Kurzanleitung zur Benutzung der Entwicklungsumgebung `Code::Blocks` für die Übung aus Programmieren 1 und 2

Institut für Mikroelektronik, TU Wien

Version März 2018

Dieses Dokument ist für den schnellen Einstieg in die Benutzung der Entwicklungsumgebung `Code::Blocks` gedacht. Die folgende Beschreibung bezieht sich auf die aktuelle `Code::Blocks`-Version 13.12, welche Sie über *http://www.codeblocks.org/* beziehen können, und die auch auf allen Rechnern im Übungsraum installiert ist.

1 Anlegen eines neuen Projektes

Nach dem Start von `Code::Blocks` erscheint das in Abb. 1 gezeigte Fenster. Auf der linken Seite sehen Sie das Projektverwaltungsfenster (*Management*), das Ihnen unter *Projects* alle geöffneten Projekte anzeigt. Anfangs ist diese Liste leer, da zuerst ein neues Projekt erstellt werden muss. Um ein neues Projekt anzulegen, wählen Sie im Menü *File* unter dem Menüpunkt *New* den Untereintrag *Project...* aus (siehe Abb. 2).

Als nächstes erscheint das in Abb. 3 gezeigte Fenster, das Sie auffordert, eine Projektvorlage auszuwählen. Während der gesamten Übung zu Programmieren 1 und 2 benötigen Sie aussschließlich die Vorlage *Console Application*. Wählen Sie das zugehörige Symbol aus und bestätigen Sie die Auswahl durch einen Mausklick auf *Go*.

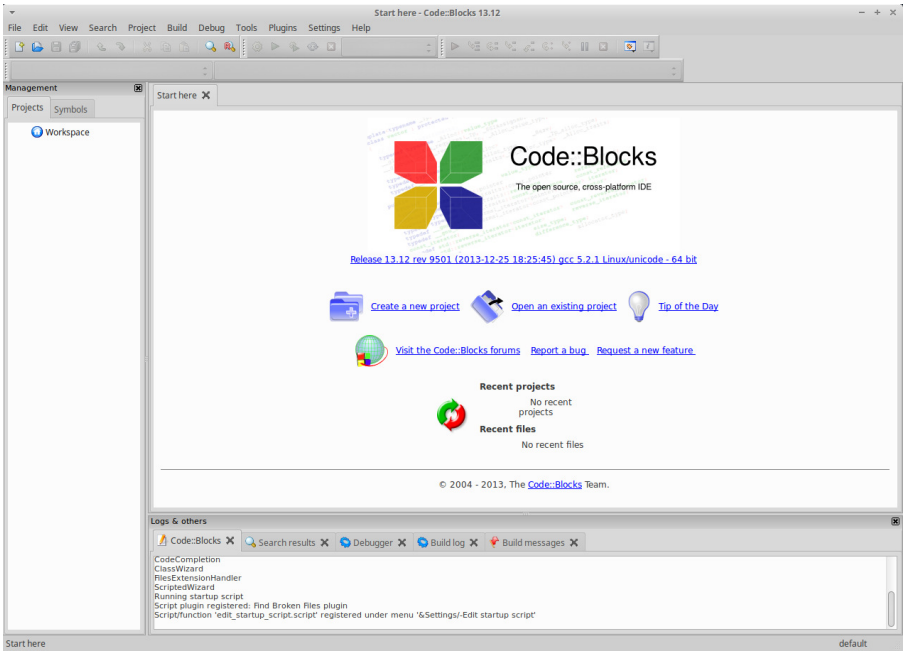


Abbildung 1: Die Entwicklungsumgebung `Code::Blocks`.

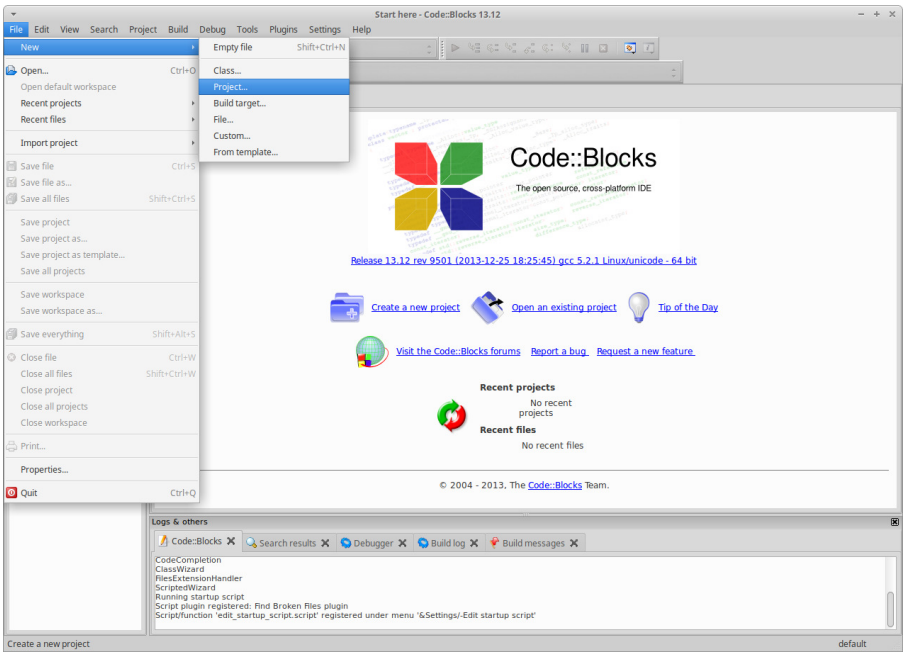


Abbildung 2: Erstellen eines neuen Projektes.

