

Starthilfe für Java, C# und C++

Inhaltsverzeichnis

Einrichten der Arbeitsumgebungen	3
Datenspeicher für Projekte	3
IDEs auf Netzlaufwerken	3
Speicher für die Konfiguration der IDEs	4
Java	4
Allgemeines	4
Bezugsquellen.....	4
Joe.....	4
Java	4
Installation.....	5
Vorschlag für eine Ordnerstruktur	5
Globale Konfiguration	6
Arbeit mit Klassen.....	7
Beispiel Code: „Hello World“	7
Arbeit mit der Entwicklungsumgebung.....	8
Kompilieren des Quellcodes	8
Ausführen der Anwendung	8
Klassenbrowser	9
Mehrere Klassen verwenden.....	10
C#.....	11
Allgemeines	11
Bezugsquellen.....	11
Freitag, 9. Dezember 2011	1

Informatik 3 Praktikum

SharpDevelop	11
.NET Runtime	11
.NET SDK	11
Installation	11
Reihenfolge.....	11
Vorschlag für eine Ordnerstruktur	12
Arbeit mit Projekten	12
Anlegen eines neuen Projekts	12
Beispiel Code: „Hello World“	14
Zusicherungen unter C#	14
Kompilieren des Projektes.....	15
Arbeit mit der Entwicklungsumgebung.....	15
Ausführen der Anwendung	15
Klassenbrowser	15
C++	16
Allgemeines	16
Bezugsquellen.....	16
Installation	16
Vorschlag für eine Ordnerstruktur	20
Arbeit mit Projekten	21
Anlegen eines neuen Projekts	21
Projektoptionen.....	22
Beispiel Code: „Hello World“	24
Arbeit mit der Entwicklungsumgebung.....	24
Klassenbrowser	24
Übersetzen	25

Einrichten der Arbeitsumgebungen

Für Ihre Arbeit stellt das RZ Ihnen insgesamt drei unterschiedliche Netzlaufwerke zur Verfügung.

- **Laufwerk U:** / wird bei jedem Start von Windows automatisch eingebunden und dient als Verwaltungsserver von Programmspezifischen Konfigurationsdateien.
- **Laufwerk N:** / muss von Ihnen als Netzlaufwerk eingebunden werden und dient der Ablage von Projekten und Ihren persönlichen Daten.
 - Der Server ist unter der folgenden Adresse zu finden:
`\\nasdat\<benutzername>`
- **Laufwerk J:** / muss ebenfalls von Hand eingebunden werden und enthält Dev-Cpp und Java weiteres dazu siehe [IDEs auf Netzlaufwerken](#).
 - Der Server ist unter der folgenden Adresse zu finden:
`\\adserv\programme\`

Datenspeicher für Projekte

Um Ihre Projekte abzulegen sollten Sie keinesfalls den lokalen Speicher des Rechners benutzen da er publik und deshalb flüchtig und unsicher ist. Folgende Möglichkeiten stehen zur Auswahl:

- 1.) privater mobiler Speicher (USB-Stick etc.)
- 2.) privater Netzwerkspeicher

Das Rechenzentrum bietet jedem Benutzer einen Netzwerkspeicher zur freien Verwendung an. Einige Entwicklungsumgebungen benötigen eine fixe Stelle an der sie Projekte und temporäre Dateien abspeichern können. Unabhängig von den Voraussetzungen der Entwicklungsumgebungen ist es ratsam ein Schema für die Ablage der persönlich erzeugten Praktikumsübungen vorzubereiten. Hier ein Vorschlag mit zugeordnetem Laufwerk N:\:

```
N:\inf3praktikum\          cpp\<Projektname>\
                           java\
                           csharp\<Projektname>\
```

IDEs auf Netzlaufwerken

Einige der Programme sind auf einem Netzlaufwerk installiert. Die müssen mit dem Laufwerksbuchstaben J:\ abgebildet werden, da die Arbeitsmaschinen genau auf diese konfiguriert sind. Die IDEs Joe (für Java) und Dev C++ sind auf diesen Pfaden installiert:

```
J:\inf3praktikum\cpp\dev-cpp
J:\inf3praktikum\java\joe
```

Speicher für die Konfiguration der IDEs

Das RZ hat für jeden Windows Benutzer einen Bereich eingerichtet, der in Grenzen benutzbar für Konfigurationen der Anwendungen ist. Dieser Bereich soll mit **U:** zugeordnet werden. Dev-C++ ist so konfiguriert dass im Labor 3-201 dieses Laufwerk benutzt wird, um globale Konfigurationen des Benutzerapplikationen abzulegen. Dieses Laufwerk wird beim Login automatisch verbunden:

```
U:\inf3praktikum\dev-cpp
```

Dieser Arbeitsbereich muss existieren, sonst startet die IDE nicht!

Arbeiten mit Konsolenanwendungen, siehe StarthilfeVisualEiffel.pdf.

Java

Allgemeines

Für die Bearbeitung von Aufgaben mit Java wird Joe empfohlen. Joe ist ein einfacher Editor für Java. Das Installationspaket enthält nur den **Editor ohne** den **Übersetzer** oder **Bibliotheken**. Dafür ist das Installationspaket nur ~1,3 MByte groß. Joe wird leider seit 2003 nicht weiterentwickelt.

Da Joe ein reiner Editor ist, ist zum Programmieren noch das **Java Development Kit (JDK)** erforderlich, das neben dem Compiler auch die benötigten Bibliotheken enthält. Da Java Programme in einer Virtuellen Umgebung ausgeführt werden, muss zusätzlich noch das **Java Runtime Environment (JRE)** installiert sein.

Neben den Installationsdateien für JDK und JRE empfiehlt es sich auch, die Java-Dokumentation herunter zu laden. Somit hat man auch offline einen Zugriff auf die Java-API.

Bezugsquellen

Joe

Den Editor Joe kann man von dieser URL **direkt** runterladen:

http://www.javaeditor.de/download/files/joe_250beta1.zip

Die Größe des Installationspaketes ist etwa 1,3 MByte.

Java

Das JDK und die Dokumentation findet man **indirekt** auf dieser URL:

<http://java.sun.com/javase/downloads/index.jsp>

Aktuell sind diese Installationspakete auf dieser Seite:

“JDK 6 Update 13”	<code>jdk-6u13-windows-i586-p.exe</code>	~73 MByte
“Java SE 6 Documentation”	<code>jdk-6-doc.zip</code>	~53 MByte

Installation

Es gibt keine Reihenfolge der Installation. Alle Dateien sind unabhängig voneinander und können nacheinander installiert werden. Joe versucht nach der Installation den Ort des JDKs zu finden. Meist schlägt das fehl. Deshalb sollte in der Dialogbox (Abbildung 1) die Frage danach ablehnt werden. Wenn alle Pakete installiert sind, wird Joe manuell gestartet und die Verbindung zum JDK manuell konfiguriert.

