów tekstowych, zawartość tych modułów należy zapisać w plikach za pomocą opcji „Konwertuj na łącze do pliku”, aby zmniejszyć zużycie pamięci w systemie. Jeśli usuniesz fragment kodu lub łącze do pliku, zostanie on przeniesiony do kategorii .trash; jeśli przytrzymasz klawisz Shift, element zostanie usunięty.

2.6 Doxyblocks

DoxyBlocks to wtyczka do Code::Blocks, która integruje doxygen z IDE. Umożliwia tworzenie dokumentacji, wstawianie bloków komentarzy oraz uruchamianie dokumentów HTML lub CHM. Zapewnia również konfigurację niektórych częściej używanych ustawień i dostęp do `doxywizard` w celu bardziej szczegółowej konfiguracji.

Ustawienia w pasku narzędzi DoxyBlocks mają następujące znaczenie:


Uruchom doxywizarda. Ctrl-Alt-D



Rysunek 2.10: Widok CodeSnipets (fragmentów kodu)

Wyodrębnij dokumentację dla bieżącego projektu. Ctrl-Alt-E

/** Wstaw blok komentarza w bieżącym wierszu. Dodatkowo DoxyBlocks spróbuje inteligentnie odczytać, czy w wierszu, do którego dodawany jest komentarz, istnieje metoda. Ctrl-Alt-B

```
/** \ brief
*
* \param bar bool
* \ return void
*
*/
void MyClass : : Foo (bool bar)
{
fooBar ( bar );
}
```

*< Wstaw komentarz do linii w bieżącej pozycji kursora. Ctrl-Alt-L

```
void MyClass : : Foo ( bool bar )
{
fooBar(bar);  /* *<  */
}
```

 Wyświetl wygenerowaną dokumentację HTML. Ctrl+Alt+H

 Wyświetl wygenerowaną dokumentację pomocy HTML. Ctrl+Alt+C

 Otwórz preferencje DoxyBlocks. Ctrl-Alt-P

Doxyblocks działa tylko wtedy, gdy w twoim systemie jest zainstalowany doxygen. Potrzebujesz przynajmniej plików wykonywalnych doxygen i doxywizard (dostępnych w oficjalnej dystrybucji doxygen na http://www.doxygen.nl/). Opcjonalnie możesz mieć wykonywalny „dot" (kropka) z pakietu graphviz (patrz https://graphviz.gitlab.io/. W systemie Windows do generowania plików chm można użyć kompilatora pomocy (hhc).

Uwagi

W preferencjach masz pole wyboru, aby zezwolić lub nie pozwolić DoxyBlocks na nadpisanie pliku doxy. Domyślnie, jeśli plik doxy już istnieje, nie jest on nadpisywany w celu ochrony zmian wprowadzonych poza DoxyBlocks, jednak to zachowanie zapobiega zapisywaniu zmian dokonanych w ramach DoxyBlocks w istniejącym pliku doxy.

Jeśli pole tekstowe jest puste w „Preferences" (Preferencjach), DoxyBlocks założy, że odpowiedni plik wykonywalny jest dostępny gdzieś na ścieżce twojego środowiska. Możesz użyć makr, takich jak $(CODEBLOCKS) na swojej ścieżce, a zostaną one automatycznie rozwinięte.

KATALOG WYJŚCIOWY Służy do określenia (względnej lub bezwzględnej) ścieżki bazowej, w której zostanie umieszczona wygenerowana dokumentacja. Jeśli zostanie wprowadzona ścieżka względna, będzie ona odnosić się do miejsca, w którym uruchomiono doxygen. Jeśli pozostanie puste, zostanie użyty bieżący katalog. DoxyBlocks użyje podanej tutaj nazwy ścieżki do utworzenia katalogu względem<preoject dir> (<katalog projektu>). Pozwala to na tworzenie oddzielnych katalogów doxygen dla projektów, które znajdują się w tym samym katalogu lub po prostu użyj innej nazwy katalogu. Jeśli to pole jest puste, dokumenty zostaną utworzone w (<katalogu projektu>) <project dir>/doxygen. Wprowadź nazwy katalogów bez kropek, wiodących separatorów, nazw woluminów itp. DoxyBlocks sprawdza poprawność nazwy ścieżki i usuwa zbędne znaki.

Przykłady:

```
[blank]          -> <project dir>/doxygen.
"docs"           -> <project dir>/docs.
"docs/sub1/sub2" -> <project dir>/docs/sub1/sub2.
"doxygen/docs"   -> <project dir>/doxygen/docs.
```

JĘZYK WYJŚCIOWY Służy do określenia języka, w którym napisana jest cała dokumentacja generowana przez doxygen. Doxygen wykorzysta te informacje do wygenerowania wszystkich stałych danych wyjściowych we właściwym języku. Domyślnym językiem jest angielski. Obsługiwane są inne języki.

Więcej informacji w plikach pomocy doxygen

2.7 Wtyczka Modyfikacje Edytora (Editor Tweaks)

Wtyczka EditorTweaks zawiera kilka różnych funkcji. Na bazie pliku kontroluje:

- zawijanie tekstu;
- numery linii;
- emisje klawiszy tabulacji (znaki tabulacji lub spacje);
- liczba spacji emitowanych przez klawisz tabulacji;
- znaki końca wiersza (powrót karetki + wysunięcie wiersza; powrót karetki; znak wysuwu);
- widoczność znaków końca linii;
- na żądanie paskowanie końcowych białych znaków;
- synchronizacja na żądanie znaków końca linii;
- wstawienie tłumienia klawiszy.

Od scalenia z wtyczką Aligner, ma możliwość uczynienia sekcji kodu bardziej czytelnymi poprzez wyrównanie określonego znaku. Na przykład, wyrównywanie „=” in

```
int var = 1 ;
int longVarName = 2 ;
int foobar = 3 ;
```

spowoduje:

```
int var = 1;
int longVarName = 2;
int foobar = 3;
```

2.8 Menedżer plików (FileManager) i wtyczka PowerShell

Eksplorator plików Rysunek 2.11 na stronie 49 jest zawarty we wtyczce FileManager i można go znaleźć w zakładce „Files” (Pliki). Skład Eksploratora plików pokazano na rysunku 2.11 na stronie 49.

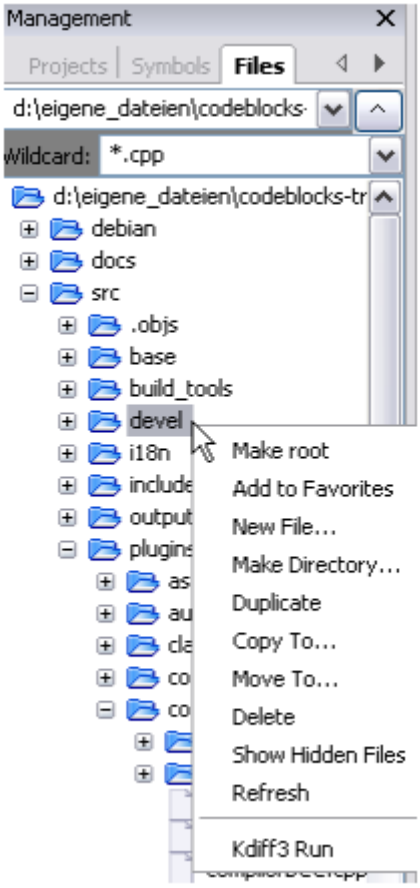
Na górze znajdziesz pole do wpisania ścieżki. Klikając przycisk na końcu tego pola, rozwijane pole wyświetla historię poprzednich wpisów, po których można nawigować za pomocą paska przewijania. Strzałka w górę po prawej stronie pola przesuwa w górę strukturę katalogów o jeden katalog.

W polu „Wildcard” możesz wprowadzić termin filtrowania dla wyświetlanego pliku. Pozostawienie pola pustego lub wpisanie * powoduje wyświetlenie wszystkich plików. Wprowadzanie *.c ;*. h, na przykład spowoduje wyświetlenie wyłącznie źródeł C i plików nagłówkowych. Otwarcie rozwijanego pola ponownie wyświetli historię ostatnich wpisów.

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

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

Make Root (Utwórz Korzeń) definiuje bieżący katalog jako katalog główny.



Rysunek 2.11: Menedżer plików

Add to Favorites (Dodaj do Ulubionych) ustawia znacznik dla katalogu i przechowuje go jako ulubiony. Ta funkcja pozwala szybko nawigować między często używanymi katalogami, także na różnych dyskach sieciowych

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

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

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

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

Copy To (Kopiuj do) otwiera okno dialogowe wprowadzania katalogu docelowego, w którym ma być przechowywany skopiowany plik/katalog.

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

Delete (Usuń) usuwa wybrane pliki/katalogi.

Show Hidden Files (Pokaż ukryte pliki) włącza/wyłącza wyświetlanie ukrytych plików systemowych. Po aktywacji ta pozycja menu jest zaznaczona.

Refresh (Odśwież) aktualizuje wyświetlanie drzewa katalogów.

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

Open in CB Editor (Otwórz w edytorze CB) otwiera wybrany plik w edytorze Code::Blocks.

Rename (Zmień nazwę) zmienia nazwę wybranego pliku.

Add to active project (Dodaj do aktywnego projektu) dodaje plik(i) do aktywnego projektu.

Uwaga:
Dostęp do plików/katalogów wybranych w Eksploratorze plików można uzyskać we wtyczce PowerShell za pośrednictwem zmiennej mpaths.

Funkcje zdefiniowane przez użytkownika można określić za pomocą polecenia menu „Settings" (Ustawienia) → „Environment" (Środowisko) → „PowerShell". W masce PowerShell nowa funkcja, którą można nazwać losowo, jest tworzona za pomocą przycisku „New" (Nowa). W polu „ShellCommand Executable" podany jest program wykonywalny, a w polu na dole okna można przekazać do programu dodatkowe parametry. Klikając funkcję w menu kontekstowym lub menu PowerShell, funkcja zostanie uruchomiona, a następnie przetworzy wybrane pliki/katalogi. Dane wyjściowe są przekierowywane do oddzielnego okna powłoki (Shell).

Na przykład wpis menu w „PowerShell" → „SVN" oraz 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 na stronie 79).

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

To i każde kolejne polecenie spowoduje utworzenie podmenu, w tym przypadku o nazwie „Extensions" (Rozszerzenia) → „SVN" → „Add" (Dodaj). Menu kontekstowe jest odpowiednio rozszerzone. Kliknięcie polecenia w menu kontekstowym spowoduje, że polecenie SVN Add przetworzy wybrane pliki/katalogi.

TortoiseSVN to szeroko rozpowszechniony program SVN z integracją z eksploratorem. Program TortoiseProc.exe TortoiseSVN można uruchomić w wierszu poleceń i wyświetlić okno dialogowe do zbierania danych wejściowych użytkownika. Możesz więc wykonywać polecenia, które są dostępne jako menu kontekstowe w eksploratorze również w linii poleceń. Dlatego możesz zintegrować go również z rozszerzeniem powłoki w Code::Blocks. Na przykład polecenie:

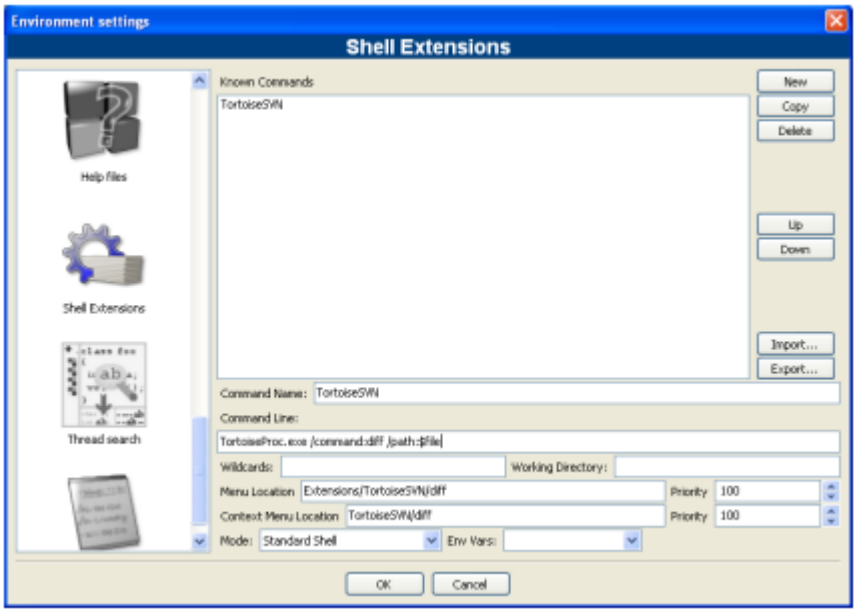
```
TortoiseProc.exe /command:diff /path:$file
```

porówna wybrany plik w eksploratorze plików Code::Blocks z bazą SVN. Zobacz rysunek 2.12 na stronie 51, jak zintegrować to polecenie.

Uwaga:
W przypadku plików, które są pod kontrolą SVN, eksplorator plików pokazuje ikony nakładek, jeśli są one aktywne w menu „View" (Widok) → „SVN Decorators" (Dekoratory SVN).

Przykład:

Możesz użyć eksploratora plików do porównywania plików lub katalogów. Wykonaj następujące kroki:



Rysunek 2.12: Dodaj rozszerzenie powłoki do menu kontekstowego

1. Dodaj nazwę za pomocą menu „Settings” (Ustawienia) → „Environment” (Środowisko) → „PowerShell”. Jest to pokazane jako wpis w menu tłumacza i menu kontekstowym.

2. Wybierz bezwzględną ścieżkę pliku wykonywalnego Diff (np. kdiff3). Dostęp do programu uzyskuje się za pomocą zmiennej `$interpreter`.

3. Dodaj parametry tłumacza:

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

To polecenie zostanie wykonane przy użyciu wybranych plików lub katalogów jako parametru. Wybór jest dostępny poprzez zmienną `$mpaths`. Jest to łatwy sposób na porównywanie plików lub katalogów.

Uwaga:
Wtyczka obsługuje użycie zmiennych Code::Blocks w rozszerzeniu powłoki (shell).

`$interpreter`	Wywołuje ten plik wykonywalny.
`$fname`	Nazwa pliku bez rozszerzenia.
`$fext`	Extension of the selected file.
`$file`	Nazwa pliku.
`$relfile`	Nazwa pliku bez informacji o ścieżce.
`$dir`	Nazwa wybranego katalogu
`$reldir`	Nazwa katalogu bez informacji o ścieżce.
`$path`	Ścieżka bezwzględna.

`$relpath`	Względna ścieżka pliku lub katalogu.
`$mpaths`	Lista aktualnie wybranych plików lub katalogów.
`$inputstr{<msg>}`	Łańcuch, który jest wprowadzany w oknie wiadomości.
`$parentdir`	Katalog nadrzędny (../).

Uwaga:
Wpisy rozszerzenia powłoki są również dostępne jako menu kontekstowe w edytorze Code::Blocks.

2.9 Edytor szestnastkowy (HexEditor)

Jak można otworzyć plik w HexEditor w ramach Code::Blocks: ’’ →’’

1. „File” (Plik) → „’ Open with HexEditor” (Otwórz za pomocą HexEditor).

2. Menu kontekstowe Nawigatora projektu („Open with” (Otwórz za pomocą) → „Hex editor” (Edytor szesnastkowy)

3. Wybierz zakładkę „Files” (Pliki) w Management Panel (Panelu Zarządzania). Wybierając plik w menedżerze plików i uruchamiając menu kontekstowe „Open With Hex editor” (Otwórz za pomocą edytora Hex), ten plik jest otwierany w edytorze HexEditor.

Podział okien:

po lewej to widok HexEditor, a po prawej to wyświetlacz, jako ciągi.

Górny wiersz: Aktualna pozycja (wartość w systemie dziesiętnym/szesnastkowym) i procent (stosunek bieżącej pozycji kursora do całego pliku).

Przyciski:

Przycisk Goto (Idź do): Skocz do pozycji bezwzględnej. Format dziesiętny lub szesnastkowy. Względny skok do przodu lub do tyłu poprzez określenie znaku.

Przycisk Search (Szukaj): Wyszukaj wzorce szesnastkowe w widoku HexEditor lub ciągi w podglądzie pliku.

Konfiguracja ilości kolumn: Exactly (Dokładnie), Multiple of (Wielokrotność), Power of (Potęga)

Tryb wyświetlania: Hex (heksagonalnie), Binary (binarnie)

Bytes (Bajty): Wybierz, ile bajtów ma być wyświetlanych w kolumnie.

Wybór Endianess: BE: Big Endian LE: Little Endian.

Value Preview (Podgląd wartości): Dodaje dodatkowy widok w HexEditor. Dla wybranej wartości w HexEditor, wartość jest również wyświetlana jako Word, Dword, Float, Double.

Expression Input (Wejście wyrażenia): Umożliwia wykonanie operacji arytmetycznej na wartości w edytorze HexEditor. Wynik operacji jest wyświetlany na prawym marginesie.

Calc: Tester wyrażenia

Edycja pliku w HexEditor:

Zawiera historię cofania i ponawiania.

Na przykład przesuń kursor do widoku ciągu: Wstaw spacje za pomocą klawisza Insert. Usuń znaki, naciskając klawisz Del.

Wprowadzając tekst, istniejąca treść jest nadpisywana jako ciąg.

Wprowadzając liczby w widoku HexEditor, wartości są nadpisywane, a podgląd jest aktualizowany.

2.10 Wyszukiwanie przyrostowe (Incremental Search)

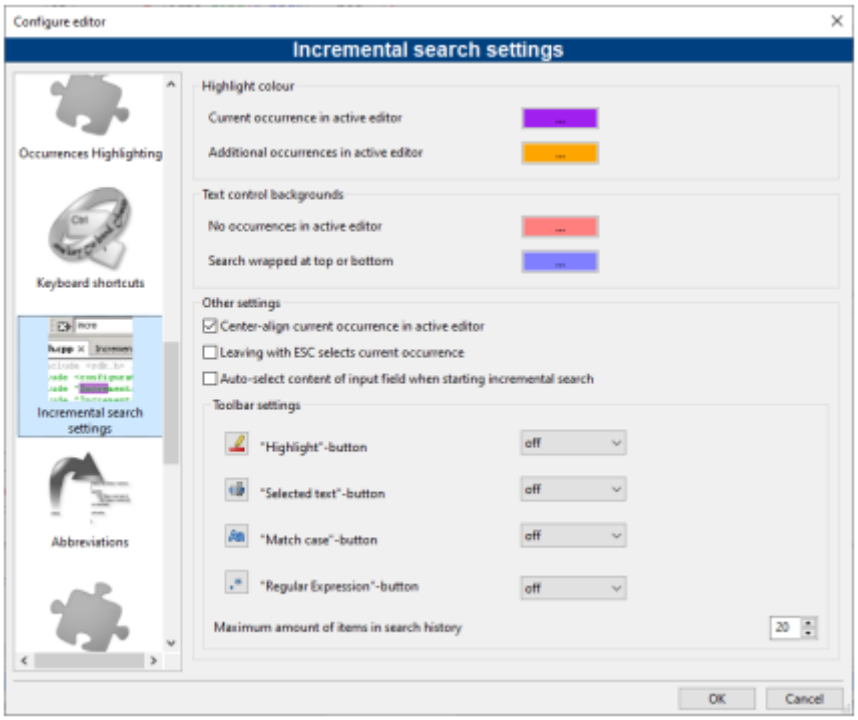
W celu wydajnego wyszukiwania w otwartych plikach Code::Blocks zapewnia tak zwane wyszukiwanie przyrostowe. Ta metoda wyszukiwania jest inicjowana dla otwartego pliku za pomocą menu „Search” (Szukaj) → „Incremental Search” (Wyszukiwanie przyrostowe) lub za pomocą skrótu klawiaturowego Ctrl-I. Fokus jest wtedy automatycznie ustawiany na maskę wyszukiwania odpowiedniego paska narzędzi. Gdy tylko zaczniesz wprowadzać wyszukiwany termin, tło maski wyszukiwania zostanie dostosowane zgodnie z wystąpieniem terminu. Jeśli trafienie zostanie znalezione w aktywnym edytorze, odpowiednia pozycja w tekście jest zaznaczana kolorem. Domyślnie, aktualne trafienie zostanie podświetlone na zielono. To ustawienie można zmienić w „Settings” (Ustawienia) → „Editor” (Edytor) → „Incremental Search” (Wyszukiwanie przyrostowe) (patrz Rysunek 2.13 na stronie 54). Naciśnięcie klawisza Return spowoduje przejście wyszukiwania do następnego wystąpienia szukanego ciągu w pliku. Za pomocą klawisza Shift Return można wybrać poprzednie wystąpienie. Ta funkcja 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 szukanego ciągu nie można znaleźć w aktywnym pliku, fakt ten jest wyróżniany czerwonym tłem maski wyszukiwania.

ESC Opuść tryb wyszukiwania przyrostowego.



Rysunek 2.13: Ustawienia wyszukiwania przyrostowego

ALT-DELETE Wyczyść dane wejściowe z pola wyszukiwania przyrostowego.

Ikony na pasku narzędzi wyszukiwania przyrostowego mają następujące znaczenie:

Usuwanie tekstu w masce wyszukiwania paska narzędzi wyszukiwania przyrostowego.

Nawigacja między wystąpieniami ciągu wyszukiwania.

Kliknięcie tego przycisku spowoduje podświetlenie wszystkich wystąpień szukanego ciągu w edytorze, a nie tylko pierwszego wystąpienia.

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

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

W polu wejściowym wyszukiwania przyrostowego można użyć wyrażenia regularnego.

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

2.11 Wtyczka NassiShneiderman

Wtyczka NassiShneiderman umożliwia tworzenie diagramów Nassi Shneiderman w ramach Code::Blocks ([, → NassiShneiderman]).

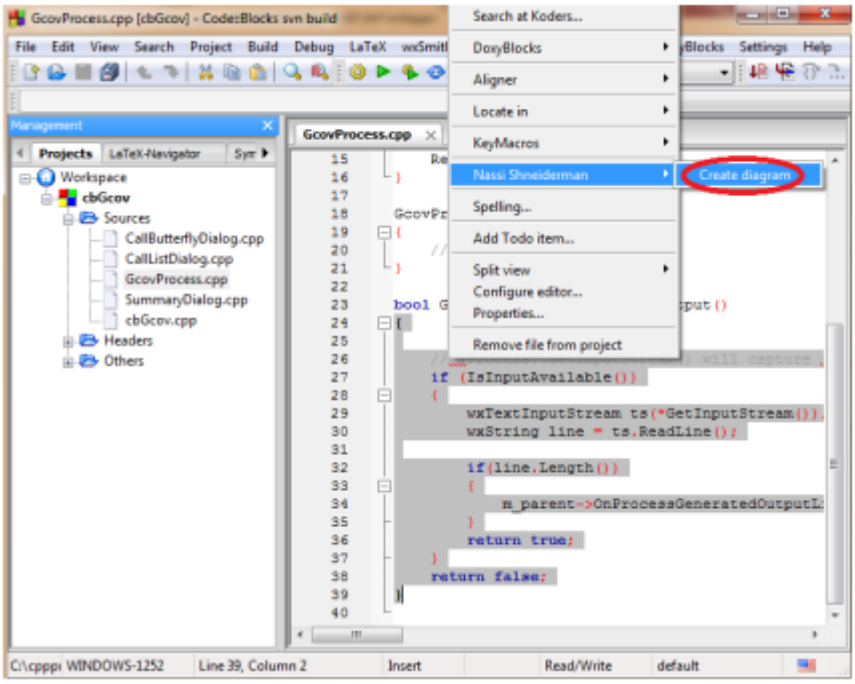
2.11.1 Tworzenie diagramu

Istnieją dwie możliwości stworzenia diagramu.

1. Aby utworzyć pusty wykres, wybierz opcje menu „File” (Plik) → „New” (Nowy) → „Nassi Shneiderman diagram” (Wykres Nassi Shneidermana).

2. Drugą opcją jest utworzenie diagramu z kodu źródłowego C/C++.

W oknie edytora podświetl kod, z którego chcesz utworzyć diagram. Na przykład treść funkcji/metody od nawiasów otwierających do nawiasów zamykających. Następnie kliknij prawym przyciskiem myszy i wybierz „Nassi Shneiderman” → „Create diagram” (Utwórz diagram) (patrz Rysunek 2.14 na stronie 55).

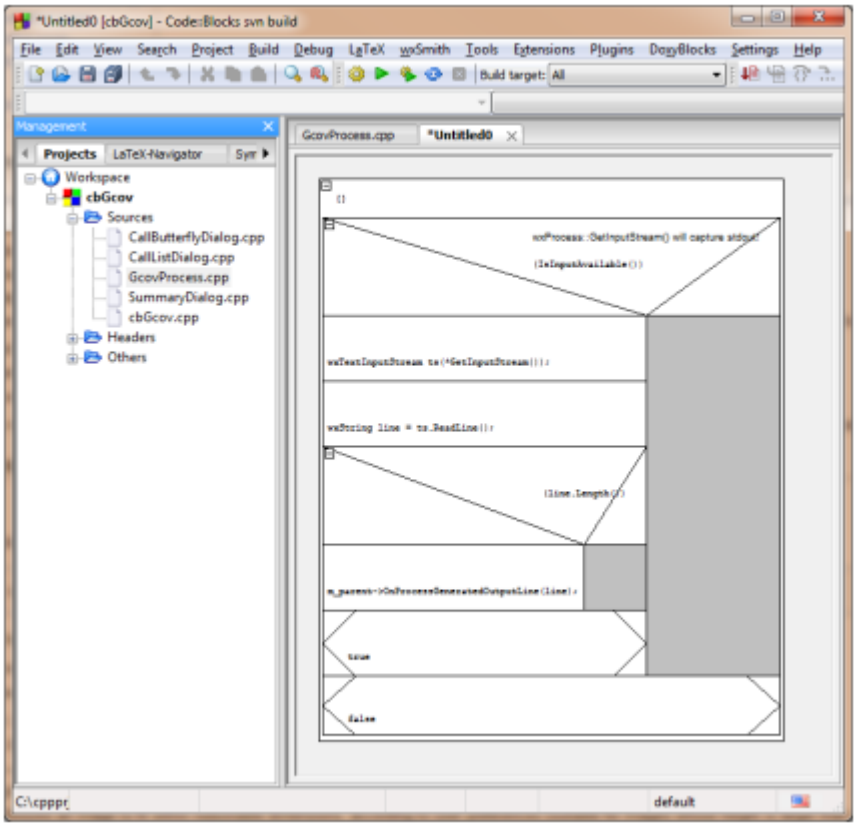


Rysunek 2.14: Tworzenie NassiShneiderman

Powinieneś otrzymać nowy diagram (patrz Rysunek 2.15 na stronie 56).

Parser ma pewne ograniczenia:

- Komentarze nie mogą znajdować się na końcu gałęzi.
- Z definicji funkcji jest w stanie przeanalizować tylko treść, a nie deklarację.
- Na pewno znajdziesz o wiele więcej...



Rysunek 2.15: Przykład diagramu NassiShneidermana

2.11.2 Edycja struktogramów

2.11.2.1 Co zrobić z diagramem?

Za pomocą struktogramu możesz zrobić wiele rzeczy:

1. Przechowuj do późniejszego wykorzystania. Diagram można zapisać jako „File” (Plik) → „Save file” (Zapisz plik) lub „File” (Plik → „Save file as...” (Zapisz plik jako...).

2. Możliwy jest eksport do różnych formatów „File” (Plik) → „Export” (Eksport).

   - „Export source...” (Eksportuj źródło...), aby zapisać jako plik źródłowy C.
   - „StrukTeX” do wykorzystania w dokumentacji w LaTeX-ie.
   - „PNG” lub „PS” i ewentualnie „SVG”, aby mieć diagram w formacie obrazu znanym z wielu innych narzędzi.

3. Bezpośrednio wstaw kod do edytora: Otwórz lub utwórz diagram. Wróć do okna edytora, kliknij prawym przyciskiem myszy i wybierz „Nassi Shneiderman” → „insert from xy” (wstaw z xy) (tutaj znajdziesz listę wszystkich otwartych diagramów).

4. Przeciągnij i upuść diagram (lub jego części) do innych narzędzi. Na przykład do OpenOffice Writer, aby uzyskać obraz w swojej dokumentacji.

Jeśli wybrany diagram ma wybranie, eksport lub generowanie kodu zajmuje tylko tę część schematu.

2.11.2.2 Rozszerzenia

Wtyczka NassiShneiderman obsługuje niektóre rozszerzenia diagramów Nassi-Shneidermana:

- **break** ma specjalny klocek ze strzałką w prawo
- **continue** (dalej) ma specjalny klocek ze strzałką w lewo
- Aby móc tworzyć diagramy z instrukcji **switch c/c++**, zaznaczenie nie jest niejawnie przerywane przed wielkością liter. Różne przypadki są wyrównywane w pionie. Obsługuje C i C++.

2.12 LibFinder

Jeśli chcesz korzystać z niektórych bibliotek w swojej aplikacji, musisz skonfigurować swój projekt, aby z nich korzystał. Taki proces konfiguracji może być trudny i denerwujący, ponieważ każda biblioteka może używać niestandardowego schematu opcji. Innym problemem jest to, że konfiguracja różni się na platformach, co powoduje niekompatybilność między projektami uniksowymi i windowsowymi.

LibFinder zapewnia dwie główne funkcjonalności:

- Wyszukiwanie bibliotek zainstalowanych w twoim systemie
- Dołączanie biblioteki do projektu za pomocą kilku kliknięć myszą, dzięki czemu jest niezależna od platformy projektu

2.12.1 Wyszukiwanie bibliotek

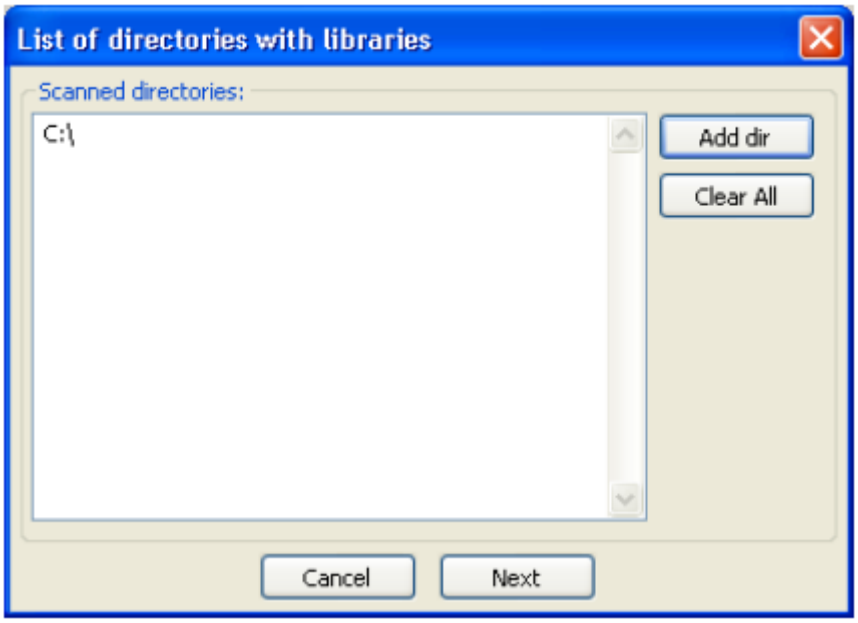
Wyszukiwanie bibliotek jest dostępne w menu „Plugins” (Wtyczki) → „Library finder” (Wyszukiwarka bibliotek). Jego celem jest wykrycie bibliotek zainstalowanych w twoim systemie i przechowywanie wyników w bazie danych LibFindera (zauważ, że te wyniki nie są zapisywane w plikach projektu Code::Blocks). Wyszukiwanie rozpoczyna się od dialogu, w którym można podać zestaw katalogów z zainstalowanymi bibliotekami. LibFinder przeskanuje je rekursywnie, 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 2.16 na stronie 58).

