Einleitung

Für die Bearbeitung der Übungsaufgaben der Lehrveranstaltung benötigst du Zugang zu einem Rechner mit einem C++-Compiler und einer entsprechenden Entwicklungsumgebung. Der Compiler (zusammen mit dem Linker) übersetzt den in Textform vorliegenden Programmcode in eine ausführbare Datei. Die Entwicklungsumgebung ist im Prinzip nur ein Texteditor mit einigen Zusatzfunktionen, die den Programmierer beim Erstellen des Programmcodes unterstützt und auf Knopfdruck den Compiler startet. Wir haben in den Poolräumen den *GCC Compiler* und einen Texteditor bereitgestellt.

In diesem Dokument findest du eine Anleitung, wie du auf einem Poolrechner oder auf deinem Laptop das erste Programm kompilieren und ausführen kannst. Der Aufbau ist wie folgt:

- *Abschnitt 0.1* führt Schritt für Schritt durch die Installation der benötigten Programme *auf deinem eigenen Rechner*. Diese sind für Rechner in Poolräumen *nicht* nötig.
- *Abschnitt 0.2* und *Abschnitt 0.3* beschreiben die nötigen Schritte zur Verwendung der Rechner in den Poolräumen.
- *Abschnitt 0.4* komplettiert die Einrichtung der Entwicklungsumgebung auf allen Rechnern. Dies ist *sowohl in Poolräumen als auch auf eigenen Rechnern* notwendig!
- *Abschnitt 0.5* rundet die Einrichtung mit einer Übungsaufgabe ab.
- *Abschnitt 0.6* erklärt, wo die Übungsblätter und weiteren Lernmaterialien herunter geladen werden können.

Teilnehmern an Übungen in Poolräumen empfehlen wir dringend, wie in Abschnitt 0.2 und 0.5 beschrieben eine Entwicklungsumgebung auf dem eigenen Rechner einzurichten. Diese wird für die Zusatzaufgaben der Übungsblätter sowie für das selbstständige Lernen benötigt. Programmieren lernt man nur durch Selbermachen, ohne Praxis werdet ihr die Klausur nicht bestehen.

> **Zusatzinformation** Im gegensatz zu den letzten Jahren, verwenden wir keine integrierte Entwicklungsumgebung (wie z.B. Eclipse) benutzen. Alle erstellten Programme werden über die Konsole kompiliert und ausgeführt (ggf. mit Hilfe von Makefiles). Daher ist es dir im Grunde freigestellt welchen Editor oder Compiler du verwenden möchtest. Wenn du aber einen anderen Compiler als den hier vorgeschlagenen verwendest (z.B. Clang, XCode oder MSVC), können wir dir bei auftretenden Problemen nicht weiterhelfen.

0.1 Für Laptopübungen: Eigenen Laptop vorbereiten

> **Nur für den eigenen Computer** Auf den Poolrechnern ist die Software (GCC, MinGW) bereits installiert. Je nachdem in welchem Raum ihr seid, ist auch bereits ein Texteditor (Notepad++, Atom, emacs) installiert. Dort könnt ihr direkt mit dem Abschnitt 0.3 weitermachen. Die folgenden Anweisungen sind also nur für die Installation auf euren eigenen Computern nötig.

Installation unter MacOS Unterstützen können wir euch leider nur für die Betriebssysteme Linux und Windows. Unter MacOS ist die C++-Programmierung mit Clang allerdings ebenfalls möglich.

Für diejenigen, die es selbstständig versuchen wollen, hier die Kurzanleitung zur Installation:

1. Öffne ein Terminal-Fenster (cmd-Taste + Leertaste, dort `terminal` eingeben und Enter drücken)
2. Dort eingeben: `xcode-select --install` (und Enter drücken)
3. Auf *Installieren* klicken. Dies installiert den clang compliler und die nötigen Bibliotheken.
4. Atom in der Variante für MacOS herunterladen (https://atom.io/).
5. Die heruntergeladene Setupdatei ausführen.