Dadurch starten Sie die in Abb. 4 dargestellte Folge von Dialogfenstern, die Ihnen bei der Erstellung einer neuen Konsolenanwendung assistiert. Das erste Fenster (Abb. 4a) ist lediglich ein Begrüßungsfenster. Durch einen Mausklick auf *Next* gelangen Sie zum zweiten Fenster (Abb. 4b), das die Auswahl der Programmiersprache erlaubt. In dieser Übung wird ausschließlich die Programmiersprache `C` verwendet. Dementsprechend sollten Sie immer diese Programmiersprache auswählen.

Im dritten Fenster (Abb. 4c) werden Sie in der ersten Eingabezeile nach einem Namen für das neue Projekt und in der zweiten Eingabezeile nach einem Verzeichnis, in dem das Projekt gespeichert werden soll, gefragt. Ein Dialogfenster zur Auswahl eines Verzeichnisses kann durch einen Mausklick auf den kleinen Button rechts neben der Eingabezeile aufgerufen werden. Die letzten beiden Eingabezeilen werden automatisch von `Code::Blocks` mit dem Namen des Projektes und dem Verzeichnis verändert. Wir empfehlen Ihnen, diese Vorschläge beizubehalten. Im in Abb. 4c gezeigten Beispiel erhält das neue Projekt den Namen `Uebung1`. Das Verzeichnis ist in diesem Fall `/home/prog001`, welches auf Linux-Betriebsystemen das Home-Verzeichnis des Benutzers `prog001` darstellt.

Das letzte Fenster (Abb. 4d) erlaubt die Wahl des Übersetzers (Compilers) sowie von voreingestellten Build-Konfigurationen. Auch hier sollten Sie die Standardeinstellungen übernehmen. Mit einem Klick auf *Finish* wird schlussendlich das neue Projekt angelegt.

Das neue Projekt wird im Projektverwaltungsfenster unter *Projects* angezeigt und beinhaltet bereits eine Quelltextdatei mit dem Namen `main.c`, welche durch Expandieren des Projektes sowie des *Sources*-Ordner eingesehen werden kann (siehe Abb. 5). Der Inhalt der Datei wird durch einen Doppelklick auf den Dateinamen im Editor-Fenster angezeigt. Die Datei `main.c` enthält bereits den fertigen Quelltext für ein einfaches *Hello World!*-Programm.

2 Übersetzen und Ausführen eines Programms

Um ein ausführbares Programm aus einer Quelltextdatei zu erzeugen, muss diese zuerst kompiliert werden. Wählen Sie dazu aus dem Menü *Build* den Eintrag *Build* aus (siehe Abb. 6). Wenn der Kompiliervorgang fehlerlos (*0 errors, 0 warnings*) durchgeführt werden konnte, wird Ihnen dies im Fenster *Logs & others* unter *Build log* mitgeteilt (vgl. Abb. 6).

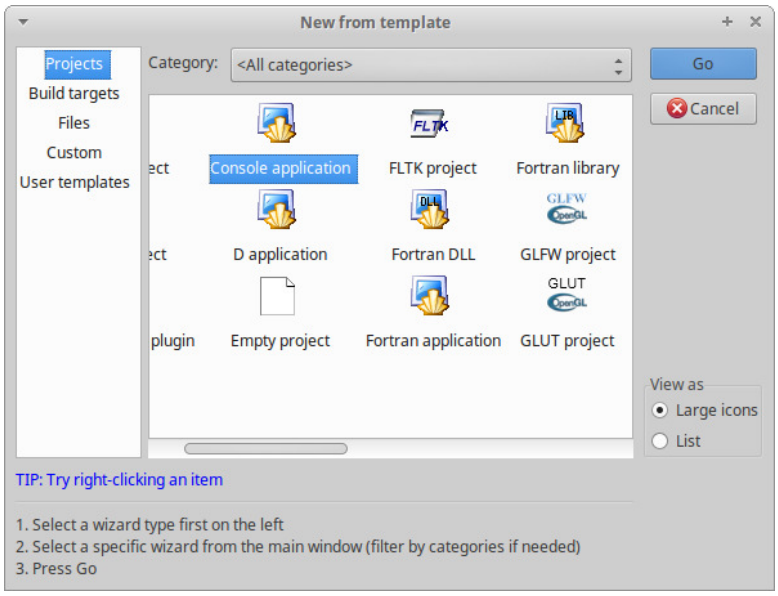


Abbildung 3: Auswahl der Projektvorlage.

Falls während des Kompiliervorgangs Fehler aufgetreten sind, werden Ihnen diese im Fenster *Logs & others* unter *Build messages* aufgelistet. Zu jeder Fehlermeldung wird Ihnen ein Dateiname sowie eine Zeilennummer angezeigt, sowie die entsprechende Stelle im Quelltext durch einen roten Balken markiert (siehe Abb. 7). Mit der Markierung ist jedoch nicht die fehlerhafte Stelle im Quelltext gefunden. Die Ursache kann auch viele Zeilen vor dieser Stelle liegen. Die Markierung schränkt lediglich die Suche des Fehlers auf den Quelltext bis zu dieser Markierung ein.

Lassen Sie sich nicht von einer Flut an Fehlermeldungen entmutigen. Viele Fehlermeldungen sind nur die Folge anderer Fehler oder haben dieselbe Ursache. Daher empfiehlt es sich, die Fehlermeldungen systematisch abzuarbeiten. Es sollte immer mit der ersten Fehlermeldung begonnen werden. Erst wenn die Ursache dieses Fehlers beseitigt ist, und die Fehlermeldung bei einem erneuten Kompiliervorgang nicht mehr auftritt, sollte mit der Beseitigung des nächsten Fehlers begonnen werden.