Gdy LibFinder skanuje w poszukiwaniu bibliotek, używa specjalnych reguł do wykrywania obecności biblioteki. Każdy zestaw reguł znajduje się w pliku xml. Obecnie LibFinder może wyszukiwać wxWidgets 2.6/2.8, Code::Blocks SDK i GLFW – lista zostanie rozszerzona w przyszłości

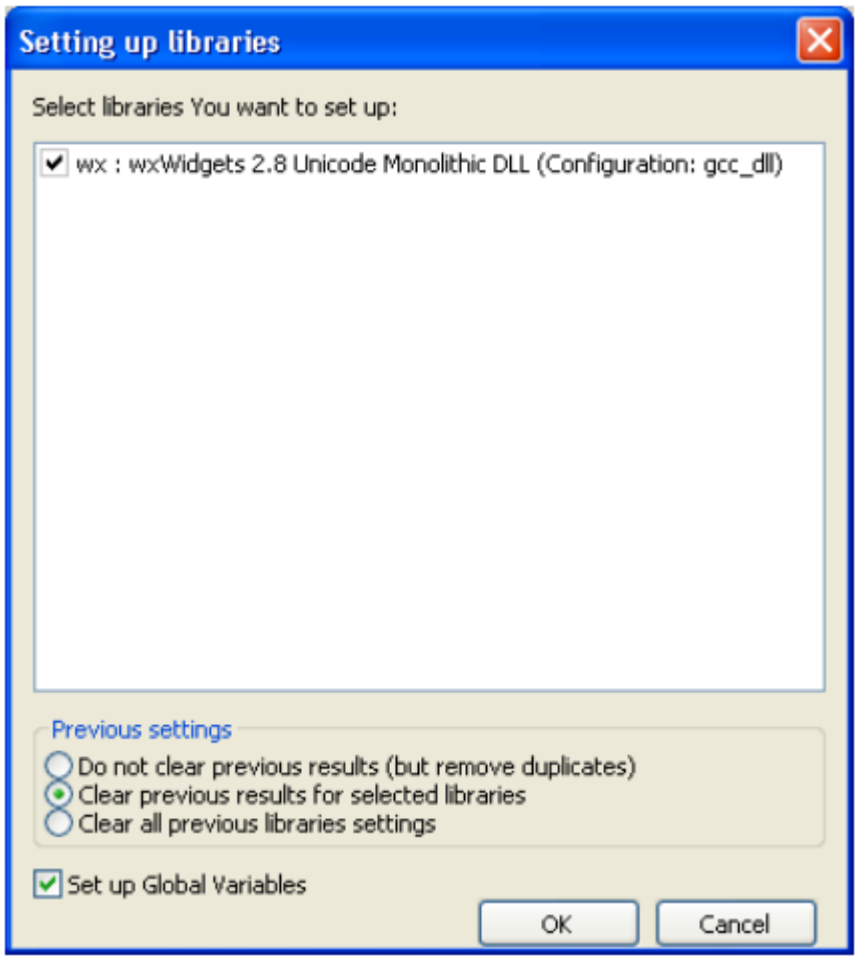
Uwaga:
Aby uzyskać więcej informacji na temat dodawania obsługi bibliotek do LibFinder, przeczytaj `src/plugins/contrib/lib_finder/lib_finder/readme.txt` w źródłach Code::Blocks.

Po zakończeniu skanowania LibFinder pokazuje wyniki (patrz Rysunek 2.17 na stronie 58).

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



Rysunek 2.16: Lista katalogów



Rysunek 2.17: Wyniki wyszukiwania

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

Do not clear previous results (Nie usuwaj 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. To jest 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 LibFindera.

Inną opcją w tym oknie dialogowym jest „Set up Global Variables" (Ustaw zmienne globalne). Gdy zaznaczysz tę opcję, LibFinder spróbuje automatycznie skonfigurować zmienne globalne, które są również używane do pomocy w obsłudze bibliotek.

Jeśli masz zainstalowany pkg-config w swoim systemie (jest on instalowany automatycznie w większości wersji linuksowych), LibFinder zapewni również biblioteki z tego narzędzia. Nie ma potrzeby ich skanowania – są one automatycznie ładowane po uruchomieniu Code::Blocks.

2.12.2 Uwzględnianie bibliotek w projektach

LibFinder dodaje dodatkową zakładkę we właściwościach projektu „Biblioteki" – ta zakładka pokazuje biblioteki używane w projekcie oraz biblioteki znane w LibFinder. Aby dodać bibliotekę do swojego projektu, wybierz ją w prawym okienku i kliknij przycisk <. Aby usunąć bibliotekę z projektu, wybierz ją w lewym panelu i kliknij przycisk > (patrz Rysunek 2.18 na stronie 60).

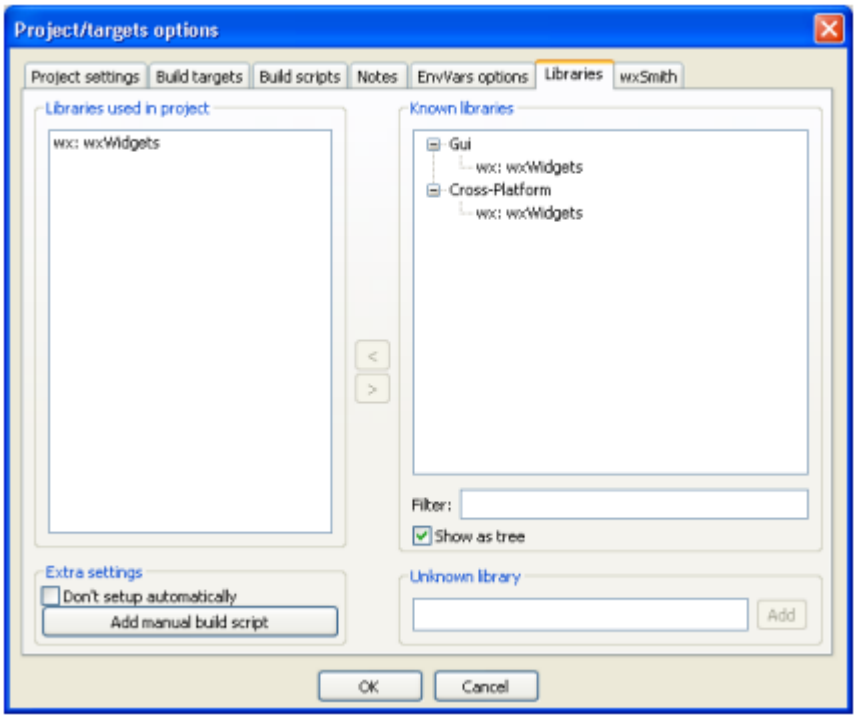
Możesz filtrować biblioteki znane LibFinderowi, dostarczając filtr wyszukiwania. Pole wyboru „Show as Tree" (Pokaż jako drzewo) pozwala przełączać się między widokiem skategoryzowanym i nieskategoryzowanym.

Jeśli chcesz dodać bibliotekę, która nie jest dostępna w bazie LibFindera, możesz użyć Pole „Nieznana biblioteka". Zauważ, że powinieneś wpisać krótki kod biblioteki (który zwykle odpowiada nazwie zmiennej globalnej) lub nazwę biblioteki w pkg-config. Lista sugerowanych skrótów znajduje się w Global Variables. Korzystanie z tej opcji jest zalecane tylko podczas przygotowywania projektu do zbudowania na innych maszynach, na których taka biblioteka istnieje i jest poprawnie wykrywana przez LibFinder. Możesz uzyskać dostęp do zmiennej globalnej w Code::Blocks, takiej jak: `$(#NAZWA_ZMIENNEJ_GLOBALNEJ.incude)`

Zaznaczenie opcji „Don't setup automatically" (Nie konfiguruj automatycznie) powiadomi LibFinder, że nie powinien automatycznie dodawać bibliotek podczas kompilowania tego projektu. W takim przypadku LibFinder można wywołać ze skryptu budującego. Przykład takiego skryptu jest generowany i dodawany do projektu po naciśnięciu „Add manual build script" (Dodaj skrypt ręcznej budowy).

2.12.3 Korzystanie z LibFindera i projektów wygenerowanych z kreatorów

Kreatory tworzą projekty, które nie używają LibFindera. Aby zintegrować je z tą wtyczką, będziesz musiał ręcznie zaktualizować opcje kompilacji projektu. Można to łatwo osiągnąć poprzez:



Rysunek 2.18: Konfiguracja projektu

... usunięcie wszystkich ustawień specyficznych dla biblioteki i dodanie biblioteki poprzez zakładkę „Libraries” (Biblioteki) we właściwościach projektu.

Taki projekt staje się wieloplatformowy. Dopóki używane biblioteki są zdefiniowane w bazie danych LibFinder, opcje budowania projektu będą automatycznie aktualizowane, aby dopasować ustawienia biblioteki specyficzne dla platformy.

2.13 Wtyczka sprawdzania pisowni (SpellChecker)

Wtyczka do sprawdzania pisowni ciągów i komentarzy.

2.13.1 Wprowadzenie

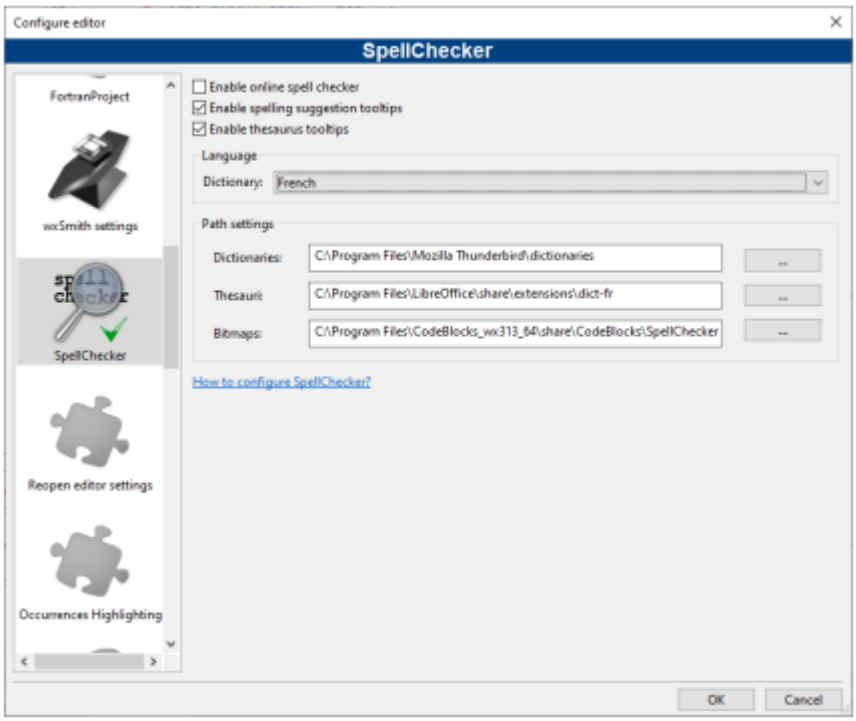
Wtyczka do sprawdzania pisowni ciągów i komentarzy. Pisownia jest sprawdzana podczas pisania. Dodatkowo dostępny jest tezaurus. Dostęp do obu można uzyskać na żądanie, wybierając odpowiednie słowo, a następnie wybierając opcję Pisownia... lub Tezaurus... z menu Edycja (operację można przypisać do skrótu klawiszowego za pomocą wtyczki Skróty klawiaturowe (Keyboard Shortcuts)). Menu kontekstowe (kliknij słowo prawym przyciskiem myszy) zawiera sugestie dotyczące pisowni.

2.13.2 Konfiguracja

Konfiguracja odbywa się w menu „Settings” (Ustawienia) → „Editor” (Edytor). Opcja sprawdzania pisowni znajduje się w połowie listy po lewej stronie.

Znaczenie kontrolek to:

Enable online spell checker (Włącz sprawdzanie pisowni online) Włącz lub wyłącz sprawdzanie pisowni.



Rysunek 2.19: Konfiguracja sprawdzania pisowni

Language (Język). Język używany do sprawdzania pisowni i tezaurusa wybiera się, wybierając słownik. Można to również zmienić na pasku stanu.

Path settings, Dictionaries (Ustawienia ścieżki, słowniki) Wtyczka szuka w tej ścieżce plików słowników.

Path settings, Thesauri (Ustawienia ścieżki, tezaurus) Wtyczka szuka w tej ścieżce plików potrzebnych tezaurusowi.

Path settings, Bitmaps (Optional) (Ustawienia ścieżki, mapy bitowe (opcjonalnie)) Wtyczka szuka w tej ścieżce flag, które mają być wyświetlane na pasku stanu.

Uwaga:
Możesz używać makr w powyższych trzech ustawieniach ścieżki, takich jak `$(CODEBLOCKS)/share/codeblocks/SpellChecker`. Zobacz Rozwijanie zmiennych, aby uzyskać więcej informacji. Jest to całkiem wygodne, jeśli używasz przenośnego Code::Blocksa

2.13.3 Słowniki

SpellChecker wykorzystuje bibliotekę o nazwie hunspell. Hunspell to narzędzie do sprawdzania pisowni OpenOffice.org, Mozilla Firefox i innych projektów. Słowniki dostępne dla tych aplikacji są kompatybilne z tą wtyczką.

Open Office udostępnia do pobrania zbiór słowników dla kilku języków i dialektów. Rozszerzenia OOo 3.x (*.oxt) to skompresowane archiwa, które można otworzyć za pomocą ulubionego archiwizatora (na przykład 7-Zip lub File Roller). Skopiuj plik .aff i plik .dic do katalogu skonfigurowanego w „Path settings, Dictionaries” (Ustawienia ścieżki, Słowniki) (patrz wyżej).

Jeśli używasz Linuksa, prawdopodobnie masz już zainstalowane kompatybilne słowniki. Zajrzyj do /usr/share/hunspell lub mój wybór to /usr/share/myspell/dicts. Powodem, dla którego lubię pliki myspell, jest to, że zawierają już pliki tezaurusa, które są nazwane poprawnie, aby działały z tezaurusem, a wszystko jest w jednym miejscu. Nie kopiuj tych plików. Po prostu wskaż sprawdzanie pisowni, gdzie znajdują się już pliki.

Rozumiem, że w systemach Windows, Firefox i Thunderbird również instalują się pliki słowników. Można je znaleźć w... `C:\Program Files\Mozilla Firefox\dictionaries` lub `C:\Program Files\Mozilla Thunderbird\dictionaries`. Ponadto zarówno OpenOffice.org, jak i LibreOffice instalują pliki słowników do `C:\Program Files\(Open/Libre)Office\share\extensions\dict-*`.

Przeglądarka Google Chrome również instaluje słowniki, ale są one w formacie .bdic i Wtyczka sprawdzania pisowni Code::Blocks nie będzie z nimi działać.

2.13.4 Pliki tezaurusa

Pliki tezaurusa są również dostępne w OOo, podobnie jak słowniki. Skopiuj pliki tezaurusa (th *.dat i th *.idx) do katalogu skonfigurowanego w „Path settings, Thesauri" (Ustawienia ścieżki, Tezaurus) (patrz wyżej) i zmień ich nazwy, aby pasowały do nazwy słownika, ale dodaj przedrostek „th " i pozwól na rozszerzenie jakie jest.

Przykład: Jeśli pliki słownika (dla jednego języka) to "en GB.aff" i "en GB.dic", to pliki tezaurusa to "th en GB.idx" i "th en GB.dat"

W moim systemie Linux znalazłem już zainstalowane pliki tezaurusa w /usr/share/myspell/dicts i /usr/share/mythes. Ponownie nie przenoś plików. Ustaw sprawdzanie pisowni tak, aby używało plików z ich bieżącej lokalizacji.

W systemie Windows, jeśli jest zainstalowany OpenOffice.org lub LibreOffice, często zawierają one pliki tezaurusa w `C:\Program Files\(Open/Libre)Office\share\extensions\dict-*`.

2.13.5 Bitmapy (flagi)

Bitmapa aktualnie wybranego języka jest pokazana na pasku stanu. Jeśli nie zostanie znaleziona żadna mapa bitowa, wyświetlana jest nazwa języka. Bitmapa musi być obrazem PNG. Wybierz flagę z ikon flag famfamfam i skopiuj ją do katalogu skonfigurowanego w „Path settings, Bitmaps" (Ustawienia ścieżki, mapy bitowe) (patrz wyżej) i zmień jej nazwę, aby pasowała do nazwy słownika, ale pozwól na rozszerzenie png.

2.13.6 Style do sprawdzenia

Sprawdzany jest tylko tekst o określonych stylach (na przykład tylko komentarze i ciągi). Style są automatycznie ustawiane przez Scintilla (komponent edycyjny CodeBlocks).

Plik OnlineSpellChecking.xml zawiera listę z indeksami stylów do sprawdzenia. Indeksy różnią się dla różnych języków programowania, więc plik zawiera listę dla każdego języka programowania. Aby dodać style, poszukaj nazwy języka programowania i indeksów w odpowiednim pliku lexer *.xml i dodaj te informacje do pliku OnlineSpellChecking.xml.

Na przykład, aby sprawdzić pisownię w skryptach powłoki bash (pliki *.sh), dodaj linię:

<Language name="Bash"index="2,5,6"/>

2.14 Eksporter kodu źródłowego (Source Code Exporter)

Często pojawia się konieczność przeniesienia kodu źródłowego do innych aplikacji lub na e-maile. Jeśli tekst zostanie po prostu skopiowany, formatowanie zostanie utracone, co sprawi, że tekst będzie bardzo nieczytelny. Remedium na takie sytuacje jest funkcja eksportu Code::Blocks. Wymagany format pliku eksportu można wybrać za pomocą opcji „File" (Plik) → „Export" (Eksportuj). Program przyjmie wtedy nazwę pliku i katalog docelowy z otwartego pliku źródłowego i zaproponuje je do zapisania pliku eksportu. Odpowiednie rozszerzenie pliku w każdym przypadku będzie określone przez format eksportu. Dostępne są następujące formaty:

html Format tekstowy, który można wyświetlić w przeglądarce internetowej lub w edytorach tekstu.

rtf Format Rich Text Format to format tekstowy, który można otwierać w aplikacjach do przetwarzania tekstu, takich jak Word lub OpenOffice.

odt Format Open Document Text jest standardowym formatem, który został określony przez Sun i O'Reilly. Ten format może być przetwarzany przez Word, OpenOffice i inne aplikacje do przetwarzania tekstu.

pdf Portable Document Format może być otwierany przez aplikacje takie jak Acrobat Reader.

2.15 Obsługa SVN

Uwaga:
To rozszerzenie jest już przestarzałe. Więc prawdopodobnie nie znajdziesz go już w ostatnich wersjach Code::Blocks.

Obsługa systemu kontroli wersji SVN jest zawarta we wtyczce Code::Blocks TortoiseSVN. Za pomocą menu „TortoiseSVN" → „Plugin settings" (Ustawienia wtyczek) możesz skonfigurować dostępne polecenia svn w zakładce „Integration" (Integracja).

Menu Integration Dodaj wpis TortoiseSVN z różnymi ustawieniami na pasku menu.

Project manager (Menadżer projektu) Aktywuj komendy Tortoise SVN w menu kontekstowym menadżera projektu.

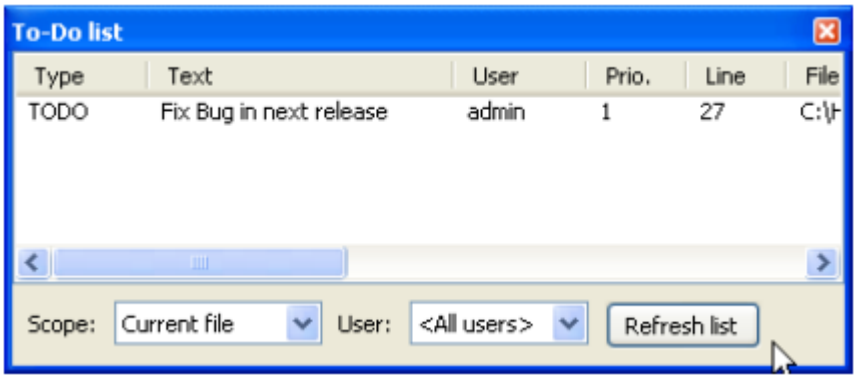
Editor (Edytor) Aktywuj polecenia TortoiseSVN w menu kontekstowym edytora.

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

Uwaga:
Eksplorator plików w Code::Blocks używa różnych nakładek ikon do wskazywania stanu svn. Polecenia TortoiseSVN są zawarte tutaj w menu kontekstowym.

2.16 Lista rzeczy do zrobienia (ToDo List)

W złożonych projektach oprogramowania, w których zaangażowani są różni użytkownicy, często wymagane jest wykonanie różnych zadań przez różnych użytkowników. W tym celu Code::Blocks oferuje listę rzeczy do zrobienia. Listę tę można otworzyć poprzez „View” (Widok) → „ToDo List” (Lista zadań do wykonania) i zawiera ona zadania do wykonania wraz z ich priorytetami, typami i odpowiedzialnymi użytkownikami. Listę można filtrować pod kątem zadań, użytkowników i/lub plików źródłowych. Sortowanie według kolumn można uzyskać, klikając nagłówek odpowiedniej kolumny.



Rysunek 2.20: Wyświetlanie listy rzeczy do zrobienia

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 w okienku wiadomości) z menu „Settings” (Ustawienia) → „Environment” (Środowisko).

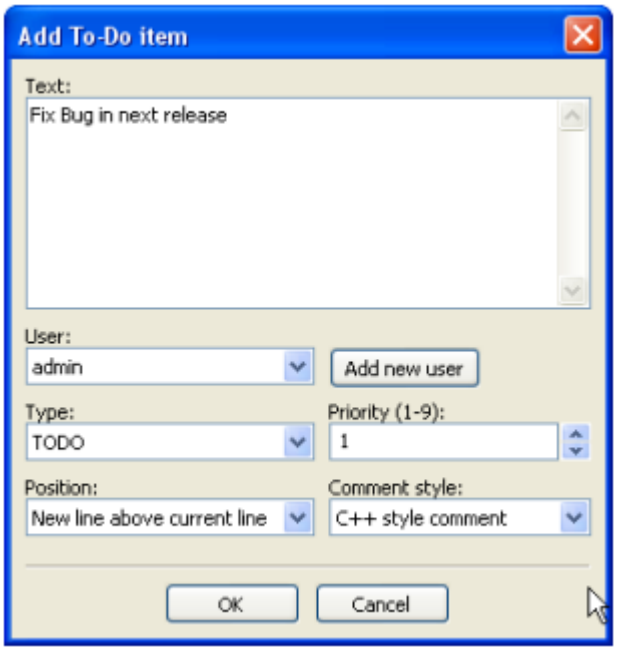
Jeśli źródła są otwarte w Code::Blocks, zadanie można dodać do listy za pomocą polecenia menu kontekstowego „Add To-Do item” (Dodaj element do zrobienia). Komentarz zostanie dodany w wybranej linii kodu źródłowego.

```
// TODO (user #1#): add new dialog for next release
```

Podczas dodawania zadania do wykonania pojawi się okno dialogowe, w którym można dokonać następujących ustawień (patrz Rysunek 2.21 na stronie 65).

User (Użytkownik) Nazwa użytkownika <user> w systemie operacyjnym. W tym miejscu można również tworzyć zadania dla innych użytkowników. W ten sposób odpowiednia nazwa użytkownika musi zostać utworzona przez Add (Dodaj) nowy. Przypisanie Todo odbywa się następnie poprzez wybór wpisów wymienionych dla Użytkownika.

Uwaga:
Zauważ, że Users (Użytkownicy) nie mają nic wspólnego z Personalities (osobowościami) użytymi w Code::Blocks.



Rysunek 2.21: Okno dialogowe dodawania ToDo

Type (Typ) Domyślnie typ jest ustawiony na Todo.

Priority (Priorytet) Ważność zadań można wyrazić za pomocą priorytetów (1 - 9) w Code::Blocks.

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

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

2.17 Narzędzia+ (Tools+)

Tworzenie nowego narzędzia jest dość proste i można je wykonać w kilku prostych krokach. Najpierw otwórz „Tools (+)" (Narzędzia (+)) → „Configure Tools..." (Konfiguruj narzędzia...), aby uzyskać dostęp do okna dialogowego Tools (Narzędzia) zdefiniowane przez użytkownika.

Tool Name (Nazwa narzędzia)

Jest to nazwa, która będzie wyświetlana w menu rozwijanym Tools(+) (Narzędzia(+)). Będzie również wyświetlana jako nazwa zakładki dla narzędzi, które przekierowują do okna danych wyjściowych Tools (Narzędzia).

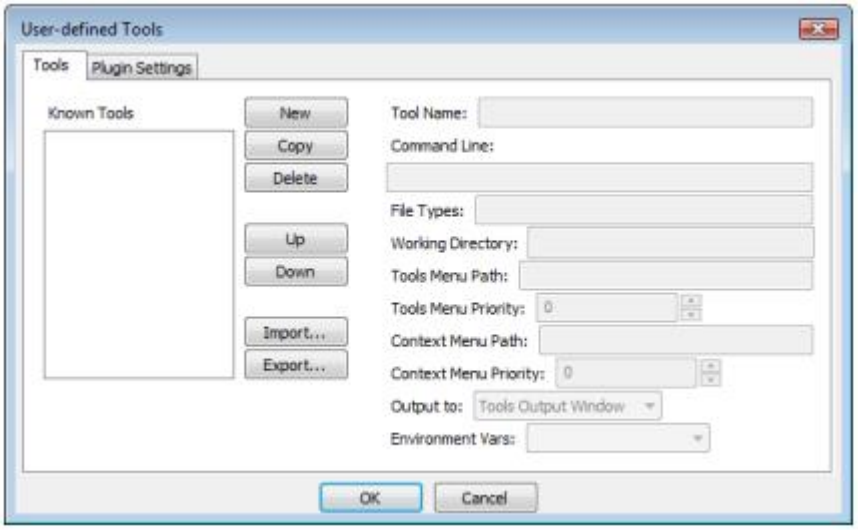
Command Line (Wiersz poleceń)

W tym miejscu można umieścić dowolną prawidłową funkcję wiersza poleceń i przełączniki. Akceptowane jest również zastępowanie zmiennych. Poniższa lista zawiera bardziej przydatne zmienne (pełna lista znajduje się w sekcji 3.2 na stronie 79).

`$relfile, $file` Odpowiednio względna i bezwzględna nazwa wybranego pliku.

`$reldir, $dir` Odpowiednio względna i bezwzględna nazwa wybranego katalogu.

`$relpath, $path` Względna i bezwzględna nazwa wybranego pliku lub katalogu.



Rysunek 2.22: Okno dialogowe Narzędzia zdefiniowane przez użytkownika

`$mpaths` Lista wybranych plików lub katalogów (tylko ścieżki bezwzględne).

`$fname, $fext` Nazwa bez rozszerzenia i rozszerzenie bez nazwy wybranego pliku.

`$inputstr{prompt}` Prosi użytkownika o wprowadzenie ciągu tekstowego, który zostanie podstawiony w wierszu poleceń.

`$if(condition){true clause}{false clause}`
Rozwiązuje klauzulę false, jeśli warunek jest pusty, 0 lub fałszywy; inaczej true (prawdziwa) klauzula.

File Types (Typy plików)

Wyrażenia wieloznaczne oddzielone średnikami ograniczą populację menu prawego przycisku myszy pliku, katalogu lub wielu ścieżek w drzewie projektu, Eksploratorze plików lub okienku edytora do określonych typów. Pozostaw puste, aby obsłużyć wszystkie typy plików/katalogów.

Working Directory (Katalog roboczy)

Katalog, z którego wykonywane jest polecenie. Dostępne są zmienne Code::Blocks, zmienne projektu i zmienne globalne. Również,

1. Jeśli w wierszu poleceń podano $dir, to $dir może być również tutaj użyty.

2. $parentdir jest dostępny dla $relfile , $file , $reldir, $dir, $relpath, $path, $fname, $fext, przeliczając na bezwzględną ścieżkę katalogu zawierającego element.

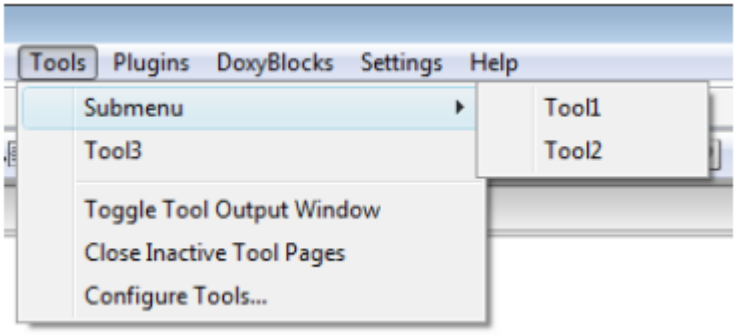
Tools Menu Path (Ścieżka menu narzędzi)

Kontroluje umieszczenie polecenia w menu Tools (+) (Narzędzia (+)), dając możliwość dodania podmenu (dozwolonych jest wiele poziomów).

- Submenu/Tool1 ˆ
- Submenu/Tool2

- Tool3

Stworzy tę strukturę

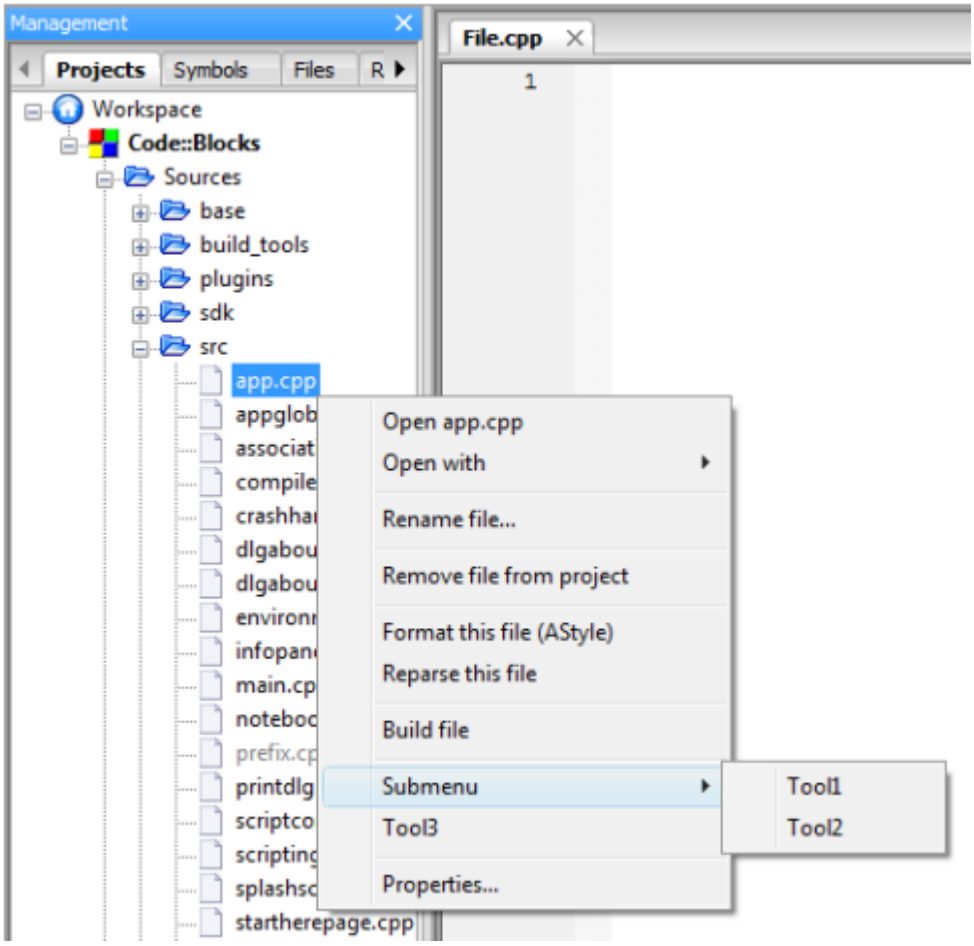


Rysunek 2.23: Struktura menu Narzędzia

Nazwa polecenia zostanie użyta, jeśli ten wpis jest pusty. Jeśli pierwszym znakiem jest kropka, polecenie zostanie ukryte.

Context Menu Path (Ścieżka menu kontekstowego)

Kontroluje to umieszczenie polecenia w menu prawym przyciskiem myszy na kartach Projects (Projekty) i Files (Pliki) panelu Management (Zarządzanie). Obowiązują tutaj te same zasady struktury ze ścieżką menu narzędzi.



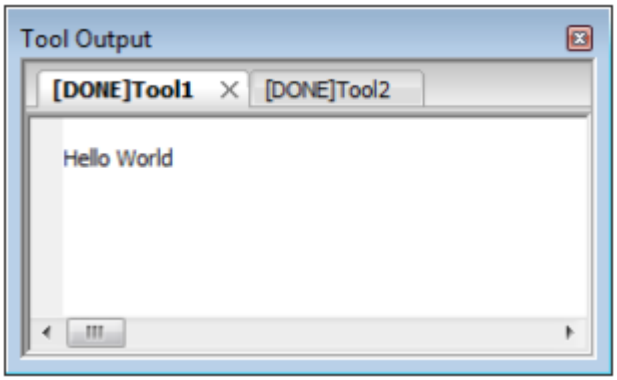
Rysunek 2.24: Struktura menu kontekstowego

Pamiętaj, że polecenie nie pojawi się w menu kontekstowym, chyba że wiersz poleceń zawiera jedno lub więcej z poniższych: $relfile , $file , $reldir, $dir, $relpath, $path, $fname i $fext.

