



Erste Schritte in der C++ Programmierung Installation und grundlegende Benutzung des Compilers MinGW und der Entwicklungsumgebungen Code::Blocks und Visual Studio

Dipl.-Ing. Adnene Gharbi, Dipl.-Ing. Christoph Roth Dipl.-Ing.(FH) Tobias Schwalb, Dipl.-Ing. Michael Tansella cand. B.Sc. Andreas Kleff, cand. B.Sc. Felix Mauch

Institutsleitung Prof. Dr.-Ing. K. D. Müller-Glaser Prof. Dr.-Ing. J. Becker Prof. Dr. rer. nat. W. Stork

KIT - Universität des Landes Baden Württemberg und nationales Forschungszentrum in der Helmholtz-Gemeinschaft



Vorwort

Dieses Dokument soll Ihnen helfen Ihre Kenntnisse, die Sie in der Übung und der Vorlesung INFORMATI-ONSTECHNIK erlernt haben, praktisch umzusetzen.

Sie werden lernen wie Sie den Compiler MinGW installieren und wie Sie diesen nutzen. Sie werden außerdem die einfache integrierte Entwicklungsumgebung Code::Blocks kennen lernen. Sie werden lernen, diese zu installieren und zu nutzen.

Anschließend werden Sie einen Einstieg in die Entwicklungsumgebung Visual Studio erhalten.

Die notwendigen Schritte werden alle exemplarisch unter Windows7 erläutert, unter anderen Betriebssystemen laufen sie eventuell etwas anders ab. Visual Studio lässt sich ohne Umwege nur unter Windows verwenden.

Es werden Hinweise für Mac OS X und Linux angegeben, für die Durchführung bitten wir jedoch um Selbstrecherche.

Verbesserungsvorschläge, sowie jede Art von Kritik zu diesem Dokument, sind sehr erwünscht (michael.tansella@kit.edu, tobias.schwalb@kit.edu).

©ITIV, Karlsruhe 2011 2. Auflage



Inhaltsverzeichnis

1	Min	ıGW	4
	1.1	Installation des Compilers	4
		1.1.1 MinGW herunterladen	4
		1.1.2 Ausführen der Installationsdatei	4
		1.1.3 Setzen der Umgebungsvariablen	6
	1.2	Die Kommandozeile und Bedienung des Compilers	8
	1.3	Fehlermeldungen	9
2	Cod	le::Blocks	0
	2.1	Code::Blocks installieren	0
		2.1.1 Code::Blocks herunterladen	0
		2.1.2 Ausführen der Installationsdatei	0
	2.2	Code::Blocks-Schnelleinstieg	2
	2.3	Compilermeldungen	5
	2.4	Compilieren und ausführen	6
	2.5	Der Debugger	6
	2.6	Editor anpassen	8
3	Ver	wendung von Visual Studio 1	9
	3.1	Visual Studio installieren	9
		3.1.1 Visual Studio herunterladen	9
		3.1.2 Installation	9
	3.2	Ein C++Projekt erstellen	1
	3.3	Compilieren und ausführen	3
	3.4	Der Debugger	4
	3.5	Editor anpassen	5



Kapitel 1 MinGW

In diesem Kapitel werden Sie den Compiler MinGW installieren und ein erstes einfaches C++ Programm erstellen.

1.1 Installation des Compilers

Der erste Schritt vor dem Programmieren ist die Installation eines Compilers, also des Übersetzers von Programmcode in maschinenlesbare Nullen und Einsen. Für dieses Skript wird der GNU C++ Compiler verwendet, welcher ursprünglich für UNIX entwickelt wurde und heute für alle Plattformen verfügbar ist. Das Paket *MinGW* stellt den GNU Compiler unter Windows zur Verfügung, unter Linux müssen Sie lediglich das Paket g++ und dessen Abhängigkeiten installieren.

1.1.1 Download von MinGW

Loggen Sie sich bei http://www.e-studium.org ein und wählen Sie die Seite der Veranstaltung Informationstechnik. Dort finden Sie im Abschnitt über Links den Verweis zum Download von MinGW. Klicken Sie den Link (http://www.mingw.org) an und laden Sie die MinGW Installationsdatei herunter.

1.1.2 Ausführen der Installationsdatei

Führen Sie die heruntergeladene Installationsdatei aus und wählen Sie als Installationsort am Besten C:\Programme\MinGW. Es ist darauf zu achten, dass im Pfad zum MinGW Ordner keine Leerzeichen enthalten sind. Durchlaufen Sie die Installation, wie auf den Abbildungen 1.1 bis 1.9 (Seiten 4 bis 6) zu sehen ist.





Klicken Sie auf Next. Abbildung 1.1: Beginn der Installation

Wählen Sie *Download and Install* und klicken Sie auf Next.

Abbildung 1.2: Auswahl der Installationsart









1.1.3 Setzen der Umgebungsvariablen

Nachdem die Installation vollendet ist, müssen Sie noch die sogenannten Umgebungsvariablen setzen. Folgen Sie hierzu den Anweisungen in den Abbilgungen 1.10 bis 1.17.

Programme (5)	
Systemsteuerung (46)	
👰 <u>System</u>	
deteien und -einstellungen	
system ×	Herunterfahren

Klicken Sie auf das *Windowssymbol* und geben Sie *System* ein und wählen Sie *System* aus.

Abbildung 1.10: Bei Win7 & WinVista: Schritt 1





Wählen Sie Erweiterte Systemeinstellungen aus. Abbildung 1.11: Bei Win7 & WinVista: Schritt 2



Systemeigensch:	aften				×			
Computername	Hardware	Erweitert	Computerschutz	Remote				
Sie müssen als Administrator angemeldet sein, um diese Änderungen durchführen zu können. Leistung								
Visuelle Effel Speicher	kte, Prozess	orzeitplanun	g, Speichernutzun	g und virtueller				
				Einstellungen				
Benutzerprof	ile				- I			
Desktopeins	tellungen be	ezüglich der /	Anmeldung					
				Einstellungen]			
_ Starten und ∖	Wiederherst	ellen			- I			
Systemstart,	Systemfehle	er und Debug	ginformationen					
				Einstellungen]			
Umgebungsvariablen								
)K Abbre	chen Überne	ehmen			

Klicken Sie mit der rechten Maustaste auf das Arbeitsplatzsymbol und wählen Sie aus dem erscheinenden Kontextmenü den Punkt *Eigenschaften*.