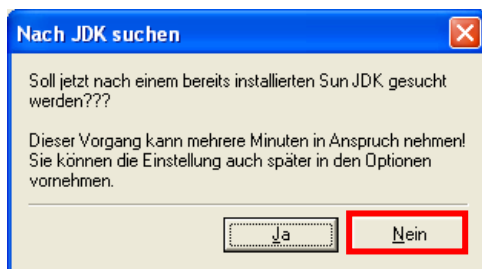


Abbildung 1: Dialogbox mit Frage nach der JDK-Pfadsuche

Vorschlag für eine Ordnerstruktur

Der folgende Vorschlag für eine Ordnerstruktur ist mit Laufwerksangaben der gestellten Rechner versehen. Diese können jedoch auch auf das eigene System (Notebook) mit kleineren Anpassungen übertragen werden.

1. Wie im Abschnitt „Datenspeicher für Projekte“ bereits erwähnt sollten Sie einen Ordner nur für Java-Programmieraufgaben erstellen.

N: /inf3praktikum/java

2. Legen Sie in diesem Ordner einen Ordner namens `sources`. Dieser enthält später die Quellcode-Dateien (Dateien mit der Endung `.java`) die vom Java-Kompilierer übersetzt werden.

N: /inf3praktikum/java/sources

Jede Quellcode-Datei enthält in der Regel eine Java-Klasse, möglich wären jedoch auch mehrere Klassen. Dabei gilt

Dateiname = Name der ersten Klasse

- Legen Sie in Ihrem Java-Ordner nun einen weiteren Ordner für Bytecode-Dateien (Dateien mit der Endung `.class`) an mit dem Namen `bytecode`.

N: /inf3praktikum/java/bytecode

Diese Dateien benötigt später der Interpreter um Ihre Anwendung ausführen zu können.

- Sollten Sie die Arbeit mit `packages` bevorzugen so legen sie in jedem der genannten Ordner einen `packages`-Ordner an.

N: /inf3praktikum/java/sources/packages

N: /inf3praktikum/java/bytecode/packages

Globale Konfiguration

Nach Abschluss der Installationen muss Joe konfiguriert werden, damit das JDK benutzt werden kann. Dazu im Joe-Menü

Optionen /Einstellungen

auswählen. Siehe Abbildung 3 und Abbildung 2. Die Ziel-VM bezeichnet die JRE bei der das Kompilat benutzt werden soll. Bei JDK6 ist als Version „1.6“ einzugeben (Das Menü bietet keine Auswahl. Die Version 1.6 muss von Hand eingegeben werden).

Zusätzlich kann ein Zielpfad für die erstellten Kompilate eingegeben werden. Dieses Vorgehen ist zu

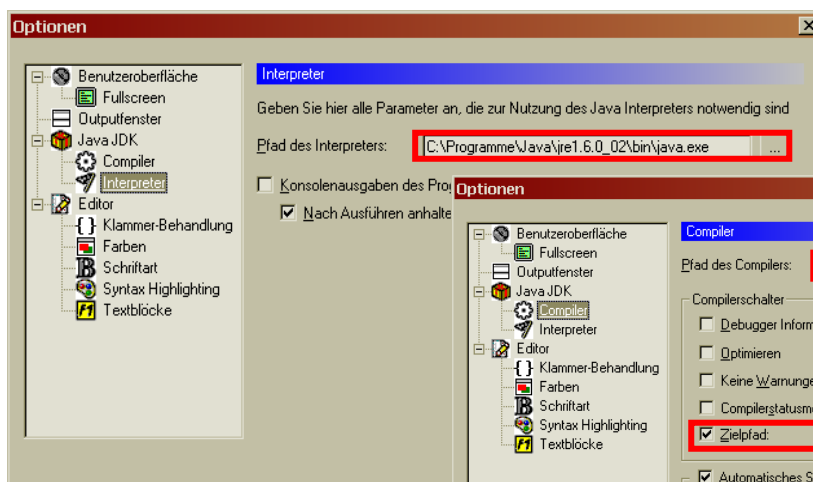
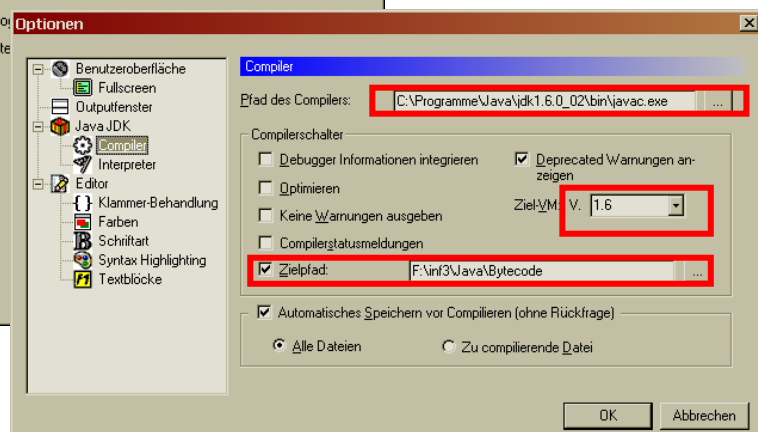


Abbildung 2: Pfad des Übersetzers (Compilers)

Abbildung 3: Pfad der virtuellen Maschine (Interpreter) eingeben



empfehlen damit Quellcode und Bytecode-Dateien nicht gemischt in einem Verzeichnis liegen.

`Javac.exe` ist der Kompilierer und `java.exe` ist der Interpreter. Sie sind beide im `bin`-Verzeichnis der JDK-Installation zu finden.

Es ist sinnvoll, die JDK-Dokumentation ins JDK-Installationsverzeichnis zu entpacken. Nun kann mit einem Internetbrowser die Schnittstellenbeschreibung (API) durchsucht werden. Der Start für die Benützung der API ist z.B.:

```
C:\Programme\Java\jdk1.6.0_02\docs\api\index.html
```

Arbeit mit Klassen

Da in Java jede Quelle mindestens eine eigenständige Klasse enthält die auch einzeln ausführbar ist, existiert in JOE keine Projektstruktur. Für jede benötigte Klasse in der Regele eine neue Datei angelegt. Dies kann weder per Klick auf „NEU“, über die Tastenkombination „STRG + N“ oder über „Datei -> Neu“ geschehen. Daraufhin öffnet sich ein leeres Editorfenster, in das der Quellcode eingegeben wird. Nach jedem Erzeugen einer neuen Klasse sollte diese gleich im Quellcode-Ordner abgespeichert werden.

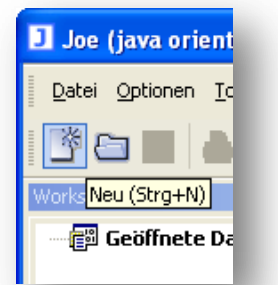


Abbildung 4: JOE neue Klasse anlegen

Beispiel Code: „Hello World“

```
class HelloWorld {  
public static void main(String[] args) {  
  
    // Definiere eine Zeichenkette.  
    String message="Hello World!";  
  
    // Gebe die Zeichenkette an der Konsole aus.  
    System.out.println(message);  
}  
}
```

Diese Klasse wird unter `HelloWorld.java` im Ordner `N:/inf3praktikum/java/sources` abgespeichert. Nach dem Kompilieren erscheint eine Datei mit dem Namen `HelloWorld.class` im Ordner `N:/inf3praktikum/java/bytecode`.