Output to (Wyjście do)

Określa to, dokąd zostaną przekierowane dane wyjściowe polecenia. Cel i funkcja polecenia określą, który z nich najlepiej wybrać. **Tools Output Window** (Okno danych wyjściowych narzędzi).

Narzędzia, które wyświetlają tylko polecenie wyników (i nie wymagają wprowadzania danych) zazwyczaj używają tego ustawienia. Program zostanie uruchomiony w sposób niewidoczny, a wszelkie dane wyjściowe zostaną przekierowane do odpowiedniej zakładki okna wyników narzędzi. Tekst [DONE] (GOTOWE) zostanie dodany po zakończeniu pracy narzędzia.



Rysunek 2.25: Okno danych wyjściowych narzędzia

Uwaga:
Jeśli okno danych wyjściowych narzędzi jest otwarte, gdy Code::Blocks jest zamknięte, może to spowodować awarię Code::Blocks.

Konsola Code::Blocks (Console Code::Blocks)

Spowoduje to uruchomienie programu przez wykonywalny program uruchamiający konsoli `cb_console_runner` (ten sam program, który jest uruchamiany po kompilacji i uruchomieniu). Jest on zwykle używany w przypadku narzędzi wiersza poleceń z bardziej zaawansowanymi interakcjami z użytkownikiem, chociaż można również używać programów z graficznym interfejsem użytkownika (zwłaszcza jeśli program jest niestabilny i/lub pozostawia komunikaty na standardowym wyjściu). Program uruchamiający konsolę zatrzyma okno (uniemożliwi jego zamknięcie), wyświetli czas działania i kod wyjścia po zakończeniu programu.

Standard Shell (Powłoka standardowa)

Jest to to samo, co umieszczenie polecenia w skrypcie wsadowym lub powłoki, a następnie jego uruchomienie. Program będzie działał w dowolnej domyślnej metodzie, a po zakończeniu jego okno zostanie zamknięte. To ustawienie jest przydatne do uruchamiania programu (na przykład pliku lub przeglądarki internetowej), który musi pozostać otwarty po zamknięciu Code::Blocks.

Uwaga:
Ponieważ wtyczka Tools+ nie jest jeszcze gotowa, niektóre funkcje, w szczególności priorytet menu i zmienne środowiskowe — nie są dostępne.

2.17.1 Przykładowe narzędzia

Otwórz przeglądarkę plików do wybranego pliku

- Windows Explorer
  - Tools Menu

```
explorer /select,"$(PROJECTFILE)"
```

  - Context Menu

```
explorer /select,"$path"
```

- Dolphin
  - Tools Menu

```
dolphin --select "$(PROJECTFILE)"
```

  - Context Menu

```
dolphin --select "$path"
```

Uwaga:
Poniższe trzy polecenia menu kontekstowego obsługują tylko foldery (ale nie pliki).

- Nautilus
  - Tools Menu

```
nautilus --no-desktop --browser "$(PROJECTDIR)"
```

  - Context Menu

```
nautilus --no-desktop --browser "$dir"
```

- Thunar
  - Tools Menu

```
thunar "$(PROJECTDIR)"
```

  - Context Menu

```
thunar "$dir"
```

- PCMan File Manager
  - Tools Menu

```
pcmanfm "$(PROJECTDIR)"
```

  - Context Menu

```
pcmanfm "$dir"
```

Zaktualizuj katalog Subversion

- Windows
  - Tools Menu

```
"path_to_svn\bin\svn" update "$inputstr{Directory}"
```

  - Context Menu

```
"path_to_svn\bin\svn" update "$dir"
```

- Linux
  - Tools Menu

```
svn update "$inputstr{Directory}"
```

  - Context Menu

```
svn update "$dir"
```

Eksportuj plik makefile

Uwaga:
używa narzędzia wiersza poleceń cbp2make.

- Windows
  - Tools Menu

```
"path_to_cbp2make\cbp2make" -in "$(PROJECTFILE)"^
```

- Linux
  - Tools Menu

```
"path_to_cbp2make/cbp2make" -in "$(PROJECTFILE)"
```

Skompresuj aktywny projekt do archiwum

- Windows

– 7z or zip - Tools Menu (w jednej linii)

```
"path_to_7z\7z" a -t$if(zip == $inputstr{7z or zip?}){zip -
mm=Deflate -mmt=on -mx9 -mfb=128 -mpass=10}{7z -m0=LZMA -mx9-
md=64m -mfb=64 -ms=on} -sccUTF-8 "-w$(PROJECTDIR).."
"$(PROJECTDIR)..\$(PROJECT_NAME)" "$(PROJECTDIR)*"
```

– tar.gz or tar.bz2 - Tools Menu (w jednej linii)

```
cmd /c ""path_to_7z\7z" a -ttar -mx0 -sccUTF-8 "-
w$(PROJECTDIR).." "$(PROJECTDIR)..\$(PROJECT_NAME)"
"$(PROJECTDIR)*" && "path_to_7z\7z" a -t$if(gz == $inputstr{gz
or bz2?}){gzip -mx9 -mfb=128 -mpass=10 -sccUTF-8 "-
w$(PROJECTDIR).."
"$(PROJECTDIR)..\$(PROJECT_NAME).tar.gz}{bzip2 -mmt=on -mx9 -
md=900k -mpass=7 -sccUTF-8 "-w$(PROJECTDIR).."
"$(PROJECTDIR)..\$(PROJECT_NAME).tar.bz2}"
"$(PROJECTDIR)..\$(PROJECT_NAME).tar" && cmd /c del
"$(PROJECTDIR)..\$(PROJECT_NAME).tar""
```

Uwaga:
Interpreter wiersza poleceń systemu Windows został wywołany bezpośrednio tutaj (cmd /c), umożliwiając łączenie wielu poleceń w jednym wierszu. Jednak powoduje to niepowodzenie wykonania polecenia w konsoli Code::Blocks.

- Linux

  – 7z or zip - Tools Menu

```
7z a -t$if(zip == $inputstr{7z or zip?}){zip -mm=Deflate -
mmt=on -mx9 -mfb=128 - mpass=10}{7z -m0=LZMA -mx9 -md=64m -
mfb=64 -ms=on} -sccUTF-8 "-w$(PROJECTDIR).."
"$(PROJECTDIR)../$(PROJECT_NAME)" "$(PROJECTDIR)*"
```

  – tar.gz or tar.bz2 - Tools Menu

```
tar -cf "$(PROJECTDIR)../$(PROJECT_NAME).tar.$if(gz ==
$inputstr{gz or bz2?}){gz" -I 'gzip}{bz2" -I 'bzip2} -9'
"$(PROJECTDIR)*"
```

2.18 Wyszukiwanie wątków (Thread Search)

Za pomocą menu „Search” (Wyszukaj) → „Thread Search” (Wyszukaj wątki) odpowiednią wtyczkę można wyświetlić lub ukryć jako zakładkę w konsoli wiadomości. W Code::Blocks można wyświetlić podgląd występowania 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 Code::Blocks.

Uwaga:
Zakres rozszerzeń plików, które mają być uwzględnione w wyszukiwaniu, jest wstępnie ustawiony i może wymagać dostosowania.

2.18.1 Funkcje

Wtyczka ThreadSearch oferuje następujące funkcje:

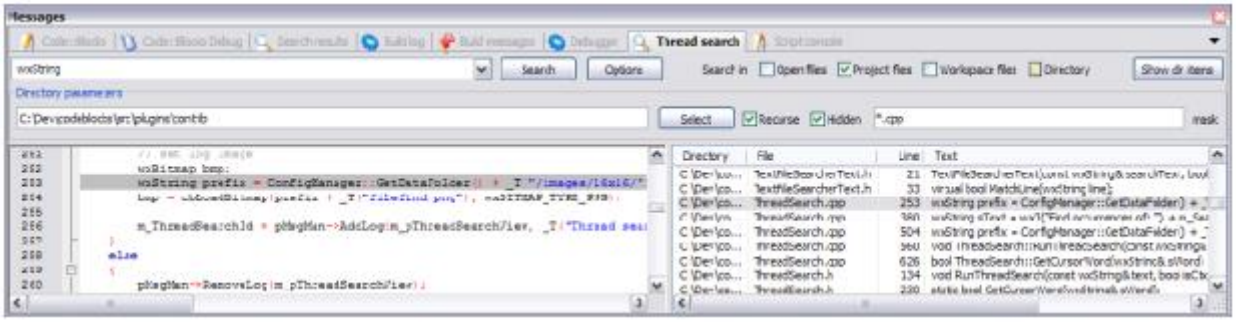
- Wielowątkowe „Search in files” (Wyszukiwanie w plikach)
- Wewnętrzny edytor tylko do odczytu do podglądu wyników
- Plik otwarty w notatniku edytorów
- Wysoce konfigurowalny
- Menu kontekstowe „Find occurrences” (Znajdź wystąpienia), aby rozpocząć wyszukiwanie w plikach ze słowem pod kursorem.

2.18.2 Użytkowanie

1. Skonfiguruj preferencje wyszukiwania (patrz Rysunek 2.27 na stronie 74)

   Po zainstalowaniu wtyczki istnieją 4 sposoby na uruchomienie wyszukiwania:

   a) Wpisz/Wybierz słowo w polu kombi wyszukiwania i naciśnij klawisz Enter lub kliknij Search (Szukaj) w panelu wyszukiwania wątków w notatniku Messages (Wiadomości).



Rysunek 2.26: Panel wyszukiwania wątków

b) Wpisz/Wybierz słowo w polu kombi 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/Thread search, aby znaleźć aktualne słowo w aktywnym edytorze.

Uwaga:
Pozycje 1, 2 i 3 mogą nie być dostępne w obecnej konfiguracji.

2. Kliknij ponownie przycisk wyszukiwania, aby anulować bieżące wyszukiwanie.

3. Pojedyncze kliknięcie elementu wyniku wyświetla go w edytorze podglądu we właściwym miejscu.

4. Dwukrotne kliknięcie elementu wyniku otwiera lub ustawia edytor w notatniku edytora we właściwym miejscu.

2.18.3 Konfiguracja

Aby uzyskać dostęp do panelu konfiguracji wtyczki ThreadSearch, kliknij (patrz Rysunek 2.27 na stronie 74):

1. Przycisk Options (Opcje) w notatniku Messages (Wiadomości) panelu wyszukiwania wątków.

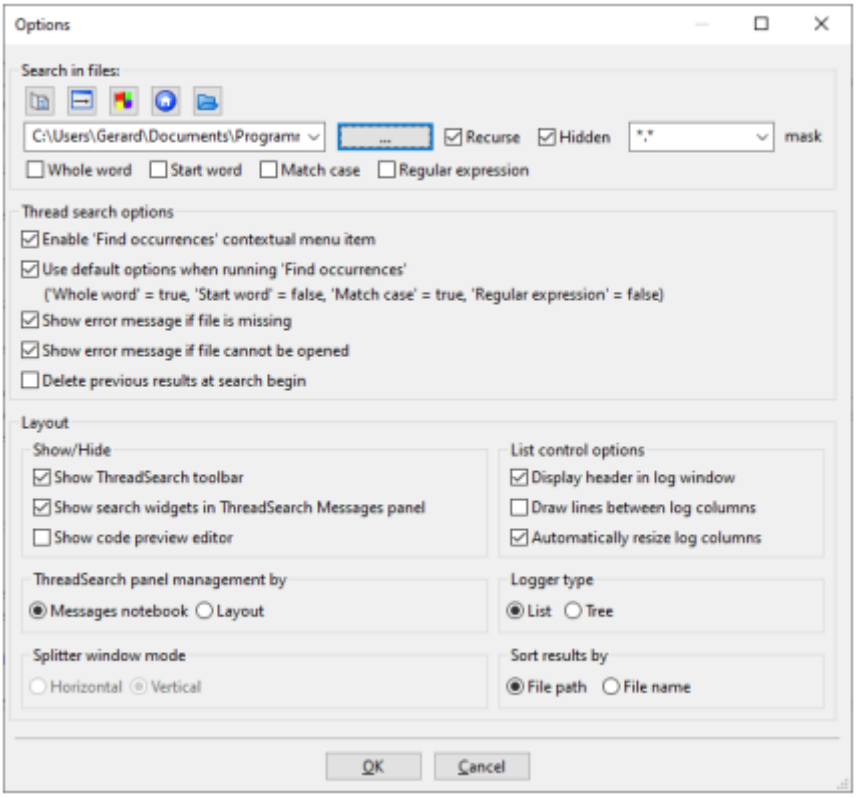
2. Przycisk Options (Opcje) na pasku narzędzi wyszukiwania wątków.

3. Element menu Settings/Environment (Ustawienia/Środowisko), a następnie element Thread search (Wyszukiwanie wątków) w lewej kolumnie.

Uwaga:
Pozycje 1, 2 i 3 mogą nie być dostępne w obecnej konfiguracji.

Search (Szukaj) w części, określa zestaw plików, które będą analizowane.

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



Rysunek 2.27: Konfiguracja wyszukiwania wątków

2.18.4 Opcje (Options)

Whole word (Całe słowo), jeśli zaznaczone, wiersz pasuje do wyrażenia wyszukiwania, jeśli zostanie znalezione wyrażenie wyszukiwania bez alfanumerycznego +'_' przed i po.

Start word (Słowo początkowe), jeśli zaznaczone, wiersz pasuje do wyrażenia wyszukiwania, jeśli wyrażenie wyszukiwania znajduje się na początku słowa, tj. bez 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, będziesz musiał ustawić opcję „Use Advanced RegEx searches” (Użyj zaawansowanego wyszukiwania RegEx) w menu „Settings” (Ustawienia) → „Editor” (Edytor) → „General Settings” (Ustawienia ogólne).

2.18.5 Opcje wyszukiwania wątków (Thread search options)

Enable „Find occurrences contextual menu item” (Włącz opcję „Znajdź elementy menu kontekstowego wystąpień”) Jeśli ta opcja jest zaznaczona, do menu kontekstowego edytora dodawane jest Find occurrences of 'Focused word' entry (Znajdź wystąpienia wpisu „Słowo skupione”).

Use default options when running 'Find occurrences' (Użyj opcji domyślnych podczas uruchamiania „Znajdź wystąpienia”) ' Jeśli ta opcja jest zaznaczona, do wyszukiwań uruchamianych za pomocą pozycji menu kontekstowego „Find occurrences” (Znajdź wystąpienia) stosowany jest zestaw opcji domyślnych. Domyślne opcje „Whole word” (Całe słowo) i „Match case” (Dopasuj wielkość liter) są włączone.

Delete previous results at search begin (Usuń poprzednie wyniki na początku wyszukiwania) Jeśli ThreadSearch jest skonfigurowany z „Tree View” (Widokiem drzewa), wyniki wyszukiwania będą wyświetlane hierarchicznie.

- pierwszy węzeł zawiera wyszukiwane hasło
- powyżej wymienione są pliki zawierające wyszukiwane hasło
- na tej liście wyświetlany jest numer wiersza i odpowiadająca mu treść zdarzenia

Jeśli przeszukujesz różne terminy, lista stanie się myląca, dlatego poprzednie wyniki wyszukiwania, mogą zostać wyczyszczone przy rozpoczęciu wyszukiwania, za pomocą tej opcji.

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

2.18.6 Układ (Layout)

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

Draw lines between columns (Rysuj linie między kolumnami) Rysuje linie między kolumnami w trybie listy (List mode).

Show ThreadSearch toolbar (Pokaż pasek narzędzi wyszukiwania wątków) Wyświetl pasek narzędzi wtyczki wyszukiwania wątków.

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 (oszczędza miejsce).

Show code preview editor (Pokaż edytor podglądu kodu) Podgląd kodu można ukryć za pomocą tego pola wyboru lub dwukrotnym kliknięciem środkowej krawędzi okna rozdzielacza. Tutaj można go ponownie pokazać.

2.18.7 Panel Zarządzanie (Panel Management)

Możesz wybrać różne tryby zarządzania oknem ThreadSearch. Po ustawieniu „Message Notebook” 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.

2.18.8 Typ rejestratora (Logger Type)

Widok wyników wyszukiwania można wyświetlać na różne sposoby. Ustawienie „List” (Lista) wyświetla wszystkie wystąpienia jako listę. Drugi tryb „Tree” (Drzewo” gromadzi wszystkie wystąpienia w pliku jako węzeł.

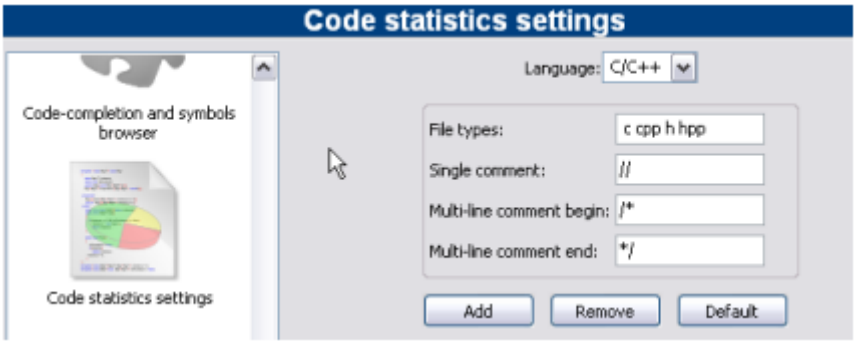
2.18.9 Okno Trybu rozdzielacza (Splitter Window Mode)

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

2.18.10 Sortuj wyniki wyszukiwania (Sort results by)

Widok wyników wyszukiwania może być posortowany według ścieżki lub nazwy pliku.

2.19 Statystyki kodu (Code statistics)



Rysunek 2.28: 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” (Statystyka kodu).

2.20 Profiler kodu (Code profiler)

Prosty interfejs graficzny do GNU GProf Profiler.

2.21 Wtyczka do importowania projektów (ProjectsImporter)

ProjectsImporter importuje zagraniczne projekty i obszary robocze z Dev-C++, MSVC6, MSVC7 i MSVC8 do użytku jako projekt Code::Blocks.

2.22 Wyszukiwanie dostępnego kodu źródłowego

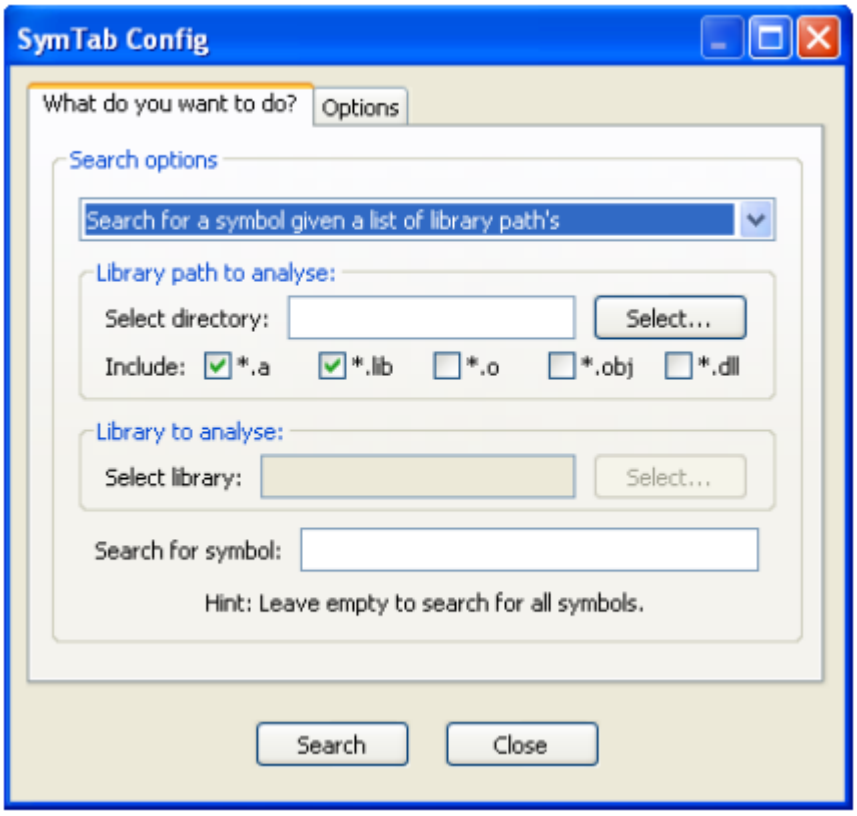
Ta wtyczka umożliwia wybranie terminu w edytorze i wyszukanie go za pomocą menu kontekstowego „Search at Koders” (Szukaj w Koders) w bazie danych [?]. Okno dialogowe oferuje dodatkowe możliwości filtrowania według języków programu i licencji.

Uwaga:
Wydaje się, że Koders i jego następca, BlackDuck, zniknęli lub zmienili swoją stronę internetową! Więc ta wtyczka już nie działa. Czekam na aktualizację...

To przeszukiwanie bazy danych pomoże Ci znaleźć kod źródłowy pochodzący z innych światowych projektów uniwersytetów, konsorcjów i organizacji, takich jak Apache, Mozilla, Novell Forge, SourceForge i wiele innych, który można ponownie wykorzystać bez konieczności ponownego wymyślania koła za każdym razem . Proszę zwrócić uwagę na licencję kodu źródłowego w każdym indywidualnym przypadku.

2.23 Wtyczka tabeli symboli (Symbol Table Plugin)

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



Rysunek 2.29: Konfiguracja tabeli symboli

Kliknięcie przycisku „Szukaj” powoduje wyświetlenie wyników wyszukiwania programu NM w osobnym oknie o nazwie „SymTabs Result”. Nazwy obiektów lub bibliotek zawierających symbol znajduje się pod tytułem „NM’s Output” (Wynik NM).

3 Rozszerzenie zmiennej

Code::Blocks rozróżnia kilka typów zmiennych. Te typy służą do konfiguracji środowiska do tworzenia programu, a jednocześnie do poprawy łatwości konserwacji i przenośności. Dostęp do zmiennych Code::Blocks uzyskuje się poprzez `$<nazwa>`.

Environment Variable (Zmienne środowiskowe) są ustawiane podczas uruchamiania Code::Blocks. Mogą modyfikować systemowe zmienne środowiskowe, takie jak PATH. Może to być przydatne w przypadkach, gdy do tworzenia projektów niezbędne jest określone środowisko. Ustawienia zmiennych środowiskowych w Code::Blocks dokonuje się w „Settings" (Ustawienia) → „Environment" (Środowisko) → „Environment Variables" (Zmienne środowiskowe).

Builtin Variables (Zmienne wbudowane) są predefiniowane w Code::Blocks i można uzyskać do nich dostęp poprzez ich nazwy (szczegóły w sekcji 3.2 na stronie 79).

Command Macros (Makra poleceń) Ten typ zmiennych służy do sterowania procesem budowania. Więcej informacji znajduje się w rozdziale 3.4 na stronie 84.

Custom Variables (Zmienne niestandardowe) to zmienne zdefiniowane przez użytkownika, które można określić w opcjach budowania projektu. Tutaj możesz na przykład zdefiniować swoją pochodną jako zmienną MCU i przypisać jej odpowiednią wartość. Następnie ustaw opcję kompilatora -mcpu=$(MCU) i Code::Blocks automatycznie zastąpi zawartość. Dzięki tej metodzie można dodatkowo sparametryzować ustawienia projektu.

Global Variables (Zmienne globalne) są używane głównie do tworzenia Code::Blocks ze źródeł lub rozwoju aplikacji wxWidgets. Te zmienne mają bardzo szczególne znaczenie. W przeciwieństwie do wszystkich innych, jeśli ustawisz takie zmienne i udostępnisz swój plik projektu innym, którzy *nie* skonfigurowali, ten GV Code::Blocks poprosi użytkownika o ustawienie zmiennej. Jest to bardzo łatwy sposób, aby upewnić się, że „inny programista" wie, co łatwo skonfigurować. Code::Blocks zapyta o wszystkie potrzebne ścieżki.

3.1 Składnia

Code::Blocks traktuje jako zmienne następujące funkcjonalnie identyczne sekwencje znaków w krokach przed kompilacją, po kompilacji lub kompilacji:

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

Nazwy zmiennych muszą składać się ze znaków alfanumerycznych i nie jest w nich rozróżniana wielkość liter. Zmienne zaczynające się pojedynczym znakiem hash (#) są interpretowane jako globalne zmienne użytkownika (szczegóły w sekcji 3.7 na stronie 84). 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 zawiedzie.

Użycie tych zmiennych może wyglądać następująco dla daty:

```
#include "include/manager.h"
wxString strdate = Manager::Get()=>GetMacrosManager()=>ReplaceMacros(_T("$TODAY"));
```

Uwaga:
Definicje na cele mają pierwszeństwo przed definicjami na projekty.

3.2 Lista dostępnych funkcji wbudowanych

Wymienione tutaj zmienne są wbudowanymi zmiennymi Code::Blocks. Nie można ich używać w plikach źródłowych.

3.2.1 Code::Blocks obszar roboczy

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

`$(WORKSPACENAME)`, `$(WORKSPACE_NAME)`
Nazwa obszaru roboczego wyświetlana w zakładce Projects (Projekty) panelu Management (Zarządzanie).

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

3.2.2 Pliki i katalogi

`$(PROJECT_FILENAME)`, `$(PROJECT_FILE_NAME)`, `$(PROJECT_FILE)`, `$(PROJECTFILE)` Nazwa pliku aktualnie skompilowanego projektu.

`$(PROJECT_NAME)`, `$(PROJECTNAME)` Nazwa aktualnie kompilowanego projektu.

`$(PROJECT_DIR)`, `$(PROJECTDIR)`, `$(PROJECT_DIRECTORY)`, `$(PROJECTDIRECTORY)` Wspólny katalog najwyższego poziomu aktualnie skompilowanego projektu.

`$(ACTIVE_EDITOR_FILENAME)` Nazwa pliku otwartego w aktualnie aktywnym edytorze.

`$(ACTIVE_EDITOR_LINE)` Zwróć bieżącą linię 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)` Podstawowa nazwa (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 działającego wystąpienia Code::Blocks.

`$(DATAPATH), $(DATA_PATH), $(DATA-PATH)` „Wspólny" katalog aktualnie działającej instancji Code::Blocks.

`$(PLUGINS)` Katalog wtyczek aktualnie działającej instancji Code::Blocks.

`$(TARGET_COMPILER_DIR)` Katalog instalacyjny kompilatora tzw. ścieżka główna.

3.2.3 Budowanie celów

zamień FOOBAR na nazwę docelową

`$(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.

`$(FOOBAR_PARAMETERS)` Parametry wykonania określonego celu.

`$(TARGET_OUTPUT_DIR)` Katalog wyjściowy bieżącego celu.

`$(TARGET_OBJECT_DIR)` Katalog obiektów bieżącego celu.

`$(TARGET_NAME)` Nazwa aktualnego 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 kompilacji (kompilator, linker itp.) bieżącego celu.

`$(TARGET_COMPILER_DIR)` Katalog narzędzi do budowania bieżącego celu (kompilator, linker itp.).

3.2.4 Język i kodowanie

`$(LANGUAGE)`	Język systemu w prostym języku.
`$(ENCODING)`	Kodowanie znaków w prostym języku.

3.2.5 Godzina i data

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

3.2.6 Zależność od platformy

`$(PLATFORM)`	Rozszerza się do msw na Windows i Unix na Linux i Mac (od wersji r11793)

3.2.7 Rozszerzenia wiersza poleceń

Te polecenia mogą być używane w wierszu poleceń dla określonej platformy.

`$(CMD_CP)`	Rozwinie się do polecenia kopiowania dla określonej platformy. (na Windows copy i Unix cp - -preserve=timestamps)
`$(CMD_RM)`	usuń polecenie
`$(CMD_MV)`	polecenie przesunięcia
`$(CMD_NULL)`	Urządzenie NULL (do przekierowywania strumieni)
`$(CMD_MKDIR)`	utwórz katalog
`$(CMD_RMDIR)`	usuń katalog

3.2.8 Wartości losowe

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

3.2.9 Ścieżka standardowa

$(GET_DATA_DIR)	Unix: prefiks/udział/nazwa aplikacji ; Windows: ścieżka EXE.
$(GET_LOCAL_DATA_DIR)	Unix: /etc/nazwa_aplikacji ; Windows: ścieżka EXE.
$(GET_CONFIG_DIR)	Unix: /etc ; Windows: C:\Documents and Settings\All Users\Dane aplikacji
$(GET_USER_CONFIG_DIR)	Unix: ; Windows: C:\Documents and Settings\nazwa użytkownika\Dane aplikacji\nazwa aplikacji
$(GET_USER_DATA_DIR)	Unix: /.nazwaaplikacji ; Windows: C:\Documents and Settings\nazwa użytkownika\Dane aplikacji
$(GET_USER_LOCAL_DATA_DIR)	Unix: /.nazwaaplikacji ; Windows: C:\Documents and Settings\nazwa użytkownika\Ustawienia lokalne\Dane aplikacji\nazwa aplikacji
$(GET_TEMP_DIR)	WSZYSTKIE platformy: Zapisywalny, tymczasowy katalog.

3.2.10 Wbudowane funkcje do konwersji ścieżek

Istnieją wbudowane funkcje makr, aby uprościć generowanie ścieżek

$TO_UNIX_PATH{}	przekonwertuj ścieżkę na ścieżkę unixową (użyj „/” jako separatora)
$TO_WINDOWS_PATH{}	przekonwertuj ścieżkę do windows (użyj '\' jako separatora)
$TO_NATIVE_PATH{}	przekonwertuj na ścieżkę natywną, na której działa instancja codeblocks

Stosowanie

$TO_UNIX_PATH{$(TARGET_OUTPUT_FILE)}	zwraca bieżący docelowy plik wyjściowy jako ścieżkę unix.

3.2.11 Ocena warunkowa

$ if (warunek){prawdziwa klauzula}{fałszywa klauzula}

Ocena warunkowa rozstrzygnie swoją prawdziwą klauzulę, jeśli:

- warunek jest niepustą sekwencją znaków inną niż 0 lub fałsz
- warunek to niepusta zmienna, której nie można rozwiązać na 0 lub fałsz
- warunek to zmienna, której wynikiem jest prawda (niejawna przez poprzedni warunek)

Ocena warunkowa rozwiąże swoją fałszywą klauzulę, jeśli:

- warunek jest pusty
- warunek to 0 lub fałsz
- warunek to zmienna, która jest pusta lub ma wartość 0 lub fałsz

Uwaga:
Należy pamiętać, że ta konstrukcja nie obsługuje wariantów składni zmiennych %if (...) ani $(if)(...).

Przykład

Na przykład, jeśli używasz kilku platform i chcesz ustawić różne parametry w zależności od systemu operacyjnego. W poniższym kodzie polecenia skryptu [[ ]] są oceniane i wykonywane jest <polecenie>. Może to być przydatne na etapie post-kompilacji (w jednej linii).

[[if (PLATFORM == PLATFORM_MSW) {print(_T(”cmd /c”)) ; } else { print (_T(”sh”)) ; }]]
<polecenie>

3.3 Rozszerzanie skryptów

Aby uzyskać maksymalną elastyczność, możesz osadzać skrypty przy użyciu operatora [[ ]] jako szczególnego przypadku rozwijania zmiennych. Osadzone skrypty mają dostęp do wszystkich standardowych funkcji dostępnych dla skryptów i działają podobnie jak cudzysłowy apostrofowe (z wyjątkiem dostępu do przestrzeni nazw Code::Blocks). Jako takie, skrypty nie ograniczają się do generowania tekstu wyjściowego, ale mogą również manipulować stanem Code::Blocks (projekty, cele itp.).

Uwaga:
Manipulowanie stanem Code::Blocks powinno być zaimplementowane raczej za pomocą skryptu przed kompilacją niż za pomocą skryptu.

Przykład z cudzysłowem apostrofowym

objdump -D ‘find. -nazwa*. elf‘ > nazwa.dis

Wyrażenie w apostrofach 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 nazwa.dis. Dzięki temu procesy można zautomatyzować w prosty sposób bez konieczności programowania jakichkolwiek pętli.

Przykład z użyciem skryptu

Tekst skryptu jest zastępowany dowolnymi danymi wyjściowymi wygenerowanymi przez skrypt lub odrzucany w przypadku błędu składni.

Ponieważ ocena warunkowa jest uruchamiana przed rozszerzeniem skryptów, ocena warunkowa może być używana do funkcji preprocesora. Zmienne wbudowane (i zmienne użytkownika) są rozwijane po skryptach, dzięki czemu możliwe jest odwoływanie się do zmiennych w wyniku skryptu.

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

wstawia tytuł aktywnego projektu do wiersza poleceń.