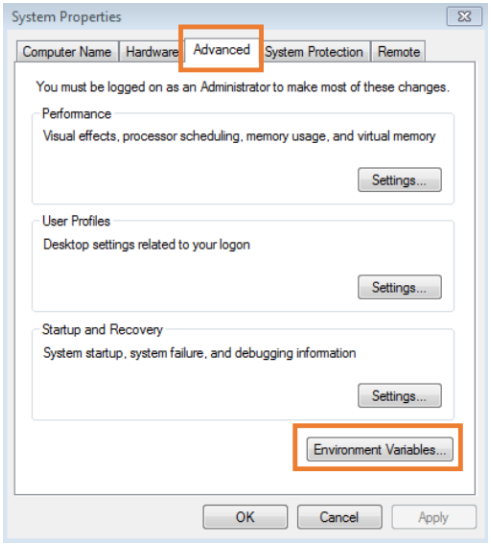
Um für eine Übung den eigenen Laptop nutzen zu können, musst du einen C++-Compiler und einen geeigneten Texteditor (z.B. Notepad++, Atom) installiert haben. Im Rahmen der Übungen verwende bitte die zur Verfügung gestellte Version von GCC – andere Compiler können auch funktionieren, wir können dafür aber keinerlei Unterstützung anbieten. Bitte beachte, dass wir hier von einem 64bit-Betriebssystem (Windows) ausgehen. Dies sollte schon seit längerem der Standard sein.

Installation von MinGW (Windows)

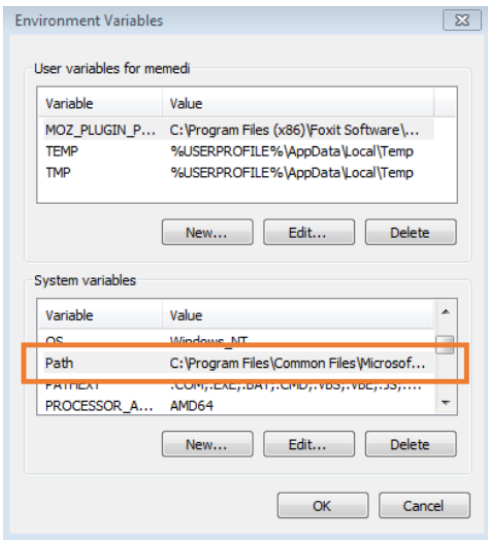
Beim Programmieren greifen wir auf bestehende Programmteile und Funktionen, sogenannte Bibliotheken, zu, um nicht ständig das Rad neu erfinden zu müssen. Die wichtigsten Bibliotheken, die zusammen mit C++ definiert sind, nennt man die Standardbibliothek. Um die Standardbibliothek zusammen mit dem GCC Compiler zu installieren, kommt *MinGW* zum Einsatz.

1. Lade MinGW-w64 herunter: https://sourceforge.net/projects/mingw-w64/files/latest/download
2. Starte die Installation (`mingw-w64-install.exe`)
3. Im `Settings` Fenster wähle folgende Einstellungen:
   - Version: 8.1.0 (mindestens 5.1.0)
   - Architecture: x86_64
   - Threads: posix
   - Exception: seh
   - Build revision: höchste verfügbare Zahl
4. Als Zielorder gebe `C:\mingw-w64` an.
5. Du solltest nun die Datei `g++.exe` im Verzeichnis `C:\mingw-w64\mingw64\bin` finden.

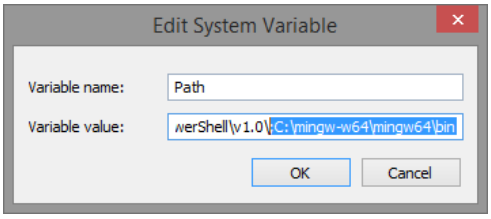
6. MinGW kann sich nicht selbst dem PATH hinzufügen. Führe daher die folgenden (und in Abbildung 1 gezeigten Schritte) aus, um dies manuell zu erledigen.



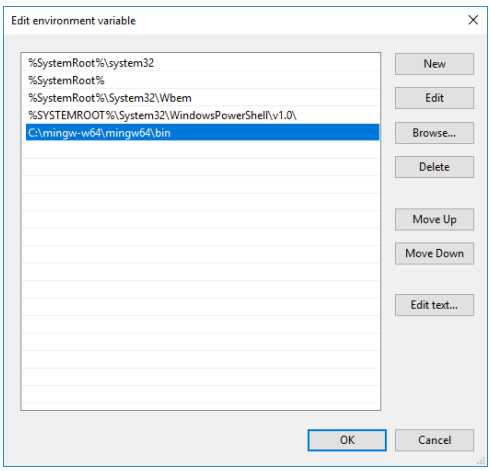
Schritt 1



Schritt 2



Schritt 3, Variante A



Schritt 3, Variante B

Abbildung 1: Umgebungsvariablen (Environment Variables) setzen.