Arbeit mit der Entwicklungsumgebung

Kompilieren des Quellcodes

Zum Kompilieren des Quellcodes genügt ein Klick auf das Icon zum Kompilieren.

Ausführen der Anwendung

Navigieren mittels der „CMD“ zu dem Ordner in dem Ihr Bytecode gespeichert wird (siehe Vorschlag für eine Ordnerstruktur

Der folgende Vorschlag für eine Ordnerstruktur ist mit Laufwerksangaben der gestellten Rechner versehen. Diese können jedoch auch auf das eigene System (Notebook) mit kleineren Anpassungen übertragen werden.

5. Wie im Abschnitt „Datenspeicher für Projekte“ bereits erwähnt sollten Sie einen Ordner nur für Java-Programmieraufgaben erstellen.

N: /inf3praktikum/java

6. Legen Sie in diesem Ordner einen Ordner namens `sources`. Dieser enthält später die Quellcode-Dateien (Dateien mit der Endung `.java`) die vom Java-Kompilierer übersetzt werden.

N: /inf3praktikum/java/sources

Jede Quellcode-Datei enthält in der Regel eine Java-Klasse, möglich wären jedoch auch mehrere Klassen. Dabei gilt

Dateiname = Name der ersten Klasse

7. Legen Sie in Ihrem Java-Ordner nun einen weiteren Ordner für Bytecode-Dateien (Dateien mit der Endung `.class`) an mit dem Namen `bytecode`.

N: /inf3praktikum/java/bytecode

Diese Dateien benötigt später der Interpreter um Ihre Anwendung ausführen zu können.

8. Sollten Sie die Arbeit mit `packages` bevorzugen so legen sie in jedem der genannten Ordner einen `packages`-Ordner an.

N: /inf3praktikum/java/sources/packages

N: /inf3praktikum/java/bytecode/packages

Globale Konfiguration >>Zielpfad des Compilers). Der Java-Interpreter wird mit `java` aufgerufen. Er erwartet als Argument einen Klassennamen und sucht den Bytecode dazu im aktuellen Verzeichnis. Der Aufruf des Hello World Programms würde dann folgendermaßen aussehen.

```
java HelloWorld
```

Der Klassenname muss ohne Endung eingegeben werden, sonst wird der Befehl mit der folgenden Fehlermeldung abgebrochen:

```
D:\paharukov\Eigene Dateien\Javo Projects\Bytecode>java HelloWorld.class
Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorld/class
Caused by: java.lang.ClassNotFoundException: HelloWorld.class
    at java.net.URLClassLoader$1.run(Unknown Source)
    at java.security.AccessController.doPrivileged(Native Method)
    at java.net.URLClassLoader.findClass(Unknown Source)
    at java.lang.ClassLoader.loadClass(Unknown Source)
    at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
    at java.lang.ClassLoader.loadClass(Unknown Source)
    at java.lang.ClassLoader.loadClassInternal(Unknown Source)
Could not find the main class: HelloWorld.class. Program will exit.
D:\paharukov\Eigene Dateien\Javo Projects\Bytecode>_
```

Abbildung 5: Fehlermeldung bei der Eingabe des Klassennamens mit der Endung `.class`

Um die Prüfung von Zusicherungen zu veranlassen, muss der Interpreter mit dem Parameter `-ea` (Enable Assertions) aufgerufen werden. Für das obere Beispiel sieht es dann folgendermaßen aus:

```
java -ea HelloWorld
```

Klassenbrowser

JOE besitzt leider keinen Klassenbrowser. Eine Online-Version der Dokumentation der Java-Bibliotheken finden Sie jedoch unter der folgenden URL:

<http://java.sun.com/javase/6/docs/api/>

Desweiteren finden Sie unter:

https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jdk-6u10-docs-oth-JPR@CDS-CDS_Developer

eine Offline-Version der Dokumentation. Diese können Sie dann in Ihre IDE einbinden.

Mehrere Klassen verwenden

Um eine weitere Klasse zu verwenden, kann von ihr ein Objekt erzeugt werden. Dann können die öffentlichen Methoden und Eigenschaften der Klasse über die Punktnotation aufgerufen werden. Hierzu ein Beispiel:

```
class HelloWorld {  
  
    public static void main(String[] args) {  
  
        //Erzeugen eines neuen Objektes vom Typ TestClass  
        TestClass myTestClass = new TestClass();  
  
        String message="Hello World!";  
  
        //Aufruf einer Methode des erzeugten Objektes  
        myTestClass.run();  
  
        System.out.println(message);  
    }  
}  
class TestClass {  
  
    public void run(){  
        System.out.println("Hello from Testclass");  
    }  
}
```

C#

Allgemeines

Zur Programmierung in C# wird Sharp Develop empfohlen. SharpDevelop ist eine freie Entwicklungsumgebung und die IDE ist sogar quelloffen verfügbar.

Zur Benutzung der Entwicklungsumgebung wird die Installation der Laufzeitumgebung .NET 2.0 vorausgesetzt. Empfohlen wird außerdem das Software Development Kit von Microsoft, welches noch zusätzliche Werkzeuge, Dokumentationen und Beispiele enthält.

Bezugsquellen

SharpDevelop

Die Entwicklungsumgebung ist momentan in der Version 3.0 vorhanden und kann von der folgenden Seite bezogen werden:

<http://www.icsharpcode.net/OpenSource/SD/Download/#SharpDevelop30>

Die Installationsdatei umfasst etwa 18 MB. Für die Bearbeitung der Aufgaben kann auch die Version 2.0 genutzt werden. Diese umfasst lediglich 8 MB

.NET Runtime

Die .NET Laufzeitumgebung kann über den untenstehenden Link direkt von Microsoft bezogen werden und umfasst etwa 2.8 MB:

<http://www.microsoft.com/downloads/details.aspx?FamilyId=AB99342F-5D1A-413D-8319-81DA479AB0D7&displaylang=de>

.NET SDK

Das Software Development Kit kann von Microsoft über den untenstehenden Link direkt von Microsoft bezogen werden und umfasst etwa 345 MB:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=FE6F2099-B7B4-4F47-A244-C96D69C35DEC&displaylang=de>

Installation

Reihenfolge

Nach der Installation der Runtime (s.o.) kann SharpDevelop direkt installiert werden. Oder man will die Vorteile des SDKs nutzen und installiert in dieser Reihenfolge:

- 1.) Runtime
- 2.) SDK
- 3.) SharpDevelop

Am Schluss der Installation fragt ein Dialog ob eine „Code Completion“ Datenbank angelegt werden soll. Das ist sinnvoll und sollte bewusst durchgeführt werden.

Vorschlag für eine Ordnerstruktur

Der folgende Vorschlag für eine Ordnerstruktur ist mit Laufwerksangaben der gestellten Rechner versehen. Diese können jedoch auch auf das eigene System (Notebook) mit kleineren Anpassungen übertragen werden.