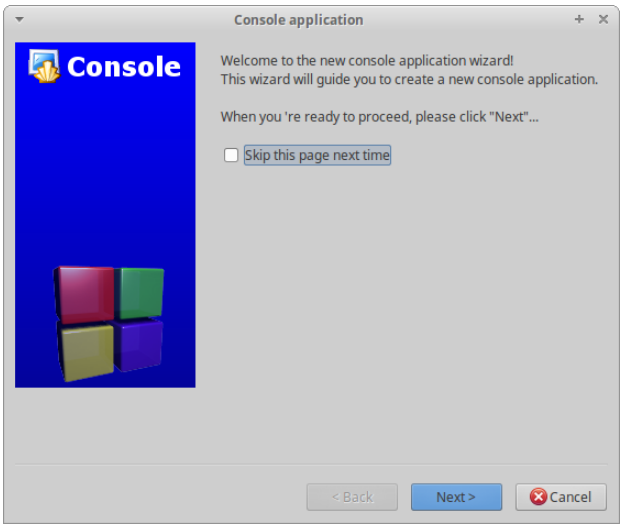
Wurden alle Quelltextdateien eines Programms erfolgreich kompiliert, so kann das Programm durch Aufrufen des Menüpunktes *Run* im Menü *Build* ausgeführt werden. Dabei erscheint ein Konsolenfenster, das alle Bildschirmausgaben des ausgeführten Programms anzeigt (Abb. 8).

Ein sehr nützlicher Befehl ist *Build and run* im Menü *Build*, welcher das Programm kompiliert, und – im Falle einer fehlerlosen Durchführung – sofort danach das Programm startet und ausführt.

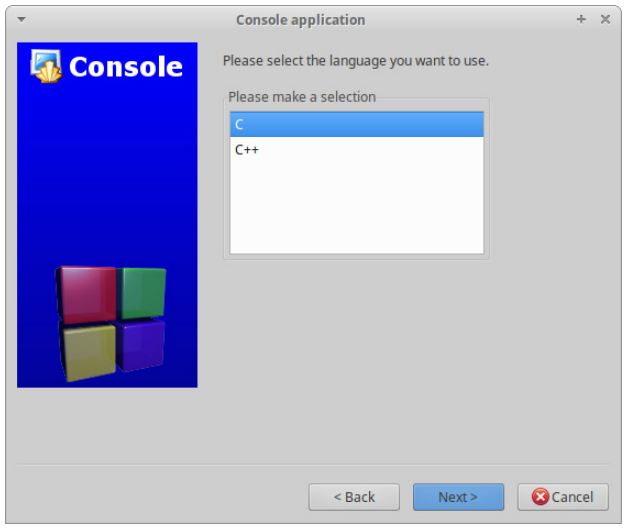
3 Einbindung von Bibliotheken

Im Verlauf der Übungen aus Programmieren 1 benötigen Sie des öfteren Funktionen aus der Mathematikbibliothek. Um die Mathematikbibliothek oder andere Bibliotheken zu verwenden, muss die entsprechende Header-Datei mit einer `#include`-Anweisung in den Quelltext eingebunden werden. Die Header-Datei der Mathematikbibliothek nennt sich `math.h` (siehe Quelltext in Abb. 9).

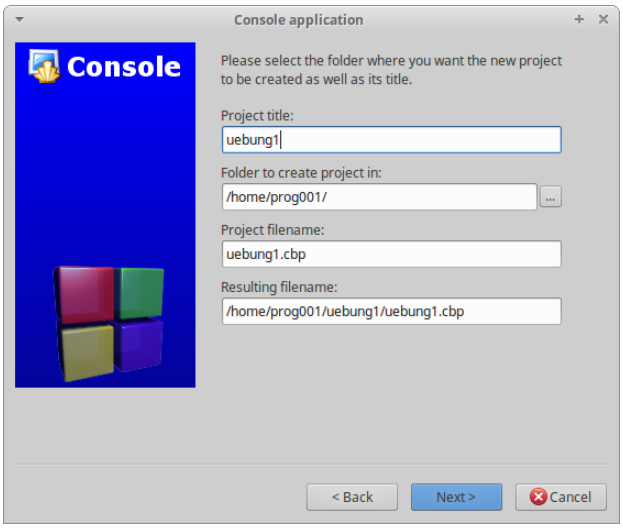
Außerdem muss dem Linker mitgeteilt werden, dass die Mathematikbibliothek eingebunden werden soll. Dazu wählen Sie im Menü *Project* den Punkt *Build options...* aus (Abb. 9). Es erscheint das in Abb. 10 gezeigte Fenster. Wählen Sie auf der linken Seite den Namen Ihres Projektes aus (in diesem Fall *Uebung1*), sodass sämtliche Änderungen der Einstellungen für alle Compiler-Konfigurationen übernommen werden. Dann wählen Sie wie gezeigt den Tab *Linker settings* aus. Unter *Other linker options* fügen Sie `-lm` ein und bestätigen die Änderungen der Linker-Einstellungen durch einen Klick auf *OK*.



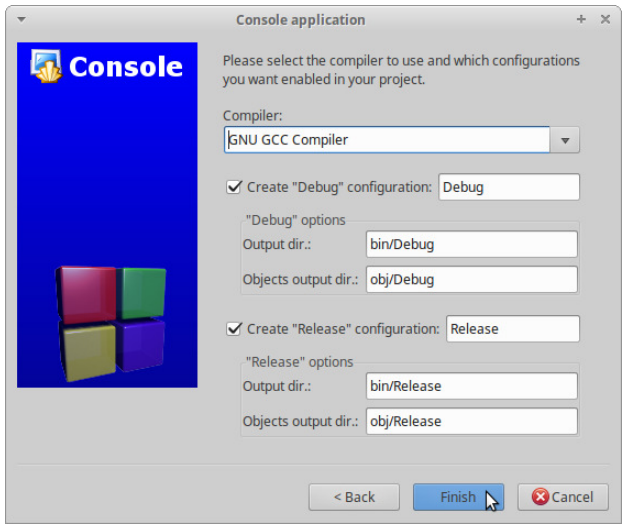
(a) Begrüßungsfenster.



