

# Installation Eclipse/CDT

Mit Eclipse CDT (C/C++ Development Tooling) steht ein Eclipse-PlugIn für die Entwicklung mit C/C++ zur Verfügung.

Homepage: <http://www.eclipse.org/cdt>

Es besteht die Möglichkeit dieses PlugIn nachträglich in eine bestehende Eclipse-Installation (z.B. *Eclipse Standard* für die Java-Entwicklung) zu installieren.

Als Alternative stehen auch bereits mit CDT vorinstallierte Eclipse-Versionen zum Download bereit.

Vor der Inbetriebnahme von Eclipse/CDT müssen aber zuerst entsprechende C/C++-Entwicklungs-Tools (Kompiler, Debugger, make-Tool, etc.) installiert sein.

Wichtig: Diese sind nicht Teil des CDT-PlugIn's!

## Installation C/C++-Entwicklungs-Tools

### Linux und Mac OS

Jeweils vor der Installation von Eclipse/CDT sicherstellen, dass die Entwickler-Packages mit C/C++-Kompiler, Makefile-Support und gdb-Debugger installiert sind.

### MS Windows

Unter MS Windows kann z.B. *Visual Studio C++*, *MinGW* oder *Cygwin* installiert werden.

Nachfolgend wird die Installation von *Cygwin* auf Windows10 beschrieben.

#### Installation von Cygwin

Download des *Setup-Programmes* (*setup-x86.exe* oder *setup-x86 64.exe*) von der Cygwin-Homepage: <http://www.cygwin.com>

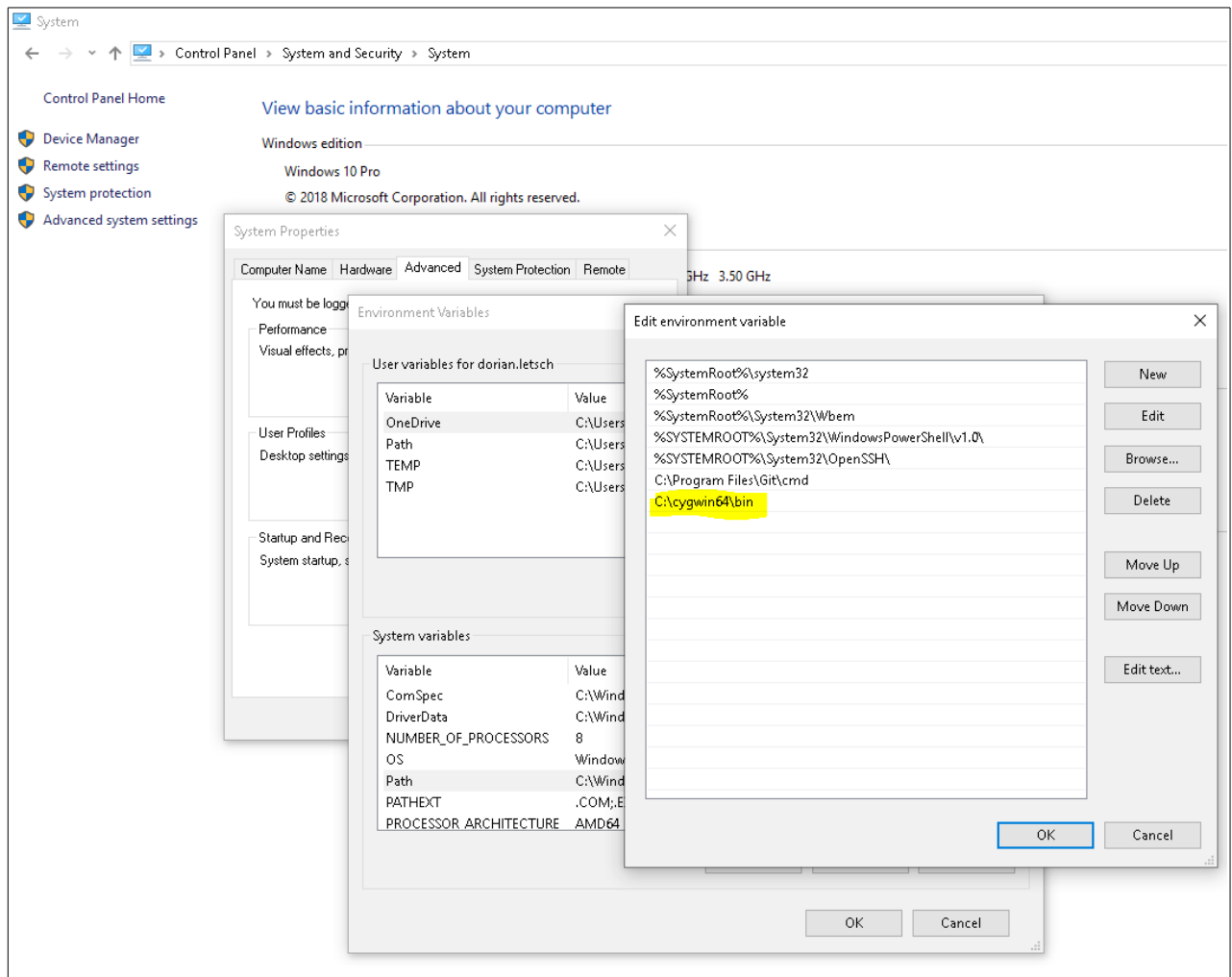
Setup starten, jeweils die Standard-Vorschläge übernehmen (als Mirror am besten: <http://www.pirbot.com>) und bei der Auswahl der Software-Paketen zusätzlich zu den Grundeinstellungen aus der Kategorie *Devel* mindestens folgende Pakete zusätzlich selektieren:

- *gcc-g++*
- *gdb*
- *make*

Ein eventuelles PopUp-Window nach der Installation mit "This program might not have installed correctly" ist mit "This program installed correctly" zu quittieren ;-)

Danach ist der Pfad zu den Cygwin-Binaries (wo sich die Binaries befinden, z.B. `g++.exe`) in den Such-Pfad (*PATH*-Environment-Variable) von Windows einzutragen. Dieser Pfad ist normalerweise: `C:\cygwin64\bin`

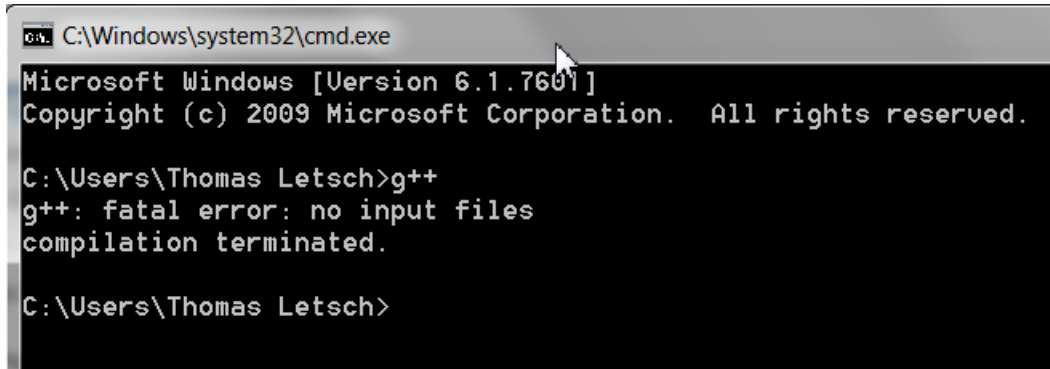
*Control Panel > System and Security > System > Advanced system settings > Environment Variables > System variables > Path > Edit > New...* :  
Den Cygwin-Pfad vorne an den bestehenden Pfad hinzufügen, also z.B.:  
`C:\cygwin64\bin;C:\Windows\system32; ...`



Dann mit 'Move Up' den Cygwin-Pfad an den Anfang verschieben.