1. Wie im Abschnitt „Datenspeicher für Projekte“ bereits erwähnt sollten Sie einen Ordner nur für C# - Projekte erstellen.

N: /inf3praktikum/csharp/

Diesen Ordner sollten Sie nun immer unter dem Punkt „Verzeichnis“ (siehe Abschnitt „Anlegen eines neuen Projekts“) angeben. Weitere Schritte sind nicht Notwendig das Projektmanagement wird vollständig von der IDE übernommen.

Arbeit mit Projekten

C# arbeitet, wie Visual Eiffel, mit Projektstrukturen. Diese werden hier als Arbeitsmappen bezeichnet. Ebenfalls Eiffel ähnlich ist die Auswahl eines Projekttyps, mit dem Unterschied dass hier viel mehr Anwendungsbereiche abgedeckt werden als bei Visual Eiffel.

Anlegen eines neuen Projekts

Zum Anlegen eines neuen Projekts klickt man in der IDE auf „Neu -> Projektmappe“. Es erscheint ein Dialogfenster in dem Sie aufgerufen werden der Projektmappe einen Namen zu geben und auszuwählen welche Art von Anwendung sie erzeugen möchten.

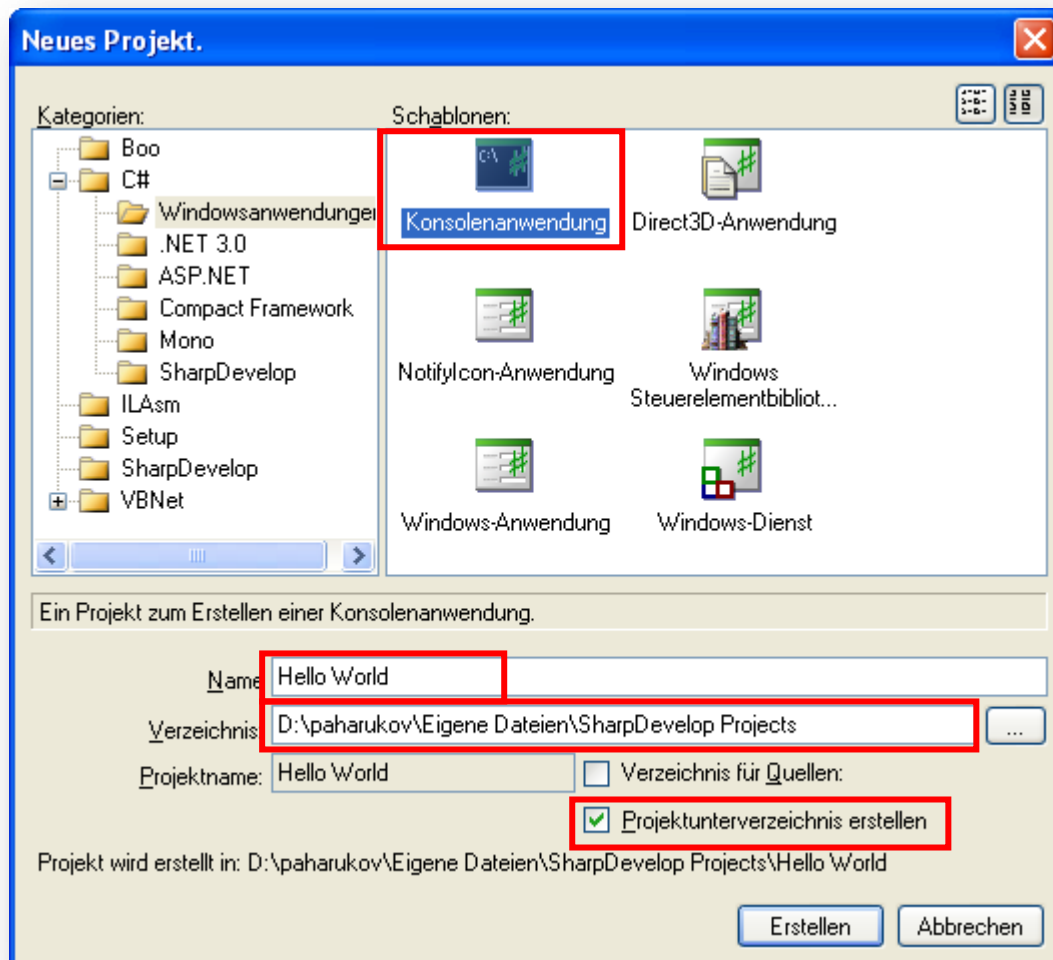


Abbildung 6: C# Anlegen eines neuen Projekts (Hello World)

Alle anderen Einstellungen können im Normalfall so belassen werden wie sie sind.

Sollten Sie nicht am eigenen Notebook arbeiten, passen Sie bitte das Projektverzeichnis so wie im Abschnitt „Vorschlag für eine Ordnerstruktur“ erklärt.

Weitere Einstellungen in den Projektoptionen sind nicht nötig da diese den Ansprüchen des I3-Praktikums vollkommen genügen.

Beispiel Code: „Hello World“

Der folgende Codeabschnitt wird von SharpDevelop automatisch erzeugt und wird hier nicht weiter erläutert.

```
using System;

namespace Hello_World
{
    class Program
    {
        public static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");

            // TODO: Implement Functionality Here

            Console.Write("Press any key to continue . . . ");
            Console.ReadKey(true);
        }
    }
}
```

Zusicherungen unter C#

Um Zusicherungen auch unter C# verwenden zu können muss in der jeweiligen Klasse zunächst eine weitere Systembibliothek eingebunden werden. Diese ist in der IDE bereits enthalten. Das Einbinden würde folgendermaßen aussehen:

```
using System.Diagnostics;
```

Der konkrete Befehl lautet dann

```
Debug.Assert()
```

Dieser kann in folgenden Formen aufgerufen werden:

```
Debug.Assert(Boolean)
Debug.Assert(Boolean, String)
Debug.Assert(Boolean, String, String)
```

Hierzu ein Ausschnitt aus der Schnittstellenbeschreibung:

Name	Description
Debug.Assert (Boolean)	Checks for a condition and outputs the call stack if the condition is false . Supported by the .NET Compact Framework.

<code>Debug.Assert (Boolean, String)</code>	Checks for a condition and displays a message if the condition is false . Supported by the .NET Compact Framework.
<code>Debug.Assert (Boolean, String, String)</code>	Checks for a condition and displays both specified messages if the condition is false . Supported by the .NET Compact Framework.

Kompilieren des Projektes

Zum Übersetzen des Projekts klicken Sie auf „Erstellen ->Erstelle Mappe neu“ oder drücken Sie „ALT + F8“. Es wird empfohlen das Projekt immer komplett neu zu erstellen um Kompilierungsfehler zu vermeiden.

Bei diesem Vorgang werden in Ihrem Projektordner zwei zusätzliche Ordner erstellt. Dabei enthält der Ordner „bin -> Debug“ die .exe- Datei, die zum Ausführen der Anwendung benötigt wird.