(b) Auswahl der Programmiersprache.



(c) Festlegen des Namens und des Speicherorts des neuen Projekts.



(d) Wahl des Übersetzers.

Abbildung 4: Der Assistent zum Erstellen einer neuen Konsolenanwendung.

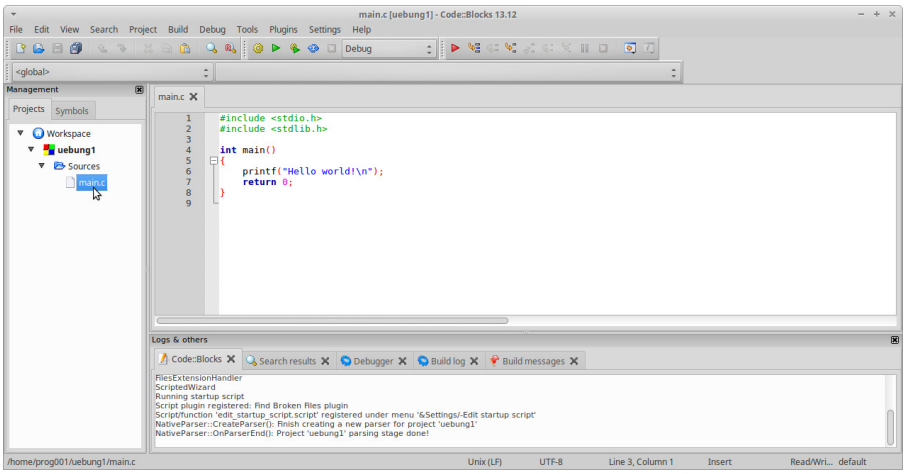


Abbildung 5: Das neu erstellte Projekt mit der automatisch erstellten Datei `main.c`, welche bereits den Quelltext für ein einfaches *Hello World!*-Programm enthält.

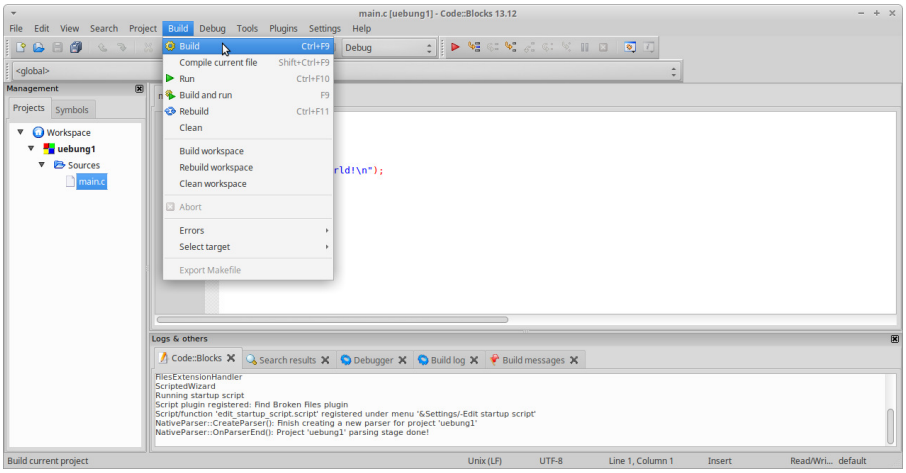


Abbildung 6: Starten des Kompiliervorgangs. Im unten angezeigten *Build log*-Fenster erfahren Sie ob dieser erfolgreich ausgeführt werden konnte.

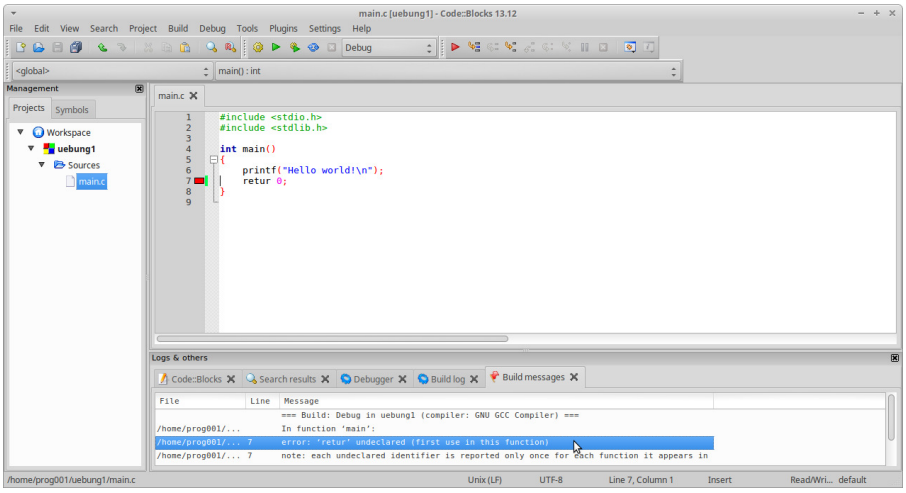


Abbildung 7: Fehlermeldungen nach einem erfolglosen Versuch den Quelltext zu kompilieren.

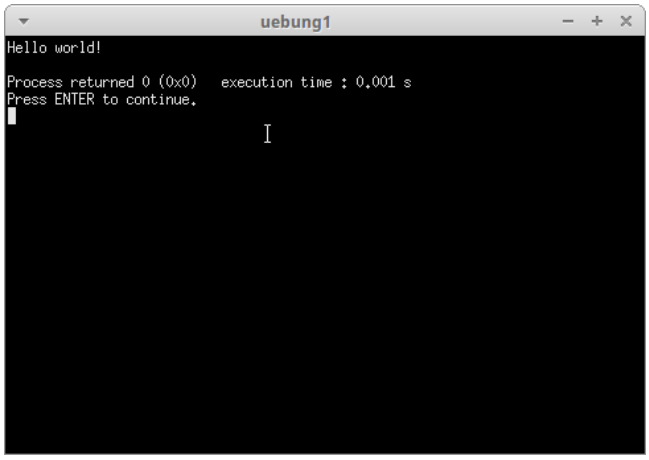


Abbildung 8: Die Bildschirmausgabe wird in einem Konsolenfenster angezeigt.