### Verifikation:

In der *DOS-Kommando-Shell* (nicht *Bash* von Cygwin!) muss nun bei der Eingabe von `g++` eine entsprechende Reaktion auftreten:



```

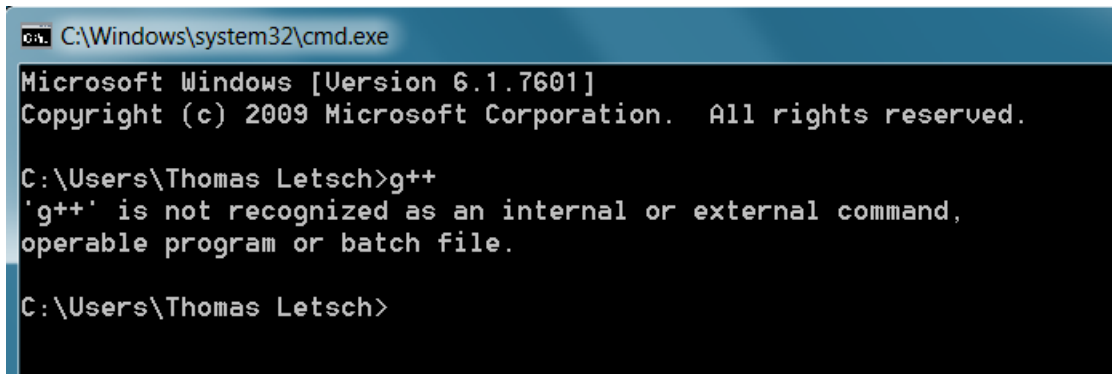
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Thomas Letsch>g++
g++: fatal error: no input files
compilation terminated.

C:\Users\Thomas Letsch>
  
```

Offensichtlich hat die Kommando-Shell den entsprechenden Eintrag im Dateisystem gefunden: `C:\cygwin64\bin\g++.exe`

Falsch wäre:



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Thomas Letsch>g++
'g++' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Thomas Letsch>
  
```

### Wichtig:

Dies muss unbedingt **vor** der Inbetriebnahme von Eclipse/CDT sichergestellt werden. Eclipse sucht aufgrund der PATH-Environment-Variable nach einem installierten C++-Compiler und passt sich dann automatisch daran (und kann dann auch mit symbolischen Links richtig umgehen).

## Installation Eclipse/CDT

Nun kann Eclipse/CDT installiert werden.

Es wird empfohlen die offizielle Eclipse Version für C/C++ Entwickler zu verwenden. Diese kann unter folgendem Link heruntergeladen werden:

<https://www.eclipse.org/downloads/packages> : Eclipse IDE for C/C++ Developers

## Verifikation und Konfiguration von Eclipse/CDT

### Hello World

Das obligate *Hello-World* :

Menü *File:New > Project... > C/C++ > C++ Project*

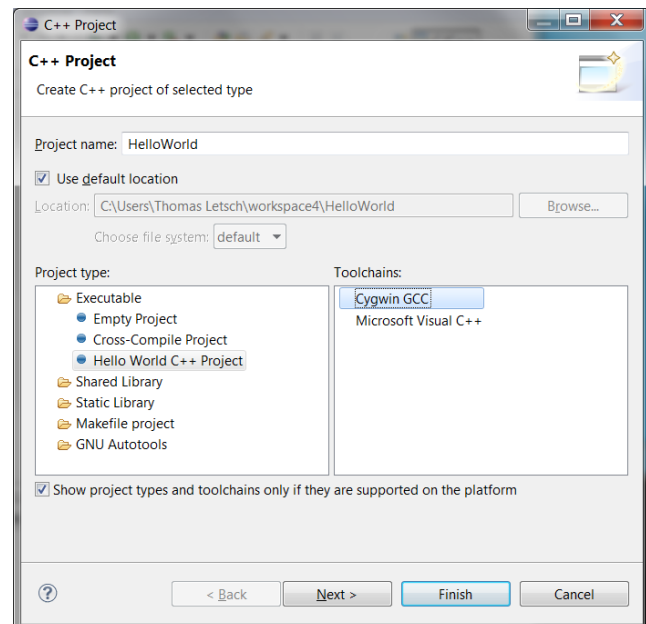
*Project type:*  
***Hello World C++ Project***

Wichtig ist der Eintrag **Cygwin GCC** (oder entsprechend auf anderen Betriebssystemen/Umgebungen) in **Toolchains**.

Wenn dieser nicht vorhanden ist, wurden die C/C++-Tools während der Installation nicht erkannt!

Dieser Eintrag muss selektiert sein →

***Finish***



Dann in der Toolbar den 'Hammer' oder Menü **Project:Build Project** .  
In der View **Console** sieht man den Kompilations- und Link-Vorgang:



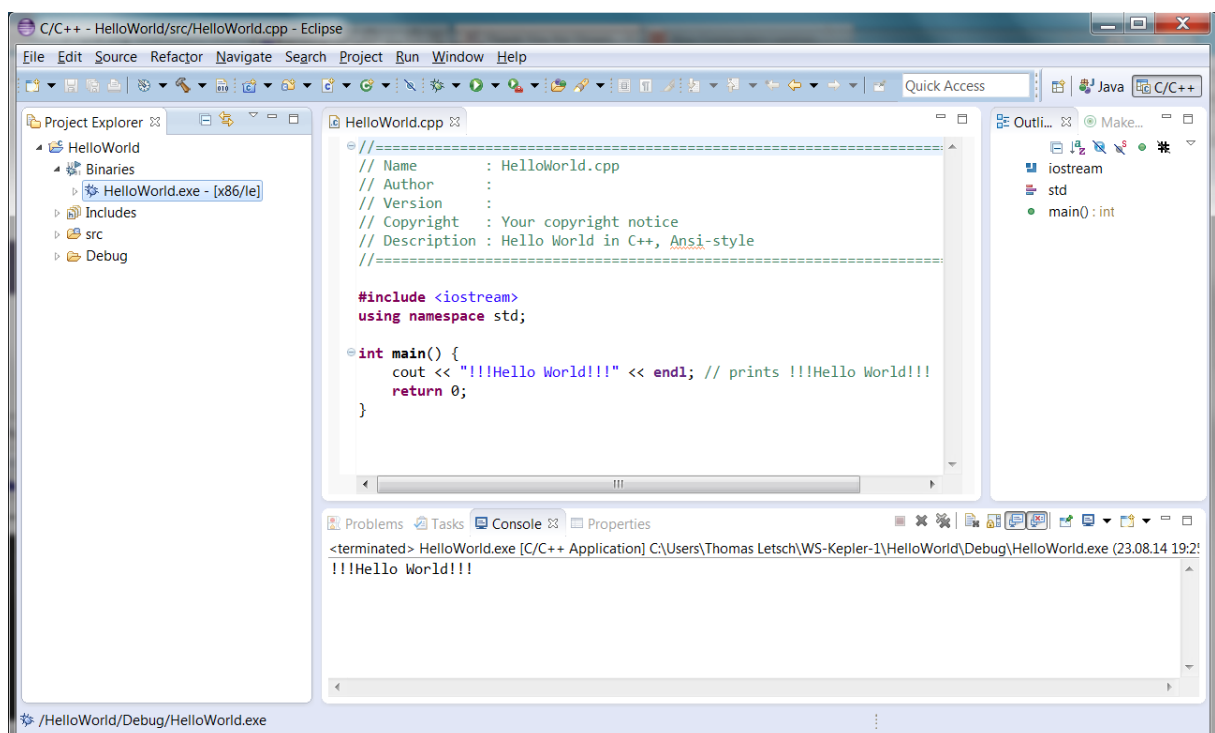
```

CDT Build Console [HelloWorld]
19:21:56 **** Build of configuration Debug for project HelloWorld ****
make all
Building file: ../src/HelloWorld.cpp
Invoking: Cygwin C++ Compiler
g++ -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/HelloWorld.d" -MT"src/HelloWorld.d" -o "src/HelloWorld.o" "../src/HelloWorld.cpp"
Finished building: ../src/HelloWorld.cpp