a) Start→*Rechtsklick* auf Computer→Eigenschaften→Erweiterte Systemeinstellungen
(Alternativ drücke `[Windows-Taste] + R` und gebe dort `sysdm.cpl` ein.)

b) Im Reiter *Erweitert*, klicke auf *Umgebungsvariablen*

c) Suche im *unteren Teil des Fensters* nach der Variablen `PATH` und klicke auf *Bearbeiten*.

d) Füge den folgenden den Pfad hinzu `C:\mingw-w64\mingw64\bin` indem du wie folgt vorgehst
(je nach Windows-Version muss Variante A oder Variante B durchgeführt werden):

A) Füge den folgenden Text an den *Wert der Variablen* an: `;C:\mingw-w64\mingw64\bin`
Achte darauf, den Text nur anzuhängen und den bestehenden Text *nicht* zu entfernen oder überschreiben!
Achte weiterhin darauf, das Semikolon (vor dem C) nicht zu vergessen!

B) Klicke auf *Neu* und füge `C:\mingw-w64\mingw64\bin` ein.

7. Um die Installation zu überprüfen, öffne die Kommandozeile (Windows-Taste + r und `cmd` eingeben) und gib `g++ --version` ein. Du solltest in etwa die Ausgabe wie in Abbildung 2 erhalten. In manchen Fällen musst du evtl. vorher Windows neu starten.

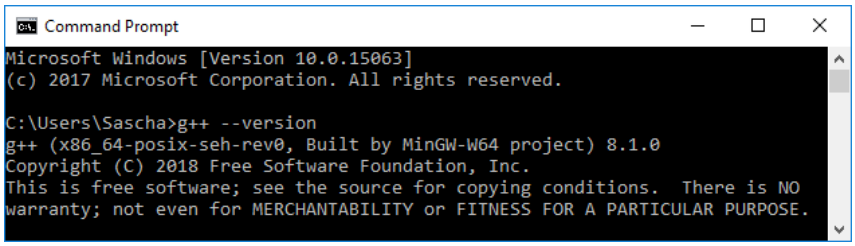


Abbildung 2: Überprüfung der Installation von MinGW

Installation eines Texteditors

Grundsätzlich kannst du jeden beliebigen Texteditor verwenden. Wir empfehlen allerdings die Verwendung eines Texteditors mit *Code-Highlighting*, um die Lesbarkeit deines Programmcodes zu erhöhen. Solche Editoren sind beispielsweise Notepad++ (https://notepad-plus-plus.org/downloads/v7.7.1/) oder Atom (https://atom.io/download/windows_x64).

1. Lade einen Texteditor deiner Wahl herunter.

2. Führe die Installation mit den Standardeinstellungen durch.

0.2 Für Poolräume: Freischalten des Netzlaufwerks und des Windows Logins

Jeder Nutzer mit gültigem IMT-Login verfügt über einen Speicherbereich im IMT. Dieser wird benötigt um in einem Poolraum arbeiten zu können und Daten (wie z.B. die Übungsaufgaben) dauerhaft speichern zu können. Um auf dieses Laufwerk zugreifen zu können, muss jeder Nutzer seinen Bereich zunächst einmalig selber freischalten.

1. Dazu brauchst du einen Rechner mit Internetzugriff. (Wenn du dich selbst nicht einloggen kannst, frage einfach einen hilfsbereiten Kommilitonen der sich bereits anmelden kann).

2. Dann öffnest du im Web-Browser den Bereich Benutzerverwaltung des IMT:
   https://serviceportal.uni-paderborn.de/web/portal/willkommen
   Auf dieser Seite loggst du dich mit deinem IMT-Login und Passwort an.

3. Unter "Benutzerverwaltung"wähle den Punkt „Netzwerkspeicher“. Hier kannst du das Verzeichnis für private und öffentliche Daten freischalten. Wenn hier steht, wie viel Speicherplatz auf deinem Laufwerk frei ist, hast du diesen Schritt bereits erledigt.