3.4 Makra poleceń

`$compiler`	Dostęp do nazwy pliku wykonywalnego kompilatora.
`$linker`	Dostęp do nazwy pliku wykonywalnego konsolidatora.
`$options`	Flagi kompilatora
`$link_options`	Flagi łączące
`$includes`	Zawarte ścieżki kompilatora
`$c`	Zawarte ścieżki linkera
`$libs`	Biblioteki linkera
`$file`	Plik źródłowy (pełna nazwa)
`$file_dir`	Katalog plików źródłowych bez nazwy pliku i rozszerzenia nazwy pliku.
`$file_name`	Nazwa pliku źródłowego bez informacji o ścieżce i rozszerzenia nazwy pliku.
`$exe_dir`	Katalog plików wykonywalnych bez nazwy pliku i rozszerzenia nazwy pliku.
`$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
`$exe_output`	Wykonywalny plik wyjściowy
`$objects_output_dir`	Katalog wyjściowy obiektów

3.5 Kompilacja pojedynczego pliku

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

3.6 Połącz pliki obiektowe z plikiem wykonywalnym

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

3.7 Globalne zmienne kompilatora

W tej sekcji opisano, jak pracować ze zmiennymi globalnymi.

3.7.1 Streszczenie

Praca jako programista nad projektem, który opiera się na bibliotekach innych firm, wiąże się z wieloma niepotrzebnymi, powtarzającymi się zadaniami, takimi jak konfigurowanie zmiennych kompilacji zgodnie z lokalnym układem systemu plików. W przypadku plików projektu należy zachować ostrożność, aby uniknąć przypadkowego zatwierdzenia lokalnie zmodyfikowanej kopii. Jeśli ktoś nie zwraca uwagi, może się to zdarzyć łatwo, na przykład po zmianie flagi kompilacji, aby utworzyć kompilację (release) wydania.

Koncepcja globalnych zmiennych kompilatora jest unikalnym nowym rozwiązaniem dla Code::Blocks, które rozwiązuje ten problem. Globalne zmienne kompilatora umożliwiają jednorazowe skonfigurowanie projektu, a dowolna liczba programistów korzystających z dowolnej liczby różnych układów systemu plików jest w stanie skompilować i opracować ten projekt. Żadne informacje o lokalnym układzie nie muszą być zmieniane więcej niż raz.

3.7.2 Nazwiska i członkowie

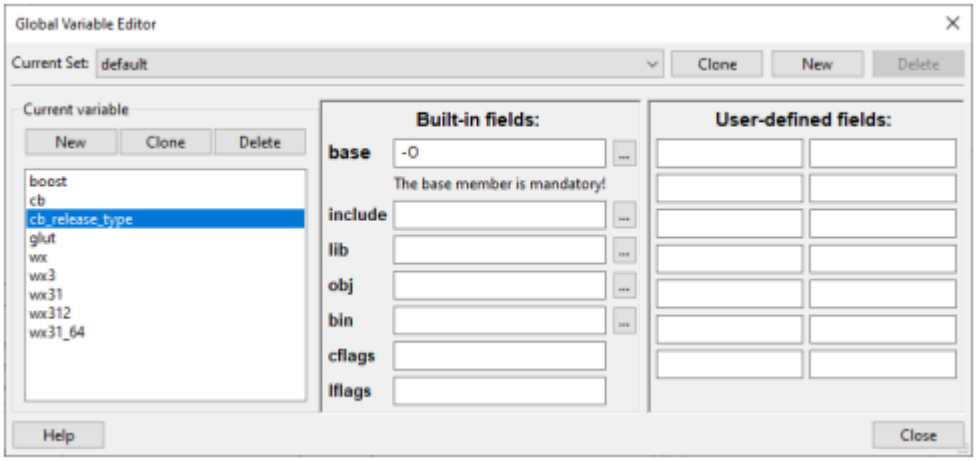
Globalne zmienne kompilatora w Code::Blocks są odróżniane od zmiennych w projekcie przez wiodący znak hasz (#). Globalne zmienne kompilatora mają strukturę; każda zmienna składa się z nazwy i opcjonalnego elementu członkowskiego. Nazwy można dowolnie definiować, a niektórzy członkowie są wbudowani w IDE. Chociaż w zasadzie możesz wybrać dowolną nazwę zmiennej, zaleca się wybranie znanego identyfikatora dla popularnych pakietów. W ten sposób minimalizowana jest ilość informacji, które użytkownik musi podać. Zespół Code::Blocks udostępnia listę zalecanych zmiennych dla znanych pakietów.

Podstawa elementu członkowskiego jest rozpoznawana na taką samą wartość, jakiej używa nazwa zmiennej bez elementu członkowskiego (aliasu).

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

Ogólnie zaleca się używanie składni $(#variable.include) zamiast $(#variable)/include, ponieważ zapewnia ona dodatkową elastyczność i poza tym jest dokładnie identyczna pod względem funkcjonalności (patrz podrozdział 3.7.6 na stronie 88 i Rysunek 3.1 na strona 86, aby uzyskać szczegółowe informacje).

Elementy cflags i lflags są domyślnie puste i mogą być używane do dostarczania tego samego spójnego zestawu flag kompilatora/konsolidatora do wszystkich kompilacji na jednej maszynie. Code::Blocks umożliwia zdefiniowanie niestandardowych elementów zmiennych oprócz wbudowanych.



Rysunek 3.1: Globalne zmienne środowisko

3.7.3 Ograniczenia

- Zarówno nazwy zmiennych kompilatora zbiorowego, jak i globalnego 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 Code::Blocks otrzyma nieprawidłowe sekwencje znaków jako nazwy, może je zastąpić bez pytania.

- Każda zmienna wymaga zdefiniowania swojej podstawy. Wszystko inne jest opcjonalne, ale podstawa jest absolutnie obowiązkowa. Jeśli nie określisz podstawy zmiennej, nie zostanie ona zapisana (niezależnie od tego, jakie inne zdefiniowałeś).

- Nie możesz zdefiniować niestandardowego członka, który ma taką samą nazwę jak wbudowany członek. Obecnie niestandardowy element członkowski zastąpi wbudowany element członkowski, ale ogólnie zachowanie w tym przypadku jest niezdefiniowane.

- Wartości zmiennych i składowych mogą zawierać dowolne sekwencje znaków, z zastrzeżeniem następujących trzech ograniczeń:

  – Nie możesz definiować zmiennej przez wartość, która odwołuje się do tej samej zmiennej lub któregokolwiek z jej członków.
  – Nie możesz zdefiniować członka za pomocą wartości, która odwołuje się do tego samego członka
  – Nie można definiować składowej lub zmiennej przez wartość, która odwołuje się do tej samej zmiennej lub składowej poprzez zależność cykliczną.

Code::Blocks wykryje najbardziej oczywiste przypadki definicji rekurencyjnych (co może się zdarzyć przez przypadek), ale nie przeprowadzi dogłębnej analizy każdego możliwego nadużycia. Jeśli wprowadzisz bzdury, wtedy dostaniesz bzdury; jesteś teraz ostrzeżony.

Przykłady

Zdefiniowanie `wx.include` jako `$(#wx)/include` jest zbędne, ale całkowicie legalne Zdefiniowanie `wx.include` jako `$(#wx.include)` jest nielegalne i zostanie wykryte przez Code::Blocks Zdefiniowanie `wx.include` jako `$(#cb.lib)`, który ponownie jest zdefiniowany jako `$(#wx.include)` stworzy nieskończoną pętlę.

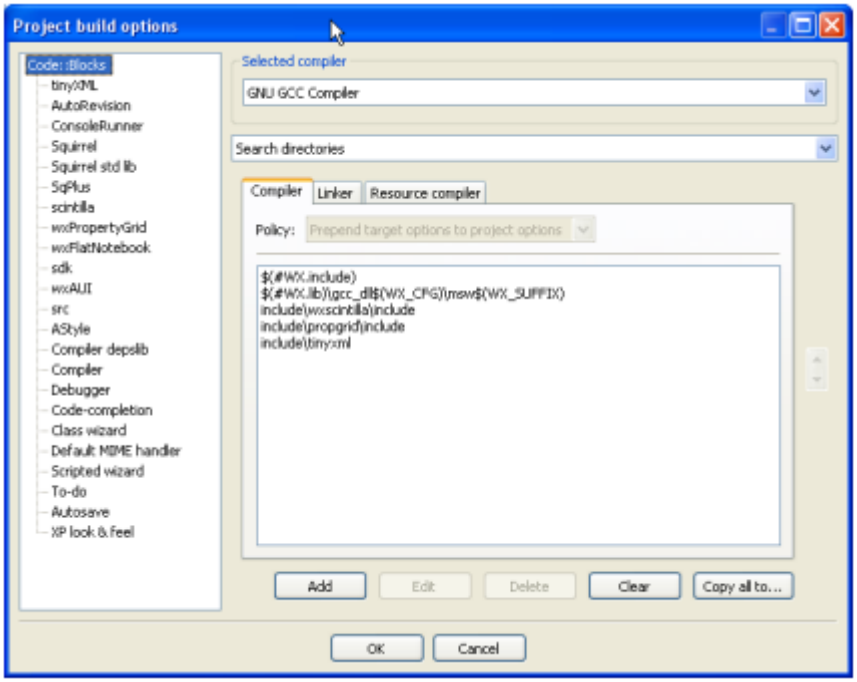
3.7.4 Używanie globalnych zmiennych kompilatora

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

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

Jeśli chcesz później zmodyfikować lub usunąć zmienną, możesz to zrobić w menu ustawień.

Przykład



Rysunek 3.2: Zmienne globalne

Powyższy obraz przedstawia zarówno zmienne na projekt, jak i globalne. WX_SUFFIX jest zdefiniowany w projekcie, ale WX jest globalną zmienną użytkownika.

3.7.5 Zestawy zmiennych

Czasami chcesz używać różnych wersji tej samej biblioteki lub tworzysz dwie gałęzie tego samego programu. Chociaż można dogadać się z globalną zmienną kompilatora, może to stać się nużące. W tym celu Code::Blocks obsługuje zestawy zmiennych. Zestaw zmiennych to niezależny zbiór zmiennych identyfikowanych 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 tych samych zmiennych, a identyczne zmienne w różnych zestawach nie muszą mieć tych samych wartości ani nawet tych samych elementów niestandardowych.

Kolejną pozytywną rzeczą w zestawach jest to, że jeśli masz tuzin zmiennych i chcesz mieć nowy zestaw z jedną z tych zmiennych wskazującą inną lokalizację, nie musisz ponownie wprowadzać wszystkich danych. Możesz po prostu utworzyć klon swojego bieżącego zestawu, który następnie zduplikuje wszystkie twoje 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.

3.7.6 Mini-samouczek dotyczący członków niestandardowych

Jak wspomniano powyżej, pisanie `$(#var.include)` i `$(#var)/include` jest domyślnie dokładnie tym samym. Dlaczego więc miałbyś chcieć napisać coś tak nieintuicyjnego jak `$(#var.include)`?

Weźmy na przykład standardową instalację Boost w systemie Windows. Generalnie można by oczekiwać, że fikcyjny pakiet ACME będzie miał swoje pliki dołączane w ACME/include, a jego biblioteki w ACME/lib. Opcjonalnie może umieścić swoje nagłówki w jeszcze innym podfolderze o nazwie acme. Tak więc po dodaniu poprawnych ścieżek do opcji kompilatora i konsolidatora, można się spodziewać `#include<acme/acme.h>` i połączyć się z `libacme.a` (lub czymkolwiek to jest).

Jednak Boost domyślnie instaluje nagłówki w `C:\Boost\include\boost-1_33_1\boost` i jego bibliotekach w `C:\Boost\lib`. Wydaje się to niemożliwe, aby uzyskać to pod jednym dachem bez konieczności dostosowywania wszystkiego na każdym nowym komputerze, zwłaszcza jeśli musisz pracować również pod Linuksem lub innym systemem operacyjnym.

W tym miejscu ujawnia się prawdziwa moc globalnych zmiennych użytkownika. Definiując wartość zmiennej `#boost` idziesz o krok dalej niż zwykle. Element członkowski należy zdefiniować jako `C:\Boost\include\boost-1_33_1\boost`, a element członkowski lib jako `C:\Boost\lib`, odpowiednio. Twoje projekty używające `$(#boost.include)` i `$(#boost.lib)` będą magicznie działać na każdym komputerze bez żadnych modyfikacji. Nie musisz wiedzieć dlaczego, nie chcesz wiedzieć dlaczego.

4 Praca z Code::Blocks

Ten rozdział zajmuje się podstawową wiedzą, aby móc pracować z Code::Blocks. Niektóre akapity, tutaj bezpośrednio zaczerpnięte z Wiki, nakładają się, ale z nieco inną prezentacją z treścią pierwszego rozdziału.

4.1 Proces budowania Code::Blocks

Na tych stronach szczegółowo wyjaśniono proces kompilacji. To, co dzieje się za kulisami i „kiedy" jest przemilczane. Mam nadzieję, że będzie to ciekawa lektura :).

4.1.1 Kolejność budowania

Jak zapewne się domyślasz, Code::Blocks nie uruchamia poleceń budowania w sposób losowy, ale jako dobrze przemyślaną i przygotowaną sekwencję. Ale najpierw zobaczmy, jakie komponenty są przedmiotem kompilacji:

Workspace (Obszar roboczy): zawiera jeden lub więcej projektów

Project (Projekt): zawiera co najmniej jeden cel kompilacji. Zawiera również pliki projektu.

Build target (Cel kompilacji): przypisane są do niego pliki projektu, które są budowane jako grupa i generują jedno wyjście binarne. Dane wyjściowe mogą być plikiem wykonywalnym, biblioteką dynamiczną lub biblioteką statyczną. **UWAGA**: istnieje jeden rodzaj celu kompilacji, który nie generuje binarnych danych wyjściowych bezpośrednio, ale po prostu uruchamia kroki przed/po kompilacji (co może generować dane binarne zewnętrznie).

Podzielmy te tematy na sekcje i wyjaśnijmy je bardziej szczegółowo.

4.1.2 Workspace (Obszar roboczy)

Obszar roboczy to element kontenera najwyższego poziomu do organizowania projektów. Ponieważ w danym momencie może być otwarty tylko jeden obszar roboczy, tak naprawdę nie ma dla nich problemu z kolejnością kompilacji. To tylko jeden, więc właśnie powstał ;).

Użyj menu „Build" (Buduj) → „Build workspace" (Buduj obszar roboczy), aby zbudować obszar roboczy (tj. wszystkie zawarte w nim projekty).

4.1.3 Projects (Projekty)

Tutaj zaczyna się robić ciekawie :).

Kolejność kompilacji projektów różni się w zależności od tego, czy użytkownik ustawił zależności projektu, czy nie. Proszę czytaj dalej.

Bez zależności projektowych

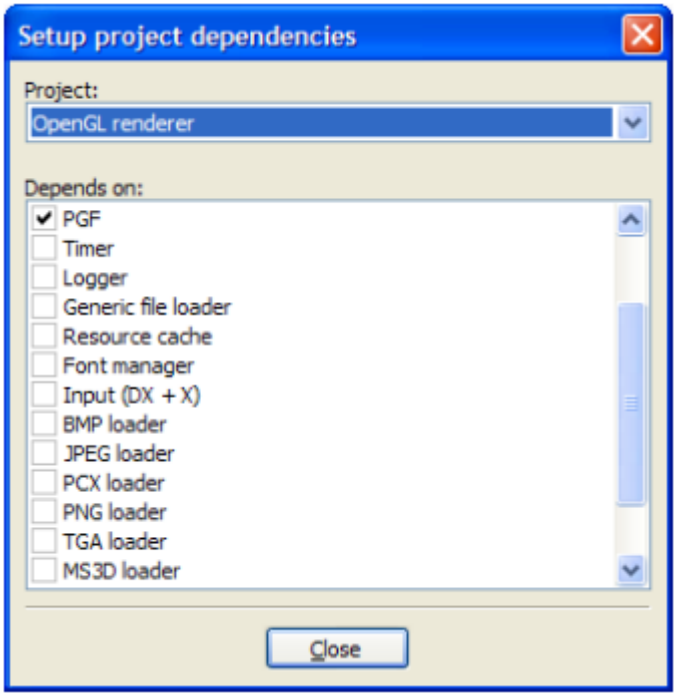
W tym przypadku projekty są budowane w kolejności ich wyglądu, od góry do dołu. Jednak w większości projektów (przynajmniej nie tych „hello world") chciałoby się skonfigurować zależności projektu.

Korzystanie z zależności projektu

Zależności projektu to prosty sposób na poinformowanie Code::Blocks, że dany projekt „zależy” od innego (zawsze w tym samym obszarze roboczym).

Zatem wyobraź sobie w swoim obszarze roboczym, że masz projekt biblioteki i projekt wykonywalny, który zależy od biblioteki. Wtedy możesz (i powinieneś) poinformować Code::Blocks o tej zależności. W tym celu należy wybrać „Project” (Projekt) → „Properties” (Właściwości) i kliknąć przycisk „Project’s dependencies..” (Zależności projektu).

Należy zauważyć, że informacje o zależnościach są zapisywane w pliku obszaru roboczego, a nie w pliku projektu, ponieważ jest to zależność między dwoma projektami w obszarze roboczym.



Rysunek 4.1: Konfigurowanie zależności projektu

Korzystanie z tego okna dialogowego jest bardzo łatwe. Wybierz projekt, do którego chcesz dodać zależność, a następnie umieść znacznik wyboru na wszystkich projektach, od których ten projekt zależy. Zapewni to, że wszystkie sprawdzone projekty zostaną zbudowane przed projektem, który od nich zależy, zapewniając zsynchronizowaną kompilację.

Wskazówka: nie musisz zamykać tego okna dialogowego i ponownie uruchamiać właściwości innego projektu, aby ustawić ich zależności. W tym samym oknie dialogowym możesz ustawić wszystkie zależności projektów. Po prostu wybierz inny projekt z listy rozwijanej :).

Kilka rzeczy do zapamiętania:

- Zależności są ustalane bezpośrednio lub pośrednio. Jeśli projekt A zależy bezpośrednio od projektu B, a projekt B zależy od projektu C, to projekt A pośrednio zależy również od projektu C.

- Code::Blocks jest wystarczająco sprytny, aby uważać na zależności kołowe i zabraniać ich. Zależność cykliczna powstaje, gdy projekt A zależy bezpośrednio lub pośrednio od projektu B, a projekt B zależy bezpośrednio lub pośrednio od projektu A.

- Zależności obowiązują zarówno w przypadku budowania całego obszaru roboczego, jak i pojedynczego projektu. W takim przypadku zostaną zbudowane tylko zależności potrzebne do budowania projektu.

4.1.4 Budowanie celów

Kolejność budowania celów kompilacji zależy od kilku rzeczy.

1. Jeśli użytkownik wybrał określony cel kompilacji w polu rozwijanym paska narzędzi kompilatora, budowany jest tylko ten cel kompilacji. Jeśli dla projektu zawierającego ten cel są skonfigurowane zależności projektu, wszystkie projekty, od których zależy, również skompilują cel o tej samej nazwie. Jeśli taki cel nie istnieje, projekt jest pomijany.

2. Jeśli wybrany jest wirtualny cel „All" (Wszystko), to wszystkie cele w projekcie (i wszystkie projekty, od których on zależy) są budowane w kolejności od góry do dołu. Istnieje kilka wyjątków od tego:

    - Cel nie jest budowany z „All" (Wszystko), jeśli opcja celu (na stronie właściwości projektu „Targets" (Cele)) „Build this target with All" (Buduj ten cel ze wszystkimi) nie jest zaznaczona.

    - Jeżeli żadne cele w projekcie nie mają zaznaczonej powyższej opcji, to na liście nie pojawi się żaden wirtualny cel „All" (Wszystko).

4.1.5 Faza przetwarzania wstępnego

Przed rozpoczęciem rzeczywistego procesu kompilacji (tj. Rozpoczęcie wykonywania poleceń kompilatora/konsolidatora), uruchamiany jest etap przetwarzania wstępnego, który generuje wszystkie wymagane wiersze poleceń dla całego procesu kompilacji. Ten krok przechowuje w pamięci podręcznej większość generowanych informacji, dzięki czemu kolejne kompilacje działają szybciej.

Ten krok uruchamia również dołączone skrypty kompilacji

4.1.6 Rzeczywiste wykonanie poleceń

Jest to etap, na którym właściwie rozpoczyna się proces budowania, z punktu widzenia użytkownika. Pliki zaczynają się kompilować i ostatecznie łączyć, aby wygenerować różne wyjścia binarne, które definiują cele kompilacji.

W tym kroku wykonywane są również kroki przed kompilacją i po kompilacji.

4.1.7 Kroki przed i po kompilacji

Są to polecenia, które można skonfigurować na poziomie projektu i/lub celu. Są to polecenia powłoki, które m.in. kopiują pliki lub jakąkolwiek inną operację, którą możesz wykonać w zwykłej powłoce systemu operacyjnego.

Zmienne określone w artykule Rozszerzanie zmiennych (rozdział 3 na stronie 78) mogą być używane w skryptach do pobierania takich rzeczy, jak docelowy katalog wyjściowy, katalog projektu, typ celu i inne.

Oto podział kolejności wykonywania kroków przed/po kompilacji dla wyimaginowanego projektu z dwoma celami (Debug (debugowanie)/Release (uwalnianie):

1. Kroki przed kompilacją projektu

   a) Docelowe etapy „Debugowania" przed kompilacją

   b) Docelowe pliki kompilacji „Debug"

   c) Obierz cel „Debug", łącz pliki i wygeneruj dane binarne

   d) Docelowe etapy „debugowania" po kompilacji (patrz uwagi poniżej)

   e) Docelowe etapy „uwalniania" przed budowaniem

   f) Docelowe pliki kompilacji „Release"

   g) Obierz za cel „Release", łącz pliki i wygeneruj wyjście binarne

   h) Docelowe etapy „Release" po zakończeniu budowy (patrz uwagi poniżej)

2. Kroki po zbudowaniu projektu

Mam nadzieję, że to wszystko wyjaśnia :)

Uwaga:
Kroki przed kompilacją są zawsze wykonywane. Kroki po kompilacji będą uruchamiane tylko wtedy, gdy projekt/cel, do którego są dołączone, nie jest aktualny (tj. ma zostać zbudowany). Możesz to zmienić, wybierając „Always execute, even if target is up to date" (Zawsze wykonuj, nawet jeśli cel jest aktualny) w opcjach kompilacji.

Przykłady skryptów

Skrypt pokompilacyjny, który kopiuje plik wyjściowy do folderu C:\Program\bin w systemie Windows:

cmd / c copy "$ (PROJECT_DIR) $ (TARGET_OUTPUT_FILE)" "C: \ Program\ bin "

Uruchom skrypt bash „copyresources.sh" w systemie Linux:

```
/bin /sh copyresources.sh
```

Utwórz nowy katalog w katalogu wyjściowym:

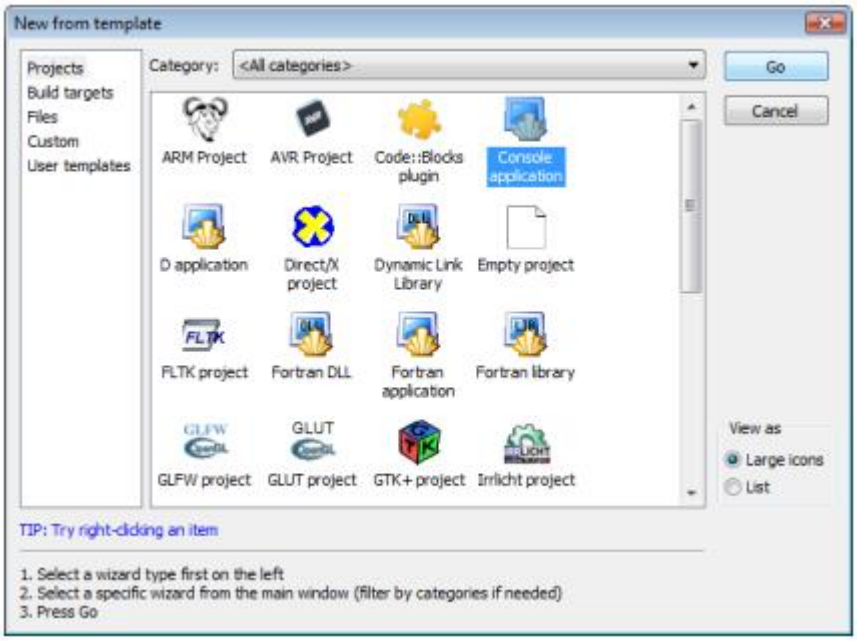
```
mkdir $ (TARGET_OUTPUT_DIR) / data
```

4.2 Tworzenie nowego projektu

Te strony są przewodnikiem po wielu początkowych (i niektórych pośrednich) funkcjach tworzenia i modyfikacji projektu Code::Blocks. Jeśli jest to Twoje pierwsze doświadczenie z Code::Blocks, oto dobry punkt wyjścia.

4.2.1 Kreator projektu

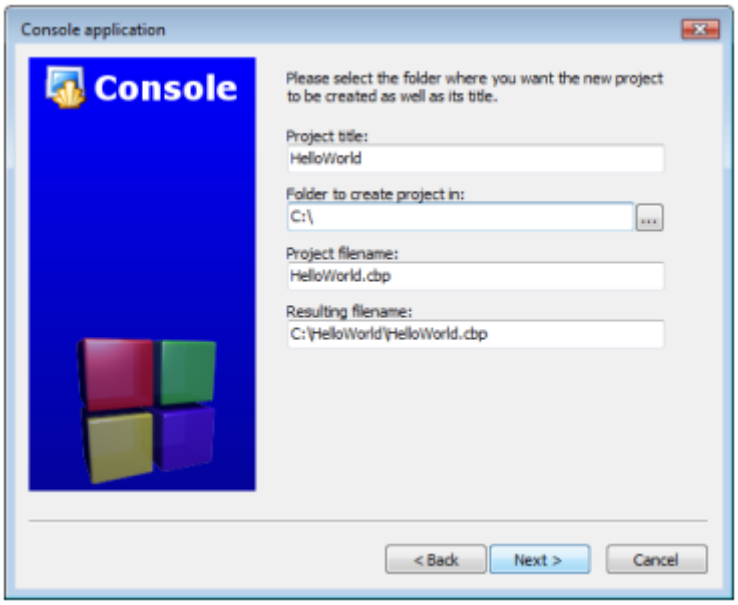
Uruchom kreatora projektu poprzez „File” (Plik) → „New” (Nowy) → „Project...” (Projekt...), aby rozpocząć nowy projekt. Tutaj jest wiele wstępnie skonfigurowanych szablonów dla różnych typów projektów, w tym możliwość tworzenia własnych szablonów. Wybierz aplikację **Console application** (aplikacja konsolowa), ponieważ jest to najczęstsza aplikacja do celów ogólnych, i kliknij **Go** (Przejdź).



Rysunek 4.2: Kreator projektu

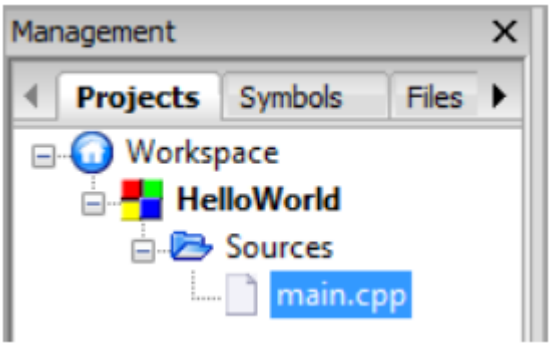
Uwaga:
czerwony tekst zamiast czarnego tekstu pod dowolną z ikon oznacza, że korzysta z niestandardowego skryptu kreatora.

Następnie pojawi się kreator aplikacji konsolowej. Kontynuuj przeglądanie menu, wybierając C++ po wyświetleniu monitu o język. Na następnym ekranie nadaj projektowi nazwę i wpisz lub wybierz folder docelowy. Jak widać poniżej, Code::Blocks wygeneruje pozostałe wpisy z tych dwóch.



Rysunek 4.3: Aplikacja konsolowa

Na koniec kreator zapyta, czy ten projekt powinien używać domyślnego kompilatora (zwykle GCC) i dwóch domyślnych kompilacji: Debug i Release. Wszystkie te ustawienia są w porządku. Naciśnij Finish, a projekt zostanie wygenerowany. Główne okno zmieni kolor na szary, ale to nie problem, wystarczy otworzyć plik źródłowy. W zakładce Projects (Projekty) panelu Management (Zarządzanie) po lewej stronie rozwiń foldery i kliknij dwukrotnie plik źródłowy main.cpp, aby otworzyć go w edytorze.



Rysunek 4.4: Wybierz źródło

Ten plik zawiera następujący standardowy kod:

main.cpp

```
#include <iostream>
using namespace std;
int main ( )
{
cout << ”Hello world!” << endl;
return 0;
}
```

4.2.2 Zmiana składu pliku

Pojedynczy plik źródłowy ma niewielkie zastosowanie w programach o dowolnej użytecznej złożoności. Aby sobie z tym poradzić, Code::Blocks ma kilka bardzo prostych metod dodawania dodatkowych plików do projektu.

Dodawanie pustego pliku

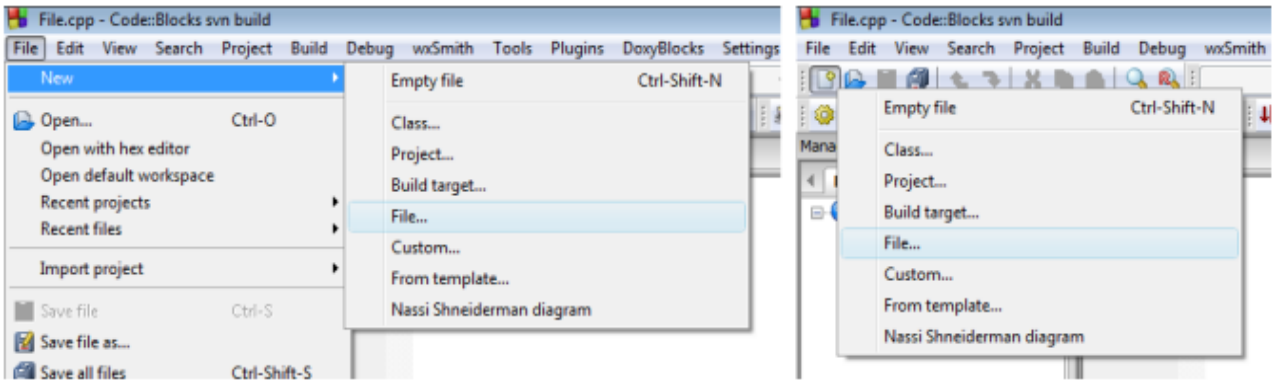
w tym przykładzie podzielimy funkcję

main.cpp
```
cout << ”Hello world!” << endl;
```

do osobnego pliku.

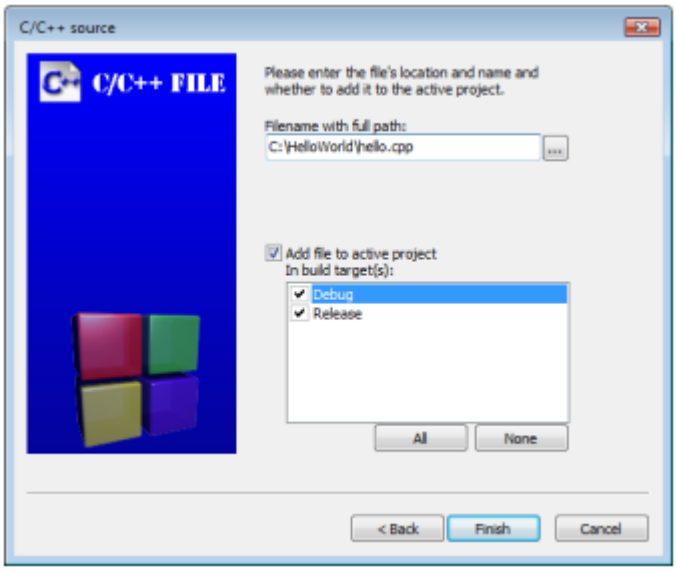
Uwaga:
generalnie niewłaściwym stylem programowania jest tworzenie tak małej funkcji; zrobiono to tutaj, aby dać prosty przykład.

Aby dodać nowy plik do projektu, wywołaj kreatora szablonu pliku przez „File” (Plik) → „New” (Nowy) → „File...” (Plik...) lub „Main Toolbar” (Główny pasek narzędzi) → „New file (button)” (Nowy plik (przycisk)) → „File...” (Plik...). Użyj menu „Build” (Buduj) → „Build workspace” (Buduj obszar roboczy), aby zbudować obszar roboczy (tj. wszystkie zawarte w nim projekty).