Building target: HelloWorld.exe
Invoking: Cygwin C++ Linker
g++ -o "HelloWorld.exe" ./src/HelloWorld.o
Finished building target: HelloWorld.exe

19:21:58 Build Finished (took 2s.355ms)
  
```

Das Programm resp. Binary kann gestartet werden mit:  
Im **ProjektExplorer** : Context-Menü auf **Binaries>HelloWorld.exe** und dann:  
**Run As > Local C/C++ Application**



## Debugger

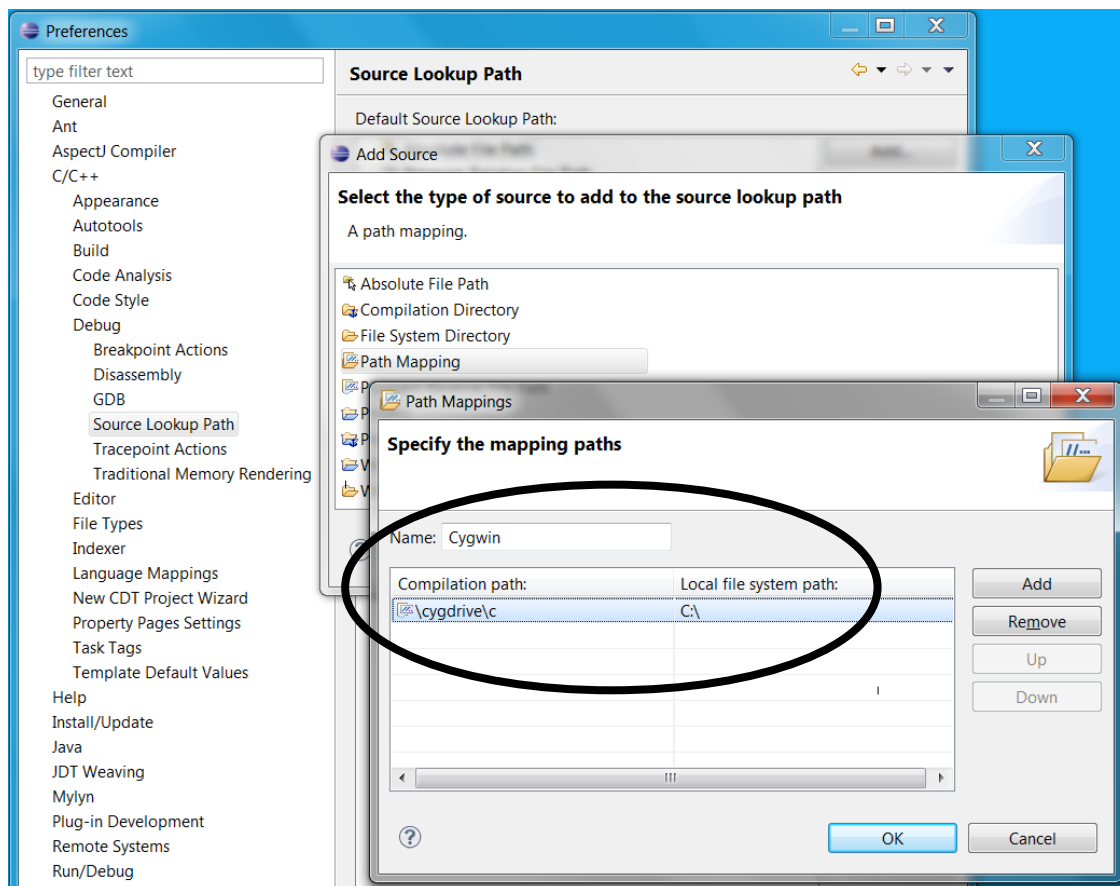
Beim Einsatz des Debuggers unter *MS Windows* müssen die Pfad-Angaben vom Cygwin-g++-Kompiler (mit */cygdrive/c* etc.) entsprechend auf normale Windows-Pfade umgewandelt werden.