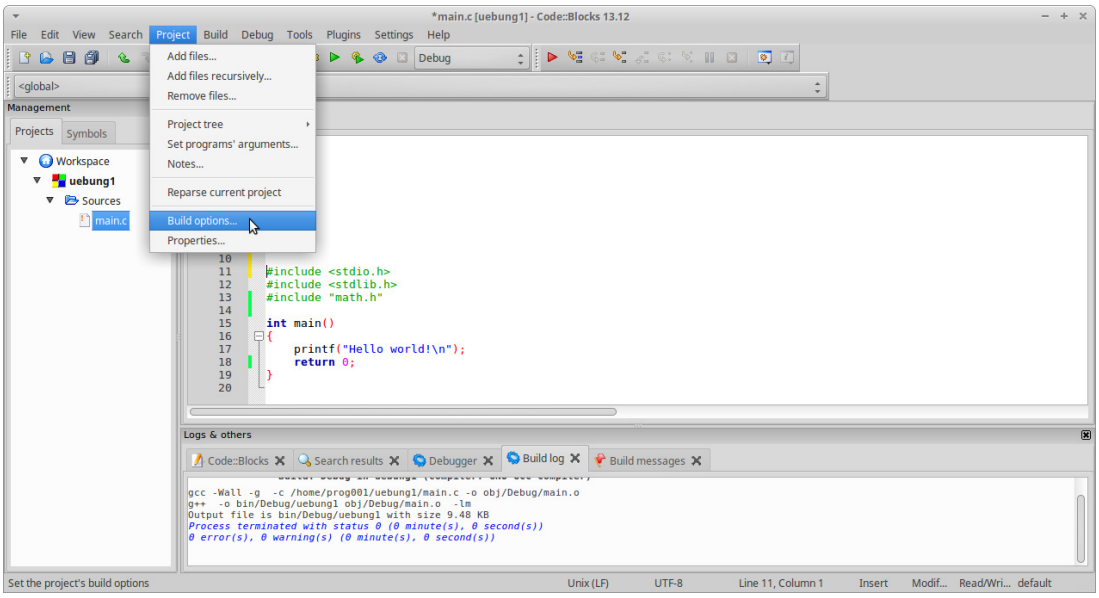


Abbildung 9: Verwenden der Mathematikbibliothek und Aufrufen der *Build Options...*

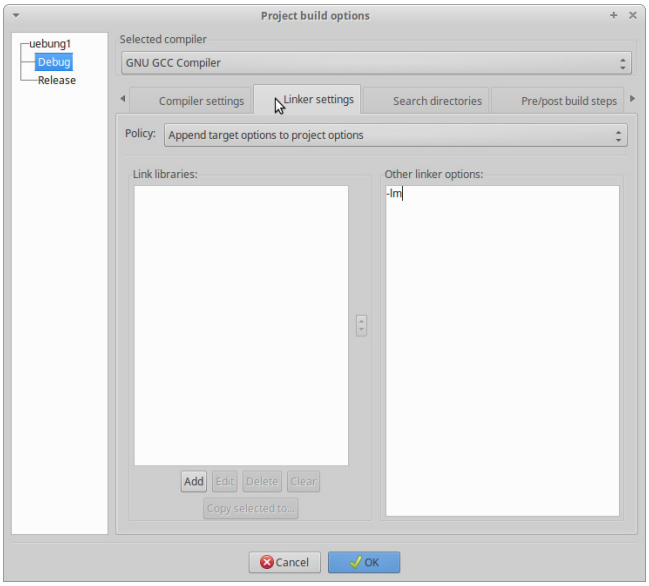


Abbildung 10: Hinzufügen von `-lm` zu den Linker-Einstellungen, um die Mathematikbibliothek einzubinden.

4 Verwalten von Projekten mit mehreren Header- und Quelltextdateien

Im Rahmen der Übungen aus Programmieren 2 müssen Sie Projekte mit mehreren Header- oder Quelltextdateien anlegen. Um dem Projekt eine neue Datei hinzuzufügen, wählen Sie im Menü *File* unter *New* den Eintrag *File...* aus (vgl. Abb. 2). Darauf erscheint das in Abb. 11 dargestellte Fenster.

4.1 Hinzufügen einer Header-Datei

Wollen Sie dem Projekt eine neue Header-Datei hinzufügen, so wählen Sie *C/C++ header* aus. Durch einen Klick auf *Go* starten Sie den Assistenten, der Ihnen bei der Erstellung der neuen Header-Datei hilft. Das erste Fenster (Abb. 12a) ist lediglich ein Begrüßungsfenster. Im nächsten Fenster (Abb. 12b) müssen Sie unter *Filename with full path:* das Verzeichnis, in dem die neue Header-Datei gespeichert werden soll, sowie den Namen der Header-Datei inklusive Dateiendung `.h` angeben. Es wird empfohlen die neue Datei im zugehörigen Projektverzeichnis zu speichern.