Abbildung 1.12: Bei WinXP: Arbeitsplatz Kontextmenü aufrufen

Den Reiter *Erweitert* wählen und auf Umgebungsvariablen klicken.

Abbildung 1.13: Erweitert-Reiter in den Systemeigenschaften



Umgebungsvariable	n 💌	
- Benutzervariablen f	ür Andreas	
Variable	Wert	
TEMD	% I ISEP DD OETI E%) (uppData) (ocali Temp	
TMP	%USERPROFILE%\AppData\Local\Temp	
	Neu Bearbeiten Löschen	
Systemvariablen		
Variable ComSpec FP_NO_HOST_C. NUMBER_OF_P OS	Wert C:\Windows\system32\cmd.exe NO 1 Windows_NT	
	Neu Bearbeiten Löschen	Klicken Sie im oberen Teil auf Neu.
	OK Abbrechen	Abbildung 1.14: Umgebungsvariablen: Übersichtsfenster
Benutzervariable b Name der Variablen Wert der Variablen:	earbeiten	Geben Sie als Namen PATH und als Wert C:\Programme\MinGW\bin (XP) C:\Program Files\MinGW\bin (Win7, WinVista) oder\ bin entsprechend dem von Ihnen gewählten Installationspfad ein. Abbildung 1.15: Anlegen einer neuen Umgebungsvariable
Programme (1) Cmd.exe Weitere Ergeonne Cmd	X Herunterfahren >	Starten Sie den Computer neu, starten Sie die <i>Ein- gabeaufforderung</i> (Abs. 1.2). Bei Win7 & WinVista siehe Abbildung. WinXp: <i>Start→Ausführen</i> und <i>cmd</i> eingeben. Abbildung 1.16: Testen der Umgebungsvariable
C:\Windows\system Microsoft Windo Copyright (c) 21 C:\Users\Andrea g++: no input f C:\Users\Andrea	n32\cmd.exe ws [Version 6.1.7600] 009 Microsoft Corporation. Alle Rec s>g++ iles s>_	Geben Sie g++↔ ein. Die Ausgabe sollte g++: no input files sein. Abbildung 1.17: Testen der Umgebungsvariable

1.2 Die Kommandozeile und Bedienung des Compilers

Heute wird üblicherweise eine integrierte Entwicklungsumgebung zur Programmierung verwendet, da eine solche einheitliche und einfache Programmoberfläche die Arbeit erleichtert. Für die ersten Gehversuche reicht allerdings auch das, was sie bis jetzt installiert haben.

Um ein C++ Programm zu erstellen, öffnen Sie nun den Windows Editor und schreiben Sie zum Beispiel das Programm aus dem Kompendium. Speichern Sie nun diese Datei als *HelloWorld.cpp*.

Jetzt muss ein Kommandozeilenfenster geöffnet werden, um darin die Programme zur Übersetzung aufzurufen. Dazu wählen Sie $Start \rightarrow Ausführen$ und tippen cmd ein, so dass sich nach einem Klick auf Ausführen ein leeres Kommandozeilenfenster öffnet.

Die Kommandozeile war vor der Einführung der grafischen Oberfläche die einzige Art, einen Computer



direkt zu bedienen. Programme werden aufgerufen, indem man ihren Namen eintippt und beim Programmaufruf noch die nötigen Parameter und Optionen übergibt. Heute bietet jedes Betriebsystem eine Kommandozeile an, über die der Computer alternativ bedient werden kann. Im weiteren Verlauf ist das Kommando cd (cd für *change directory*) nötig mit dem Sie auf der Kommandozeile in verschiedene Ordner navigieren können.

```
1 cd C:\Windows\
2 cd ..
3 cd Windows
```

Dieser Befehl zum Beispiel navigiert die Kommandozeile erst in den Windows Ordner auf dem Laufwerk C. Der zweite Befehl navigiert einen Ordner höher, also in das Laufwerk C selber. Schließlich navigiert der dritte Befehl die Kommandozeile wieder in den Windows Ordner im aktuellen Pfad. Der aktuelle Pfad wird in der Kommandozeile immer links in der aktuellen Zeile angezeigt.

Um den C++ Compiler der GNU Compiler Collection aufzurufen müssen Sie den Befehl g++ verwenden. Dieser Befehl muss vom Namen einer C++ Datei gefolgt sein, welche übersetzt werden soll. Außerdem können Sie mit -NameDerOption Optionen angeben. Zum Beispiel listet g++ -help Ihnen alle möglichen Kommandozeilenoptionen auf oder die g++ -v NameDerC++Datei.cpp bewirkt, dass beim Compilieren alle Teilschritte genau angezeigt werden.

Ein Aufruf von g++ HelloWorld.cpp erstellt das ausführbare Programm in einer Datei mit dem Namen *a.exe.* Mit der Option -o NameDerAusführbarenDatei kann ein anderer Name gewählt werden und der vollständige Aufruf lautet somit: g++ HelloWorld.cpp -o HelloWorld.exe¹.

Mit diesem Befehl wird das Programm als ausführbare Datei *HelloWorld.exe* generiert, es sei denn, der Compiler stößt während des Übersetzens auf Fehler im Quellcode.