Rysunek 4.5: Nowy plik

Wybierz źródło **C/C++** i kliknij **Go** (Przejdź). Kontynuuj przeglądanie poniższych okien dialogowych, podobnie jak przy tworzeniu oryginalnego projektu, wybierając **C++** po wyświetleniu monitu o język. Na ostatniej stronie zobaczysz kilka opcji. Pierwsze pole określi nową nazwę pliku i lokalizację (jak wspomniano, wymagana jest pełna ścieżka). Możesz opcjonalnie użyć przycisku ..., aby wyświetlić okno przeglądarki plików, aby zapisać lokalizację pliku. Zaznaczenie opcji **Add file to active project** (Dodaj plik do aktywnego projektu) spowoduje zapisanie nazwy pliku w folderze **Sources** (Źródła) zakładki **Projects** (Projekty) panelu **Management** (Zarządzanie). Zaznaczenie dowolnego z celów kompilacji spowoduje ostrzeżenie Code::Blocks, że plik powinien zostać skompilowany i połączony z wybranymi celami. Może to być przydatne, jeśli na przykład plik zawiera określony kod debugowania, ponieważ pozwoli to na włączenie (lub wykluczenie) z odpowiednich celów kompilacji. Jednak w tym przykładzie funkcja hello ma kluczowe znaczenie i jest wymagana w każdym celu, więc zaznacz wszystkie pola i kliknij przycisk **Finish** (Zakończ), aby wygenerować plik.



Rysunek 4.6: Konfiguracje programu Hello

Nowo utworzony plik powinien otworzyć się automatycznie; jeśli nie, otwórz go, klikając dwukrotnie jego plik w zakładce **Projects** (Projekty) panelu **Management** (Zarządzanie). Teraz dodaj kod dla funkcji, którą wywoła **main.cpp**.

hello.cpp

```
#include <iostream>
using namespace std;

void hello( )
{
        cout << ” Hello world! ” << endl ;
}
```

Dodawanie istniejącego pliku

Teraz, gdy funkcja **hello()** znajduje się w osobnym pliku, funkcja musi być zadeklarowana, aby **main.cpp** z niej korzystał. Uruchom edytor zwykłego tekstu (na przykład Notatnik lub Gedit) i dodaj następujący kod:

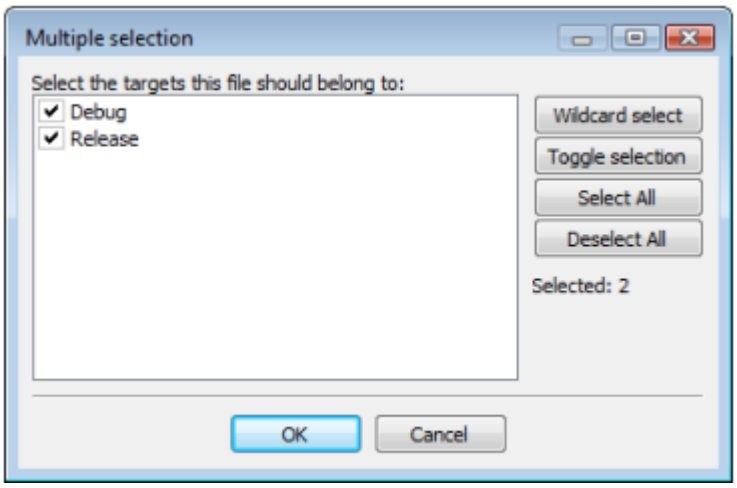
hello.h

```
#ifndef HELLO_H_INCLUDED
#define HELLO_H_INCLUDED

void hello();

#endif   // HELLO_H_INCLUDED
```

Zapisz ten plik jako nagłówek (**hello.h**) w tym samym katalogu, co inne pliki źródłowe w tym projekcie. Wróć do Code::Blocks, kliknij „Project” (Projekt) → „Add files...” (Dodaj pliki...), aby otworzyć przeglądarkę plików. Tutaj możesz wybrać jeden lub wiele plików (używając kombinacji klawiszy Ctrl i Shift). (Opcja „Project” (Projekt) → „Add files recursively...” (Dodaj pliki rekursywnie...) przeszuka wszystkie podkatalogi w podanym folderze, wybierając odpowiednie pliki do włączenia.) Wybierz **hello.h** i kliknij **Open** (Otwórz), aby wyświetlić okno dialogowe z zapytaniem, do których celów budowania, do których powinny należeć pliki. W tym przykładzie wybierz oba cele.



Rysunek 4.7: Przynależność do celu

Uwaga:
jeśli bieżący projekt ma tylko jeden cel kompilacji, to okno dialogowe zostanie pominięte

Wracając do głównego źródła (main.cpp), dołącz plik nagłówkowy i zastąp funkcję cout, aby dopasować nową konfigurację projektu.

main.cpp

```
#include ”hello.h”

int main()
{
hello();
return 0;
}
```

Naciśnij Ctrl-F9, „Build” (Buduj) → „Build” (Buduj) lub „Compiler Toolbar” (Pasek narzędzi kompilatora) → „Buduj (przycisk – koło zębate)”, aby skompilować projekt. Jeśli następujące dane wyjściowe zostaną wygenerowane w dzienniku budowy (w dolnym panelu), wszystkie kroki zostały wykonane poprawnie.

```
———————— Build: Debug in HelloWorld ————————

Compiling: main.cpp
Compiling: hello.cpp
```

```
Linking console executable: bin\Debug\HelloWorld.exe
Output size is 923.25 KB
Process terminated with status 0 (0 minutes, 0 seconds)
0 errors, 0 warnings (0 minutes, 0 seconds)
```

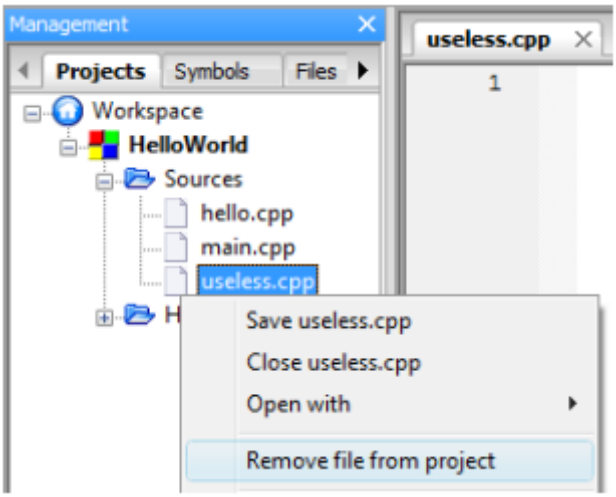
Plik wykonywalny można teraz uruchomić, klikając przycisk „Run” (Uruchom) lub naciskając Ctrl-F10.

Uwaga:
opcja F9 (buduj i uruchamiaj) łączy te polecenia i może być bardziej użyteczna w niektórych sytuacjach.

Zobacz proces kompilacji Code::Blocks, aby dowiedzieć się, co dzieje się za kulisami podczas kompilacji.

Usuwanie pliku

Korzystając z powyższych kroków, dodaj do projektu nowy plik źródłowy C++, **useless.cpp**. Usunięcie tego niepotrzebnego pliku z projektu jest proste. Po prostu kliknij prawym przyciskiem myszy **useless.cpp** w zakładce „**Projects**” (Projekty) panelu „**Management**” (Zarządzanie) i wybierz „**Remove file from project**” (Usuń plik z projektu).

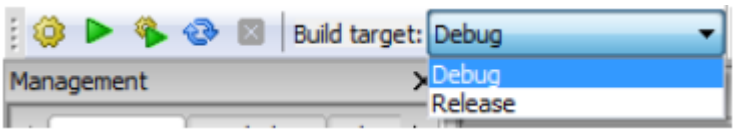


Rysunek 4.8: Usuń plik z projektu

Uwaga:
usunięcie pliku z projektu nie powoduje jego fizycznego usunięcia; Code::Blocks usuwa go tylko z zarządzania projektem.

4.2.3 Modyfikowanie opcji kompilacji

Do tej pory cele kompilacji pojawiały się kilka razy. Zmiana między dwoma domyślnymi wygenerowanymi - **Debug** (debugowanie) i **Release** (zezwolenie na publikację) - można po prostu wykonać za pomocą listy rozwijanej na pasku narzędzi **Compiler Toolbar** (pasek narzędzi kompilatora). Każdy z tych celów może być innego typu (na przykład: biblioteka statyczna; aplikacja konsolowa), zawierać inny zestaw plików źródłowych, zmienne niestandardowe, różne flagi kompilacji (na przykład: symbole debugowania -p; optymalizacja rozmiaru -Os ; optymalizacja czasu linkowania -flto) i kilka innych opcji.

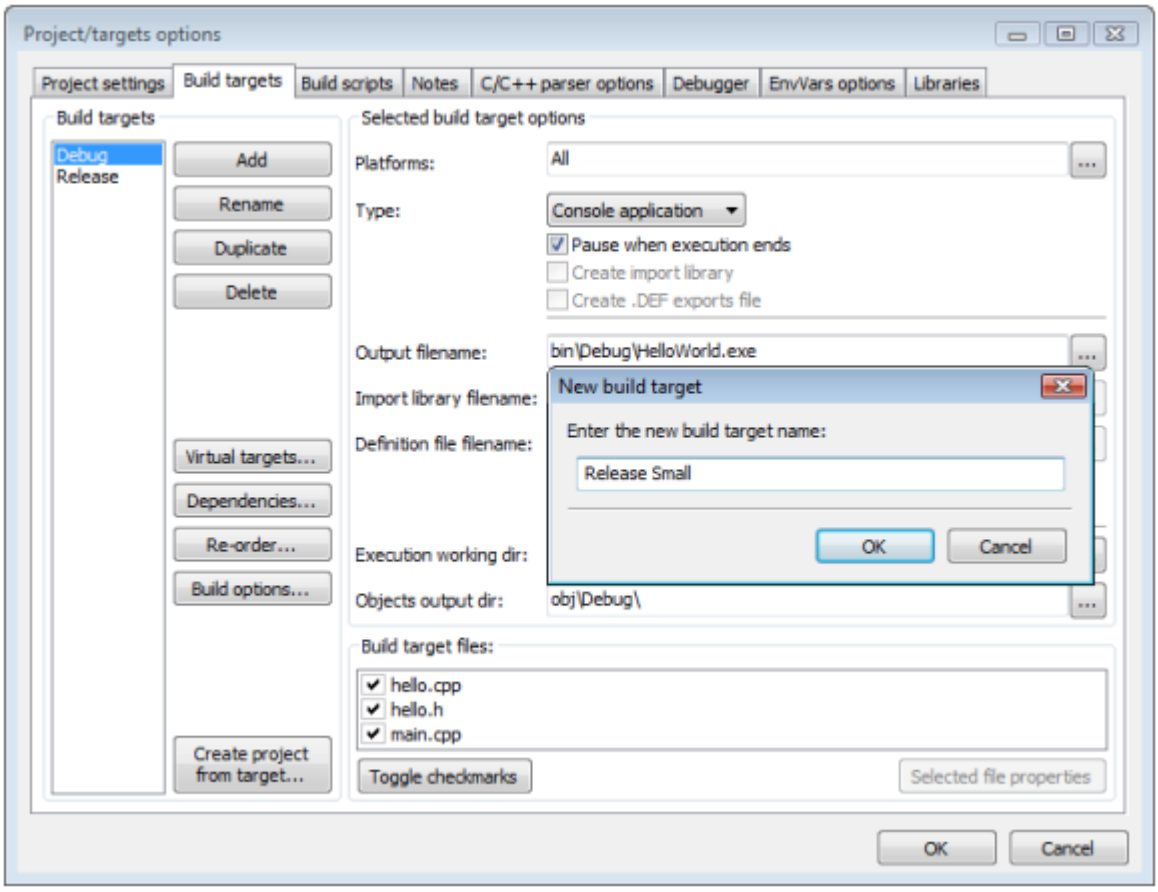


Rysunek 4.9: Wybór celu

„Open Project” (Otwórz projekt) → „Properties...” (Właściwości...), aby uzyskać dostęp do głównych właściwości aktywnego projektu **HelloWorld**. Większość ustawień na pierwszej zakładce, **Project Settings** (Ustawienia projektu), jest rzadko zmieniana. **Title** (Tytuł): umożliwia zmianę nazwy projektu. Jeśli **Platforms** (Platformy): zostanie zmienione na coś innego niż domyślne **All** (Wszystko), Code::Blocks zezwoli na budowanie projektu tylko na wybranych obiektach docelowych. Jest to przydatne, jeśli na przykład kod źródłowy zawiera interfejs API systemu Windows i dlatego byłby nieważny w dowolnym miejscu poza systemem Windows (lub w innych sytuacjach specyficznych dla systemu operacyjnego). Opcje **Makefile**: są używane tylko wtedy, gdy projekt powinien używać makefile zamiast wewnętrznego systemu budowania Code::Blocks (zobacz Code::Blocks i Makefiles [sekcja 4.5 na stronie 111], aby uzyskać więcej informacji).

Dodawanie nowego celu kompilacji

Przejdź do zakładki **Build targets** (Cele kompilacji). Kliknij przycisk **Add** (Dodaj), aby utworzyć nowy cel kompilacji i nadaj mu nazwę **Release Small**. Podświetlenie w lewej kolumnie powinno automatycznie przełączyć się na nowy cel (jeśli nie, kliknij go, aby zmienić fokus). Jako domyślne ustawienie dla **Type** (Typu): - „GUI application” (Aplikacja GUI) - jest niepoprawne dla programu **HelloWorld**, zmień go na „Console application” (Aplikacja konsoli) za pomocą listy rozwijanej. Nazwa pliku wyjściowego **HelloWorld.exe** jest w porządku, z wyjątkiem tego, że spowoduje to, że plik wykonywalny zostanie wyprowadzony w katalogu głównym. Dodaj przed nim ścieżkę „bin\ReleaseSmall\” (Windows) lub „bin/ReleaseSmall/” (Linux), aby zmienić katalog (jest to ścieżka względna od katalogu głównego projektu). **Execution working dir** (Katalog roboczy wykonania): odnosi się do miejsca, w którym program zostanie wykonany po wybraniu opcji **Run** (Uruchom) lub **Build and run** (Buduj i uruchom). Ustawienie domyślne ”.” jest w porządku (odnosi się do katalogu projektu). **Objects output dir** (Katalog wyjściowy obiektów): należy zmienić na ”obj\ReleaseSmall\” (Windows) lub ”obj/ReleaseSmall/” (Linux) w celu zachowania spójności z pozostałą częścią projektu. **Build target files** (Pliki docelowe kompilacji): obecnie nie ma nic zaznaczonego. Jest to problem, ponieważ nic nie zostanie skompilowane, jeśli ten cel zostanie zbudowany. Zaznacz wszystkie pola.



Rysunek 4.10: Opcje celu

Następnym krokiem jest zmiana ustawień celu. Kliknij Build options... (Opcje kompilacji...), aby uzyskać dostęp do ustawień. Pierwsza karta, która się pojawi, zawiera szereg flag kompilatora dostępnych za pośrednictwem pól wyboru. Wybierz „Strip all symbols from binary” (Usuń wszystkie symbole z binarnych) i „Optimize generated code for size” (Optymalizuj wygenerowany kod pod kątem rozmiaru). Flagi tutaj zawierają wiele powszechniejszych opcji, jednak można przekazać niestandardowe argumenty. Przejdź do zakładki Other options (Inne opcje) i dodaj następujące przełączniki:

```
-fno-rtti
-fno-exceptions
-ffunction-sections
-fdata-sections
-flto
```

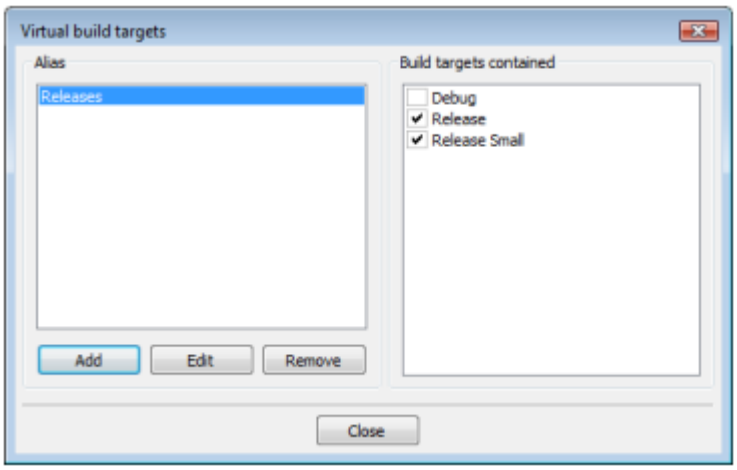
Teraz przejdź do zakładki **Linker settings** (Ustawienia konsolidatora). Pole **Link libraries** (Połącz biblioteki): zapewnia miejsce do dodawania różnych bibliotek (na przykład wxmsw28u dla wersji monolitycznej dll wxWidgets dla systemu Windows Unicode). Ten program nie wymaga żadnych takich bibliotek. Przełączniki niestandardowe z poprzedniego kroku wymagają ich odpowiedników w czasie połączenia. Dodaj

```
-flto
-Os
-Wl,--gc-sections
-shared-libgcc
-shared-libstdc++
```

do **Other linker options** (innych opcji linkera): zakładka. (Aby uzyskać więcej informacji na temat działania tych przełączników, zobacz dokumentację GCC dotyczącą opcji optymalizacji i opcji konsolidatora).

Wirtualne cele

Kliknij **OK**, aby zaakceptować te zmiany i powrócić do poprzedniego okna dialogowego. Teraz, gdy istnieją dwie kompilacje wydania, skompilowanie obu będzie wymagało dwóch oddzielnych uruchomień kompilacji (**Build**) lub kompilacji i uruchomienia (**Build and run**). Na szczęście Code::Blocks zapewnia opcję łączenia wielu kompilacji razem. Kliknij Wirtualne cele... (**Virtual targets...**), a następnie Dodaj (**Add**). Nazwij wirtualny cel Wydania (**Releases**) i kliknij **OK**. W prawym polu zawierającym cele kompilacji (**Build targets contained**) wybierz zarówno Zwolnij (**Release**), jak i Zwolnij małe (**Release small**). Zamknij to pole i naciśnij **OK** w głównym oknie.



Rysunek 4.11: Wirtualne cele

Wirtualny cel „Releases” będzie teraz dostępny z paska narzędzi kompilatora; budowanie tego powinno skutkować następującymi danymi wyjściowymi:

```
-------------- Build: Release in HelloWorld ---------------

Compiling: main.cpp
Compiling: hello.cpp
Linking console executable: bin\Release\HelloWorld.exe
Output size is 457.50 KB

-------------- Build: Release Small in HelloWorld ---------------

Compiling: main.cpp
Compiling: hello.cpp
Linking console executable: bin\ReleaseSmall\HelloWorld.exe
Output size is 8.00 KB
Process terminated with status 0 (0 minutes, 1 seconds)
0 errors, 0 warnings (0 minutes, 1 seconds)
```

4.3 Debugowanie za pomocą Code::Blocks

W tej sekcji opisano, jak pracować w trybie debugowania

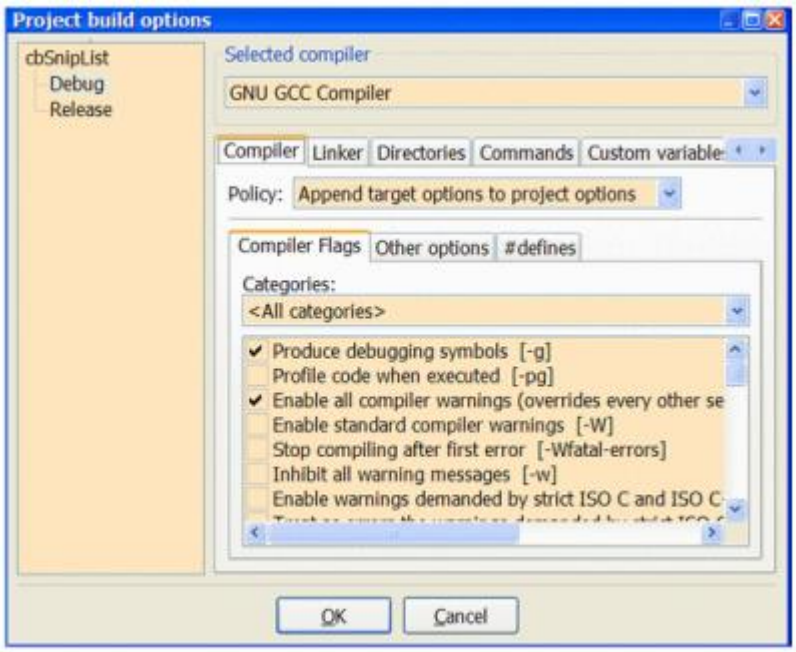
4.3.1 Zbuduj wersję debugowania swojego projektu

Upewnij się, że projekt jest skompilowany z włączoną opcją kompilatora -g (symbole debugowania) i wyłączoną opcją -s (symbole pasków). Gwarantuje to, że plik wykonywalny zawiera symbole debugowania.

Przełączniki optymalizacji kompilatora powinny być wyłączone, symbole usuwania (-s) **muszą** być wyłączone.

Należy pamiętać, że może być konieczne ponowne skompilowanie projektu, ponieważ w przeciwnym razie aktualne pliki obiektowe mogą nie zostać ponownie skompilowane za pomocą opcji -g. Należy pamiętać, że w kompilatorach innych niż GCC, -g i/lub -s mogą być różne przełączniki (-s może nie być w ogóle dostępne).

„Menu” → „Project” (Projekt) → „Build Options” (Opcje budowania)



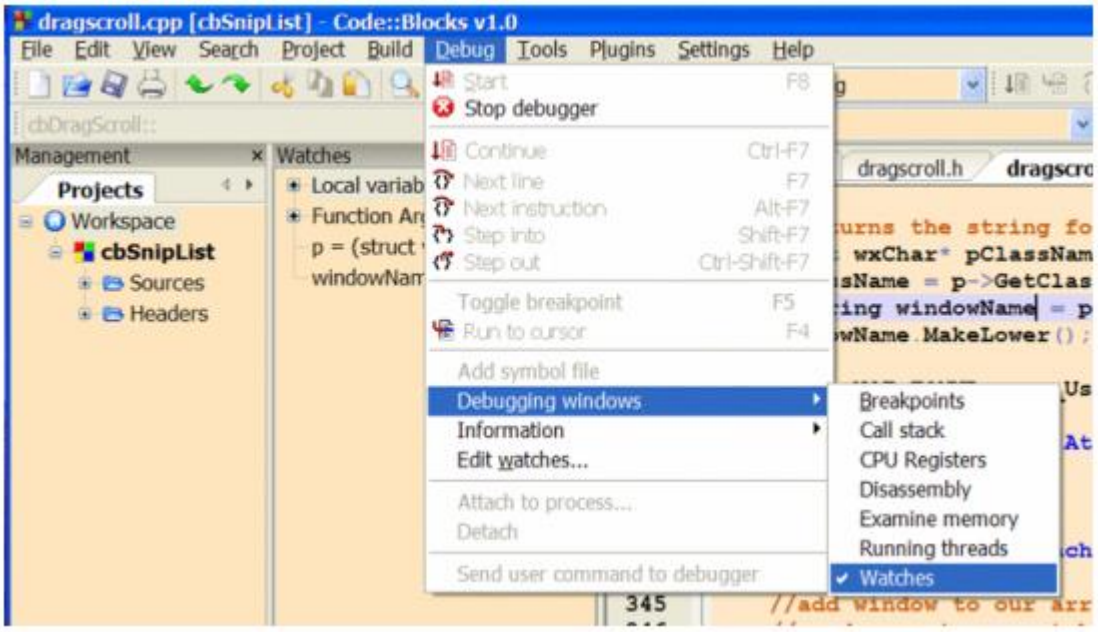
Rysunek 4.12: Opcje budowania projektu debugera

4.3.2 Dodaj obserwatorów.

W wersji 10.05

Uwaga:
To bardzo stara wersja. Nie powinieneś jej więcej używać.

Otwórz okno obserwatorów debugera (Debugger Watches Window)



Rysunek 4.13: Otwórz okno obserwatorów debuggera

Listę obserwatorów można zapisać do pliku, a później ponownie wczytać. Aby to zrobić, kliknij prawym przyciskiem myszy na liście obserwatorów i wybierz „save watch file” (zapisz plik obserwowany) (i „load watch file” (załaduj plik obserwowany), aby je ponownie załadować).

Rysunek 4.14: Zapisz okno obserwacji

Od 12.11 lub najnowszych nocnych kompilacji

W najnowszych nocnych kompilacjach okno obserwatorów zostało przeprojektowane i działa inaczej niż w 10.05.

Obecnie istnieją trzy sposoby dodawania w nim zegarków:

1. Kliknij pusty ostatni wiersz w oknie obserwatorów, wpisz nazwę zmiennej (lub pełne wyrażenie) i naciśnij Enter.

2. Gdy debugger zatrzymał się w punkcie przerwania, wybierz nazwę zmiennej lub pełne wyrażenie, kliknij prawym przyciskiem myszy, aby otworzyć menu kontekstowe, a następnie wybierz „Add watch 'expression'" (Dodaj obserwatora 'wyrażenie').

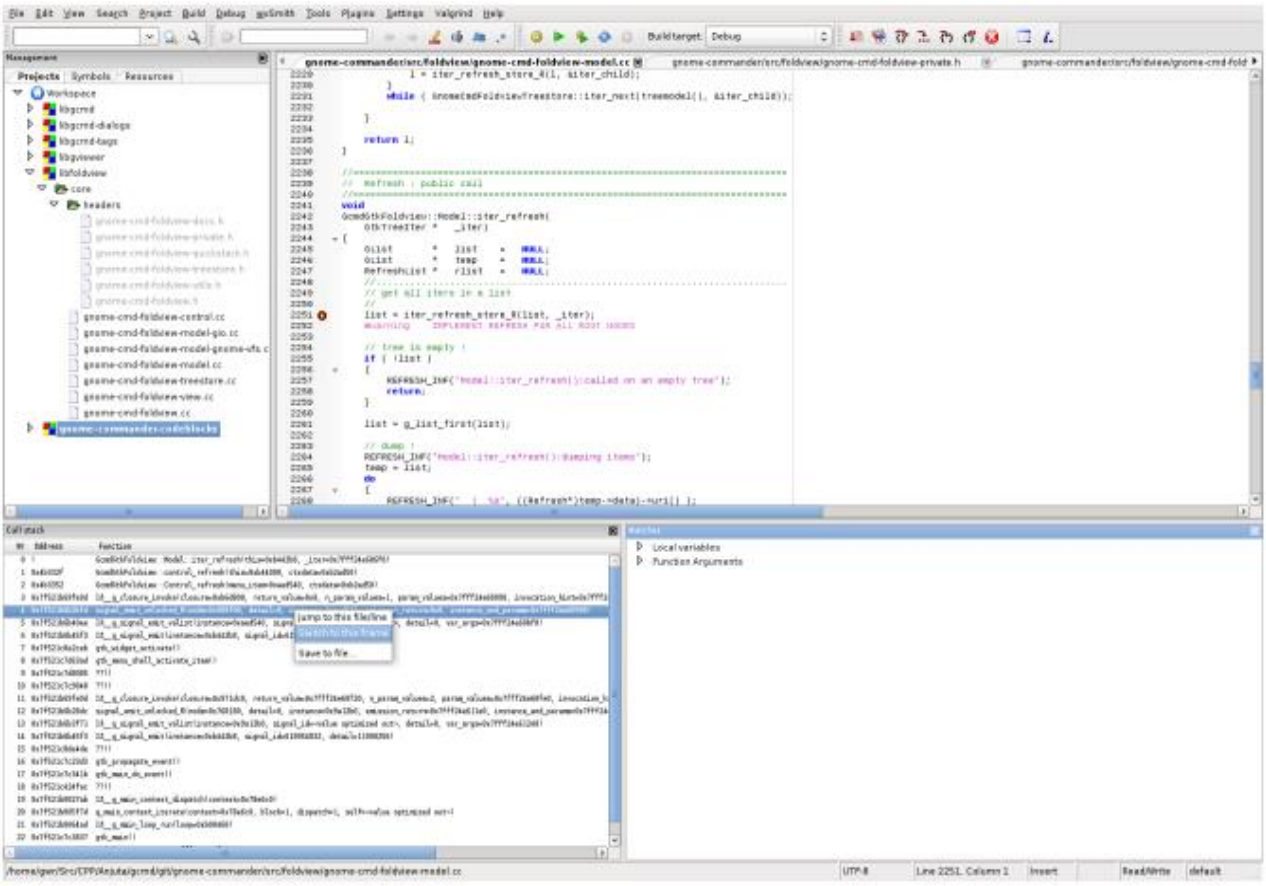
3. Wybierz wyrażenie w edytorze, przeciągnij i upuść w oknie obserwatorów.

Automatyczne włączanie zmiennych lokalnych i argumentów funkcji zostało ponownie zaimplementowane w wersji 13.12.

4.3.3 Podwójne kliknięcie w oknie Stos wywołań (Call stack)

Uwaga:
podczas debugowania, dwukrotne kliknięcie ramki w oknie debugowania „stosu wywołań" nie aktualizuje automatycznie zmiennych wyświetlanych w oknie debugowania „watches" (obserwatorzy).

Musisz kliknąć prawym przyciskiem myszy ramkę w oknie debugowania „call stack" (stosu wywołań) i wybrać „Switch to this frame" (Przełącz na tę ramkę).

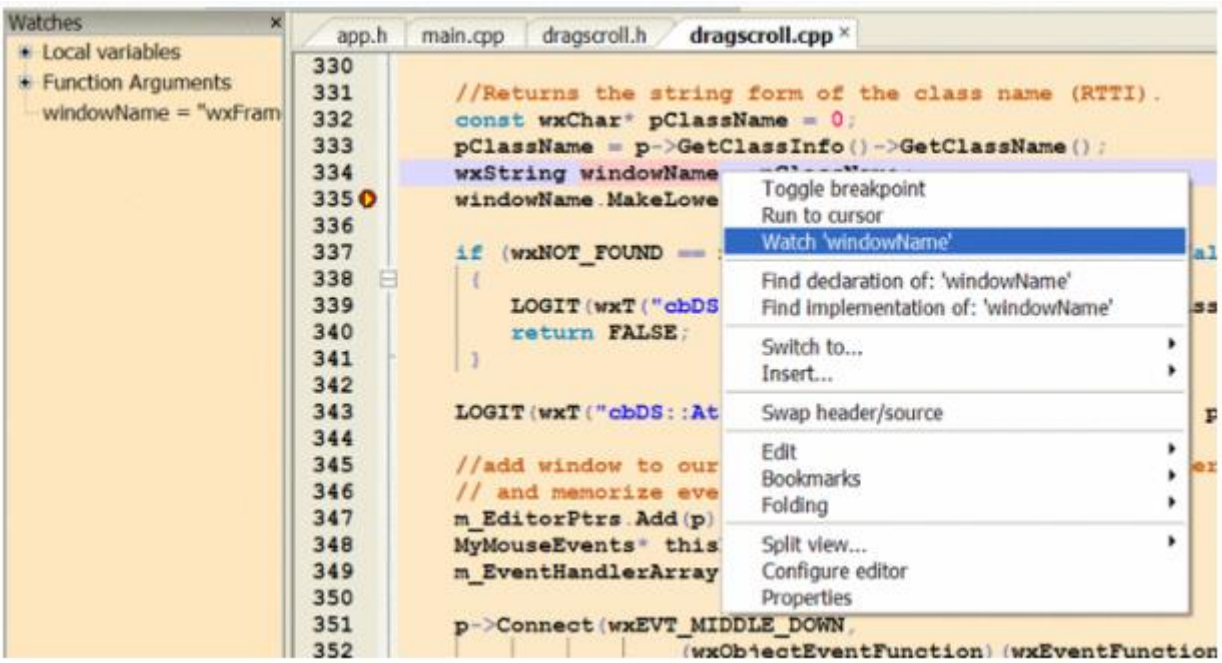


Rysunek 4.15: Okno obserwacji (Watch Window)

4.3.4 Ustaw punkty przerwania

Znajdź wiersz zawierający zmienną, która ma być obserwowana. Ustaw punkt przerwania w pozycji, która pozwoli Ci obserwować wartość zmiennej.

„Menu” → „Debug” (Debugowanie) → „Toggle Breakpoint” (Przełącz punkt przerwania).



Rysunek 4.16: Ustawianie zmiennych obserwacji

Uruchom debuger, aż zostanie osiągnięty punkt przerwania. Kliknij zmienną prawym przyciskiem myszy, aby ustawić obserwatora w oknie Watch Window.

Punkty przerwania można również przełączać lewym przyciskiem myszy na lewym marginesie edytora.