Arbeit mit der Entwicklungsumgebung

Ausführen der Anwendung

Navigieren Sie in der „CMD“ zu Ihrem Projektordner. Hier befindet sich nun der Ordner „bin“ der einen Unterordner „Debug“ enthält. Navigieren Sie zum Debug-Ordner und führen Sie hier die <Projektname>.exe aus.

Klassenbrowser

Die IDE verfügt über einen integrierten Klassenbrowser, der hier als „Dynamische Hilfe“ bezeichnet wird. Dieser kann über „Hilfe -> Dynamische Hilfe“ eingeblendet werden. Um Informationen über ein Objekt aufzurufen genügt es den Text-Cursor in den Bezeichner des Gewünschten Typs zu platzieren. Die Informationen werden dann in der Dynamischen Hilfe angezeigt.

C++

Allgemeines

Zur Arbeit mit C++ wird die IDE Dev C++ empfohlen. Dev C++ ist eine freie Entwicklungsumgebung für Windows und wird samt Compiler und den Standard Bibliotheken zur Verfügung gestellt.

Bezugsquellen

Die aktuelle Version von Dev-C++ ist die Version 5.0.0.8 mit dem Übersetzer. Über diese folgende Referenz kommt man direkt auf die Installationsdatei (Größe: ~21 MB):

<http://sourceforge.net/projects/orwelldevcpp/>

Zusätzliche Informationen zur Entwicklungsumgebung und dem Übersetzer findet man unter:

<http://www.bloodshed.net/dev/index.html>

Wir benutzen allerdings den inoffiziellen Release, da man den Quellcode sonst selbst kompilieren müssten: <http://orwellengine.blogspot.com/2011/11/dev-c-5007-released.html>

Installation

Die Installation wird hier für Windows-Betriebssysteme beschrieben. Inzwischen ist diese Beschreibung in 2 Teile getrennt, da sich herausgestellt hat, dass Dev C++ mit Windows 7 nicht mehr „out of the box“ funktioniert, sondern ein paar Änderungen manuell vorgenommen werden müssen.

Windows allgemein

Nach Starten der Installationsdatei entpackt und installiert sich Dev C++ auf bis zu 60 MByte Plattenplatz. Beim ersten Start werden Menüsprache und Aussehen abgefragt. Danach folgt Dialog für die „Code Completion“ (Abbildung 7). Sie dient als Hilfe für die Interfacebenutzung.

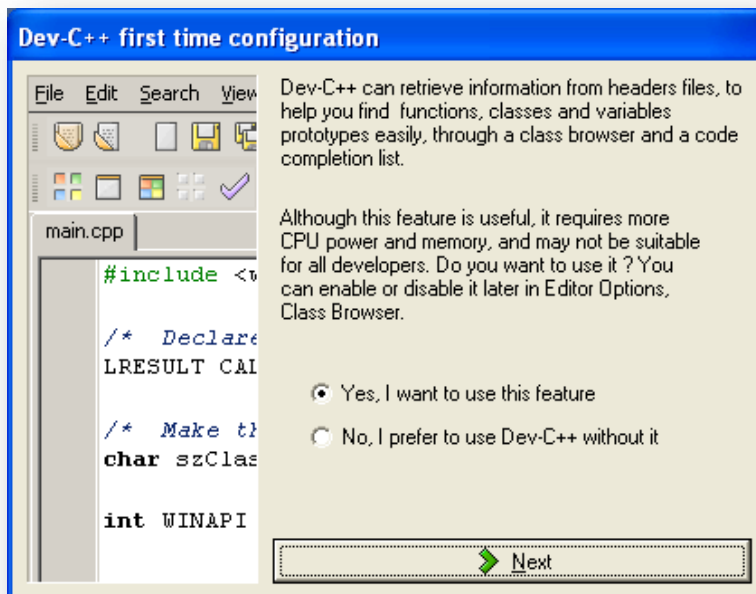


Abbildung 7: Code Completion Dialog

Windows 7

Unter Windows 7 zusätzliche Schritte notwendig, da der mitgelieferte Compiler MinGW in der Form nicht funktioniert.

Ein Workaround dafür ist, MinGW selbst zu installieren. Die aktuelle Version vom 18.11.2011 kann hier heruntergeladen werden:

<http://sourceforge.net/projects/mingw/files/Installer/mingw-get-inst/mingw-get-inst-20111118/>

Diese wird, einfach dem Setup folgend, installiert, z.B. in das Verzeichnis:
„C:\Tools\Entwicklung\MinGW“.

Um diesen Compiler dann in Dev C++ nutzen zu können, muss unter „Tools → Compiler Options“ in der Menü-Leiste, eine neue Compiler-Konfiguration angelegt werden (s. Abbildung 8Abbildung 7).

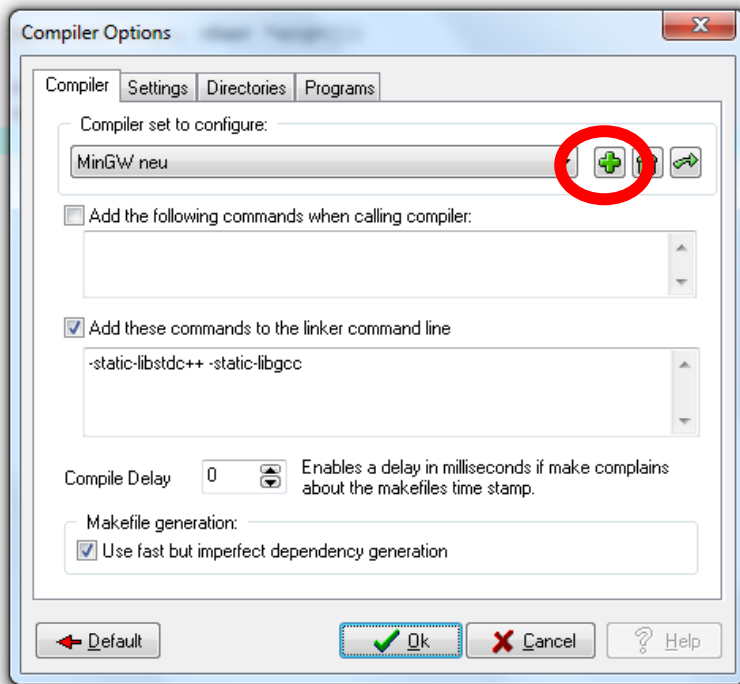


Abbildung 8: Compiler Options

Dann müssen alle Verzeichnisse auf den MinGW-Installationsordner geändert werden:

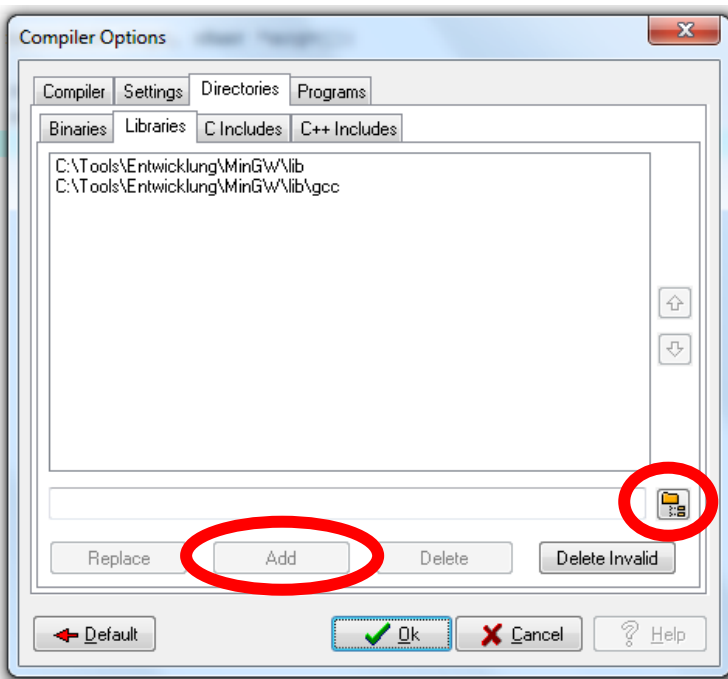


Abbildung 9: Directories

Ebenso müssen die Programme auf die Exe-Dateien aus der neuen MinGW-Installation eingerichtet werden. Also auf das Ordner-Symbol klicken, dann den Ordner der Binaries aus der MinGW-Installation (hier: C:\Tools\Entwicklung\MinGW\bin) auswählen und zum Schluss bei allen die jeweilige Exe-Datei auswählen:

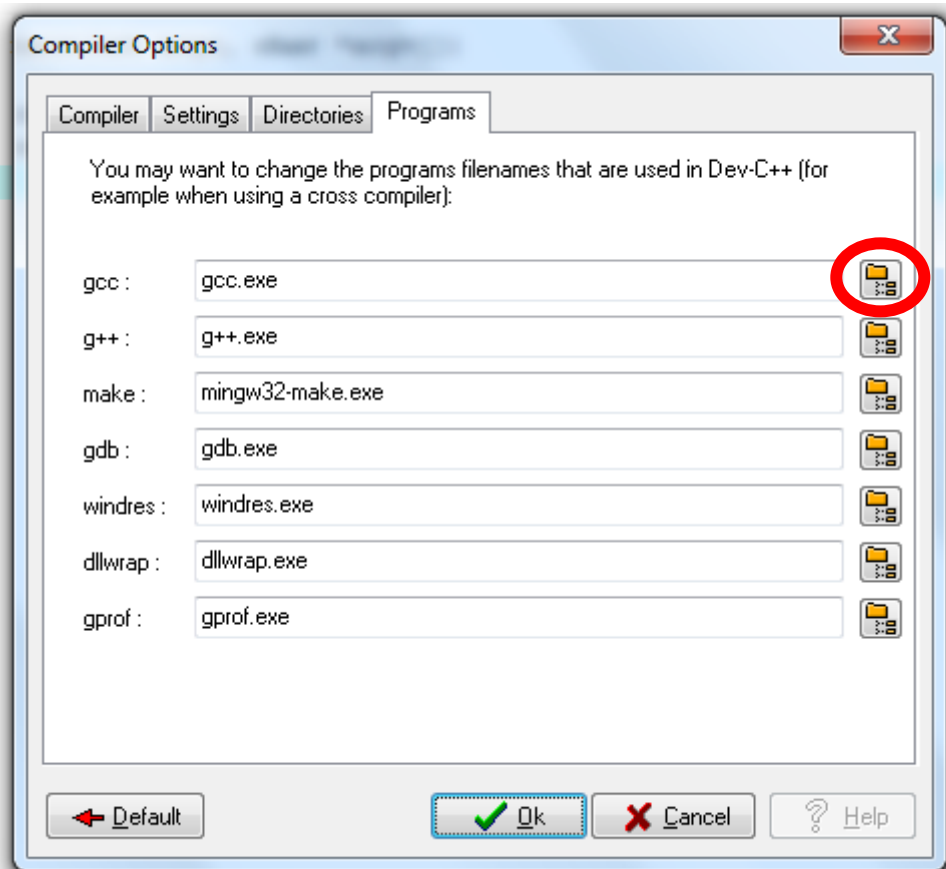


Abbildung 10: MinGW-Binaries

Nachdem die Compiler-Einstellungen komplett sind, muss noch der Compiler dem Projekt zugewiesen werden. Und zwar Rechtsklick auf das Projekt im Project-Explorer und dann „Project Options“ anklicken. Im Reiter „Compiler“ kann nun der neue ausgewählt werden:

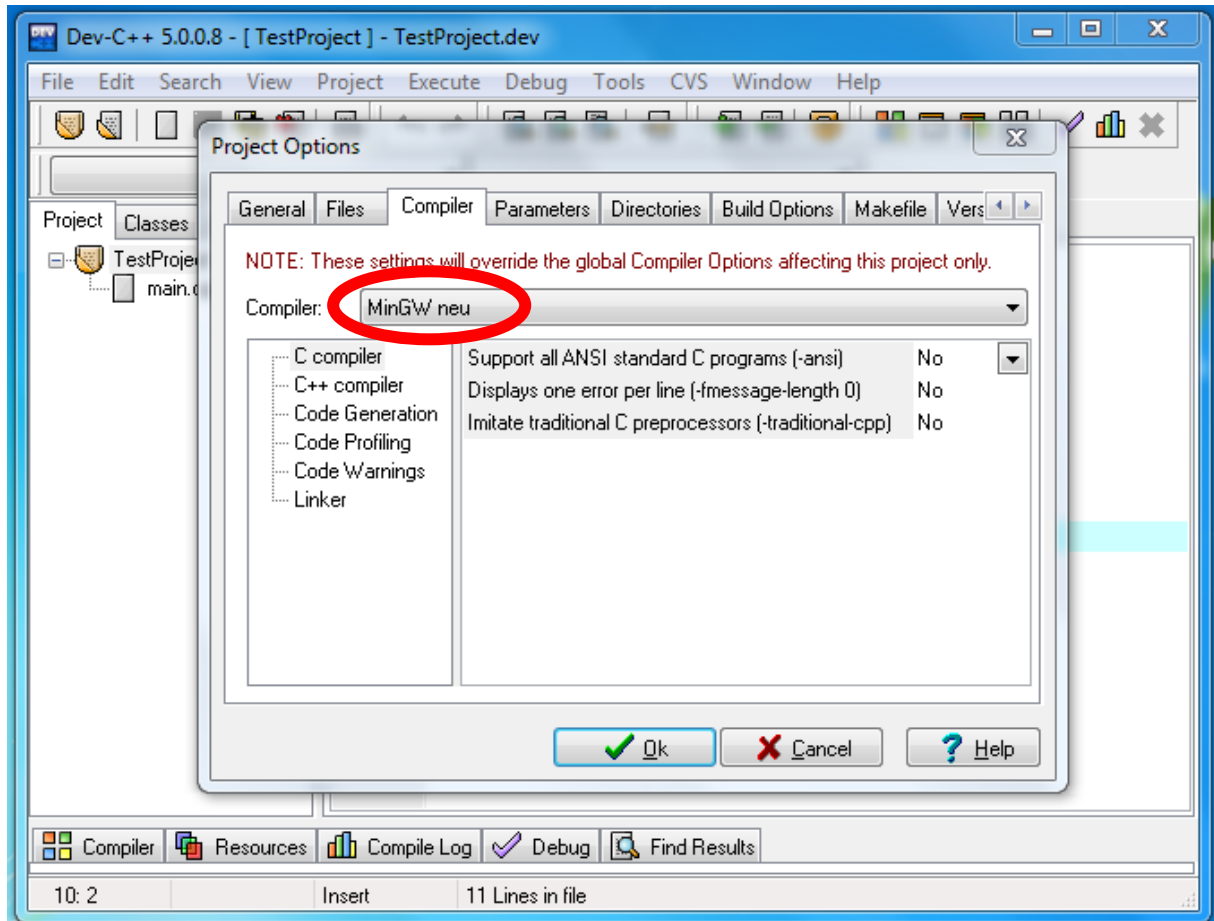


Abbildung 11: Projekt-Einstellungen

Vorschlag für eine Ordnerstruktur

Der folgende Vorschlag für eine Ordnerstruktur ist mit Laufwerksangaben der gestellten Rechner versehen. Diese können jedoch auch auf das eigene System (Notebook) mit kleineren Anpassungen übertragen werden.

1. Wie im Abschnitt „Datenspeicher für Projekte“ bereits erwähnt sollten Sie einen Ordner nur für C++-Programmieraufgaben erstellen.

N: /inf3praktikum/c++

2. Legen Sie in diesem Ordner einen Ordner namens `_cpp`. Dieser enthält später die Quellcode-Dateien (Dateien mit der Endung `.cpp`) die vom C++-Kompilierer übersetzt werden.

N: /inf3praktikum/c++/_cpp

- Legen Sie in Ihrem C++-Ordner nun einen weiteren Ordner für Header-Dateien (Dateien mit der Endung `.h`) an mit dem Namen `_h`.

N: /inf3praktikum/c++/_h

Diese Dateien benötigt später enthalten später die Schnittstellenbeschreibungen der jeweiligen Klassen.

- Legen Sie in Ihrem C++-Ordner noch einen weiteren Ordner für Objekt-Dateien (Dateien mit der Endung `.o`) an mit dem Namen `_o`.

N: /inf3praktikum/c++/_o

Diese Dateien benötigt später der Linker um dieser zu einer einzelnen, ausführbaren Datei zusammenzuführen.

- Legen Sie nun in Ihrem C++-Ordner noch den letzten Ordner für Ausführbare-Dateien (Dateien mit der Endung `.exe`) an mit dem Namen `_exe`.

N: /inf3praktikum/c++/_exe

Diese Datei können Sie später per Doppelklick oder, bevorzugt, über die Eingabeaufforderung ausführen.

Arbeit mit Projekten

DevC++ arbeitet, wie Visual Eiffel und C#, mit Projektstrukturen. Ebenfalls Eiffel ähnlich ist die Auswahl eines Projekttyps, mit dem Unterschied dass hier viel mehr Anwendungsbereiche abgedeckt werden als bei Visual Eiffel.

Anlegen eines neuen Projekts

Zum Anlegen eines neuen Projekts klickt man in der IDE auf „Datei ->Neu-> Projekt...“. Es erscheint ein Dialogfenster in dem Sie aufgerufen werden dem Projekt einen Namen zu geben und auszuwählen welche Art von Anwendung sie erzeugen möchten.

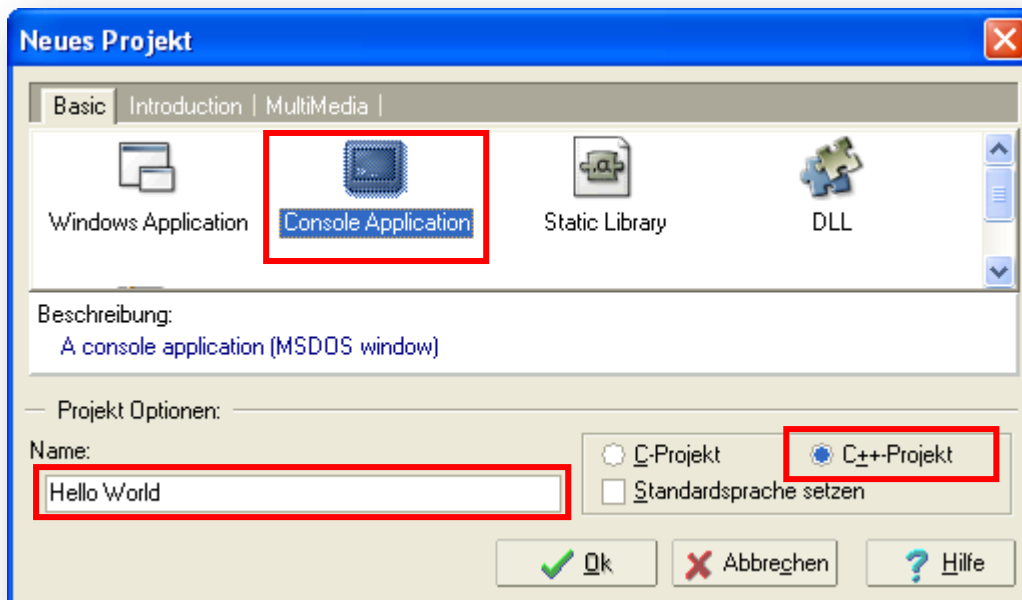


Abbildung 12: C++ Anlegen eines neuen Projektes (Hello World)

Projektoptionen

Bevor man mit der Arbeit anfängt sollte man für jedes neue Projekt die Projektoptionen einstellen. Das Dialogfenster für die Projektoptionen wird mit Klick auf „Projekt“ -> „Projekt Optionen“ geöffnet. Anders als bei Eiffel müssen hier jedoch lediglich die erstellten Ordner für `.cpp`, `.o`, `.exe` und `.h` eingebunden werden. Abbildung 13 bis Abbildung 16 zeigen die Bereiche in den diese Einstellungen vorgenommen werden müssen.

Bei jedem neuen Projekt sind die in Abbildung bis gezeigten Bereiche zunächst leer. Um hier einen Ordner hinzuzufügen muss man entweder den kompletten Pfad für den Ordner in das, in Abbildung 13 abgebildete, Eingabefeld eingeben oder auf das Icon rechts daneben klicken und zu dem Ordner navigieren. Welche Ordner in welchem Bereich eingebunden werden müssen wird in den folgenden Schritten erläutert.