1. Zeile: Compilieren

3. Zeile: Ausführen

Abbildung 1.18: Compiler verwenden

1.3 Fehlermeldungen

Findet der Compiler oder der Linker Fehler in Ihrem Quellcode, wird das Kompilieren abgebrochen und eine Fehlermeldung ausgegeben. Diese Fehlermeldungen sind meistens nicht einfach zu verstehen und von Compiler zu Compiler unterschiedlich. Es ist jedoch wichtig, die Fehlermeldungen deuten zu können, um die eigenen Fehler im Programmcode zu finden. Außerdem gibt der Compiler noch zu jedem Fehler die Zeilennummer an, in welcher der Fehler aufgetreten ist. Diese Information ist sehr nützlich, um den Fehler im Code lokalisieren zu können, kann jedoch auch irreführend sein. Hier sind einige typische Fehlermeldungen aufgelistet:

- Error: "xyz" was not declared in this scope. Diese Fehlermeldung tritt auf, wenn der Compiler einen Namen findet, den er weder einer Variable noch einer Funktion zuweisen kann. Haben Sie also einen Funktionenname oder Variablenname nicht richtig geschrieben, wird Ihnen dieser Fehler erscheinen.
- Error: expected ', ' or '; ' before 'xyz' Diese Fehlermeldung tritt auf, wenn Sie vergessen haben ein Semikolon am Ende einer Anweisung zu schreiben. Achtung: Die angegebene Zeile ist die Zeile nach derjenigen ohne Semikolon, dieses wird vor dem Text xyz gesucht.
- Error: expected primary-expression before '{' token Diese Fehlermeldung tritt zum Beispiel auf, wenn Sie Klammern oder geschwungene Klammern von Funktionen oder Abfragen nicht richtig gesetzt haben oder diese fehlen.

Oft resultiert ein Versuch zu kompilieren in einer ganzen Menge Fehler, welche teilweise jedoch von einander abhängen. Deshalb ist es ratsam, immer den ersten Fehler zu entfernen und danach neu zu kompilieren, denn die Fehleranzahl sinkt dann häufig sehr schnell.

 $^{^1}$ Auf Windows-Systemen ist das Suffix .exe notwendig, um ausführbare Dateien zu erkennen. Auf Linux, Mac OS X und anderen Betriebssystemen ist das nicht notwendig.



Kapitel 2 Code::Blocks

2.1 Code::Blocks installieren

Nachdem Sie nun in Kapitel 1 den GNU Compiler installiert haben, sind Sie in der Lage, Programme im Windows-Editor zu schreiben und diese in der Kommandozeile zu kompilieren.

In diesem Abschnitt werden Sie die integrierte Entwicklungsumgebung (IDE) Code::Blocks installieren. Eine IDE stellt Ihnen verschiedene Funktionalitäten zur Verfügung. Unter anderem sind diese:

- Integrierter Editor mit Syntaxhervorhebung
- Einfacher Aufruf des Compilers
- Einen Debugger (Abschnitt 2.5, Seite 16)
- Komfortable Projektverwaltung

2.1.1 Download von Code::Blocks

Loggen Sie sich bei http://www.e-studium.org ein und wählen Sie die Seite der Veranstaltung Informationstechnik. Dort finden Sie im Abschnitt über Links den Verweis zu Download von Code::Blocks.Klicken Sie den Link (http://www.codeblocks.org/downloads/5) an und laden Sie die MinGW Installationsdatei mit dem Dateinamen codeblocks-x.xx-setup.exe herunter. Auf dieser Seite finden Sie auch die Installationsdateien für Mac OS X, sowie für verschiedene Linux Distributionen. Bei den meisten Distributionen müssten Sie Code::Blocks allerdings auch aus den Standardpaketquellen installieren können.

2.1.2 Ausführen der Installationsdatei

Folgen Sie zur Installation den Anweisungen in den Abbildungen 2.1 bis 2.7.



Klicken Sie auf Next.

Abbildung 2.1: Beginn der Installation











Klicken Sie hier auf Nein. Sie können das Programm später wie gewohnt über das Startmenü starten.

Abbildung 2.6: Code::Blocks starten?

Klicken Sie auf Finish. Abbildung 2.7: Installation ist abgeschlossen

2.2 Code::Blocks-Schnelleinstieg

Um Ihr erstes Programm in der Entwicklungsumgebung Code::Blocks zu erstellen, starten Sie das Programm und folgen Sie anschließend den Anweisungen von Abb. 2.8 bis Abb. 2.17

Compilers auto-detection		
Compiler	Status	▲ Set as default
GNU GCC Compiler	Detected	
Microsoft Visual C++ Toolkit 2003		
Microsoft Visual C++ 2005/2008		
Borland C++ Compiler (5.5, 5.82)		=
Digital Mars Compiler		
OpenWatcom (W32) Compiler		
GNU GCC Compiler for MSP430		
Cygwin GCC		
Intel C/C++ Compiler		
SDCC Compiler		
Tiny C Compiler		
GDC D Compiler		-
Diskel Mana D. Cassallan		
Default compiler: GNU GCC Compiler		
	ОК	



Beim ersten Start von Code::Blocks werden Sie gefragt, mit welchem Compiler Code::Blocks arbeiten soll. Wählen Sie hier den GNU GCC Compiler und klicken Sie auf OK.

Abbildung 2.8: Auswahl des Compilers

Schließen Sie das "Tip of the Day"-Fenster mit dem Close-Button.

Abbildung 2.9: Der erste Start von Code::Blocks





Klicken Sie im Hauptfenster auf Create a new project.

Abbildung 2.10: Der Hauptbildshirm von Code::Blocks





Hier können Sie zwischen verschiedenen Vorlagen wählen. Für diese Vorlesung ist die Vorlage *Console application* völlig ausreichend. Wählen Sie also diese aus und klicken Sie auf OK.

Abbildung 2.11: Die Vorlagen-Auswahl

Wenn Sie diese Seite in Zukunft nicht mehr sehen wollen, setzen Sie den Haken bei "Skip this page next time" Klicken Sie auf Next.

Abbildung 2.12: Der Projektassistent





Console application		×
Console	Please select the language you want to use. Please make a selection	
	< Back Next > Canc	el





Wählen Sie C++ aus und klicken Sie auf Next. Abbildung 2.13: Wahl der Programmiersprache

Geben Sie in der ersten Zeile den Namen des Projekts ein, für das erste Beispiel **helloCB**. In der Zeile darunter müssen Sie den Speicherort für das Projekt eingeben. Geben Sie einen beliebigen Speicherort an, an dem Sie das Projekt speichern möchten. Code::Blocks erzeugt dort automatisch einen Unterordner mit dem Namen des Projekts. Der Assistent füllt nun automatisch die letzten beiden Zeilen aus. Klicken Sie auf Next.