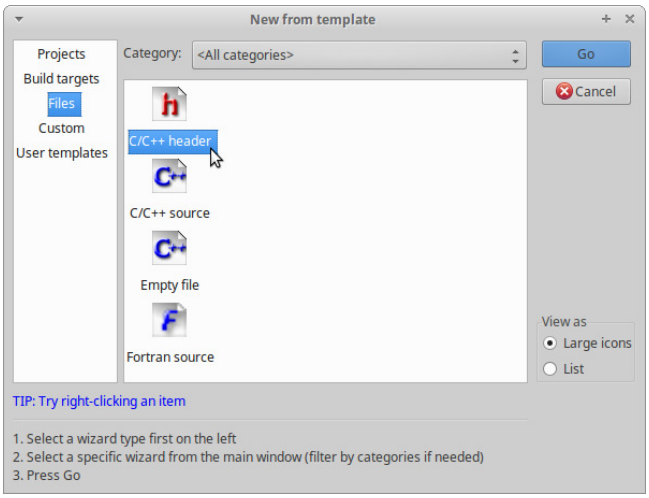
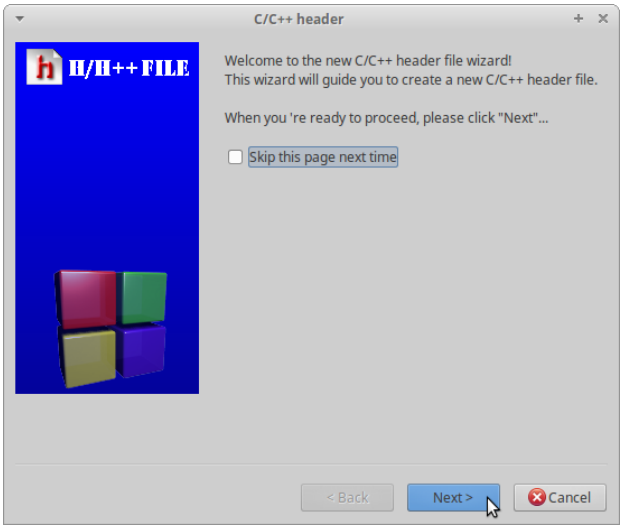
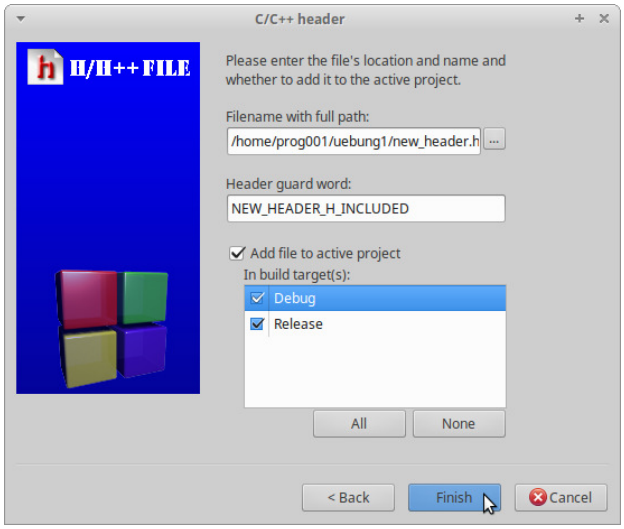


Abbildung 11: Hinzufügen einer neuen Datei.



(a) Das Begrüßungsfenster des Assistenten zum Erstellen einer neuen Header-Datei.



(b) Angabe des vollständigen Pfads und des Dateinamens sowie der Buil-Konfigurationen, denen die Datei angehören soll.

Falls Sie sich nicht sicher sind, wie dieses Verzeichnis heißt, klicken Sie auf den Button rechts neben der Eingabezeile. Daraufhin erscheint ein Fenster, das die Auswahl eines Verzeichnisses erlaubt. Dort ist standardmäßig das Projektverzeichnis eingestellt, so dass Sie lediglich nur noch den neuen Namen der Header-Datei (inklusive der Dateiendung `.h`) im Textfeld *Name:* angeben müssen, um eine korrekte Formatierung für das Textfeld *Filename with full path:* zu erhalten.

Das zweite Textfeld wird bei der Angabe des Dateinamens automatisch gefüllt und kann unverändert übernommen werden. Zum Schluss müssen Sie angeben, bei welchen Build-Konfigurationen die neue Datei berücksichtigt werden soll. Hier sollten Sie durch einen Klick auf den Button *All* alle markieren. Nach dem Drücken von *Finish* wird die neue Datei erstellt und Ihrem Projekt hinzugefügt. Die neue Header-Datei wird im Projektverwaltungsfenster im Ordner *Headers* des aktuellen Projektes angezeigt.

4.2 Hinzufügen einer neuen Quelltextdatei

Das Hinzufügen einer neuen Quelltextdatei ist analog der Vorgehensweise bei einer Header-Datei. Allerdings werden Sie zusätzlich nach der Programmiersprache gefragt, wo Sie wie beim Erstellen eines neuen Projektes, die Programmiersprache `C` auswählen sollten (vgl. Abb. 4b). Der von Ihnen gewählte Dateiname sollte immer die für `C`-Quelltextdateien übliche Endung `.c` aufweisen. Die neu erstellte Quelltextdatei wird im Projektverwaltungsfenster im Ordner *Sources* des aktuellen Projektes angezeigt.

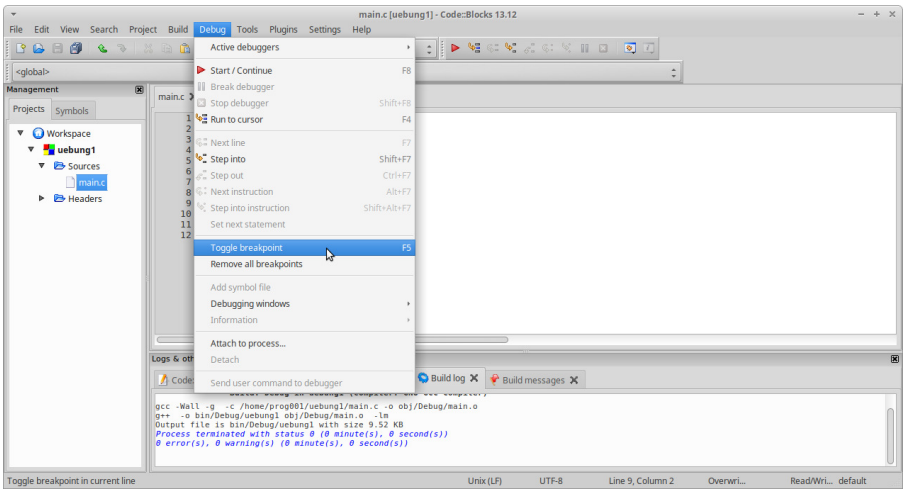


Abbildung 12: Einen Haltepunkt setzen oder entfernen.