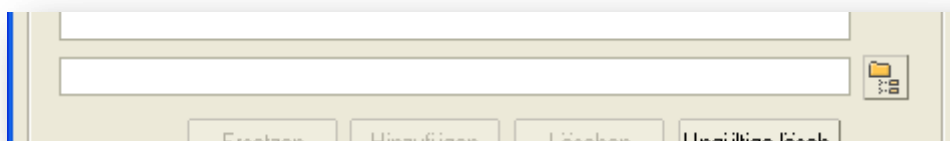


Abbildung 13: Eingabefeld für den Ordnerpfad

In den Projektoptionen existieren zwei Hauptbereiche in den Ordneinstellungen vorgenommen werden müssen. Diese heißen „Verzeic...“ und „Build O...“ und sind in Abbildung 14 rot markiert.

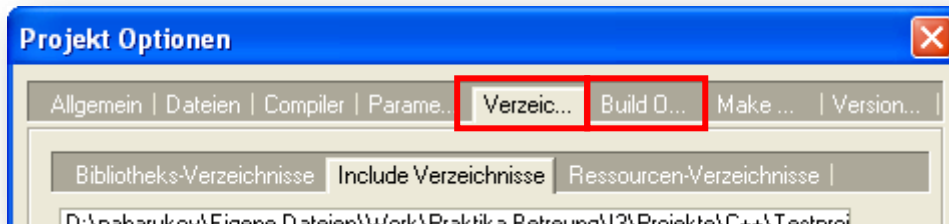


Abbildung 14: Einbinden des Ordners für Headerdateien

Im Bereich „Verzeic...“ existieren nochmals zwei Reiter unter den Einstellungen vorgenommen werden müssen diese sind

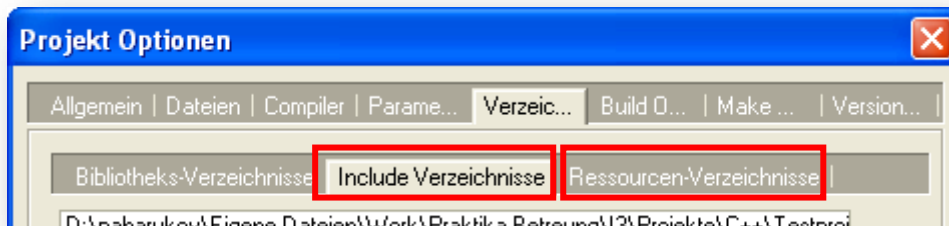


Abbildung 15: Bereiche in den `_h` und `_cpp` Ordner eingebunden werden müssen

- Include Verzeichnisse
 - o Hier muss der Ordner für die Headerdateien eingebunden werden. Auf den RZ-Rechnern wäre dies der Pfad
N: /inf3praktikum/c++/_h
- Ressourcen Verzeichnisse
 - o Hier muss der Ordner für die Quellcodedateien eingebunden werden. Auf den RZ-Rechnern wäre dies der Pfad
N: /inf3praktikum/c++/_cpp

Im Bereich „Build O...“ müssen ebenfalls zwei Einstellungen vorgenommen werden diese sind in Abbildung 16 markiert. Liegen die Ordner innerhalb des Projektverzeichnis reicht es aus hier nur die Ordernamen ohne den kompletten Pfad anzugeben.

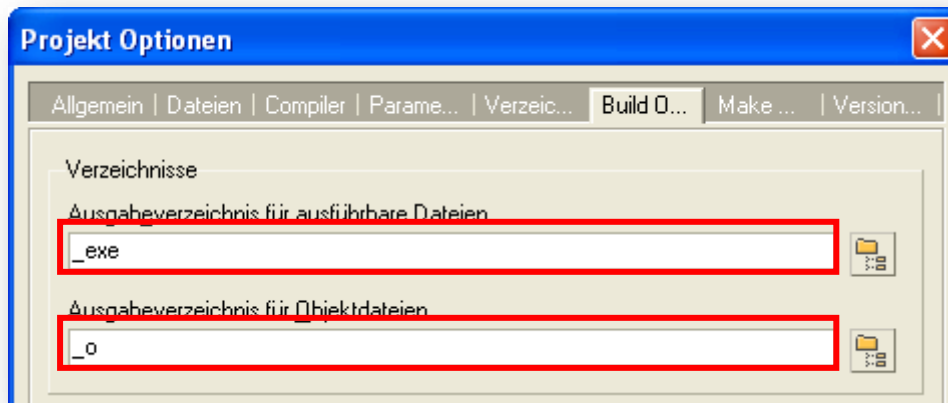


Abbildung 16: Bereiche in dem _exe und _o Ordner eingebunden werden müssen

Beispiel Code: „Hello World“

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

int main()
{
    cout<<"Hello World\n";
    cin.get();
}
```

Arbeit mit der Entwicklungsumgebung

Klassenbrowser

Der Klassenbrowser liefert Informationen zu den Klassenschnittstellen, die während der Implementierung benötigt werden. Dieser ist in der Entwicklungsumgebung eingebaut und können mit „STRG“+“Linke Maustaste“ auf den Typen aufgerufen werden. Desweiteren arbeitete DevC++ mit einer s.g. Code Completion, diese zeigt dem Programmierer bereits während der Eingabe die vorhandenen Methoden und Eigenschaften eines Objektes an. Wird die Code-Completion nicht automatisch während der Eingabe aufgerufen kann dies durch drücken von „STRG“+“Leertaste“ erzwungen werden. In der Abb ist das Code-Completion Menü zu sehen das bei der Eingabe des Buchstaben „c“ erscheint. Hier kann man nun zu der gewünschten Methode navigieren durch drücken der „Eingabetaste“ bestätigen. Sollte eine Eigenschaft oder Methode weitere Includes voraussetzen so werden diese automatisch in den Header eingefügt.

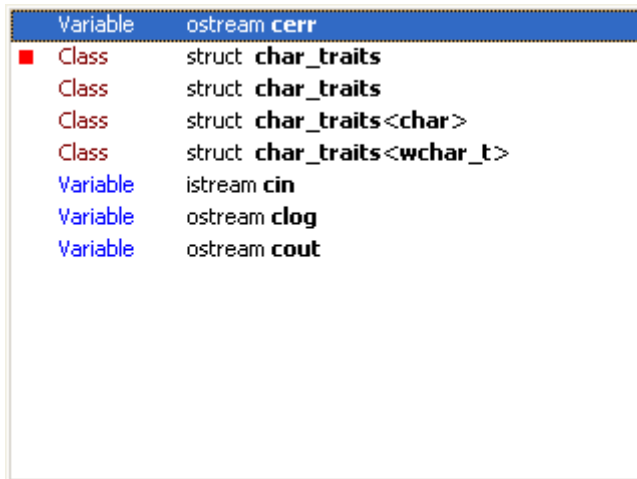


Abbildung 17: Code-Completion Menü

Übersetzen

Zum Übersetzen des Projekts klicken Sie auf „Ausführen ->Kompilieren“ oder drücken Sie „CTRL + F9“.

Bei diesem Vorgang wird in Ihrem `_exe` Ordner eine ausführbare Datei erstellt die später über die CMD aufgerufen werden kann.

Vorschläge zur Verbesserung dieser Starthilfe nehme ich gerne entgegen.