Dazu wird ein entsprechendes Mapping in den Präferenzen definiert:

Menü: *Window > Preferences > C/C++ > Debug > Source Lookup Path :*

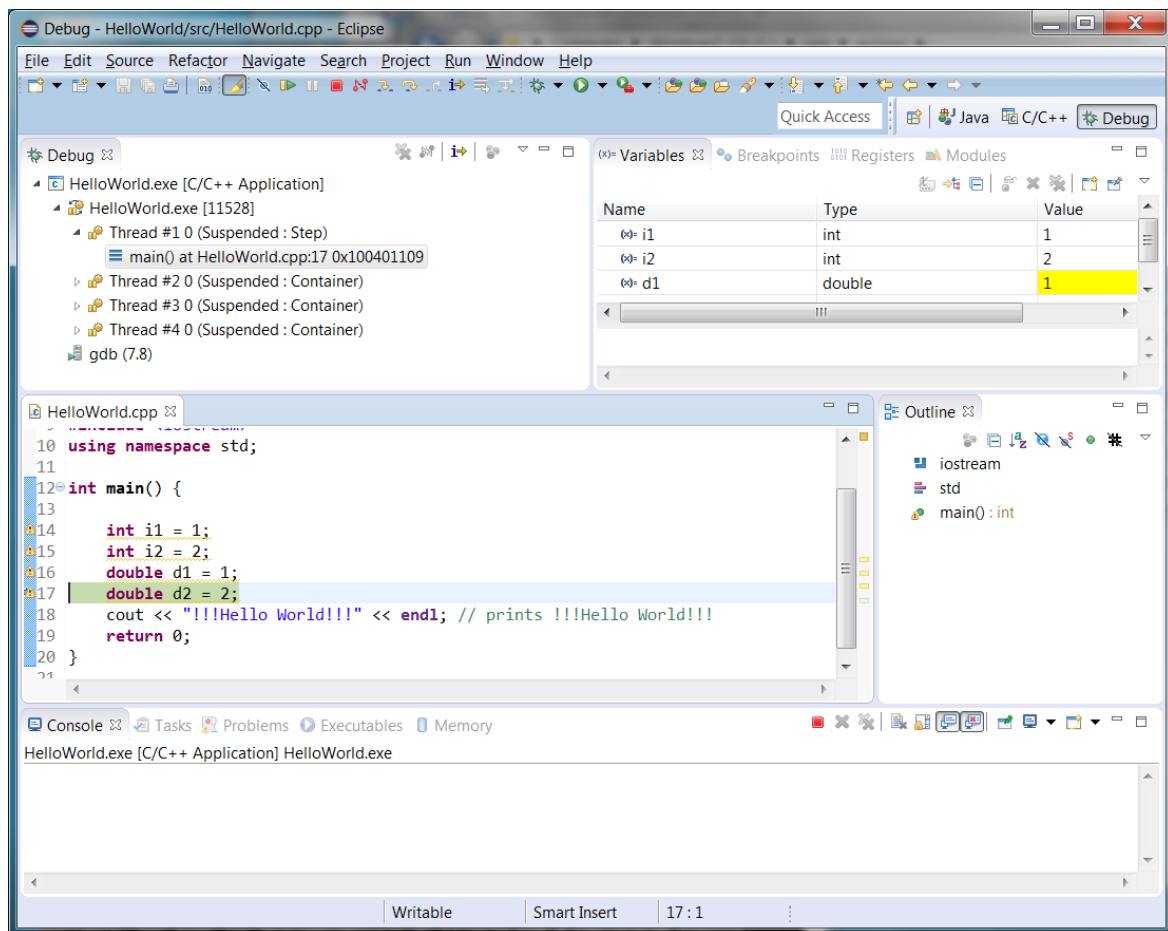
*Add... > Path Mapping :*

- **Name:** **Cygwin**
- Add :**
  - **Compilation path:** **/cygdrive/c**
  - **Local file system path:** **C:\** *(mit dem Browse-Button einstellen)*



Für weitere verwendete Drives ist dies entsprechend auch anzuwenden (z.B. für *u:* \ bei der HSR-Übungsumgebung).

Danach kann der Debugger wie in Java verwendet werden:



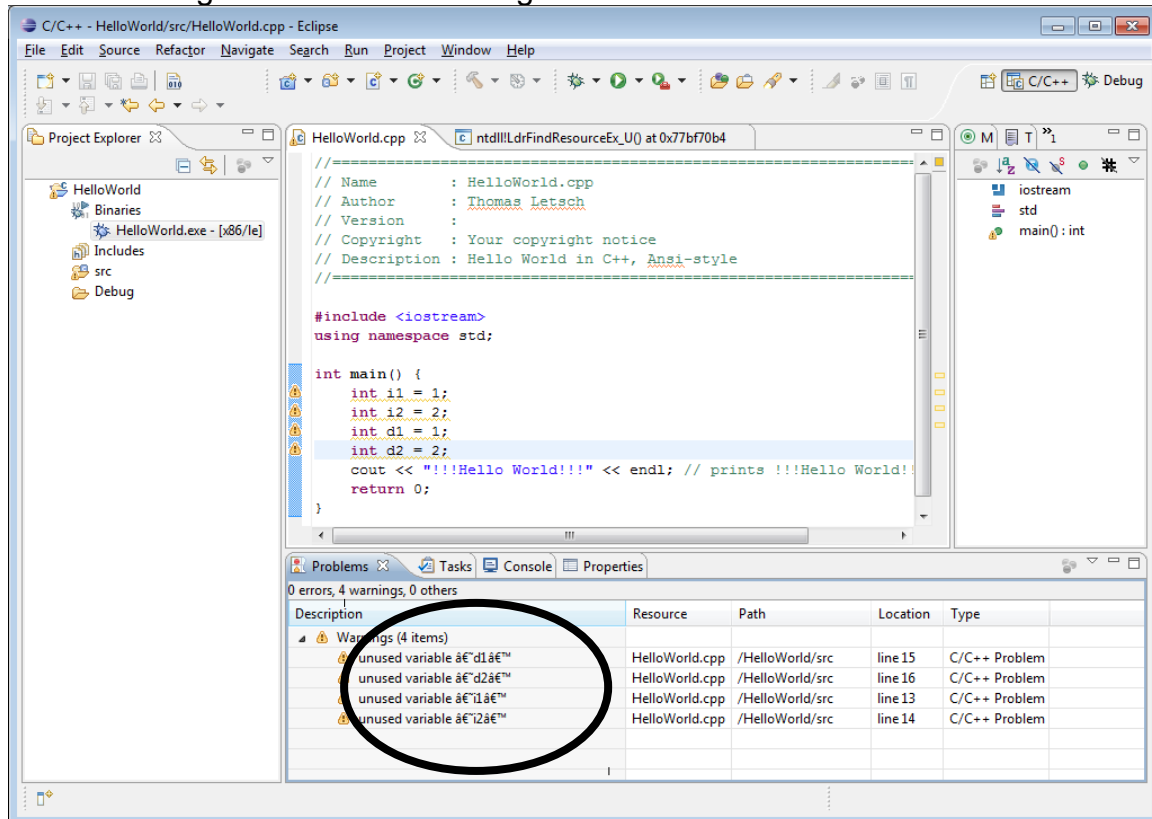
Hinweis:

Falls auf **Windows** der Debugger nicht funktioniert:

Sicherstellen, dass im *Setup*-Programm von Cygwin die Bibliothek **libexpat1** selektiert ist (in Category *Libs*)

## Encoding

Falls Warnungen und Fehlermeldungen unleserliche Zeichen enthalten:



Dann ist das Encoding nicht korrekt.

Dies kann explizit definiert werden mit dem Setzen der entsprechenden Environment-Variable:

Menü: *Window > Preferences > C/C++ > Build > Environment > Add... :*

- Variable: **LANG**
- Value: **en\_US.ISO-8859-1**