Abbildung 2.14: Dem Projekt einen Namen geben.

Lassen Sie hier alles auf den Standardwerten, sonst kann Ihr Programm später unter Umständen nicht erstellt werden. Klicken Sie auf Finish, um den Assistenten zu beenden.

Abbildung 2.15: Einstellungen des Compilers



💾 main.cpp [Hallo World] - Code::Blocks 8.02								
File Edit View Search Pro	ject Build Debu	g wxSmith Tools	Plugins Settings	Help				
1								
😒 🕨 🥵 🐼 Build target: Debug 🔹								
🚺 🏰 😚 🗟 🤭 🖑 🐼	🗔 i.							
Management X	main.cpp ×			۵ ۵				
Projects Symbols	1 \$1 2 us 4 5 in 6 ☐ (7 8 9 } 10 <	<pre>nclude <iostream "hellc="" 0;<="" <<="" cout="" ing="" main()="" namespace="" pre="" return="" st="" t=""></iostream></pre>	≥> :d; > world!" << end.	1;				
	Logs & others			×				
	🚺 Code::Blocks	Search results	ؼ Build log	< ▶				
default Line 1, Col	umn 1	Insert	Read/Write	defau				

Mit einem Doppelklick auf Sources und anschließend auf *main.cpp* im linken Bereich *Management* öffnet sich die Hauptdatei Ihres Projekts. Code::Blocks hat für Sie bereits ein **Hello World!**-Programm erstellt, wie Sie es aus dem Kompendium Kapitel 1 kennen.

Abbildung 2.16: Ihr erzeugtes Projekt

C:\C++\halloCB\bin\Debug\halloCB.exe	- • ×
Hello world! Process returned 0 (0x0) execution time Press any key to continue.	: 0.050 s

Durch Drücken der Taste **F9** erstellen Sie das Projekt und führen es gleich aus. Es erscheint ein Konsolenfenster mit der Ausgabe Ihres Programms: Hello world!.

Abbildung 2.17: Erstellen und ausführen

Nun können Sie bereits ein Projekt erstellen und dieses vom Compiler übersetzen lassen. In den folgenden Abschnitten werden Sie Code::Blocks etwas näher kennenlernen.

2.3 Compilermeldungen

In Code::Blocks erscheinen die Fehlermeldungen des Compilers in einem kleinen Fenster am unteren Bildschirmrand (Abb. 2.18). Fehler werden rot hervorgehoben, Warnungen blau. Wenn Sie auf eine Fehlermeldung klicken, springt Code::Blocks automatisch zur entsprechenden Codezeile und markiert diese mit einem roten Rechteck am Rand.

💾 main.cpp [debuggerExample] - Code::Blocks 8.02	×					
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help							
🗋 🖆 🗃 🕼 🔦 🗞 🕼 🕼 🖳 S							
main() : int							
😧 👂 🐼 🐼 Build target: Debug 💦 🕴 🖷 🐨 🔭 🥐 🖑 🐼 🗔 🚛							
Management X	main.cpp ×	4 4					
Projects Symbols	3 using namespace std;						
🖃 😡 Workspace	4						
🚊 💾 debuggerExample	5 = int main() {						
🗄 - 🔁 Sources	7						
	8 // Alphabet in das Array schreiben						
	9						
	ll -)	A.					
	12						
	13 // Alphabet ausgeben						
	14 ⊟ for(int i = 0; i < 26; i++) {						
	15 cout << alphabet[i] << " ";						
	17						
	18 cout << endl;	-					
	< <u> </u>	P.					
	Logs & others	×					
	🃝 Code::Blocks 🔍 Search results 🥎 Build log Build messages 🤤 Debugger	۹ ۵					
	File Line Message	*					
	C:\Users\Andre In function `int main()':						
	C:\Users\Andre 10 error: `alphbet' was not declared in this scop	e 🗐					
	C:\Users\Andre 10 warning: unused variable 'alphbet'						
		-					
C:\Users\Andreas default	Line 10, Column 1 Insert Read/Write default						

Abbildung 2.18: Fehlermeldungen beim Erstellen



2.4 Compilieren und ausführen

Wenn Sie ein neues Programm geschrieben haben, möchten Sie dieses vermutlich vom Compiler übersetzen lassen. Dies können Sie in Code::Blocks auf drei verschiedene Varianten machen:

- Klicken Sie auf das Menü $Build \rightarrow Build$ oder
- Drücken Sie [Strg] + [F9] oder
- Klicken Sie auf das *Build*-Icon ^S in der Werkzeugleiste.

Ihr erstelltes Programm können Sie nun wieder über das Menü ($Build \rightarrow Run$), über die Tastenkombination [Strg] + [F10] oder über das Run-Icon) ausführen. Sie können beide Schritte auch auf einmal machen:

- Klicken Sie auf das Menü
 Build $\rightarrow Build$ and run oder
- Drücken Sie **F9** oder
- Klicken Sie auf das Build+Run-Icon \Im in der Werkzeugleiste.

2.5 Der Debugger

Um logische Fehler in einem Programm zu finden, wünscht man sich oft, das Programm einmal Schritt für Schritt durchzugehen und dabei die Variablen beobachten zu können. Der sogenannte Debugger macht dies möglich. Auch Code::Blocks bietet einen solchen Debugger.

Im Folgenden wird darauf verzichtet, für alle Aktionen auch den Menüaufruf zu erläutern. Alle Funktionalitäten des Debuggers finden sich im Menü *Debug*.

Um den Debugger nun zu starten, muss man das Projekt zunächst erstellen (Abschnitt 2.4).