4.3.5 Uwagi

Obsługa skryptów

Code::Blocks natywnie używa języka skryptowego squirrel do obsługi gdb, patrz: Skrypty debugera (sekcja 4.4 na stronie 107). Ponieważ gdb 7.x obsługuje ładną drukarkę Pythona, może również używać gdb (z obsługą Pythona) do wyświetlania niektórych złożonych wartości zmiennych. zobacz wątek na forum nieoficjalny MinGW GDB gdb z wydanym Pythonem i Użyj GDB python w Codeblocks, aby uzyskać więcej informacji.

Debugowanie pojedynczego pliku

Aby debugować program, **musisz** skonfigurować projekt. Programy jednoplikowe (bez powiązanego projektu) nie są obsługiwane.

Ścieżka ze spacjami

Punkty przerwania nie mogą działać, jeśli ścieżka/folder, który umieściłeś w projekcie, zawiera spacje lub inne znaki specjalne. Dla bezpieczeństwa używaj angielskich liter, cyfr i ‘ ‘.

Rozgałęzienie

Jeśli Twoja aplikacja używa wywołania systemowego „fork”, będziesz mieć problem z zatrzymaniem debugowanego programu lub ustawieniem punktów przerwania w locie. Oto link wyjaśniający tryby forkingu GDB: http://sourceware.org/gdb/onlinedocs/gdb/Forks.html

Aktualizacja do najnowszej wersji MinGW

Od wersji gdb 6.8 wydanej w kwietniu 2008 r. Code::Blocks obsługuje wiele funkcji, które nie istnieją we wczesnych wersjach. Możesz zaktualizować, instalując binaria z pakietów SourceForge MinGW64.

Uwaga:
Pakiet TDM-Mingw był dobrym wyborem aż do wersji 5.1, ale jego rozwój został porzucony.

Użyj 32-bitowego CDB dla programów 32-bitowych i 64-bitowego CDB dla programów 64-bitowych

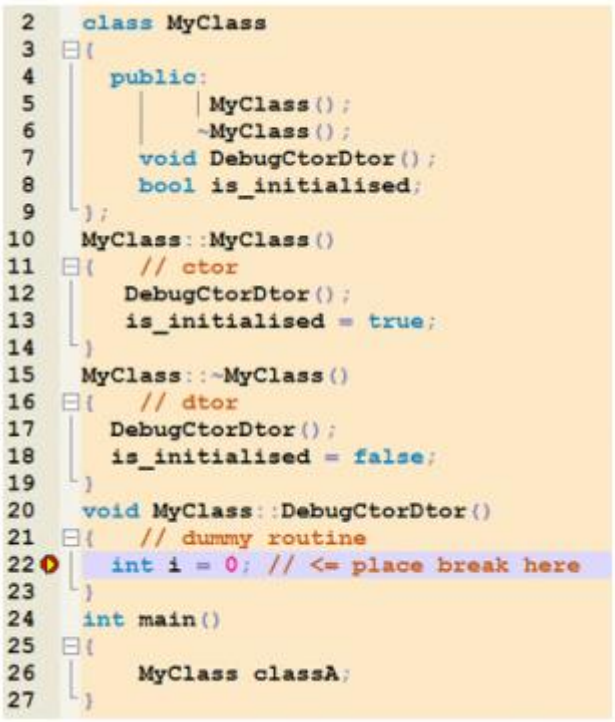
Wygląda na to, że debugowanie programu 32-bitowego z 64-bitowym CDB (Centralna Baza Danych) nie działa w systemie Windows 7 (i nie tylko?), ale 32-bitowy CDB działa idealnie.

Uwaga:
Nie powinno już tak być w przypadku Code::Blocks rev>=10920. Zobacz bilet po szczegóły: #429

Ograniczenia wczesnej wersji MinGW

Jeśli nadal używasz MinGW i gdb 6.7 z plików instalacyjnych 8.02, ustawienie punktów przerwania w konstruktorze nie zadziała. Oto kilka sztuczek:

Punkty przerwania nie działają w konstruktorach lub destruktorach w GDB 6.7 i wcześniejszych wersjach. Działają jednak w ramach wywołanych od nich rutyn. To wczesne ograniczenie GDB, a nie błąd. Możesz więc zrobić coś takiego:



```
2   class MyClass
3   {
4     public:
5         MyClass();
6         ~MyClass();
7       void DebugCtorDtor();
8       bool is_initialised;
9   };
10  MyClass::MyClass()
11  {   // ctor
12     DebugCtorDtor();
13     is_initialised = true;
14  }
15  MyClass::~MyClass()
16  {   // dtor
17    DebugCtorDtor();
18    is_initialised = false;
19  }
20  void MyClass::DebugCtorDtor()
21  {   // dummy routine
22    int i = 0; // <= place break here
23  }
24  int main()
25  {
26      MyClass classA;
27  }
```

Rysunek 4.17: Debugowanie za pomocą starego GDB

...i umieścić punkt przerwania w „DebugCtorDtor” w wierszu „int i = 0;”. Debuger przerwie się w tym wierszu. Jeśli następnie przejdziesz do debugera („Menu Debug” (Menu Debug) → „Nast5ępny wiersz” (Next Line); lub alternatywnie F7) dotrzesz do kodu w konstruktorze/destruktorze (”jest zainicjowany = true/false;”).

4.4 Skrypty debugera

Ta sekcja opisuje skrypty debugera.

4.4.1 Podstawowa zasada skryptu debuggera

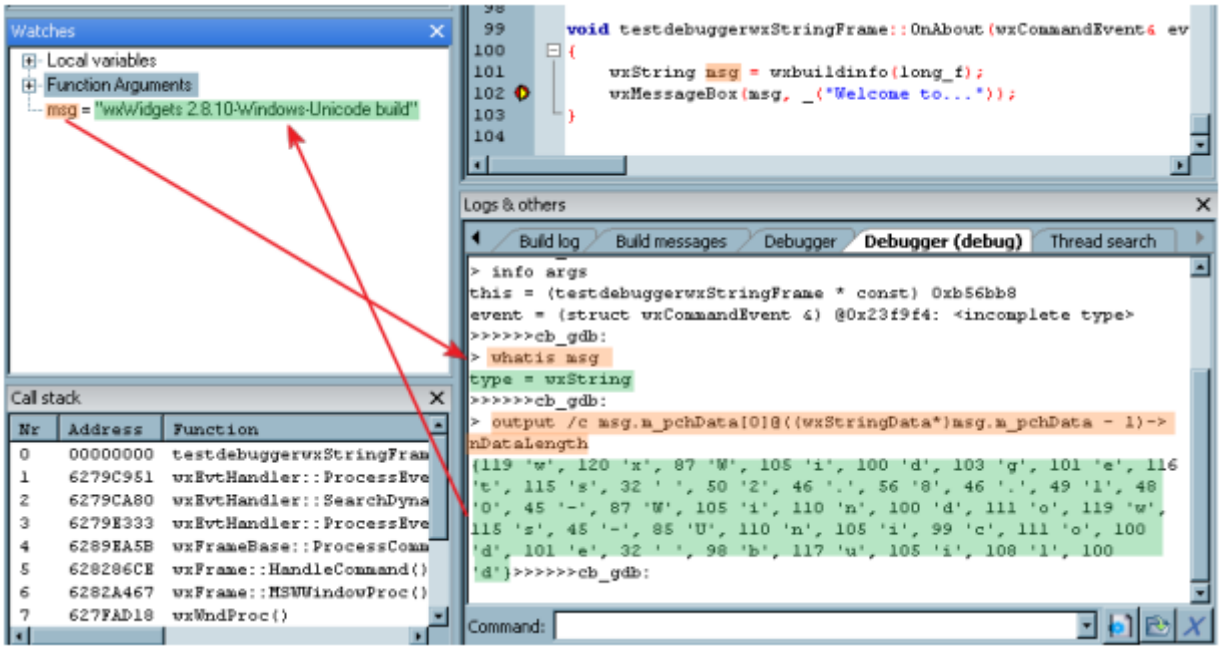


Figure 4.18: Debugger command

Spójrz na powyższy obrazek, to będzie krótkie wprowadzenie do działania skryptu debuggera. Na przykład chcesz wyświetlić zmienną „msg”. Między wtyczką debuggera a gdb są dwa uściski dłoni.

Najpierw wtyczka debuggera wysyła polecenie do gdb, aby zapytać o typ msg.

```
whatis msg
```

wtedy gdb zwróci typ

```
type = wxString
```

Po drugie, debugger sprawdza, czy wxString jest już zarejestrowany i wysyła wynik polecenia
`/c msg.m_pchData [0]@((wxStringData*)msg.m_pchData-1)->nDataLength`

Następnie gdb odpowiada następującym ciągiem:

```
{119 'w', 120 'x', 87 'W', 105 'i', 100 'd', 103 'g', 101 'e', 116 't',
115 's', 32 ' ', 50 '2', 46 '.', 56 '8', 46 '.', 49 '1', 48 '0', 45 '-',
87 'W', 105 'i', 110 'n', 100 'd', 111 'o', 119 'w', 115 's', 45 '-',
85 'U', 110 'n', 105 'i', 99 'c', 111 'o', 100 'd', 101 'e', 32 ' ',
98 'b', 117 'u', 105 'i', 108 'l', 100 'd'}
```

Na koniec wartość jest wyświetlana w oknie obserwacji (watch window).

4.4.2 Funkcje skryptów

Skrypty debugera są podobne do Visual Studio Debugger Visualizer. Pozwalają na napisanie małego fragmentu kodu, który jest wykonywany przez debuger za każdym razem, gdy próbujesz wyświetlić określony typ zmiennych i może być użyty do wyświetlenia niestandardowego tekstu z ważnymi informacjami, których potrzebujesz.

Cytat z Game_Ender - 23 marca 2006

Nie sądzę, aby można było otworzyć kolejne okno, aby coś sobie wyobrazić.

Zobaczmy, jak to działa. Wszystko znajduje się w jednym pliku w folderze scripts/ o nazwie gdb types.script :). W przyszłości planowana jest obsługa większej liczby (zdefiniowanych przez użytkownika) skryptów.

Ten skrypt jest wywoływany przez Code::Blocks w dwóch miejscach:

1. po uruchomieniu GDB. Wywołuje funkcję skryptową RegisterTypes(), aby zarejestrować wszystkie typy zdefiniowane przez użytkownika znane przez debugger Code::Blocks.

2. ilekroć GDB napotka typ zmiennej, wywołuje funkcje skryptu specyficzne dla tego typu danych (zarejestrowane w RegisterTypes() - więcej na ten temat poniżej).

To jest przegląd. Przeanalizujmy dostarczony plik gdb types.script i zobaczmy, jak dodaje obsługę std :: string do GDB.

```
// Rejestruje nowe typy ze sterownikiem
function RegisterTypes (driver)
{
// podpis:
// driver.RegisterType (type_name , regex , eval._func , parse_func);

// Ciąg STL
driver.RegisterType(
        _T(”STL String”),
        _T(”[ ^ [:alnum:] _ ]+string[ ^ [:alnum:] _ ]*”),
        _T(”Evaluate_StlString”),
        _T(”Parse_StlString”)
    );
}
```

Parametr „driver” to sterownik debuggera, ale nie musisz się tym przejmować :) (obecnie działa tylko z GDB). Ta klasa zawiera jedną metodę: RegisterType. Oto jej deklaracja C++:

```
void GDB_driver : : RegisterType ( const wxS tring& name , const wxString& regex ,
                                   const wxString&eval_func , const wxString&parse_func)
```

Tak więc w powyższym kodzie skryptu zarejestrowany jest typ „STL String” (tylko nazwa - nieważne jaka), dostarczając ciąg wyrażenia regularnego do dopasowania wtyczki debugera, a na końcu dostarcza nazwy dwóch obowiązkowych funkcji potrzebnych dla każdego zarejestrowanego typu:

1. funkcja oceny: musi zwrócić polecenie zrozumiałe dla właściwego debuggera (w tym przypadku GDB). Dla „STL string” (łańcucha STL) funkcja oceniania zwraca polecenie „output” (wyjście) GDB, które zostanie wykonane przez debugger GDB.

2. funkcja parsera: gdy debugger uruchomi polecenie zwrócone przez funkcję oceny, przekazuje wynik do tej funkcji w celu dalszego przetwarzania. To, co zwraca ta funkcja, jest faktycznie wyświetlane przez Code::Blocks (zwykle w oknie obserwacji (watches window) lub w podpowiedzi).

Zobaczmy funkcję oceny dla std :: string:

```
function Evaluate_StlString(type, astr, start, count)
{
	local oper = _T(”.”);

	if (type.Find ( _T ( ”*” ) ) > 0 )
		oper = _T ( ”->” ) ;

	local result = _T(”output”) + a_str + oper + _T(”c_str()[”)
			+ start + _T(”]@”);
	if (count != 0)
		result = result + count ;
	else
		result = result + a_str + oper + _T(”size()”);
	return result;
}
```

Nie będę wyjaśniał, co zwraca ta funkcja. Powiem tylko, że zwraca polecenie GDB, które spowoduje, że GDB wydrukuje rzeczywistą zawartość std :: string. *Tak, musisz znać swój debugger i jego polecenia, zanim spróbujesz go rozszerzyć.*

Powiem ci jednak, jakie są te argumenty funkcji.

- type: typ danych, np. ”char*”, ”const string” itp.

- a_str: nazwa zmiennej GDB, którą próbuje oszacować.

- start: offset początkowy (używany dla tablic).

- count: liczba zaczynająca się od offsetu początkowego (używana w przypadku tablic).

Zobaczmy teraz odpowiednią funkcję parsera:

```
function Parse_StlString(a_str, start)
{
	// nic nie trzeba robić
	return a_str;
}
```

- a_str: zwrócony ciąg znaków, gdy GDB uruchomił polecenie zwrócone przez funkcję oceny. W przypadku std :: string jest to zawartość ciągu.

- start: offset początkowy (używany dla tablic).

Cóż, w tym przykładzie nic nie trzeba robić. ”a_str” zawiera już zawartość std: string, więc po prostu ją zwracamy :)

Proponuję przestudiować, w jaki sposób wxString jest rejestrowany w tym samym pliku, jako bardziej złożony przykład.

4.5 Code::Blocks i pliki Makefile

Ta sekcja opisuje, jak używać pliku makefile w Code::Blocks na przykładzie wxWidgeys.

4.5.1 Artykuł Wiki

Autor : Gavrillo 22:34, 21 maja 2010 (UTC)

CB domyślnie nie używa plików makefile. Ma własne pliki projektu .cbp, które automatycznie wykonują to samo. Istnieje kilka powodów, dla których możesz chcieć użyć makefile. Być może migrujesz projekt, który ma plik makefile do Code::Blocks. Inną możliwością jest wyjęcie projektu z Code::Blocks.

Konieczność użycia preprocesora nie jest ważnym powodem do używania pliku makefile, ponieważ CB ma opcję budowania pre/post. Z menu „Project” (Projekt) → „Build Options” (Opcje budowania) pojawia się zakładka z krokami przed/po kompilacji, które można w tym celu wykorzystać.

Ten rozdział zajmuje się dokładniej plikami makefile używającymi mingw32-make 3.81, CB 8.02 i Wxwidgets 2.8 w systemie Windows Vista, chociaż jestem pewien, że większość z nich będzie miała zastosowanie do innych konfiguracji.

Jeśli zdecydujesz, że chcesz użyć własnego pliku makefile, musisz wejść na ekran z „Project” (Projekt) → „Properties” (Właściwości) i zobaczyć pole wyboru „this is a custom makefile” (to jest niestandardowy plik makefile). Zaznacz to pole, upewnij się, że nazwa tuż nad nim jest tą, którą chcesz mieć dla swojego pliku makefile.

Powinieneś także spojrzeć na „Project” (Projekt) → „Build Options” (Opcje budowania). Istnieje zakładka o nazwie „Make commands” (Wykonywanie poleceń) (musisz przewinąć karty w poziomie, aby się do niej dostać). W polu 'build project/target' powinieneś zobaczyć linię $make =f $makefile $target. Zakładając, że jesteś w trybie debugowania, $target prawdopodobnie będzie nazywał się „debug”, co prawdopodobnie nie jest tym, czego chcesz. Powinieneś zmienić $target na nazwę pliku wyjściowego (z rozszerzeniem .exe i bez początkowego $).

Innym przydatnym dodatkiem jest „Project” (Projekt) → „Project Tree” (Drzewo projektu) → „Edit File types and categories” (Edytuj typy i kategorie plików). Jeśli dodasz pliki makefile z maską *.mak (CB wydaje się preferować .mak niż .mk) będziesz mógł dodać swój plik makefile z rozszerzeniem .mak do projektu i pojawi się on w panelu zarządzania projektem po lewej stronie.

Zakładając, że zamierzasz edytować plik makefile w CB, powinieneś upewnić się, że edytor używa tabulatorów (w przeciwieństwie do spacji). Jest to ogólny problem z make, ponieważ musi rozpoczynać wiersze poleceń od znaku tabulacji, a wiele edytorów zastępuje tabulatory spacjami. Aby ustawić to w CB, otwórz okno „Settings” (Ustawienia) → „Editor” (Edytor) i zaznacz pole wyboru dla znaku tabulatora.

Jednak prawdziwe problemy zaczynają się teraz. Automatyczny makefile CB dodaje wszystkie nagłówki dla Wxwidgets, ale jeśli używasz makefile, wszystko to jest wyłączone i musisz to zrobić sam.

Na szczęście CB ma jeszcze jedną funkcję, która może ci pomóc. Jeśli przejdziesz do menu „Settings” (Ustawienia) → „Compiler and Debugger” (Kompilator i Debugger), przewiń zakładki poziomo do prawego końca, a znajdziesz zakładkę „other settings” (inne ustawienia). Tam kliknij pole wyboru „Save build to HTML ...” (Zapisz kompilację do HTML...). To spowoduje, że CB utworzy w czasie kompilacji plik HTML, który rejestruje wszystkie polecenia kompilacji.

Uwaga:
ten sposób tworzenia pliku html nie istnieje w najnowszej wersji CB, ale są inne rozwiązania

Jeśli kompilujesz (bez użycia pliku makefile - więc jeśli już wszystko zresetowałeś - przepraszam) domyślny minimalny program wxWwidgets, możesz zobaczyć polecenia kompilatora i konsolidatora, które tworzą ten plik.

Zakładając, że zamierzasz użyć tego jako podstawy swojego projektu, możesz użyć zawartości pliku HTML utworzonego jako podstawy swojego pliku makefile.

Nie możesz po prostu skopiować go z przeglądarki HTML w CB (w CB nie ma takiej funkcji), ale możesz załadować plik do przeglądarki lub edytora i skopiować go stamtąd. Można go znaleźć w katalogu projektu za pomocą <the_same_name_as_your_project>_build_log.HTML. Niestety będzie to wymagało niewielkich poprawek, jak pokazano poniżej:

Oto kopia pliku kompilacji dla podstawowego programu Wxwidgets opisanego powyżej.

Uwaga:
dla lepszej wiarygodności, długie kolejki zostały podzielone. Znak ^ wskazuje linię kontynuacji w trybie DOS, a znak \ wskazuje linię kontynuacji w pliku makefile. Ale powinieneś mieć polecenia tylko w jednej linii, o ile usuniesz znaki linii kontynuacji

```
mingw32-make.exe -f test.mak test.exe

mingw32-g++.exe -pipe -mthreads -D__GNUWIN32__ -D__WXMSW__ -DWXUSINGDLL         ^
        -DwxUSE_UNICODE -Wall -g -D__WXDEBUG__ -IC:\PF\wxWidgets2.8\include      ^
        -IC:\PF\wxWidgets2.8\contrib\include -IC:\PF\wxWidgets2.8\lib\gcc_dll\mswud      ^
        -c C:\Development\test\testApp.cpp -o obj\Debug\testApp.o

mingw32-g++.exe -pipe -mthreads -D__GNUWIN32__ -D__WXMSW__ -DWXUSINGDLL         ^
        -DwxUSE_UNICODE -Wall -g -D__WXDEBUG__ -IC:\PF\wxWidgets2.8\include      ^
        -IC:\PF\wxWidgets2.8\contrib\include -IC:\PF\wxWidgets2.8\lib\gcc_dll\mswud      ^
        -c C:\Development\test\testMain.cpp -o obj\Debug\testMain.o

windres -IC:\PF\wxWidgets2.8\include -IC:\PF\wxWidgets2.8\contrib\include        ^
        -IC:\PF\wxWidgets2.8\lib\gcc_dll\mswud -iC:\Development\test\resource.rc      ^
        -o obj\Debug\resource.coff
```

```
mingw32-g++.exe -LC:\PF\wxWidgets2.8\lib\gcc_dll -o bin\Debug\test.exe                ^
        obj\Debug\testApp.o obj\Debug\testMain.o obj\Debug\resource.coff              ^
        -lwxmsw28ud –mwindows

Process terminated with status 0 (0 minutes, 12 seconds)
0 errors, 0 warnings
```

Powyższe można przekonwertować na poniższy plik makefile. Celowo pozostawiłem go dość blisko wyjścia pliku HTML.

```
# test program makefile

Incpath1 = C:\PF\wxWidgets2.8\include
Incpath2 = C:\PF\wxWidgets2.8\contrib\include
Incpath3 = C:\PF\wxWidgets2.8\lib\gcc_dll\mswud

Libpath = C:\PF\wxWidgets2.8\lib\gcc_dll

flags = -pipe -mthreads -D__GNUWIN32__ -D__WXMSW__ -DWXUSINGDLL \ -
DwxUSE_UNICODE -Wall -g -D__WXDEBUG__

CXX = mingw32-g++.exe

test.exe : obj\Debug\testApp.o obj\Debug\testMain.o obj\Debug\resource.coff
        $(CXX) -L$(Libpath) -o bin\Debug\test.exe obj\Debug\testApp.o                 \
        obj\Debug\testMain.o obj\Debug\resource.coff -lwxmsw28ud -mwindows

obj\Debug\testMain.o : C:\Development\test\testMain.cpp
        $(CXX) $(flags) -I$(Incpath1) -I$(Incpath2) -I$(Incpath3)                    \
-c C:\Development\test\testMain.cpp -o obj\Debug\testMain.o

obj\Debug\testApp.o : C:\Development\test\testApp.cpp
        $(CXX) $(flags) -I$(Incpath1) -I$(Incpath2) -I$(Incpath3)                    \
        -c C:\Development\test\testApp.cpp -o obj\Debug\testApp.o

obj\Debug\resource.coff : C:\Development\test\resource.rc
        windres -I$(Incpath1) -I$(Incpath2) -I$(Incpath3)                            \
        -iC:\Development\test\resource.rc -oobj\Debug\resource.coff

# original output from codeblocks compilation
# note I’ve had to add compiling the .res file
#
# mingw32-g++.exe -pipe -mthreads -D__GNUWIN32__ -D__WXMSW__ -DWXUSINGDLL      ^
#       -DwxUSE_UNICODE -Wall -Wall -g -D__WXDEBUG__                           ^
#       -Wall -g -IC:\PF\wxWidgets2.8\include -IC:\PF\wxWidgets2.8\contrib\include  ^
#       -IC:\PF\wxWidgets2.8\lib\gcc_dll\mswud                                 ^
#       -c C:\Development\test\testApp.cpp -o obj\Debug\testApp.o
```

```
# mingw32-g++.exe -pipe -mthreads -D__GNUWIN32__ -D__WXMSW__ -DWXUSINGDLL     ^
#         -DwxUSE_UNICODE -Wall -Wall -g -D__WXDEBUG__                        ^
#         -Wall -g -IC:\PF\wxWidgets2.8\include -IC:\PF\wxWidgets2.8\contrib\include  ^
#         -IC:\PF\wxWidgets2.8\lib\gcc_dll\mswud                              ^
#         -c C:\Development\test\testMain.cpp -o obj\Debug\testMain.o

# mingw32-g++.exe -LC:\PF\wxWidgets2.8\lib\gcc_dll -o bin\Debug\test.exe      ^
#         obj\Debug\testApp.o obj\Debug\testMain.o                            ^
#         obj\Debug\resource.res -lwxmsw28ud -mwindows
```

Napisałem ogólny plik makefile, który testowałem tylko w systemie Windows Vista, ale powinien działać z każdym projektem rozpoczętym w sposób opisany powyżej. Musisz zmienić nazwę projektu i odpowiednio ustawić ścieżki (prawdopodobnie wystarczy zmienić Ppath i WXpath).

```
# Generic program makefile
# -- assumes that you name your directory with same name as the project file
# -- eg project test is in \test\

# Project name and version
Proj := test
Version := Debug

#paths for Project (Ppath) Object files (Opath) and binary path (Bpath)
Ppath := C:\Development\$(Proj)
Opath := obj\$(Version)
Bpath := bin\$(Version)

#Library & header paths
WXpath := C:\PF\wxWidgets2.8
IncWX := $(WXpath)\include
IncCON := $(WXpath)\contrib\include
IncMSW := $(WXpath)\lib\gcc_dll\mswud
Libpath := $(WXpath)\lib\gcc_dll

flags = -pipe -mthreads -D__GNUWIN32__ -D__WXMSW__ -DWXUSINGDLL -
DwxUSE_UNICODE \
-Wall -g -D__WXDEBUG__

CXX = mingw32-g++.exe

Obj := $(Opath)\$(Proj)App.o $(Opath)\$(Proj)Main.o $(Opath)\resource.coff

$(Proj).exe : $(Obj)
        $(CXX) -L$(Libpath) -o $(Bpath)\$(Proj).exe $(Obj) -lwxmsw28ud -mwindows
```

```
$(Opath)\$(Proj)Main.o : $(Ppath)\$(Proj)Main.cpp
	$(CXX) $(flags) -I$(IncWX) -I$(IncCON) -I$(IncMSW) -c $^ -o $@

$(Opath)\$(Proj)App.o : C:\Development\$(Proj)\$(Proj)App.cpp
	$(CXX) $(flags) -I$(IncWX) -I$(IncCON) -I$(IncMSW) -c $^ -o $@

$(Opath)\resource.coff : C:\Development\$(Proj)\resource.rc
	windres -I$(IncWX) -I$(IncCON) -I$(IncMSW) -i$^ -o $@

.PHONEY : clean

clean:
	del $(Bpath)\$(Proj).exe $(Obj) $(Opath)\resource.coff
```

Uwaga:
eksport makefile projektu Code::Blocks jest pośrednio możliwy. Można to osiągnąć za pomocą narzędzia cbp2make (patrz opis w sekcji 4.6 na stronie 115 i/lub przykład użycia w Tool+, podrozdział 2.17.1 na stronie 71).

Ostatnia uwaga. Kiedy już użyjesz makefile, próby skompilowania czegokolwiek bez makefile skutkują kilkoma ostrzeżeniami. Jedynym sposobem, w jaki znalazłem rozwiązanie tego problemu, jest ponowna instalacja CB. Jeśli potrzebujesz obu, być może możliwe jest zainstalowanie 2 wersji CB.

Uwaga:
to jest uwaga oryginalnego tekstu na Wiki. Ale jako gd on user nigdy nie miałem takiego zachowania

Ogólnie rzecz biorąc, nie zaleca się używania pliku makefile, ale jeśli istnieje jakiś ważny powód, aby to zrobić, to jest to sposób.

4.5.2 Uzupełnienia

Domyślnie Code::Blocks buduje cel „Release” i „Debug”. W twoim Makefile te cele mogą nie być obecne. Ale być może masz cel „All” (Wszystko) (lub „all”). Możesz zmienić nazwę w Code::Blocks celu (lub dodać go) na nazwę podaną w pliku Makefile.

Co więcej, Twój Makefile buduje plik wykonywalny o określonej nazwie w określonym folderze. Powinieneś dostosować w Code::Blocks ścieżkę i nazwę pliku wykonywalnego. W ten sposób Code::Blocks, ponieważ nie zna ani nie analizuje pliku Makefile, znajdzie plik wykonywalny, a zielona strzałka Execute w menu może działać (lub Ctrl-F10).

4.6 Narzędzie Cbp2make

Narzędzie do generowania plików Makefile dla Code::Blocks IDE firmy Mirai Computing. Tekst tej sekcji pochodzi z jego cbp2make Wiki na SourceForge.

Uwaga:
Cbp2make nie jest wtyczką Code::Blocks, ale pełną aplikacją konsolową, umieszczoną w głównym katalogu Code::Blocks, do generowania plików makefile z wewnętrznego systemu generującego Code::Blocks

4.6.1 Informacje

„cbp2make" to samodzielne narzędzie do budowania, które umożliwia generowanie plików makefile dla projektu GNU Make out of Code::Blocks projektu IDE lub pliku obszaru roboczego. (Zobacz także http://fora.codeblocks.org/index.php/topic,13675.0.html])

4.6.2 Użycie

Utwórz plik makefile dla pojedynczego projektu lub obszaru roboczego

Załóżmy, że masz projekt „mój projekt.cbp" i potrzebujesz pliku makefile dla tego projektu. W tym najprostszym przypadku wystarczy:

cbp2make =in my_project.cbp

To samo dotyczy obszarów roboczych.

cbp2make =in my_projects.workspace

Utwórz plik makefile z inną nazwą pliku

Domyślnie „cbp2make" doda rozszerzenie „.mak" do nazwy projektu, aby utworzyć nazwę pliku dla makefile. Jeśli chcesz to zmienić, użyj następującego polecenia:

cbp2make =in my_project.cbp =out Makefile

Utwórz plik makefile dla innej platformy

Jeśli pracujesz w systemie GNU/Linux i chcesz wygenerować plik makefile dla Windows lub odwrotnie, możesz określić jedną lub więcej platform, dla których potrzebujesz plików makefile.

cbp2make =in my_project.cbp =windows
cbp2make =in my_project.cbp =unix
cbp2make =in my_project.cbp =unix =windows =mac
cbp2make =in my_project.cbp ==all =os

"cbp2make" doda przyrostek „.unix" lub „windows" lub „mac" odpowiednio do nazwy pliku makefile dla każdej platformy.

Utwórz plik makefile dla wielu projektów lub obszarów roboczych

Jeśli masz więcej niż jeden niezależny projekt lub obszar roboczy, możesz je przetwarzać jednocześnie, ale będziesz potrzebował pliku tekstowego zawierającego listę projektów, np. „projects.lst", z jedną nazwą pliku projektu w wierszu.

```
my_project.cbp
my_other_project.cbp
```

A potem możesz je przetworzyć za pomocą polecenia:

cbp2make = list =in projects.lst

4.6.3 Konfiguracja

Niektóre opcje związane z instalacją lub projektem, głównie ustawienia łańcucha narzędzi, można zapisać w pliku konfiguracyjnym. Domyślnie (od wersji 110), cbp2make nie zapisuje żadnych ustawień w pliku konfiguracyjnym, chyba że użytkownik wyraźnie określi opcję "--config". Plik konfiguracyjny może być globalny (przechowywany w profilu użytkownika / katalogu domowym) lub lokalnym (przechowywany w bieżącym katalogu).

Należy pamiętać, że ponieważ cbp2make jest na wczesnym etapie rozwoju, stary plik konfiguracyjny może stać się niekompatybilny z nową wersją narzędzia i może być konieczne jego ręczne zaktualizowanie lub zainicjowanie nowego.

Inicjalizacja

cbp2make --config options --global
cbp2make --config options --local

Późniejsze użycie

Kiedy cbp2make jest wywoływane, najpierw próbuje załadować lokalny plik konfiguracyjny. W przypadku braku konfiguracji lokalnej następną próbą będzie wczytanie konfiguracji globalnej. Jeśli i ta próba się nie powiedzie, używana jest domyślna konfiguracja wbudowana. Kolejność wyszukiwania konfiguracji można nadpisać za pomocą opcji wiersza polecenia „--local” lub „--global”. Jeśli jedna z opcji jest dostarczona do cbp2make, nieokreślona konfiguracja nie zostanie pobrana, nawet jeśli podanej brakuje, a nieokreślona istnieje.