4. Melde dich von der Seite mittels des Abmelde-Knopfes ab.

5. Warte fünf Minuten (in der Zeit kann sich der Nächste einloggen).

6. Wenn du auf einer virtuelle Maschine angemeldet bist, fahre diese herunter (Start → Runterfahren) und starte sie neu.

Sobald du dich jetzt auf einem Windows-Rechner mit deinem IMT-Zugang eingeloggt hast, solltest du automatisch ein Laufwerk **U:** haben, dass mit deinem CIF-FS-home-Verzeichnis verknüpft ist. Hier gespeicherte Daten bleiben dauerhaft erhalten und du kannst auch von deinem eigenen Rechner darauf zugreifen (siehe dazu die Hinweise beim IMT).

Weitere Hinweise findest du beim IMT unter: https://hilfe.uni-paderborn.de/Netzwerkspeicher
Bei Problemen kannst du dich auch an das Notebook-café wenden: http://nbc.uni-paderborn.de/

0.3 Im Poolraum anmelden

Je nach Poolraum müsst ihr verschiedene Schritte durchführen, um Atom zu starten:

Raum P1.2.04 und P1.2.01.1 (Maschinenbau Poolraum) Die Rechner in diesem Raum sind Terminals, die sich mit virtuellen Maschinen verbinden – d.h. die Rechner selber dienen nur zur Anzeige, eure Programme werden im Rechenzentrum der Uni ausgeführt. Meldet euch zunächst mit eurem IMT-Account am System an. Dann bekommt ihr eine Liste von möglichen virtuellen Maschinen zur Auswahl. Wählt hier die Maschine **IMT Windows 10 NVidia** aus. Bitte weiterlesen bei Abschnitt *Virtuelle Maschine*.

Raum N5.206 Auf den Rechnern in diesem Räumen läuft ein normales Windows – in den Übungen arbeiten wir jedoch nicht direkt auf den Rechnern sondern auf einer virtuellen Maschine die im Rechenzentrum der Uni ausgeführt wird. Meldet euch zunächst mit eurem IMT-Account am System an. Startet dann das Programm **VMware Horizon View Client** auf dem Desktop und bestätigt so lange, bis ihr eine Auswahl von virtuellen Maschinen bekommt. Wählt hier die Maschine **IMT Windows 10 NVidia** aus.

Virtuelle Maschine: Standardmäßig ist euer Netzwerkspeicher (siehe Abschnitt 0.3) an das Laufwerk `U:` gebunden; wobei Daten im Unterverzeichnis `data` abgelegt werden.

Legt euch am Besten ein Verzeichnis an, in dem ihr eure Projekte abspeichert. Innerhalb dieses Verzeichnis legt ihr am Besten für jede Übungszettel ein Unterverzeichnis an. Das Ergebnis sieht dann beispielsweise so aus: `U:\data\gping\uebung-01`

In der virtuellen Maschine findet ihr auf dem Desktop eine Verknüpfung mit Notepad++ sowie eine Verknüpfung mit der Kommandozeile.

Raum P1.6.12.4 und P7.2.02.01 Auf den Rechnern in diesen Räumen läuft Linux. Du kannst dich auch hier mit deinem IMT-Account anmelden. Den Texteditor Atom findest du im Startmenü: `App` → `Development (oder Entwicklung)` → `Atom`.

Legt euch am Besten ein Verzeichnis an, in dem ihr eure Projekte abspeichert. Innerhalb dieses Verzeichnis legt ihr am Besten für jede Übungszettel ein Unterverzeichnis an. Das Ergebnis sieht dann beispielsweise so aus: `~/data/gping/uebung-01`

Falls das nicht funktioniert, du deinen Netzwerkspeicher aber wie in Abschnitt 0.3 eingerichtet hast, probiere bitte den Pfad `~/gping/uebung-01` aus.

> **Hinweise zu Dateipfaden** Unter bestimmten Umständen kann es zu Problemen beim Kompilieren und Ausführen kommen, wenn sich bestimmte Zeichen im Dateipfad befinden. Daher solltet ihr folgendes vermeiden:
>
> - Umlaute: Zeichen, die nicht im englischen Alphabet vorkommen, z.B. *ä*, *Ö* oder *ß*.
> - Leerzeichen: Z.B. `C:\Mein Ordner\main.cpp`. Verwendet stattdessen am Besten Unterstriche oder Bindestriche.
>
> Alle oben angegebenen Pfadbeispiele entsprechen diesen Regeln (z.B. `U:\data\gping\uebung-01` oder `~/data/gping/uebung-01`). Problematisch wären hingegen `U:\data\gp ing\übung-01`.