Der Debugger führt das Programm solange ganz normal aus, bis er zu einem sogenannten *Breakpoint* gelangt. Ein solcher *Breakpoint* lässt sich setzen, indem man mit der Maus rechts neben eine Zeilennummer im Code klickt. Es erscheint dann an dieser Stelle ein kleiner roter Kreis (s. Abb. 2.19). Ein erneuter Klick auf die Stelle löscht den *Breakpoint* wieder. Das Selbe lässt sich auch mit der Taste **F5** in der Zeile erreichen, in der momentan der Cursor steht.

Starten Sie nun den Debugger, indem Sie F \overline{F} drücken oder auf das *Debug*-Icon \downarrow in der Werkzeugleiste klicken.

Sie können jetzt mit dem *Debug*-Icon 🎼 oder den Tasten [Strg] + [F7] zum nächsten Breakpoint fortfahren.

Mit dem \mathfrak{F} Icon oder \mathbb{F} können Sie Zeile für Zeile fortfahren. Um nun die Werte der aktuell gültigen Variablen anzuzeigen, wählen Sie in der Menüleiste *Debug* \rightarrow *Debugging windows* \rightarrow *Watches*. Es erscheint nun ein kleines Fenster, welches Sie durch Verschieben an den rechten Rand in die Code::Blocks-Oberfläche einbetten können (Abb. 2.19). Variablen, die erst kürzlich geändert wurden, werden rot hervorgehoben.





Abbildung 2.19: Der Debugger in Aktion

Beispiel: Erstellen Sie folgendes Programm:

Listing 2.1: Ein Programm zur Ausgabe des Alphabets

```
1 #include <iostream>
 \mathbf{2}
  using namespace std;
 3
 4
  int main() {
 5
     char alphabet[26] = ""; // Ein leeres Char-Array
 6
 7
     // Alphabet in das Array schreiben
     for( int i = 0; i < 26; i++ ) {</pre>
 8
 9
       alphabet[i] = 65 + i; // '65' ist der ASCII-Code für 'A'
10
     }
11
12
     // Alphabet ausgeben
13
     for( int i = 0; i < 26; i++ ) {</pre>
       cout << alphabet[i] << " ";</pre>
14
15
     }
16
17
     cout << endl;</pre>
18
19
     return 0;
20| }
```

Die Ausgabe des Programms ist A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. Setzen Sie nun einen Breakpoint in Zeile 10. Der Debugger hält also im ersten Durchlauf der ersten For-Schleife an, es gilt i = 0 und unser Char-Array alphabet ist noch leer, also mit Leerstellen (\0) gefüllt. Wenn Sie nun den Debugger zeilenweise weiter laufen lassen (F7), können Sie beobachten, wie sich die Zählvariable immer weiter erhöht und das Char-Array langsam mit dem Alphabet gefüllt wird.

Die Position des Debuggers im Quellcode wird Ihnen hierbei mit einem gelben Dreieck direkt neben der Zeilennummer angezeigt. (s. Abb. 2.19).



2.6 Anpassung des Editors

Wie bei den meisten IDEs lässt sich auch in Code::Blocks der Editor an die persönlichen Vorlieben eines Programmierers anpassen. Beispielsweise können Sie festlegen, wie viele Zeichen ein Tabulator einnehmen soll oder Sie können Abkürzungen für bestimmte Codeabschnitte festlegen.

Im Folgenden soll nur erklärt werden, wie man die Tabulatorbreite, also die Breite der automatischen Einrückung, ändert.

Wählen Sie aus der Menüleiste Settings $\rightarrow Editor...$ Im erscheinenden Fenster können Sie nun die Tabulatorbreite einstellen (Abb. 2.20).

Configure editor									
	General settings								
	Font This is sample text	Choose							
	Default encoding when opening files:	/INDOW5-1252							
General settings	TAB options	End-of-line options							
	TAB size in spaces: 2	Strip trailing blanks End files with blank line Ensure consistent EOLs End-of-line mode: CR LF							
Folding	Indent options	Other options							
Marrins and caret	Auto indent Smart indent Backspace unindents Show indentation guides Show spaces: No	 Word wrap Use POSIX style for RegEx searches Use Advanced RegEx searches Show line numbers Highlight line under caret Home key always moves caret to fir 							
	Editor title is the file's oname only (no path information) relative filename (to the project file)								
OK Cancel									

Abbildung 2.20: Die Editoreinstellungen in Code::Blocks

Sie können Code::Blocks noch viel weiter anpassen. Alle Möglichkeiten hier zu erläutern, würde allerdings den Rahmen dieses Dokuments sprengen.



Kapitel 3 Installation von Visual Studio und Erstellen eines C++ Projekts

3.1 Visual Studio installieren

Inzwischen haben Sie MinGW, einen reinen Compiler und Code::Blocks, eine integrierte Entwicklungsumgebung (IDE), kennengelernt.

Nun werden Sie noch die IDE Visual Studio kennenlernen. Visual Studio bietet im Prinzip die selben Vorteile wie Code::Blocks. Das Visual Studio-Paket umfasst aber noch weitere Programmiersprachen, wie Java, sowie auch die Weiterentwicklung von C++, C#. Im Rechenzentrum werden Sie Visual Studio vorinstalliert finden, sodass Sie es für das Tutorium und das Projektpraktikum nutzen können.

3.1.1 Visual Studio herunterladen

Als Student des KIT können Sie sich Visual Studio umsonst herunterladen. Zuvor müssen Sie sich beim MSDNAA Karlsruhe unter https://www.stud.uni-karlsruhe.de/~usath/elms/register.php anmelden. Die Anmeldung kann einige Tage in Anspruch nehmen. Sollten Sie schon angemeldet sein, dann können Sie Visual Studio von der Seite http://msdn40.e-academy.com/micror_info herunterladen.

Hinweis: Die MSDNAA-Lizenz berechtigt Sie dazu, Visual Studio kostenlos zu nutzen, jedoch dürfen Sie diese Lizenz nicht zu kommerziellen Zwecken nutzen. Außerdem dürfen Sie das Programm nicht weitergeben. Jeder Student kann sich das Programm selbst herunterladen.