Domyślna kolejność wyszukiwania

cbp2make -in project.cbp -out Makefile}

Jawnie określona konfiguracja

cbp2make --local -in project.cbp -out Makefile
cbp2make --global -in project.cbp -out Makefile

4.6.4 Składnia wiersza poleceń

Wygeneruj plik makefile:

cbp2make –in <project_file> [-cfg <configuration>] [-out <makefile>]
[-unix] [-windows] [-mac] [--all-os] [-targets "<target1> [,<target2> [, ...]]"]
[--flat-objects] [--flat-objpath] [--wrap-objects] [--wrap-options]
[--with-deps] [--keep-objdir] [--keep-outdir] [--target-case keep|lower|upper]

cbp2make -list –in <project_file_list> [-cfg <configuration>]
[-unix] [-windows] [-mac] [--all-os] [-targets "<target1>[,<target2> [, ...]]"]
[--flat-objects] [--flat-objpath] [--wrap-objects] [--wrap-options]
[--with-deps] [--keep-objdir] [--keep-outdir] [--target-case keep|lower|upper]

Zarządzaj łańcuchami narzędzi:

cbp2make --config toolchain --add \[-unix|-windows|-mac\] –chain <toolchain>
cbp2make --config toolchain --remove \[-unix|-windows|-mac\] –chain <toolchain>

Zarządzaj narzędziami do budowania:

cbp2make --config tool --add \[-unix|-windows|-mac\] –chain <toolchain>
	-tool <tool> -type <type> <tool options>

cbp2make --config tool --remove \[-unix|-windows|-mac\] –chain <toolchain>
	-tool <tool>

Rodzaje narzędzi:

pp=preprocessor as=assembler cc=compiler rc=resource compiler
sl=static linker dl=dynamic linker el=executable linker
nl=native linker

Opcje narzędzi (wspólne):

-desc <description> -program <executable> -command <comand_template>
-mkv <make_variable> -srcext <source_extensions> -outext <output_extension>
-quotepath <yes|no> -fullpath <yes|no> -unixpath <yes|no>

Opcje narzędzi (kompilator):

	-incsw <include_switch> -defsw <define_switch> -deps <yes|no>
\end{lstlisting}

Opcje narzędzi (linker):
\begin{verbatim}
	-ldsw library_dir_switch> -llsw <link_library_switch> -lpfx <library_prefix>
	-lext <library_extension> -objext <object_extension> -lflat <yes|no>

Zarządzaj platformami:

cbp2make --config platform \[-unix|-windows|-mac\] \[-pwd <print_dir_ command>\]
	\[-cd <change_dir_command>\] \[-rm <remove_file_command>\]
	\[-rmf <remove_file_forced>\] \[-rmd <remove_dir_command>\]
	\[-cp <copy_file_command>\] \[-mv <move_file_command> \]
	\[-md <make_dir_command>\] \[-mdf <make_dir_forced>\]
	\[-make <default_make_tool>\]

Zarządzaj globalnymi zmiennymi kompilatora:

cbp2make --config variable --add \[-set <set_name>\] –name <var_name>
\[-desc <description>\] \[-field <field_name>\] –value <var_value>

cbp2make --config variable --remove \[-set <set_name>\] \[-name <var_name>\]
\[-field <field_name>\]

Zarządzaj opcjami:

cbp2make --config options --default-options "<options>"
cbp2make --config show

Wspólne opcje:

cbp2make –local // użyj konfiguracji z bieżącego katalogu
cbp2make –global // użyj konfiguracji z katalogu domowego
cbp2make –verbose // pokaż informacje o projekcie
cbp2make –quiet // ukryj wszystkie wiadomości
cbp2make –help // wyświetl tę wiadomość
cbp2make –version // wyświetl informacje o wersji

Opcje
„Generowanie plików Makefile”

-in <project_file> // określa plik wejściowy lub listę plików;

-cfg <configuration> // określa plik konfiguracyjny , zobacz także opcje „--local” i „--global”;

-out <makefile> // określa nazwę pliku makefile lub listę plików makefile

-unix // włącza generowanie plików makefile zgodnych z Unix / Linux;

-windows // włącza generowanie plików makefile zgodnych z Windows;

-mac // włącza generowanie plików makefile zgodnych z Macintosh;

--all-os // włącza wszystkie platformy docelowe jednocześnie;

-targets ”<target1>[,<target2>[, . . .]]” // określa jedyne cele kompilacji, dla których zostanie
// utworzony plik makefile;

--flat-objects // wymusza „płaskie” nazwy plików obiektowych z ograniczonym
// zestawem znaków;

```
--flat-objpath              // wymusza „płaskie” ścieżki dla plików obiektowych bez podkatalogów;

--wrap-objects              // pozwala na użycie wieloliniowych list obiektów, co ułatwia odczytanie
                            // pliku makefile;

--wrap-options              // pozwala na użycie makr wieloliniowych ;

--with-deps                 // umożliwia działanie wbudowanego skanera zależności dla projektów
                            // C/C++;

--keep-objdir               // wyłącza polecenie, które usuwa katalogi plików obiektowych w
                            // „clean” (czysty) celu;

--keep-outdir               // wyłącza polecenie kasujące katalog dla wyjściowego pliku binarnego
                            // w celu „clean”;

--target -case [ keep | lower | upper ]          // określa styl dla celów makefile ;
```

4.7 Code::Blocks. Internacjonalizacja interfejsu

W tej sekcji opisano, jak uzyskać i używać zlokalizowanej wersji Code::Blocks.

Interfejs Code::Blocks może być prezentowany w Twoim języku. Wiele łańcuchów używanych wewnętrznie w interfejsie Code::Blocks jest wprowadzonych za pomocą makra wxWidgets : . Łańcuchy, które nie zmieniają się wraz z językiem, są zwykle wprowadzane przez makro wxT() lub T(). Aby uzyskać interfejs Code::Blocks prezentowany w twoim języku, musisz po prostu powiedzieć Code::Blocks, że plik językowy jest dostępny. Aby był zrozumiały dla Code::Blocks, musi to być plik .mo uzyskany po „kompilacji” pliku .po. Takie pliki są dostępne na forum dla „French” oraz w serwisie Launchpad dla szerszego zestawu języków.

Oryginalna witryna Launchpad jest już przestarzała: https://launchpad.net/codeblocks

Tematem forum dotyczącym tłumaczenia jest
http://forums.codeblocks.org/index.php/topic,1022.0.html. Jeśli jesteś zainteresowany, możesz również znaleźć narzędzia do wyodrębniania ciągów z plików źródłowych Code::Blocks. Narzędzia te tworzą plik .pot, który można następnie uzupełnić tłumaczeniami, aby utworzyć plik .po.

Niedawno utworzono nową stronę internetową https://launchpad.net/codeblocks-gd.
Zawiera ponad 9300 ciągów, chociaż oryginalna witryna miała tylko 2173! Wiele prac zostało wykonanych na Code::Blocks

Na stronie tłumaczenia wybierz „View All languages” (Wyświetl wszystkie języki) na dole po prawej stronie.

Na tej nowej stronie zostały zaimportowane stare tłumaczenia, tylko najczęściej używane języki (obecnie 14 języków). Na życzenie istnieje możliwość dodania nowych języków (ale tłumacze będą mieli trochę więcej pracy!).
Przepraszamy za to, ale oryginalne nazwiska tłumaczy w wielu przypadkach zostały utracone :-[.
Język francuski ma największą liczbę przetłumaczonych napisów. Szablon (plik .pot) został zaktualizowany dla ostatnich wersji svn, a Launchpad zawiera wykonane do tej pory tłumaczenia.

W przypadku języka rosyjskiego użyto również całkiem nowej strony internetowej, ale nie jest ona aktualna. Wiele tłumaczeń wymaga zatwierdzenia, ale nie jestem do tego odpowiednim facetem! Strona startera jest otwierana jako „ustrukturyzowana". Powinieneś więc być w stanie zaproponować nowe tłumaczenia lub je poprawić. W niektórych przypadkach powinny one zostać przez kogoś zatwierdzone przed publikacją.
Postaram się zachować „szablon", gdy będą dostępne nowe angielskie ciągi.

Wy (tłumacze) powinniście mieć możliwość wzięcia udziału. Wystarczy mieć (lub utworzyć) konto launchpad (Ubuntu).

Inni użytkownicy mogą poprosić o pobranie pliku .po lub .mo. Jest to ten ostatni (plik .mo), forma binarna, której możesz użyć do uzyskania interfejsu Code::Blocks w swoim własnym języku: po prostu umieść go w swoim "katalogu instalacyjnym codeblocks"/share/CodeBlocks/locale/"łańcuch językowy " (dla mnie w systemie Windows to
`C:\Program Files\CodeBlocks wx313 64\share\CodeBlocks\locale\fr FR`
Następnie w menu Settings/Environment.../View (Ustawienia/Środowisko.../Widok) powinieneś być w stanie wybrać język.

Trochę więcej szczegółów dotyczących używania przetłumaczonych ciągów menu w Code::Blocks.

Tylko dla użytkowników tłumaczeń:

Pobierz plik w formacie .mo w żądanym języku. Nazwa startera może wyglądać tak: de LC MESSAGES All codeblocks.mo (tutaj w języku niemieckim).

Powinieneś umieścić ten plik w katalogu instalacyjnym codeblocks.

W systemie Windows jest to coś takiego:
C:\Program Files (x86)\CodeBlocks\share\CodeBlocks\locale\xxxx dla 32 bitów
lub
C:\Program Files\CodeBlocks\share\CodeBlocks\locale\xxxx dla 64 bitów

Ścieżka pod Linuksem jest dość podobna.

xxxx musi być dostosowany do Twojego języka. To jest:

- de_DE dla języka niemieckiego,
- es_ES dla języka hiszpańskiego,
- fr_FR dla francuskiego,
- to_IT dla języka włoskiego,
- lt_LT dla języka litewskiego,
- nl_NL dla języka niderlandzkiego,
- pl_PL dla języka polskiego,
- pt_BR dla portugalskiego brazylijskiego,
- pt_PT dla języka portugalskiego,
- ru_RU dla rosyjskiego,
- si dla syngaleskiego,
- zh_CN dla języka chińskiego uproszczonego,
- zh_TW dla chińskiego tradycyjnego.

W razie potrzeby utwórz podkatalogi. Następnie umieść pobrany plik .mo tutaj. Możesz pozostawić nazwę pliku bez zmian lub zachować tylko pierwsze litery (jak chcesz), ale zachować rozszerzenie .mo.

Następnie uruchom Code::Blocks. W Ustawieniach/Środowisku/Widoku powinieneś być w stanie zaznaczyć pole języka (internacjonalizacja) i wybrać swój język. Jeśli nie, prawdopodobnie zapomniałeś o czymś lub popełniłeś błąd.
Zrestartuj Code::Blocks, aby aktywować nowy język.

Jeśli chcesz wrócić do języka angielskiego, po prostu usuń zaznaczenie pola wyboru języka.

Dla tłumaczy:

Możesz bezpośrednio pracować w launchpadzie.

Problem: interfejs nie jest zbyt przyjazny.

Możesz także pobrać plik .po, pracować nad nim np. za pomocą poedit (darmowa wersja jest OK). Możesz przetestować swoje tłumaczenia lokalnie, kompilując (tworząc plik .mo) i instalując plik .mo w odpowiednim podkatalogu Code::Blocks.

Gdy zrobisz wystarczające postępy (to Twoja decyzja), możesz wgrać plik .po w launchpadzie. Może być konieczne zatwierdzenie Twojej pracy lub oznaczenie jej jako do recenzji.

Nie bój się: to dość długa praca. W starej witrynie do przetłumaczenia było 2173 napisów. Teraz to ponad 9300 ! Ale praca może być dzielona, Launchpad jest do tego gotowy!

Wskazówka : Zacznij od często używanych menu : szybciej zobaczysz postępy.

5 Instalacja i konfiguracja CodeBlocks z MinGW

Ten rozdział opisuje jak zainstalować i skonfigurować Code::Blocks. Proces instalacji jest tutaj opisany dla systemu Windows, ale może być dostosowany do innych systemów operacyjnych.

5.1 Instalacja najnowszej oficjalnej wersji Code::Blocks w systemie Windows

Zainstaluj kroki:

- Pobierz instalator Code::Blocks (https://codeblocks.org/downloads/26). Jeśli wiesz, że nie masz zainstalowanego MinGW lub nie wiesz, który wybrać, **pobierz pakiet, który zawiera MinGW**. Dla wersji 20.03 nazwa instalatora to: codeblocks-20.03mingw-setup.exe. Poprzednia wersja została zidentyfikowana do 17.12.

- Uruchom instalator, jest to standardowy instalator dla systemu Windows; po prostu naciśnij Next (Dalej) po przeczytaniu każdego ekranu.

- Jeśli planujesz zainstalować kompilator po zainstalowaniu Code::Blocks, przeczytaj informacje zawarte w instalatorze.

- Jeśli pobrałeś instalator, który nie jest dostarczany z MinGW, być może będziesz musiał ręcznie skonfigurować kompilator (zwykle Code::Blocks automatycznie wykrywa kompilator).

W następnej sekcji zobaczymy, jak zainstalować i skonfigurować inny kompilator.

Uwagi:

- Plik codeblocks-20.03-setup.exe zawiera Code::Blocks ze wszystkimi wtyczkami. Plik codeblocks-20.03-setup-nonadmin.exe jest udostępniany dla wygody użytkowników, którzy nie mają uprawnień administratora na swoich komputerach.

- Plik codeblocks-20.03mingw-setup.exe dodatkowo zawiera kompilator GCC/G++ i debugger GDB w wersji MinGW64 8.1.0, 64 bit, seh. Plik codeblocks20.03mingw fortran setup.exe zawiera dodatkowo kompilator Gfortran

- Pliki codeblocks-20.03(mingw)-nosetup.zip są dostarczane dla wygody użytkowników uczulonych na instalatorów. Nie pozwoli to jednak wybrać wtyczek / funkcji do zainstalowania (zawiera wszystko) i nie będzie tworzyć żadnych skrótów menu. W przypadku „instalacji” jesteś sam.

- Możliwe jest użycie nocnego buildu z Forum. Ta kompilacja nie jest dostarczana z kompilatorem!! Musisz samodzielnie zainstalować kompilator (jeśli jeszcze go nie masz). Przed instalacją zajrzyj na http://forums.codeblocks.org/index.php/topic,3232.0.html

- Dobrym rozwiązaniem jest zainstalowanie oficjalnej dystrybucji z MinGW i instalowanie na szczycie tej oficjalnej wersji co noc. Będziesz musiał uważnie postępować zgodnie z procedurą, ponieważ mogą wystąpić pewne niezgodności. Mieszanie wersji przynosi problemy

- Pełna wersja Code::Blocks jest dystrybuowana z 64-bitowym MinGW zawartym w podkatalogu codeblocks. Czasami prowadzi to do problemów, ponieważ pełna ścieżka zawiera spację (w Program Files). Tak więc dobrą praktyką jest przeniesienie tego katalogu MinGW do katalogu głównego dysku. Możesz również zmienić jego nazwę na C: \ MinGW64. Autodetekcja kompilatora w Code::Blocks powinna go tam znaleźć.

5.2 Konfiguracja MinGW

W tej sekcji opisano, jak zainstalować i skonfigurować MinGW

5.2.1 Przegląd

Łańcuch narzędzi kompilatora jest tym, czego Code::Blocks używa do zamiany wpisywanego kodu na liczby zrozumiałe dla komputera. Ponieważ łańcuch narzędzi kompilatora jest bardzo złożonym przedsięwzięciem, nie jest on częścią samego Code::Blocks, ale raczej jest oddzielnym projektem, z którego korzysta Code::Blocks. Rodzajem łańcuchów narzędzi kompilatorów, o których mowa na tej stronie, są łańcuchy narzędzi „MinGW”. Co oznacza „Minimalist GNU for Windows” (Minimalistyczne GNU dla Windows). A „GNU” rozwija się do „GNU to nie Unix”. Więcej informacji o projekcie GNU można znaleźć na stronie głównej GNU.

W przypadku większości łańcuchów narzędzi kompilatorów opartych na MinGW posiadanie łańcucha narzędzi w PATH jest ważne, ponieważ oznacza to, że podczas tworzenia biblioteki narzędzi będą domyślnie dostępne dla twoich programów podczas ich tworzenia, a także ułatwi korzystanie z narzędzi, takich jak CMake, ponieważ będzie mógł znaleźć Twój łańcuch narzędzi kompilatora. Kiedy faktycznie dystrybuujesz swoje programy na inne komputery, skopiujesz potrzebne pliki .dll z katalogu toolchain i dołączysz je jako część swojego instalatora. Na twoim komputerze są one w twojej PATH, więc zawsze je masz, na komputerach twoich użytkowników nie będą mieli łańcucha narzędzi kompilatora, więc tam dostarczasz pliki .dll ze swoim programem.

5.2.2 Łańcuchy narzędzi kompilatora MinGW

W sieci można znaleźć kilka dystrybucji MinGW. Oto lista dystrybucji, która nie jest wyczerpująca.

MinGW - The original project - http://www.mingw.org/: kompilatory tylko 32-bitowe;

TDM distribution - http://tdm-gcc.tdragon.net/: 32- i 64-bitowa, ale stara dystrybucja 5.1, teraz przestarzała. Była używana i dystrybuowana z Code::Blocks 17.12;

New TDM distribution - http://tdm-gcc.tdragon.net/: oparty na dystrybucji 9.2. Wydaje się, że ma czasem problemy, przynajmniej z kompilacją samego Code::Blocks.

MinGW 64 - https://sourceforge.net/projects/mingw-w64/files/: 32 i 64 bit, dystrybucja 8.1 w targetowaniu Toolchains. Projekt macierzysty MinGW-Builds zawiera znacznie więcej niż jest to konieczne - MinGW-Builds zazwyczaj wystarczy zamiast pełnych prac.

Dostępnych jest kilka opcji: dla kompilatora 32-bitowego możesz wybrać wersję posix, sjlj (i686-posix-sjlj), a dla kompilatora 64-bitowego możesz wybrać wersję posix, seh (x86 64-posix-seh) kompatybilny ze starą wersją TDM). To jest ta ostatnia 64-bitowa wersja posix-seh, która jest używana w ostatnich skompilowanych wersjach Nightlies Code::Blocks i dystrybuowana z wersją 20.03. Inne opcje również działają: zależy od Twoich potrzeb. gcc, g++, gfortran, gdb, lto, omp, mingw32-make są w dystrybucji;

MinGW 64 Ray_linn Personal build - https://sourceforge.net/projects/mingw-w64/files/Multilib%20Toolchains%28Targetting%20Win32%20and%20Win64%29/ray_ linn/gcc-9.x-with-ada/: 64-bitowa dystrybucja 9.1 w osobistym podkatalogu build. Używaj statycznych bibliotek (**static libraries**), więc nie musisz dystrybuować bibliotek dll z własnymi plikami wykonywalnymi, ale są one większe. ada, gcc, g++, gfortran, lto, objc, obj-c++, omp są w dystrybucji. Problem: gdb i make nie są uwzględnione.

MinGW Equation - http://www.equation.com/servlet/equation.cmd?fa=fortran: najnowsze wersje 32- i 64-bitowe (kilka wyborów). Użyj statycznych bibliotek, tak jak w powyższej wersji, tworzy większe pliki wykonywalne, ale nie ma potrzeby rozpowszechniania bibliotek dll z własnymi plikami wykonywalnymi. gcc, g++, gfortran, gdb, lto, omp, make są w dystrybucji;

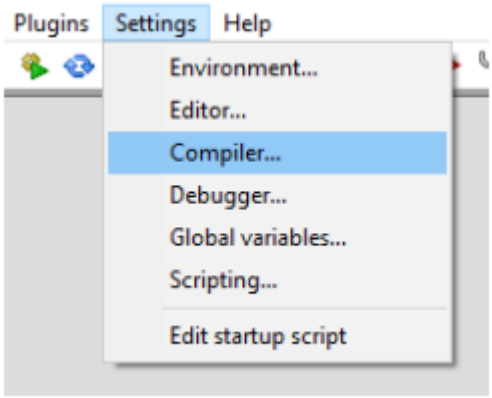
MinGW LH_Mouse version - https://gcc-mcf.lhmouse.com/: najnowsze wersje 32- i 64-bitowe (ale nie ostatnia). Bez gfortrana (?). Specjalny model gwintu (mcf). W dystrybucji znajdują się gcc, g++, gdb, lto, omp, mingw32-make. Bez gfortranu;

MinGW on Winlibs - http://winlibs.com/: najnowszy 64-bitowy kompilator (ale nie ostatni). W dystrybucji znajdują się gcc, g++, gfortran, gdb, lto, objc, obj-c++, omp, mingw32-make;

Msys2 - http://www.msys2.org/ i https://packages.msys2.org/group/mingw-w64-x86_ 64-toolchain: Wersje 32 i/lub 64-bitowe zainstalowane w C:\msys64\mingw32 i C:\msys64\mingw64. W przeciwieństwie do powyższych uniwersalnych instalatorów, konieczna jest dodatkowa praca, aby dostosować się do Twoich potrzeb, kompilatorów i łańcuchów narzędzi. Przeczytaj uważnie dokumentację. Niemniej jednak, Msys2 tworzy najnowsze wersje kompilatora i daje dostęp do narzędzia aktualizacji: pacman. ada, gcc, g++, gfortran, gdb, lto, objc, obj-c++, omp, mingw32-make są w dystrybucji lub mogą być zainstalowane przez pacmana;

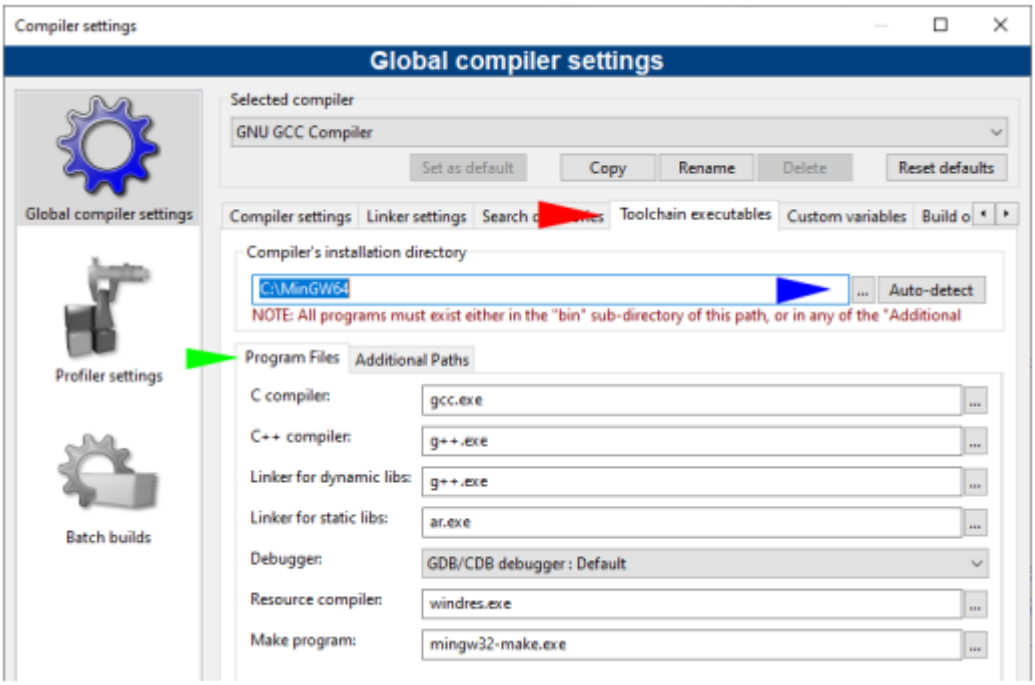
5.2.3 Konfiguracja Code::Blocks

Przejdź do ustawień kompilatora:



Rysunek 5.1: Ustawienia kompilatora

A następnie w zakładce ”Toolchain executables” (czerwona strzałka), kliknij na wielokropek (”...”, niebieska strzałka) i wybierz katalog główny, w którym zainstalowałeś MinGW (tutaj 64-bitowy). Po wybraniu tego katalogu w podzakładce „Program Files” (zielona strzałka) wypełnij pola, jak pokazano. Jeśli nie używasz 64-bitowego zestawu narzędzi MinGW, mogą występować niewielkie różnice w nazwach plików wykonywalnych. Jeśli najpierw wybierzesz niebieską strzałkę wielokropka, a następnie dla każdego wielokropka, który klikniesz w „Program Files”, będziesz już w swoim 64-bitowym katalogu bin MinGW, gdzie znajdują się rzeczywiste programy.

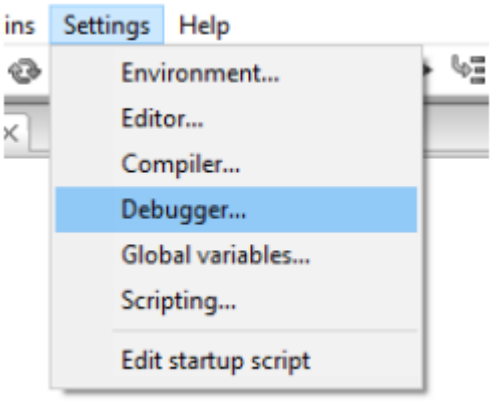


Rysunek 5.2: Code::Blocks konfiguracja łańcucha narzędzi

Uwaga: Możesz wprowadzić nazwę jako gcc.exe lub x86 64-w64 mingw32-gcc.exe lub mingw32-gcc.exe (w zależności od dystrybucji): to ten sam plik wykonywalny. Idem dla g++.

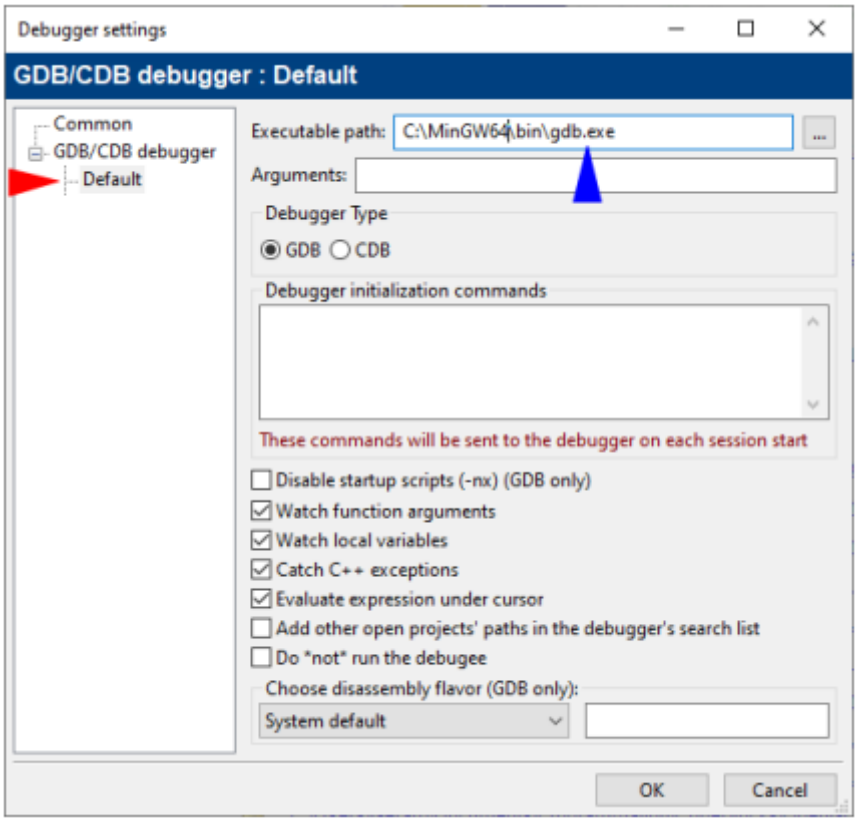
Uwaga:
Aby skonfigurować nowy kompilator, na przykład gfortran, wpisz gfortran.exe w 3 pierwsze pola tekstowe w zakładce „Pliki programów” lub dokładną nazwę taką, jaka jest w twojej dystrybucji.

Teraz przejdź do ustawień Debugera:



Rysunek 5.3: Ustawienia debugera

Wybierz domyślny debugger (czerwona strzałka), a następnie wypełnij ścieżkę do pliku wykonywalnego, jak pokazano dla MinGW 64-bit (niebieska strzałka).



Rysunek 5.4: Domyślna konfiguracja debugera

Streszczenie

Masz teraz środowisko Code::Blocks, które jest skonfigurowane do poprawnego korzystania z 64-bitowego MinGW. Używając tego przewodnika jako szablonu, możesz łatwo skonfigurować alternatywne łańcuchy narzędzi kompilatora bez względu na źródło - po prostu postępuj zgodnie z tą samą podstawową procedurą.

Narzędzia programistyczne

Zwykle nie powinieneś potrzebować wielu z tych narzędzi. ZIP jest wygodny, zwłaszcza gdy: samo budowanie Code::Blocks jest często już zawarte w MinGW, ale poza tym narzędzia te służą tylko do specjalistycznych celów.

- UnxUtils: różne narzędzia typu Unix na https://sourceforge.net/projects/unxutils/
- GnuWin32: Inne narzędzia na https://sourceforge.net/projects/gnuwin32/
- ZIP: 32-bitowy lub 64-bitowy na ftp://ftp.info-zip.org/pub/infozip/win32/ : proszę wybierz wersję zip300xn.

5.3 Nocna wersja Code::Blocks w systemie Windows

Nocne kompilacje są dostarczane „tak jak są”. Są to dystrybucje „binarne”, zwykle dostarczane codziennie, reprezentujące najnowszy i najlepszy stan źródeł Code::Blocks. Zwykle są dość stabilne, ale mogą wprowadzać nowe błędy, regresje, a z drugiej strony mogą wprowadzać nowe funkcje, poprawki błędów, ...

Zanim opiszemy, co zawierają kompilacje, może lepiej zacząć od tego, czego NIE zawierają nocne kompilacje. Na początek zadaj sobie pytanie: czym jest Code::Blocks? Cóż, jest to IDE (zintegrowane środowisko programistyczne), oznacza to, że integruje różne narzędzia i sprawia, że współpracują ze sobą. Tak więc Code::Blocks **NIE jest kompilatorem** (ani MS, ani Borland, ani GCC), nie jest debuggerem (ani MS, ani GDB), nie jest systemem makefile! Tak więc te komponenty **NIE** są częścią Code::Blocks, a zatem nie są również częścią nocnych dystrybucji kompilacji. Jednak kilka z wymienionych komponentów programistycznych można połączyć, aby ładnie ze sobą współpracowały za pośrednictwem Code::Blocks. Na przykład, możesz podłączyć kompilator GCC i debugger GDB oraz kompilować i debugować aplikacje pisane ręcznie.
Sam Code::Blocks jest kompilowany za pomocą GCC, w systemie Windows odbywa się to za pomocą portu MinGW. Ponieważ CB jest aplikacją wielowątkową, potrzebuje procedur wspierających, zapewniających mu funkcje wielowątkowości. Zapewnia to port MinGW, a dokładniej „mingwm10.dll", w każdym poście zapowiadającym nową conocną kompilację można znaleźć link do pobrania tej biblioteki. Code::Blocks posiada graficzny interfejs użytkownika (GUI). Istnieje kilka sposobów tworzenia GUI: możesz kodować używając podstawowego Windows API (działa tylko w Windows) lub możesz użyć MS-MFC (działa tylko w Windows) lub możesz użyć jakiejś abstrakcji GUI innej firmy, jak QT , wxWidgets, Tk, ...
Code::Blocks używa wxWidgets, obok abstrakcji GUI, wxWidgets oferuje o wiele więcej abstrakcji (ciągi, pliki, strumienie, gniazda, ...) i dobrą rzeczą jest to, że wxWidgets dostarcza te abstrakcje na wiele różnych platform (Windows, Linux, Apple , ...). Oznacza to, że Code::Blocks musi być wyposażony w rzeczywiste funkcje z wxWidgets (powiedzmy, że kod binarny wykonuje rzeczywistą pracę), jest to osiągane przez bibliotekę dll: "wxmsw31u gcc cb.dll" (*17.12 został zbudowany z "wxmsw28u gcc cb.dll"*). Ponownie w każdym poście ogłoszeniowym nowej nocnej kompilacji można znaleźć link do pobrania tej biblioteki dll i innych wymagań wstępnych.

Kiedy w systemie Windows zainstalujesz oficjalną wersję z dołączonym MinGW (wersja zalecana), znajdziesz katalog MinGW w C: \ Program Files \ codeblocks. Jeśli działa dobrze w większości zastosowań, nie jest to najlepsze miejsce, chociażby dlatego, że w nazwie ścieżki jest spacja i ta spacja może przeszkadzać niektórym komponentom MinGW. Tak więc w systemie Windows przenieś ten katalog MinGW na przykład do C:. Możesz nawet zmienić jego nazwę na MinGW32 na 32-bitową wersję toolchain lub MinGW64 na 64-bitową wersję.

Jak powiedziano wcześniej, dobrym rozwiązaniem do zainstalowania Nightly jest najpierw zainstalowanie oficjalnej wersji, a następnie skonfigurowanie i wypróbowanie. W ten sposób twoje pliki konfiguracyjne, skojarzenia, skróty zostaną poprawnie ustawione i będą przechowywane do nocnej instalacji. Link do wyszukiwania ostatnich nocnych wiadomości to http://forums.codeblocks.org/index.php/board,20.0.html.

Jeśli zainstalujesz nightly powyżej wersji 20.03, musisz uważnie postępować zgodnie z tą procedurą, ponieważ kilka rzeczy się zmieniło, przynajmniej są one skompilowane w 64 bitach z najnowszym kompilatorem, inną wersją wxWidgets i potrzebują dodatkowych bibliotek dll.

Normalnie nie będziesz miał takich problemów z wersją 20.03.