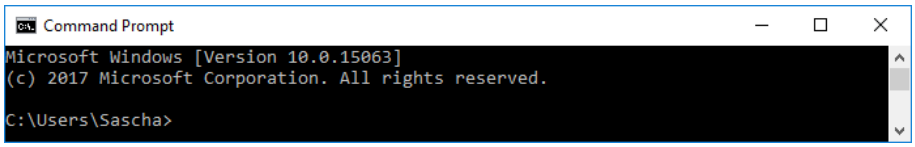
0.4 Exkursion: Die Kommandozeile

Da wir dieses Jahr unsere Programme ausschließlich über die Kommandozeile kompilieren und ausführen, gibt es hier eine Kurze Einführung mit den wichtigsten Kommandos. Die Kommandozeile (oder auch Konsole, Terminal, CMD, CLI, oder Prompt (Eingabeaufforderung)) bietet die Möglichkeiten, ohne grafische Benutzeroberfläche, Dateien und Ordner auf deinem Computer zu Bearbeiten und Programme auszuführen.

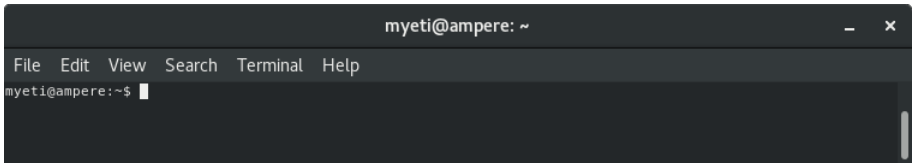
Um die Kommandozeile zu starten, gehe wie folgt vor:

- Windows 10: Drücke die Windows-Taste → gib in die Suchleiste `cmd` ein → bestätige mit Enter.
- Ältere Windows Versionen: Start Menu → Alle Programme → Zubehör → Eingabeaufforderung.
- Linux: Wahrscheinlich ist es unter Programme → Zubehör → Terminal, aber das ist von deinem System abhängig (Windows-Taste und `terminal` eingeben sollte auch funktionieren).
- OS X: Öffne das Launchpad → Andere → Terminal.

Du solltest nun ein weißes oder schwarzes Fenster sehen, das auf deine Anweisungen wartet.



Windows Kommandozeile



Linux Terminal

Anzeigen von Dateien und Unterordnern

Die wichtigsten Kommandos, die jeder beherrschen sollte, dienen dem Navigieren im Verzeichnisbaum deines Computers, sowie dem Anzeigen, Erstellen, Löschen und Kopieren von Ordnern und Dateien.

Wenn du die Kommandozeile startest, befindest du dich standardmäßig in deinem aktuellen Benutzerverzeichnis `C:\Users\DEINBENUTZERNAME` oder im Systemverzeichnis `C:\windows\system32` unter Windows, bzw. `~` oder `/home/DEINBENUTZERNAME` unter Linux/Mac (auf den Poolrechnern sieht der Pfad nochmal anders aus). Um herauszufinden in welchem Ordner du dich gerade befindest, kannst du einfach folgendes Kommando eingeben und mit Enter bestätigen.

- Windows: `cd`
  (cd steht für *change directory*)
- Linux/OS X: `pwd`
  (pwd steht für *print working directory*)

Nun wollen wir herauszufinden, welche Datein und Unterordner sich in unserem aktuellen Verzeichnis befinden. Dazu können wir die Befehle `dir` (Windows) oder `ls` (Linux) verwenden. Wenn ihr diese Kommandos eingebt, solltet ihr alle Dateien und Unterordner im aktuellen Ordner angezeigt bekommen, was in etwa wie folgt aussieht:

- Windows:

```
> dir
Directory of C:\Users\Sascha
24/09/2018  13:54 PM <DIR> .
24/09/2018  13:54 PM <DIR> ..
14/08/2018  09:33 PM <DIR> Contacts
24/09/2018  14:50 PM <DIR> Desktop
15/08/2018  09:45 PM <DIR> Documents
...
```

- Linux:

```
$ ls
data Desktop  Documents  Downloads  Music  Pictures ...
```

Wechseln des Verzeichnisses

Nun möchten wir das aktuelle Verzeichnis wechseln. Der Befehl dazu ist auf allen Betriebssystemen der gleiche: `cd VERZEICHNIS`

Wir wollen in unseren Projektordner `U:\data\gping` (Windows) bzw. `~/data/gping/` (Linux) wechseln (Sollte der Pfad noch nicht existieren, könnt ihr den wie gewohnt über den Datei Explorer anlegen, oder, wie weiter unten beschrieben, über die Kommandozeile anlegen).