3.1.2 Installation

Hier ist die Installation beispielhaft für Visual Studio 2005 durchgeführt, verläuft für andere Versionen aber analog.

Die heruntergeladene Version von Visual Studio sind .iso-Dateien. Dies sind CD-Images und können entweder über ein entsprechendes Programm eingebunden oder über ein Brennprogramm auf eine CD gebrannt werden. Wichtig! Nicht die .iso als Datei brennen sondern den Inhalt. Die meisten Brennprogramme können iso-Imagedateien öffnen.

Folgen Sie zur Installation den Anweisungen in den Abbildungen 3.1 bis 3.6:



Klicken Sie auf Visual Studio 2005 installieren. Abbildung 3.1: Visual Studio installieren







Microsoft					
VISUAI STUCIO 2005 Zu installierende Features auswählen: Standard Initallet die emplohienen Features des Produkts.	Setup Featurebeschi Bei dieser Features o	reibung: Option werder les Produkts in	n die am häu stalliert.	figsten verwer	ndeten
Vollständig Installiert alle Features des Produkts.					-
Benutzerdefiniert Frod/Hiebere auwoffen, die ein- baw, ausgeschlossen werden sollen.	Produktinstalationspfad: C:\Program Files\Microsoft Visual Studio 8\ Durchsuchen				
	Laufwerk	Datenträger	Verfügbar	Erforderlich	Verbleibend
	C:	16.0 GB	9.8 GB	2.8 GB	7.0 GB
	,	< Zurück	Instalie	ren	Abbrechen

🖟 Microsoft Visual Studio 2005-Setup - Optionsseite					
Visual Studio 2005	Setup				
Zu installierende Features auswählen: Alfreiden Studie 2005 Professional E Studie 20	Featurebeschr Visual C+ Visual C+ hervorrage eine breite • Pro Feature-Instal C\Pregram File Erforderlicher I Laufwerk C:	ebung: + 2005 bietet E ande Produktiv Paletto von T duktivität Ve abionspfad: estylaktenspeich Datenträger 16.0 GB	Entwicklern a rität, Sicherh echnologien. arwalten Sie al Studio 8WCU err: Verfügbar 9.8 GB	uf der genzen eit und Leistur große Projekti Efforderlich 1.9 GB	Welt g sowie mappen v v verbleibend 7.9 GB
۲ ())) () () () () () () (
Standard wiederherstellen		< Zurück	Installie	ren	Abbrechen

Klicken Sie auf Weiter. Abbildung 3.2: Setup Assistent

Aktivieren Sie die Checkbox. Tragen Sie Ihren Namen und gegebenenfalls den Product-Key ein. Lesen Sie die Lizenzbedingungen durch und wenn Sie diese akzeptieren, klicken Sie auf Weiter.

Abbildung 3.3: Akzeptieren des Lizenzvertrags

Verändern Sie den Installationspfad nur, wenn auf $C:\backslash$ nicht genügend Speicherplatz verfügbar ist.

Klicken Sie auf Installieren um das gesamte Visual Studio-Paket zu installieren. Überspringen Sie dann Abb. 3.5.

Wählen Sie Benutzerdefiniert, um nur die gewünschten Pakete zu installieren und klicken Sie auf Weiter.

Abbildung 3.4: Wahl der Installationsart

Deaktivieren Sie alle Auswahlfelder, bis auf das bei Visual C++, um nur C++ zu installieren. Klicken Sie auf Installieren.

Abbildung 3.5: Benutzerdefinierte Installation



Microsoft Visual Studio 2005-Setup
Legen Sie den Datenträger ein: Microsoft Visual Studio 2005 Professional Edition - DEU CD 2
OK Abbrechen

Microsoft Visual Studio 2005-Setup - Letzte Seite	
Wisual Studio 2005 Se	etup
· Fotola	
Visual Studio-Setup wurde abges	chlossen.
Visual Studio-Setup wurde abgeschlossen. Lesen Sie die Sicherheitsbinweise	③ Sicherheitshinweise:
	Es wird dringend empfohlen, auf diesem Computer die neuesten Sicherheitspatches für das Betriebssystem zu installieren. Sie finden die neuesten Updates auf der Windows Update- Website unter
	http://windowsupdate_microsoft.com. Außerdem finden Sie Updates für Windows 2000 auf der Seite <u>Windows 2000</u> , für Windows XP auf der Seite <u>Windows XP</u> und für Windows Server 2003 auf der Seite <u>Windows Server</u> 2003.
	 Verwenden Sie <u>Microsoft Baseline</u> <u>Security Analyzer (MBSA)</u> (nur auf Englisch verfugbar), um Ihren Computer auf Sicherheitsnisken hin zu überprüfen. MBSA verfügt sowohl über eine grafische als auch über eine
	< Zurück Fertig stellen Abbrechen

Nach endlicher Zeit wird die zweite CD verlangt. Legen Sie diese ein und klicken Sie auf $\mathsf{OK}.$

Abbildung 3.6: CD 2

Klicken Sie auf Fertig stellen, um die Installation abzuschließen.

Abbildung 3.7: Abschluss der Installation

3.2 Ein C++Projekt erstellen

Starten Sie Visual Studio und folgen Sie zur Erstellung eines C++ Projekts den Anweisungen in den Abbildungen 3.8 bis 3.13:



Neues Projekt				- ? - × -
Projekttypen:		Vorlagen:		
Visual C++ -ATL -CLR -Allgemein -M4C -Intelligentes Gerät -Umm2 - Ander Projekttypen		Von Vraud Studio installierte Vorlagen Win32-X-Konsolensameendung Win32-Pro Menie Vorlagen Onlinevorlagen durchsuchen	sjekt	
Ein Projekt zur Erstellu	ing einer Win32-Kon	olenanwendung.		
Name:	Hallo World			
Speicherort: C:\Users\Andreas		locuments\Visual Studio 2005\Projects	- Du	chsuchen
Projektmappenname:	Hallo World	Projektmappenver	rzeichnis erstellen	
			ОК	bbrechen

Um ein neues Projekt anzulegen drücken Sie $\boxed{1}$ und wählen Sie *Neues Projekt...* oder drücken Sie die Tasten $\boxed{Strg} + \boxed{Shift} + \boxed{N}$.