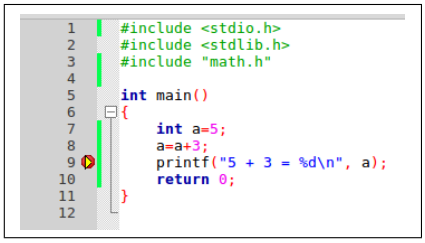


Abbildung 13: Ein Haltepunkt wird durch einen roten Punkt markiert. Der aktuelle Fortschritt der Programmausführung wird durch das gelbe Dreieck angezeigt.

5 Debuggen eines Programms

Ein Debugger unterstützt das Auffinden logischer Fehler in einem bereits erfolgreich kompilierten Programm. Während eines Debug-Vorgangs kann das Programm bis zu einem Haltepunkt ausgeführt oder auch schrittweise abgearbeitet werden. Der Debugger ermöglicht auch das Anzeigen der Werte von Variablen.

Das Setzen eines Haltepunkts erfolgt durch Platzieren der Textmarke in der gewünschten Zeile und Aufrufen von *Toggle breakpoint* im Menü *Debug* (Abb. 12). Durch ein erneutes Aufrufen von *Toggle breakpoint* in einer Zeile, in der bereits ein Haltepunkt gesetzt wurde, kann dieser wieder entfernt werden. Ein Haltepunkt wird im Editor am linken Rand durch einen roten Punkt symbolisiert (Abb. 13).

Der Debugger wird durch die Auswahl *Start* im Menü *Debug* gestartet (vgl. Abb. 12). (Wenn dabei ein Dialogfenster erscheint, das Sie fragt, ob Sie das Layout speichern wollen, lesen Sie Abschnitt 6.4.) Die Programmausführung schreitet bis zum Erreichen des ersten Haltepunkts fort. Die Stelle, bis zu der das Programm ausgeführt wurde, wird durch ein gelbes Dreieck markiert (Abb. 13). Dabei ist zu beachten, dass die durch das Dreieck markierte Zeile selbst noch nicht ausgeführt wurde.

Wenn Sie den Debugger starten wollen, aber die Ausführung bereits in der ersten Code-Zeile unterbrechen wollen, können Sie auch *Step into* aus dem Menü *Debug* aufrufen. Solange der Debugger aktiv ist, stellt das Menü *Debug* weitere Befehle zur Verfügung. Mit *Continue* setzen Sie die Ausführung bis zum nächsten Haltepunkt fort. *Next line* führt den Code der aktuellen Zeile aus und hält die Ausführung in der nächsten Code-Zeile wieder an. Mit dem Aufruf *Run to cursor* setzen Sie die Programmausführung bis zur aktuellen Position der Textmarke fort. Mit *Stop Debugger* beenden Sie den Debugger.

Um den Inhalt von Variablen während der Ausführung des Programms mit dem Debugger zu beobachten, öffnen Sie das *Watches*-Fenster. Dieses kann über das Menü *Debug* im Unterpunkt *Debugging windows* erreicht werden (siehe Abb. 14). Das *Watches*-Fenster zeigt den aktuellen Zustand von lokalen Variablen an (siehe Abb. 15) und ist somit ein nützliches Werkzeug zum Auffinden logischer Fehler.

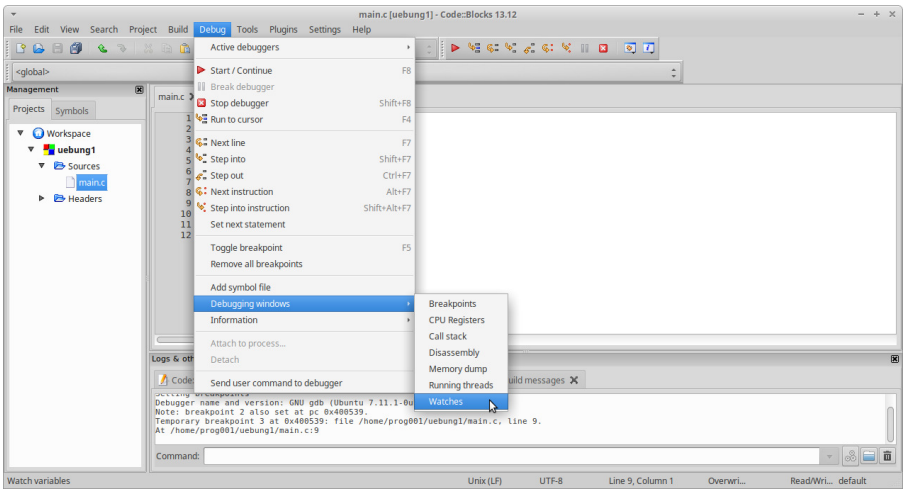


Abbildung 14: Öffnen des *Watches*-Fensters.

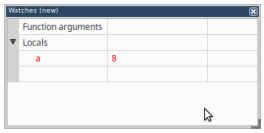


Abbildung 15: Im *Watches*-Fenster können die aktuellen Werte von Variablen eingesehen werden. Der hier gezeigte Zustand der Variable `a` wird bei der Ausführung bis zur Zeile 7 des in Abb. 13 gezeigten Codes erreicht.