Eine Besonderheit unter Windows ist, dass du nicht direkt das Laufwerk mit dem Befehl `cd` wechseln kannst. Um auf das Laufwerk `U:` zu gelangen, geben wir einfach den Laufwerkbuchstaben gefolgt von einem Doppelpunkt ein: `U:`.

Nun können wir einfach in das gewünschte Verzeichnis wechseln:

- Windows:

```
> U:
> cd data
> cd gping
> cd
U:\data\gping
```

- Linux:

```
$ cd ~
$ cd data
$ cd gping
$ pwd
/home/sascha/data/gping
```

Hinweis Eine besondere Rolle hat der Ordner mit dem Namen "..". Dieser Ordner ist ein virtueller Ordner, der dazu dient um auf den Übergeordneten Ordner zurückzukommen. Möchstest du also zurück zum vorherigen Ordner, kannst du das tun indem du einfach `cd ..` eingibst. Du kannst auch direkt in das Wurzelverzeichnis springen indem du `cd /` eingibst.

Anlegen und löschen von Ordnern

Als nächstes wollen wir einen neuen Ordner für unser erstes Programm anlegen. Ordner anlegen können wir mit dem Befehl `mkdir ORDNERNAME`. Legen wir einen Ordner mit Namen *uebung-0* an und navigieren zu diesem und legen einen weiteren Ordner *test* an (hier sind die Befehle für Windows und Linux gleich):

```
> mkdir uebung-0
> cd uebung-0
> mkdir test
```

Jetzt wollen wir den gerade erzeugten Ordner "test"wieder löschen:

- Windows:

```
> rmdir /S test
test, Are you sure <Y/N>? Y
```

- Linux:

```
$ rm -r test
```

Hier werden zusätzlich zu dem Kommando (`rmdir` bzw. `rm`) weitere Parameter (`/S` bzw. `-r`) angegeben. Diese dienen dazu, dass auch alle Unterordner und Dateien in dem Ordner mitgelöscht werden (Ansonsten erhält man eine Fehlermeldung).

Achtung Wenn du Daten mit `del`, `rmdir` oder `rm` löschst, kannst du das nicht mehr rückgängig machen, das bedeutet die gelöschten Dateien sind für immer weg! Sei also sehr vorsichtig mit diesem Befehl.

Zusammenfassung

Hier ist eine Zusammenfassung einiger nützlicher Kommandos:

Befehl (Windows)	Befehl (Linux)	Beschreibung	Beispiel
`exit`	`exit`	Fenster schließen	`exit`
`cd`	`cd`	Verzeichnis wechseln	`cd test`
`cd`	`pwd`	aktuelles Verzeichnis anzeigen	`cd`
`dir`	`ls`	Unterordner/Dateien zeigen	`dir`
`copy`	`cp`	Datei kopieren	`copy c:\test\test.txt c:\windows\test.txt`
`move`	`mv`	Datei verschieben	`move c:\test\test.txt c:\windows\test.txt`
`mkdir`	`mkdir`	neues Verzeichnis erstellen	`mkdir testdirectory`
`rmdir` (oder `del`)	`rm`	Datei löschen	`del c:\test\test.txt`
`rmdir /S`	`rm -r`	Verzeichnis löschen	`rm -r testdirectory`

Das sind nur sehr wenige der Befehle, welche du in deiner Konsole verwenden kannst, aber du wirst erstmal nicht mehr brauchen.

Falls du neugierig bist, findest du auf ss64.com eine vollständige Übersicht über alle Kommandozeilen-Befehle für alle Betriebssysteme.

0.5 Das erste Programm

> **Hinweis** Diese Anleitung setzt die Verwendung von Notepad++ oder Atom und GCC vorraus. Wenn du einen anderen Entwicklungsumgebung verwendest, sollten die Schritte aber größtenteils analog sein.

Starte Notepad++.