Abbildung 3.8: Neues Projekt

Wählen Sie *Win32-Konsolenanwendung* unter den Projekttyp *Win32*.

Geben Sie dem Projekt einen Namen und klicken OK.

Abbildung 3.9: Projektassistent







Klicken Sie auf Weiter. Abbildung 3.10: Win32-Projektassistent





Wählen Sie Konsolenanwendung und Leeres Projekt aus und klicken Sie auf Fertig stellen.

Abbildung 3.11: Win32-Projektassistent

Um eine neue cpp-Datei oder h-Datei zum Projekt hinzuzufügen, klicken Sie auf $\stackrel{\text{def}}{=}$ und wählen Sie Neues Element hinzufügen.... Oder drücken die Tasten $[Strg] + [Shift \uparrow] + [A]$.

Abbildung 3.12: Neue cpp-Datei oder h-Datei

Neues Element hir	nzufügen - Hallo World		- ? X -
Kategorien: Visual C++ U Code Determine Reformer Denstprogramm Eigenschatenblätter		Vorlagen: Von Vorlagen (한 CoreDen La Headerdeter (/h) (한 Core-Deter (CoreDen La Headerdeter (/h) 에 Mine Vorlagen Mine Vorlagen Coninevorlagen durchsuchen	
Erstellt eine Date	i mit C++-Quellcode.		
Name:	Hallo World		
Speicherort:	c:\Users\Andreas\[ocuments\Visual Studio 2005\Projects\Hallo World\Hallo World	Durchsuchen
		Hinzufügen	Abbrechen

Wählen Sie C++-Datei (.cpp) oder Haderdatei (.h)unter der Kategorie Code. Geben Sie der Datei einen Namen und klicken Sie auf Hinzufügen.

Abbildung 3.13: Datei wählen



3.3 Compilieren und ausführen

Wenn Sie ein neues Programm geschrieben haben, möchten Sie dieses vermutlich vom Compiler übersetzen lassen und ausführen. Im Gegensatz zu Code::Blocks wird das Programm sofort ausgeführt. Dies können Sie auch in Visual Studio auf unterschiedliche Arten machen:

- Klicken Sie auf das Menü Debuggen \rightarrow Starten ohne Debuggen =
- Drücken Sie Strg + F5

Wenn Sie das Programm auch gleich Debuggen (Abschnitt 3.4) möchten gibt es folgende Alternativen:

- Klicken Sie auf das Menü $Debuggen \rightarrow Debuggen \ starten$
- Drücken Sie **F5**
- Klicken Sie auf das *Debug*-Icon ▶ (s. Hinweis Kap. 3.4)

Außerdem können Sie

- Das laufende Programm abbrechen (
- Das laufende Programm abbrechen und sofort erneut starten (







Geben Sie Ihr erstes C++ Programm ein, wie im Kompendium beschrieben. Drücken Sie auf die Taste F5 oder klicken Sie auf das *Debug*-Icon , um Ihr Programm zu kompilieren und anschließend zu starten.

Abbildung 3.14: Programm starten

Nach dem Kompilieren wird im Bereich Ausgabe die Anzahl an Fehlern und Warnungen erscheinen. Sollten keine begangen worden sein wird sich direkt das Kommandofenster öffnen und das Programm wird ausgeführt.

Abbildung 3.15: Compiler-Meldung

Wenn Sie keinen Tippfehler gemacht haben, wird sich das Kommandofenster öffnen und Hello World erscheinen.

Abbildung 3.16: Ausgabe



3.4 Der Debugger

Um logische Fehler in einem Programm zu finden, wünscht man sich oft, das Programm einmal Schritt für Schritt durchzugehen und dabei die Variablen beobachten zu können. Der sogenannte Debugger macht dies möglich. Auch Visual Studio bietet wie Code::Blocks einen solchen Debugger.

Im Folgenden wird darauf verzichtet, für alle Aktionen auch den Menüaufruf zu erläutern. Alle Funktionalitäten des Debugger befinden sich im Menü *Debuggen*.

Bei Visual Studio wird im Gegensatz zu Code::Blocks das Projekt automatisch neu kompiliert, wenn Sie es neu Ausführen. Dies geschieht nur, wenn Sie Änderungen daran vorgenommen haben.

Der Debugger führt das Programm solange ganz normal aus, bis er zu einem sogenannten *Breakpoint* gelangt. Ein solcher *Breakpoint* lässt sich setzen, indem man mit der Maus links neben eine Zeilennummer oder Codezeile im Code klickt. Es erscheint dann an dieser Stelle ein kleiner roter Kreis (s. Abb. 3.17). Ein erneuter Klick auf die Stelle löscht den *Breakpoint* wieder. Das Selbe lässt sich auch mit der Taste [F9] in der Zeile erreichen, in der momentan der Cursor steht.

Starten Sie nun den Debugger, indem Sie F5 drücken oder auf das *Debug*-Icon ▶ in der Werkzeugleiste klicken.

Hinweis Damit mit dem *Debug*-Icon ▶ auch wirklich der Debugger gestartet wird, muss im Auswahlfeld neben dem Icon auch explizit *Debug* ausgewählt werden.

Sie können jetzt mit dem *Debug*-Icon boder der Taste F5 zum nächsten Breakpoint fortfahren.

Mit dem 🖘 Icon oder mit der Taste [F11] können Sie Anweisung für Anweisung fortfahren. Wenn eine Anweisung eine Funktion aufruft, wird direkt dort hin gesprungen.

Um Zeile für Zeile auszuführen können Sie stattdessen den 🖵 Icon oder die Taste F10 drücken.

Mit dem 🖆 Icon oder den Tasten [Shift] + [F11] können Sie das Programm solange ausführen lassen, bis es aus einer aufgerufenen Funktion zurück springt.