- Rozpakuj pobrane w nocy i skopiuj wszystkie pliki z podkatalogu codeblocks. Jeśli przeniosłeś swój podkatalog MinGW gdzie indziej, możesz najpierw usunąć całą zawartość podkatalogu codeblocks, aby mieć pewność, że nie będziesz mieszać wersji. Jeden wyjątek: jeśli zainstalowałeś w tym podkatalogu jakieś specyficzne rzeczy, jak np. pliki lokalizacyjne, nie usuwaj ich.

- Rozpakuj pliki dll wxWidgets znalezione w Twoim nightly. Możesz je zainstalować bezpośrednio w katalogu codeblocks lub dla większego użytku w podkatalogu bin MinGW. Sprawdź, czy ten podkatalog bin znajduje się w Twojej PATH. Powinien tam być, jeśli zainstalowałeś oficjalną wersję CB z jej instalatorem.

- Rozpakuj biblioteki dll wymagań wstępnych. Możesz je zainstalować bezpośrednio w katalogu codeblocks. Również tutaj, w celu większego wykorzystania, można je zainstalować w podkatalogu bin MinGW lub w dowolnym innym katalogu dostępnym za pośrednictwem zmiennej PATH. Ale uważaj, bo mogą już tam być, ale w innej wersji lub skompilowane z inną wersją MinGW. W takim przypadku lepiej trzymać je w podkatalogu codeblocks do prywatnego użytku i uniknąć problemów z mieszaniem wersji MinGW.

Uwaga:
dll wxWidgets i prerequisites (warunki wstępne) nie zmienia się tak często. Tak więc, jeśli zainstalujesz codzienną wersję na poprzednią, nie zawsze jest konieczne ich aktualizowanie. Przeczytaj uważnie temat forum w tym konkretnym wieczorze

Normalnie to wszystko. Twój nocny jest gotowy do użycia ...

6 Budowanie CodeBlocks ze źródeł

Ten rozdział opisuje jak zbudować samego Code::Blocksa.

Uwaga:
Ten rozdział istniał w oryginalnych plikach .tex w wersji 1, ale nie został opublikowany we wszystkich językach. Musi zostać przejrzany i wypełniony przynajmniej dla użytkowników Linuksa.

6.1 Wprowadzenie

Ta sekcja opisuje proces używany przy tworzeniu nocnych kompilacji i może być użyta jako wskazówka, jeśli chcesz samodzielnie zbudować Code::Blocks. Jest opisywany jako sekwencja działań.

Aby wykonać nasze zadania związane z budową, będziemy potrzebować kilku narzędzi. Stwórzmy listę składników do naszych eksperymentów kulinarnych.

- Kompilator
- początkowy system kompilacji
- źródła Code::Bloks
- program zip
- svn (system kontroli wersji)
- wxWidgets

6.2 Windows lub Linux

Ta sekcja jest napisana dla składni Windows, ale może być łatwo dostosowana do Linuksa. Inne szczegółowe instrukcje można znaleźć na Wiki:
http://wiki.codeblocks.org/index.php/Installing_Code::Blocks_from_source_on_Linux

Ponieważ programiści Code::Blocks budują Code::Blocks używając GCC, równie dobrze możemy użyć tego pod Windows. Najłatwiejszym i najczystszym portem jest MinGW. Jest to kompilator rozprowadzany z Code::Blocks po pobraniu PEŁNEGO oficjalnego pakietu. Z C::B 17.12 była to dystrybuowana wersja TDM 5.1.0, dość stara. Pozostaniemy przy wersji 8.1.0, która działa dobrze i która jest teraz dystrybuowana z Code::Blocks 20.03, ale możesz znaleźć nowsze (jak pokazano w MinGW Compiler Toolchains (podrozdział 5.2.2 na stronie 123). uzyskać dwie wersje MinGW: do generowania kodu 32 bitowego, lub do kodu 64 bitowego na [,→MinGW64].Dostępnych jest wiele podwyborów.Dla 32 bitów sugerowałbym i686-posix-sjlj a dla 64 bitów x86 64-posix seh.

Uwaga:
Wersja 64-bitowa może generować kod 64-bitowy i kod 32-bitowy. Po prostu dodaj opcję -m64 lub -m32 do opcji kompilacji ORAZ opcji linków. Normalnie wersja 32-bitowa generuje tylko 32-bitowy kod. Tylko uważaj, jeśli używasz bibliotek statycznych lub dynamicznych, muszą one być wygenerowane z tą samą liczbą bitów.

Najpierw krótkie objaśnienie elementów MinGW:

gcc-core - rdzeń pakietu GCC

gcc-g++ - kompilator c i c++

gfortran - kompilator fortranu. WAŻNE : gfortran 5.1 ma błąd : nie działa instrukcja open , aby otworzyć plik do odczytu danych ! Niestety, ostatnia dystrybucja TDM używała tego gfortrana i nie została poprawiona.

mingw Runtime - implementacja bibliotek wykonawczych.

mingw utils - kilka narzędzi (implementacja mniejszych programów, z których korzysta samo GCC).

win32Api - API do tworzenia programów Windows.

binutils - kilka narzędzi używanych w środowiskach budowania.

make - program make Gnu, dzięki któremu można budować z plików make.

GDB - debuger Gnu.

Sugerowałbym wypakowanie (i zainstalowanie dla GDB, jeśli to konieczne) wszystkiego, co znajduje się w katalogu `C:\MinGW`. Jeśli chcesz mieć zarówno wersję 32-bitową, jak i 64-bitową, możesz zainstalować je odpowiednio w `C:\MinGW32` i `C:\MinGW64`. W dalszej części tego artykułu założymy, że to właśnie tam umieściłeś. Jeśli masz już instalację Code::Blocks, która została dołączona do MinGW, nadal radzę zainstalować MinGW w sposób opisany tutaj. Kompilator nie należy do drzewa katalogów IDE; to dwie odrębne rzeczy. Code::Blocks po prostu wprowadza go w oficjalnych wersjach, aby przeciętny użytkownik nie musiał zawracać sobie głowy tym procesem. Niemniej jednak niektóre narzędzia Unix Like mają problemy, gdy są instalowane w ścieżce zawierającej spacje (lub nawet znaki akcentowane) w nazwach podkatalogów. W przypadku prostego użycia C::B nie będziesz mieć problemów, ale może się to zdarzyć, więc nie wahaj się przenieść `C:\Program Files\Codeblocks\MinGW` do czegoś takiego jak `C:\MinGW`.

Może być konieczne dodanie katalogu bin instalacji MinGW (i/lub MinGW32/MinGW64) do ścieżki. Prostym sposobem na to jest użycie następującego polecenia w wierszu polecenia:

```
set path=%PATH%;C:\MinGW32\bin;C:\MinGW32\i686-w64-mingw32\bin;
```

lub (dla instalacji 64-bitowej):

```
set path=%PATH%;C:\MinGW64\bin;C:\MinGW64\x86_64-w64-mingw32\bin;
```

Konieczne jest uruchomienie pliku wykonywalnego poza kontekstem Code::Blocks IDE.

Możesz także zmodyfikować globalną (lub użytkownika) zmienną środowiskową PATH.

6.2.1 Wstępny system kompilacji

Dla [,→CODEBLOCKS] dostępny jest opis projektu CodeBlocks.cbp. Jeśli załadujesz ten plik projektu w Code::Blocks, będziesz w stanie zbudować Code::Blocks ze źródeł. Wszystko, co musimy zrobić, to zdobyć gotową wersję Code::Blocks.

Najpierw pobierz kompilację na noc. Możesz dokonać wyboru z tego miejsca ([,→FORUM] w Nightly Builds). Nocne kompilacje są wersjami unicode, zawierającymi rdzeń i dodane wtyczki. Przeczytaj uważnie pierwszy post każdego nightly: zawiera on instrukcje pobierania i instalowania dodatkowych bibliotek dll, niezbędnych w najnowszych wersjach MinGW, szczególnie w wersji 8.1 używanej do tworzenia ostatnich C::B nightly.

Następnie rozpakuj plik 7-zip do dowolnego katalogu. Jeśli nie masz 7-zip, możesz pobrać go za darmo z [,→7Z].

Teraz Code::Blocks potrzebuje jeszcze jednej biblioteki dll do poprawnego działania: dll WxWidgets. Możesz go również pobrać na nocnym forum kompilacji. Po prostu rozpakuj go do tego samego katalogu, w którym rozpakowałeś nocną kompilację Code::Blocks. Potrzebuje również mingwm10.dll. Znajduje się w katalogu bin naszej instalacji MinGW. Dlatego ważne jest, aby upewnić się, że katalog bin instalacji MinGW znajduje się w zmiennej ścieżki.

Na koniec uruchom tę nową, nocną kompilację Code::Blocks. Powinien odkryć kompilator MinGW, który właśnie zainstalowaliśmy. Jeśli tak nie jest, przejdź do menu ”Settings / Compiler... / Toolchain executables” (Ustawienia / Kompilator... / Pliki wykonywalne Toolchain (łańcuch narzędzi)) i dostosuj ścieżkę MinGW do konkretnej instalacji.

6.2.2 System kontroli wersji

Aby móc pobierać najnowsze i najlepsze źródła Code::Blocks, musimy zainstalować System Kontroli Wersji (Version Control System).

Deweloperzy Code::Blocks udostępniają swoje źródła poprzez system kontroli wersji[,→Subversion]. Tak więc potrzebujemy klienta, aby uzyskać dostęp do swojego repozytorium źródeł svn. Miłym, łatwym klientem dla Windows jest [,→TortoiseSVN], który jest dostępny za darmo. Pobierz i zainstaluj, zachowując wszystkie sugerowane ustawienia.

Teraz stwórz katalog, gdziekolwiek chcesz, na przykład D:\projects\CodeBlocks. Kliknij prawym przyciskiem myszy ten katalog i wybierz z wyskakującego menu: svn-checkout. W wyświetlonym oknie dialogowym podaj następujące informacje dla adresu URL repozytorium:

svn://svn.code.sf.net/p/codeblocks/code/trunk

i pozostaw wszystkie inne ustawienia bez zmian.

Teraz bądź cierpliwy, podczas gdy TortoiseSVN pobiera najnowsze pliki źródłowe z repozytorium Code::Blocks do naszego lokalnego katalogu. TAk; wszystkie te źródła Code::Blocks są na Twojej drodze!

Aby uzyskać więcej informacji na temat ustawień SVN, zobacz informacje o ustawieniach SVN (jednak ta wtyczka C::B nie istnieje już w ostatnich wersjach Code::Blocks). Jeśli nie lubisz integracji z Explorerem lub szukasz wieloplatformowego klienta, możesz rzucić okiem na RapidSVN.

6.2.3 wxWidgets

[,→Wxwidgets] to abstrakcja platformy, która dostarcza API do obsługi wielu rzeczy, takich jak GUI, gniazda, pliki, funkcjonalność rejestru. Korzystając z tego interfejsu API, możesz stworzyć program niezależny od platformy.

Code::Blocks to aplikacja wxWidgets (dalej: wx), co oznacza, że jeśli chcesz uruchomić Code::Blocks, potrzebujesz funkcjonalności wx. Można to zapewnić na kilka sposobów. Może to być biblioteka .dll lub biblioteka statyczna. Code::Blocks używa wx jako biblioteki dll, którą można również pobrać z sekcji poświęconej nocnym kompilacjom na forum.

Jeśli jednak chcemy zbudować aplikację wx, musimy dołączyć nagłówki źródeł wx. Mówią kompilatorowi o funkcjonalności wx. Oprócz tych plików nagłówkowych, nasza aplikacja musi połączyć się z bibliotekami importu wx. Cóż, zróbmy to krok po kroku.

Wx jest dostarczany jako plik zip jego źródeł, więc musimy go zbudować sami. Kupiliśmy już kompilator MinGW, więc wszystkie potrzebne narzędzia mamy pod ręką

Następnie rozpakujmy źródła wx do C:\Projects, więc otrzymamy katalog główny wx w następujący sposób: `C:\Projects\wxWidgets-3.1.3`. Następnie rozpakuj łatkę do tego samego katalogu, aby nadpisała pliki. Zauważ, że od teraz będziemy odwoływać się do katalogu głównego wx jako <wxDir>

Teraz zbudujemy wxWidgets. Tak to robimy:

Najpierw upewnij się, że C:\MingW\bin jest na twojej ścieżce, podczas kompilacji zostaną wywołane niektóre programy, które znajdują się w katalogu MinGW\bin. Ponadto Make musi być w wersji 3.80 lub nowszej.

Teraz nadszedł czas na skompilowanie wxWidgets. Otwórz wiersz poleceń i przejdź do katalogu wxWidgets:

```
cd \build\msw
```

Jesteśmy teraz we właściwym miejscu. Najpierw wyczyścimy źródło:

```
mingw32-make -f makefile.gcc SHARED=1 MONOLITHIC=1 BUILD=release
UNICODE=1 clean
```

Gdy wszystko jest już czyste, możemy skompilować wxWidgets:

```
mingw32-make -f makefile.gcc SHARED=1 MONOLITHIC=1 BUILD=release
UNICODE=1
```

Jeśli używasz TDM-gcc 5.1.0 lub gcc 8.1.0, będziesz musiał dodać następujące opcje w tym samym wierszu poleceń:

```
USE_XRC=1
CXXFLAGS+="-fpermissive -fno-keep-inline-dllexport -std=gnu++11 -
Wno-deprecated-declarations"
LDFLAGS="-Wl,--allow-multiple-definition"
```

Code::Blocks dla Windows, od wersji SVN 11701 jest generowany do obsługi aktywacji/wymuszonego DIRECT2D (dla szybszej i lepszej jakości grafiki). Generowanie wxWidgets wymaga wprowadzenia pewnych zmian bezpośrednio w plikach setup.h. Zwykle jedyny plik, który wymaga modyfikacji, znajduje się w podkatalogu lib, utworzonym podczas pierwszego generowania wxWidgets i używanym w kolejnych generacjach. W przypadku kompilacji gcc, dll wygląda to mniej więcej tak:
`C:\wxWidgets-3.1.3\lib\gcc dll\mswu\wx`
Możesz po prostu zmodyfikować wiersz zawierający (wiersz 1593 w wxWidgets 3.1.3):

```
#define wxUSE_GRAPHICS_DIRECT2D 0
```

za pomocą

```
#define wxUSE_GRAPHICS_DIRECT2D 1
```

Inny plik setup.h może zostać zmodyfikowany, ale zwykle nie jest to wymagane.

opcjonalnie :

```
USE_OPENGL=1
DEBUG_LEVEL=0
```

Uwaga:
USE OPENGL=1 tworzy dodatkową bibliotekę dll, używaną, jeśli twój program wxWidgets wymaga okien OpenGL, szczególnie gdy używany jest wxGLCanvas

Istnieje również możliwość personalizacji nazwy produkowanej biblioteki dll:

```
VENDOR=cb
```

lub dla wersji 64-bitowej:

```
VENDOR=cb_64
```

Uwaga:
Jeśli VENDOR nie jest określony, jest to odpowiednik VENDOR=custom. VENDOR=cb jest używany przez zespół Code::Blocks. Tak więc, aby uniknąć nieporozumień, wskazane może być użycie innego identyfikatora „dostawcy”.

To zajmie trochę czasu.

Aby wykonać kompilację debugowania, wykonaj te same kroki i opcje (jedyną zmianą jest BUILD=debug):

- Wyczyść poprzednią kompilację za pomocą

```
mingw32-make -f makefile.gcc SHARED=1 MONOLITHIC=1 BUILD=debug
UNICODE=1 clean
```

- Kompiluj za pomocą

```
mingw32-make -f makefile.gcc SHARED=1 MONOLITHIC=1 BUILD=debug
UNICODE=1
```

Na koniec dodaj te same opcje, co powyżej.

Zajrzyj teraz trochę do katalogu (`<wxDir>\lib\gcc dll`). Pojawiły się biblioteki importu i biblioteka dll, a na tej pozycji powinien również znajdować się podkatalog `mswu\wx` zawierający plik `setup.h`

Możesz użyć strip.exe (dystrybuowanego z MinGW), aby zmniejszyć rozmiar bibliotek dll. na przykład:

```
strip ..\..\lib\gcc_dll\wxmsw313u_gcc_cb_64.dll
strip ..\..\lib\gcc_dll\wxmsw313u_gl_gcc_cb_64.dll
```

Gratulacje! Właśnie zbudowałeś wxWidgets!

Zróbmy kilka wstępnych zadań, zanim przejdziemy do prawdziwej kompilacji Code::Blocks.

6.2.4 Zip

Podczas kompilacji Code::Blocks kilka zasobów zostanie spakowanych w plikach zip. Dlatego proces kompilacji powinien mieć dostęp do pliku zip.exe. Musimy pobrać plik `zip.exe` (jeśli nie ma go jeszcze w twojej dystrybucji MinGW) i umieścić go gdzieś na naszej drodze. Dobre miejsce to: `MingW\bin`.

Plik zip.exe można pobrać bezpłatnie z tej strony (http://www.info-zip.org/pub/infozip/Zip.html) lub stąd (ftp://ftp.info-zip.org/pub /infozip/win32/), który jest bezpośrednim łączem ftp do katalogu zawierającego najnowsze wersje.

Po pobraniu po prostu rozpakuj plik zip.exe do odpowiedniej lokalizacji.

6.2.5 Obszar roboczy

To prowadzi nas do ostatniego zadania wstępnego. Kod Code::Blocks można podzielić na 2 główne części: rdzeń z wewnętrznymi wtyczkami i dołączone wtyczki. Zawsze musisz zbudować główne/wewnętrzne części przed zbudowaniem części wkładu.

Aby zbudować część wewnętrzną, możesz użyć pliku projektu Code::Blocks, który znajdziesz pod adresem: `<cbDir>\src\CodeBlocks.cbp`. Nawiasem mówiąc, nasz główny katalog Code::Blocks jest od tej pory wymieniany jako `<cbDir>`. Obszar roboczy to coś, co grupuje kilka projektów. Aby zbudować wtyczki contrib, można je znaleźć w `<cbDir>\src\ContribPlugins.workspace`.

Ale stwórzmy przestrzeń roboczą zawierającą wszystko. Umieśćmy ten obszar roboczy w głównym katalogu `<cbDir>`. Wystarczy skorzystać ze zwykłego edytora tekstu i utworzyć plik o nazwie „CbProjects.workspace”. Taki plik istnieje już w ostatnich wersjach C::B. Możesz nadać mu następującą treść:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<CodeBlocks_workspace_file>
  <Workspace title="CodeBlocks Workspace wx3.1.x">
        <Project filename="CodeBlocks_wx31_64.cbp" active="1" />
        <Project filename="tools/Addr2LineUI/Addr2LineUI_wx31.cbp" />
        <Project filename="tools/cb_share_config/cb_share_config_wx31.cbp" />
        <Project filename="tools/CBLauncher/CbLauncher_wx31.cbp" />
        <Project filename="tools/cbp2make/cbp2make_wx31.cbp" />
        <Project filename="plugins/codecompletion/cctest_wx31.cbp" />
        <Project filename="plugins/contrib/wxContribItems/wxContribItems_wx31.cbp" />
        <Project filename="plugins/contrib/wxSmith/wxSmith_wx31.cbp" />
        <Project filename="plugins/contrib/wxSmithContribItems/wxSmithContribItems_wx31.cbp”/>
           <Depends filename="plugins/contrib/wxContribItems/wxContribItems_wx31.cbp" />
           <Depends filename="plugins/contrib/wxSmith/wxSmith_wx31.cbp" />
        </Project>
        <Project filename="plugins/contrib/wxSmithAui/wxSmithAui_wx31.cbp">
           <Depends filename="plugins/contrib/wxSmith/wxSmith_wx31.cbp" />
        </Project>
        <Project filename="plugins/contrib/AutoVersioning/AutoVersioning_wx31.cbp" />
        <Project filename="plugins/contrib/BrowseTracker/BrowseTracker_wx31.cbp" />
        <Project filename="plugins/contrib/byogames/byogames_wx31.cbp" />
        <Project filename="plugins/contrib/cb_koders/cb_koders_wx31.cbp" />
        <Project filename="plugins/contrib/Cccc/Cccc_wx31.cbp" />
        <Project filename="plugins/contrib/codesnippets/codesnippets_wx31.cbp" />
        <Project filename="plugins/contrib/codestat/codestat_wx31.cbp" />
        <Project filename="plugins/contrib/copystrings/copystrings_wx31.cbp" />
        <Project filename="plugins/contrib/CppCheck/CppCheck_wx31.cbp" />
        <Project filename="plugins/contrib/Cscope/Cscope_wx31.cbp" />
        <Project filename="plugins/contrib/devpak_plugin/DevPakPlugin_wx31.cbp" />
        <Project filename="plugins/contrib/DoxyBlocks/DoxyBlocks_wx31.cbp" />
        <Project filename="plugins/contrib/dragscroll/DragScroll_wx31.cbp" />
        <Project filename="plugins/contrib/EditorConfig/EditorConfig_wx31.cbp" />
        <Project filename="plugins/contrib/EditorTweaks/EditorTweaks_wx31.cbp" />
        <Project filename="plugins/contrib/envvars/envvars_wx31.cbp" />
        <Project filename="plugins/contrib/FileManager/FileManager_wx31.cbp" />
        <Project filename="plugins/contrib/FortranProject/FortranProject_cbsvn_wx31.cbp" />
        <Project filename="plugins/contrib/headerfixup/headerfixup_wx31.cbp" />
        <Project filename="plugins/contrib/help_plugin/help-plugin_wx31.cbp" />
        <Project filename="plugins/contrib/HexEditor/HexEditor_wx31.cbp" />
        <Project filename="plugins/contrib/IncrementalSearch/IncrementalSearch_wx31.cbp" />
        <Project filename="plugins/contrib/keybinder/keybinder_wx31.cbp" />
        <Project filename="plugins/contrib/lib_finder/lib_finder_wx31.cbp">
```

```
            <Depends filename="plugins/contrib/wxContribItems/wxContribItems_wx31.cbp" />
        </Project>
        <Project filename="plugins/contrib/MouseSap/MouseSap_wx31.cbp" />
        <Project filename="plugins/contrib/NassiShneiderman/NassiShneiderman_wx31.cbp" />
        <Project filename="plugins/contrib/profiler/cbprofiler_wx31.cbp" />
        <Project filename="plugins/contrib/ProjectOptionsManipulator/ProjectOptionsManipulator_
                wx31.cbp" />
        <Project filename="plugins/contrib/regex_testbed/RegExTestbed_wx31.cbp" />
        <Project filename="plugins/contrib/ReopenEditor/ReopenEditor_wx31.cbp" />
        <Project filename="plugins/contrib/rndgen/rndgen_wx31.cbp" />
        <Project filename="plugins/contrib/SmartIndent/SmartIndent_wx31.cbp" />
        <Project filename="plugins/contrib/source_exporter/Exporter_wx31.cbp" />
        <Project filename="plugins/contrib/SpellChecker/SpellChecker_wx31.cbp" />
        <Project filename="plugins/contrib/symtab/symtab_wx31.cbp" />
        <Project filename="plugins/contrib/ThreadSearch/ThreadSearch_wx31.cbp">
            <Depends filename="plugins/contrib/wxContribItems/wxContribItems_wx31.cbp" />
        </Project>
        <Project filename="plugins/contrib/ToolsPlus/ToolsPlus_wx31.cbp" />
    </Workspace>
</CodeBlocks_workspace_file>
```

Uwaga:
Istnieje kilka wariantów tego pliku, w zależności od systemu operacyjnego, wersji wxWidgets, wersji 32 lub 64 bitowej.

Użyjemy tego obszaru roboczego do zbudowania całego Code::Blocks.

6.2.6 Budowanie Codeblocks

Ta sekcja opisuje końcowy proces budowania Code::Blocks.

6.2.6.1 Windows

Wreszcie doszliśmy do ostatniego kroku; nasz ostateczny cel. Uruchom kod wykonywalny Code::Blocks z nocnego pobrania kompilacji. Wybierz Otwórz z menu Plik i przejdź do naszego utworzonego powyżej obszaru roboczego i otwórz go. Bądź trochę cierpliwy, gdy Code::Blocks analizuje wszystko, a Code::Blocks poprosi nas o 3 lub 4 zmienne globalne, te zmienne globalne wskażą nocnemu Code::Blocks, gdzie może znaleźć wxWidgets (pamiętaj: pliki nagłówkowe i import biblioteki) i gdzie może znaleźć ... źródła Code::Blocks, jest to potrzebne dla wtyczek contrib, muszą wiedzieć (jak w przypadku każdej wtyczki utworzonej przez użytkownika), gdzie znajdują się sdk (pliki nagłówkowe Code::Blocks). Oto wartości w naszym przypadku:

wx <wxDir> katalog bazowy wxWidgets.: nazwa zmiennej może mieć postać wx31, wx31 64, ...

cb <cbDir>/src katalog Code::Blocks zawierający źródła.

cb release type : -O (dla wersji wydanej, zwykły przypadek. Dla programistów możesz umieścić -g, aby debugować C::B)

boost : główny katalog, w którym zainstalowany jest boost (np.: C:\boost). Używany przez wtyczkę NassiShneiderman, wypełnij taką samą wartością, jaką zawierają podsekcje i lib.

Teraz przejdź do menu projektu i wybierz (prze)buduj obszar roboczy i gotowe. Zobacz jak Code::Blocks buduje Code::Blocks.

Po utworzeniu Code::Blocks wygenerowane pliki z informacjami debugowania można znaleźć w podkatalogu `devel`. Wywołując lub wykonując w konsoli plik wsadowy update.bat z katalogu źródłowego `<cbDir>/src` (a dokładniej wersję dostosowaną do konkretnego pokolenia, jak na przykład update31 64.bat), pliki są kopiowane do podkatalogu `<cbDir>/src/output` (lub dostosowana wersja). Ponadto usunie to wszystkie symbole debugowania. *Ten krok jest bardzo ważny - nigdy, przenigdy o tym nie zapomnij.*

Teraz możesz skopiować bibliotekę wx dll zarówno do tego wyjścia, jak i opcjonalnie do katalogu devel.

Następnie możesz zamknąć Code::Blocks. To było pobrane nocne pamiętanie?

Czas to przetestować. W katalogu wyjściowym uruchom CodeBlocks.exe. Jeśli wszystko poszło dobrze, będziesz miał uruchomiony swój własny, domowy kod Code::Blocks.

6.2.6.2 Linux

(Uwaga: ta sekcja powinna zostać przejrzana i uzupełniona. Wygląda na to, że nie jest całkowicie zaktualizowana)

linux Rozpoczęcie aktualizacji revision.sh

Dzięki tej funkcji wersja SVN Nightly Builts jest aktualizowana w źródłach. Plik można znaleźć w głównym katalogu źródeł Code::Blocks.

Podczas generowania pod Linuksem konieczne są następujące kroki. W tym przykładzie zakładamy, że znajdujesz się w katalogu źródłowym Code::Blocks. W systemie Linux zmienna środowiskowa PKG_CONFIG_PATH musi być ustawiona. Katalog `<prefix>` musi zawierać plik `codeblocks.pc`.

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH: <prefix>
```

sh update_revsion.sh
./bootstrap
./configure --with-contrib=[all | plugin names separated with comma] --prefix=<install-dir>
make
make install (as root)

Możesz także budować pod Linuksem jak pod Windows, z plikiem obszaru roboczego. Niemniej jednak pkg config i wx-config muszą być ustawione poprawnie.

6.2.7 Generuj tylko wtyczki

Ta sekcja służy do generowania tylko wtyczek.

6.2.7.1 Windows

Skonfiguruj zmienne globalne poprzez „Settings” (Ustawienia) → „Global Variables” (Zmienne globalne).

Variable cb

Dla zmiennej cb ustaw podstawowy wpis do katalogu źródłowego Code::Blocks.

```
<cbDir>/codeblocks/src
```

Variable wx

Dla zmiennej wx ustaw wpis bazowy do katalogu źródłowego wx: np.

```
C:\wxWidgets-2.8.12
```

lub zbuduj nowszą wersję, wx31 (lub wx31 64)

```
C:\wxWidgets-3.1.4
```

W projekcie Code::Blocks zmienna projektu WX_SUFFIX jest ustawiona na u. Oznacza to, że podczas generowania Code::Blocks zostanie wykonane linkowanie do biblioteki *u gcc custom.dll (domyślnie). Oficjalne nocne budowania Code::Blocks zostaną połączone z gcc cb.dll. W ten sposób układ wygląda następująco.

```
gcc_<VENDOR>.dll
```

Zmienna <VENDOR> jest ustawiana w pliku konfiguracyjnym compiler.gcc lub w wierszu poleceń make, jak pokazano wcześniej. Aby upewnić się, że możliwe jest rozróżnienie pomiędzy oficjalnie wygenerowanymi blokami Code::Blocks a tymi wygenerowanymi przez Ciebie, domyślne ustawienie VENDOR=custom nigdy nie powinno być zmieniane.

Następnie utwórz obszar roboczy ContribPlugins.cbp poprzez „Project” (Projekt) → „Build workspace” (Buduj obszar roboczy). Następnie ponownie uruchom update.bat.

6.2.7.2 Linux

Skonfiguruj zmienną wx ze zmiennymi globalnymi.

base /usr

include /usr/include/wx-2.8 (lub wersja zainstalowana na twoim komputerze)

lib /usr/lib

Katalog URL

[,→7Z] 7z zip strona główna.
http://www.7-zip.org

[,→ZIP] Info-zip strona FTP.
ftp://ftp.info-zip.org/pub/infozip/win32/

[,→SourceForgeCB] Codeblocks w Sourceforge.
https://sourceforge.net/p/codeblocks/code/log/

[,→SourceForgeCBSrc] Trunk SVN C::B Źródło na Sourceforge.
https://sourceforge.net/p/codeblocks/code/HEAD/tree/trunk/

[,→FORUM] Codeblocks forum.
http://forums.codeblocks.org/

[,→WIKI] Codeblocks wiki.
http://wiki.codeblocks.org/index.php/Main_Page

[,→CODEBLOCKS] Codeblocks strona główna.
http://www.codeblocks.org/

[,→GCC] Strona główna GCC.
http://gcc.gnu.org/

[,→MinGW] MinGW strona oryginalna.
http://www.mingw.org/

[,→MinGW64] MinGW64 strona plików.
https://sourceforge.net/projects/mingw-w64/files/

[,→MinGW64-Ray-Linn] MinGW64 Ray Linn Strona osobista.
https://sourceforge.net/projects/mingw-w64/files/Multilib%20Toolchains%28Targetting%20Win32%20i%20Win64%29/ray_linn/gcc-9.x-with-ada/

[,→MinGW64-Equation] Strona plików równań MinGW64.
http://www.equation.com/servlet/equation.cmd?fa=fortran

[,→MinGW64-LHMuse] Strona plików MinGW64 LHMuse.
https://gcc-mcf.lhmouse.com/

[,→MinGW64-WinLibs] Strona MinGW64 WinLibs.
http://winlibs.com/

[,→MSys2] Strona główna Msys2.
http://www.msys2.org/

[,→MSys2-MinGW64-Toolchain] Msys2 łańcuch narzędzi dla MinGW64.
https://packages.msys2.org/group/mingw-w64-x86_64-toolchain

[,→UnxUtils] Strona plików Unix Like Utilities.
https://sourceforge.net/projects/unxutils/

[,→GnuWin32] Kolejna strona plików Unix Like Utilities.
https://sourceforge.net/projects/gnuwin32/

[,→HIGHTEC] Strona główna HighTec.
http://www.hightec-rt.com/

[,→TortoiseSVN] Strona domowa TortoiseSVN.
http://tortoisesvn.net/

[,→Subversion] Strona główna Subversion Apache.
http://subversion.apache.org/

[,→Wxwidgets] Strona główna WxWidgets.
http://www.wxwidgets.org/

[,→Wxcode] Strona główna WxCode.https://launchpad.net/codeblocks
http://wxcode.sourceforge.net/

[,→Scripts] Polecenia skryptowe.
http://wiki.codeblocks.org/index.php?title=Scripting_commands

[,→CBDoc] Dokumentacja C::B na wiki.
http://wiki.codeblocks.org/index.php/Category:Code::Blocks_Documentation

[,→UserDoc] C::B Dokumentacja użytkownika na wiki.
http://wiki.codeblocks.org/index.php/Category:User_Documentation

[,→NassiShneiderman] Strona NassiShneiderman Wiki.
https://www.wikipedia.org/wiki/Nassi-Shneiderman_diagram

[,→Launchpad Old] Stara strona z tłumaczeniem w Launchpadzie.
https://launchpad.net/codeblocks

[,→Launchpad New] Nowa strona z tłumaczeniem w Launchpadzie.
https://launchpad.net/codeblocks-gd

[,→Translation] Temat tłumaczenia na CB Forum.
http://forums.codeblocks.org/index.php/topic,1022.0.html