6 Tipps & Tricks

6.1 Schriftgröße

Um die Schriftgröße im Editor-Fenster zu verändern, wählen Sie im Menü *Settings* den Eintrag *Editor...* aus. Im erscheinenden Fenster können Sie durch Klicken des Buttons *Choose* bei den *General Settings* im Abschnitt *Font* ein Dialogfenster aufrufen, das die Änderung der Schriftgröße ermöglicht.

6.2 Verschwundene Fenster und Symbolleisten

Falls Sie irgendein Fenster, wie z.B. das Projektverwaltungsfenster, vermissen, können Sie dieses im Menü *View* wieder einblenden. Auch verschwundene Symbolleisten können Sie im selben Menü unter *Toolbars* wieder sichtbar machen.

6.3 Tastaturkürzel und Symbolleisten

Viele Befehle sind nicht nur über das Menü erreichbar, sondern auch über bereits standardmäßig festgelegte Tastaturkürzel sowie Symbolleisten. Wenn für einen Befehl ein Tastaturkürzel definiert ist, ist dieses im Menü rechtsbündig angezeigt (siehe z.B. Abb. 14). Im Menü ist außerdem ersichtlich, ob der Befehl über eine Symbolleiste angesprochen werden kann. Ist das der Fall, ist das entsprechende Symbol am linken Rand des Menüs abgebildet.

6.4 Anordnung der Fenster

Standardmäßig werden Sie von `Code::Blocks` durch Anzeigen des in Abb. 16 abgebildeten Fensters des Öfteren gefragt, ob Sie das Layout speichern wollen. Das Layout ist in diesem Fall die Anordnung der verschiedenen Fenster der Entwicklungsumgebung. Wenn Sie nicht mehr danach gefragt werden wollen, kreuzen

Sie *Don't annoy me again!* an und klicken Sie auf *No* oder *Yes*, je nachdem, ob sich `Code::Blocks` ein verändertes Layout nie oder immer merken soll.

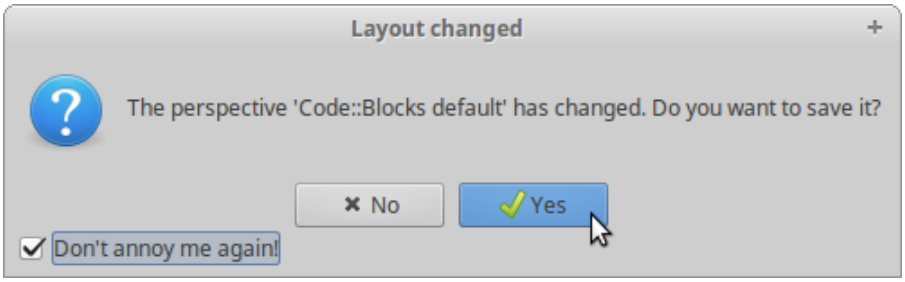


Abbildung 16: Dialogfenster mit der Frage, ob Sie das Layout speichern wollen.

6.5 Setzen von Programm-Argumenten

Dem Hauptprogramm kann bei der Ausführung durch `Code::Blocks` analog zum Aufruf von der Konsole eine Liste von Argumenten übergeben werden. Wählen Sie dazu im Menü *Project* den Punkt *Set program's arguments...* aus. Tragen Sie danach im Feld *Program arguments* die von Ihnen gewünschten Argumente ein und bestätigen Sie mit *OK* (Abb. 17).

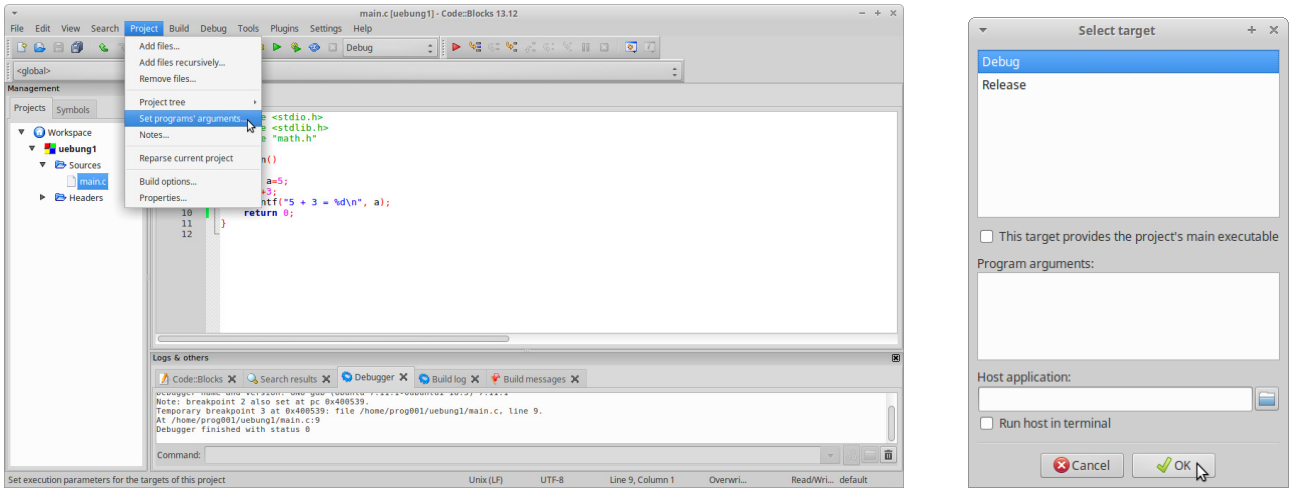


Abbildung 17: Links: Menü-Punkt zum Setzen von Programm-Argumenten
Rechts: Die Argumente werden im Feld *Program arguments* eingetragen.