Normalerweise werden im Debugger-Modus die Variablen automatisch angezeigt. Sollte dies nicht der Fall sein, so finden Sie das Fenster unter *Debuggen* \rightarrow *Fenster*. Variablen, die erst kürzlich geändert wurden, werden rot hervorgehoben. Sie haben die Folgenden Möglichkeiten, sich Variablen anzeigen zu lassen.

- Auto Visual Studio zeigt die aktuell notwendigen und zuletzt geänderten Variablen an.
- Lokal Es werden alle Variablen der aktuellen Funktion angezeigt.
- Überwachen 1-4 Hier können Sie sich selbst unterschiedliche Variablen zu Überwachung eintragen.

Beispiel: Erstellen Sie folgendes Programm:

Listing 3.1: Ein Programm zur Ausgabe des Alphabets

```
#include <iostream>
 1
  using namespace std;
 2
 3
 4
   int main() {
     char alphabet[26] = ""; // Ein leeres Char-Array
 5
 6
 7
     // Alphabet in das Array schreiben
 8
     for( int i = 0; i < 26; i++ ) {</pre>
 9
       alphabet[i] = 65 + i; // '65' ist der ASCII-Code für 'A'
10
     }
11
12
     // Alphabet ausgeben
13
     for( int i = 0; i < 26; i++ ) {</pre>
       cout << alphabet[i] << " ";</pre>
14
     }
15
16
17
     cout << endl;</pre>
18
19
     return 0;
20
  }
```



🧭 debuggerExample (Debugger	n) - Microso	oft Visual Studio					3
Datei Bearbeiten Ansicht	Projekt	Erstellen Debuggen Extras Fenster O	Community Hilfe				
i 🗊 - 🛅 - 💕 🖬 🎒 👗		i) - (i - J - E - F		~ 🖄	- 🗟	🚰 🐋	•• ∓
🗼 II 🖬 🖬 🔶 🖼 🗊	≧ <u>≣</u> Hex	🗔 🗸 👷 🗟 🍇 📐 🐼 🛱 🛱 🗐	을 🔲 위 대 위 다	æ 🔈 🤿 🖕			_
Projektmappen-Expl 👻 🖡 🗙	main.c	pp	- ×	Lokal		🚽 🕂 🗧	×
	(Globaler	r Gültigkeitsbereich) 🚽 =🏼 🗣 main()	•	Name	Wert	Тур	*
🔁 Projektmappe "debuggerE:	- 4	41		🧼 i	9	int	
📄 📅 debuqqerExample	5	5 int main() {	*	🗄 🥥 alphabet	0x0012ff0c "ABCDEFGHIJ" 🔍 🗣	char [26]	
🔤 Headerdateien	6	6 char alphabet[26] = "";	// Ein leeres C				
🚊 🗁 Quelldateien	5	7					
🔤 🚰 main.cpp	ε	B // Alphabet in das Array	y schreiben				
📖 🚞 Ressourcendateien	9	9 for(int i = 0; i < 26;	i++) {				
	10	0 alphabet[i] = 65 + :	i; // '65' ist d				
		1 }					
	12	// Alnhabet ausgeben					
	14	for i int i = 0; i < 26;	i++) {				
	15	5 cout << alphabet[i]	<< " "				
	16	6 }					
	17	7					
	18	B cout << endl;					
	19	9					
	20	return U;	Ψ				<u> </u>
Projektm	•		4	🔣 Auto 🔯 Loka	🛛 🖾 Threads 🐼 Module 👰	Uberwac	
🖓 Aufrufliste 📃 Ausgabe 🏹	Haltepunkt	te					
Bereit			Z 11	S1	Zei 1	EINFG	н

Abbildung 3.17: Der Debugger in Aktion

Die Ausgabe des Programms ist A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. Setzen Sie nun einen Breakpoint in Zeile 10. Der Debugger hält also im ersten Durchlauf der ersten For-Schleife an, es gilt i = 0 und unser Char-Array alphabet ist noch leer, also mit Leerstellen (\0) gefüllt. Wenn Sie nun den Debugger zeilenweise weiter laufen lassen (F10), können Sie beobachten, wie sich die Zählvariable immer weiter erhöht und das Char-Array langsam mit dem Alphabet gefüllt wird.

Die Position des Debuggers im Quellcode wird Ihnen hierbei mit einem gelben Pfeil direkt neben der Zeilennummer angezeigt (s. Abb. 3.17).

3.5 Anpassung des Editors

Wie bei den meisten IDEs lässt sich auch in Visual Studio der Editor an die persönlichen Vorlieben eines Programmierers anpassen. Beispielsweise können Sie festlegen, wie viele Zeichen ein Tabulator einnehmen soll.

Im Folgenden soll nur erklärt werden, wie man die Tabulatorbreite, also die Breite der automatischen Einrückung, ändert.

Wählen Sie aus der Menüleiste $Extras \rightarrow Optionen...$ Im erscheinenden Fenster können Sie nun die Tabulatorbreite einstellen (Abb. 3.18.). Wie Sie die Zeilennummern aktivieren sehen Sie in Abb. 3.19. Sie können auch bei Visual Studio noch viel weiter anpassen. Alle Möglichkeiten hier zu erläutern, würde allerdings den Rahmen dieses Dokuments sprengen.



Optionen				? 💌
Projekte und Projektmappen Umgebung Quellcodeverwaltung Text-Editor Allgemein Allgemein Tabstopps Basic C/C++ Allgemein Tabstopps Formatierung CSS HTML Nur Text PL/SQL SOL Script	nzug Keine Block Intelligent abstopp abulatorgröße: nzugsgröße:) Leerzeichen einfügen) Tabstopps beibehalten	2 2	OK 4	bbrechen

Abbildung 3.18: Die Editoreinstellungen in Visual Studio

Optionen	2 🔀
Umgebung Quellcodevenwaltung Allgemein Dateinweiterung C/C++ Allgemein Tabstopps Formatierung C/C+ Fo	Anweisungsvervollständigung
	OK Abbrechen

Abbildung 3.19: Zeilennummern aktivieren