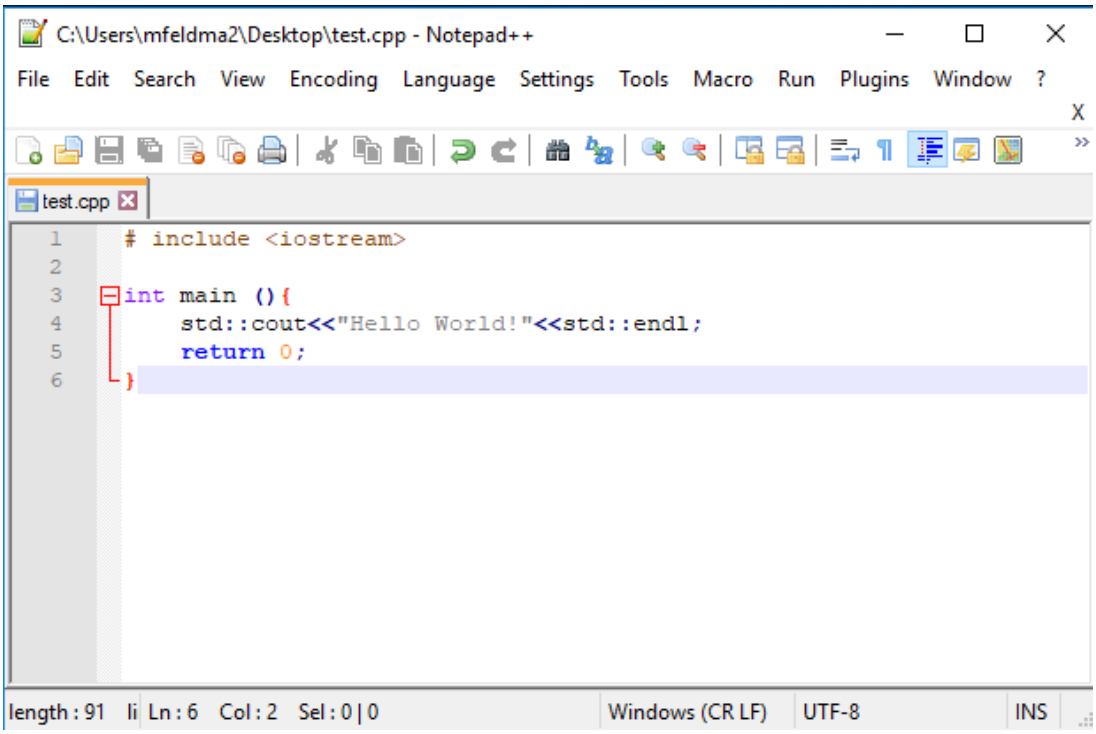
- Es sollte eine Verknüpfung auf dem Desktop angelegt worden sein. Ansonsten unter Windows zum Beispiel: *Start* → *Alle Programme* → *Notepad++* → *Notepad++*

- Unter Linux zum Beispiel: Startmenü: *App* → *Development (oder Entwicklung)* → *Atom*

Wenn Notepad++ läuft kannst du eine erste Datei erzeugen:

1. Erstelle eine neue Datei (meistens unter *File* → *New File*). Beim ersten Start von Notepad++ sollte bereits eine neue Datei geöffnet sein.

2. Speichere die Datei unter dem Namen `test.cpp` (z.B. im Ordner `U:\data\gping`, bzw. `~/data/gping`). Die Endung `.cpp` ist wichtig damit Editor und Compiler die Datei als ein C++-Programm erkennen.



3. Tippe den folgenden Code in die neu angelegte Datei und speichere (Tastenkombination `Strg+S`):

```
#include <iostream>

int main(){
    std::cout<<"Hello World!"<<std::endl;
    return 0;
}
```

Programm kompilieren und ausführen

Das vorgefertigte Programm schreibt `Hello World!` auf den Bildschirm (die klassische Ausgabe fast aller ersten Programme). Bevor das Programm ausgeführt werden kann, muss es erst vom Quellcode (Text) in ein ausführbares Binärprogramm übersetzt werden (d.h. das Programm wird kompiliert und gelinkt). Dies erfolgt darüber, dass man die entsprechenden Compiler-Befehle über die Kommandozeile eingibt. Viele Entwicklungsumgebungen stellen zwar die Funktionalität zur Verfügung das Programm direkt zu kompilieren und auszuführen, aber zum besseren Verständnis, was hinter den Kulissen passiert, werden wir unsere Programme ausschließlich über die Konsole kompilieren.

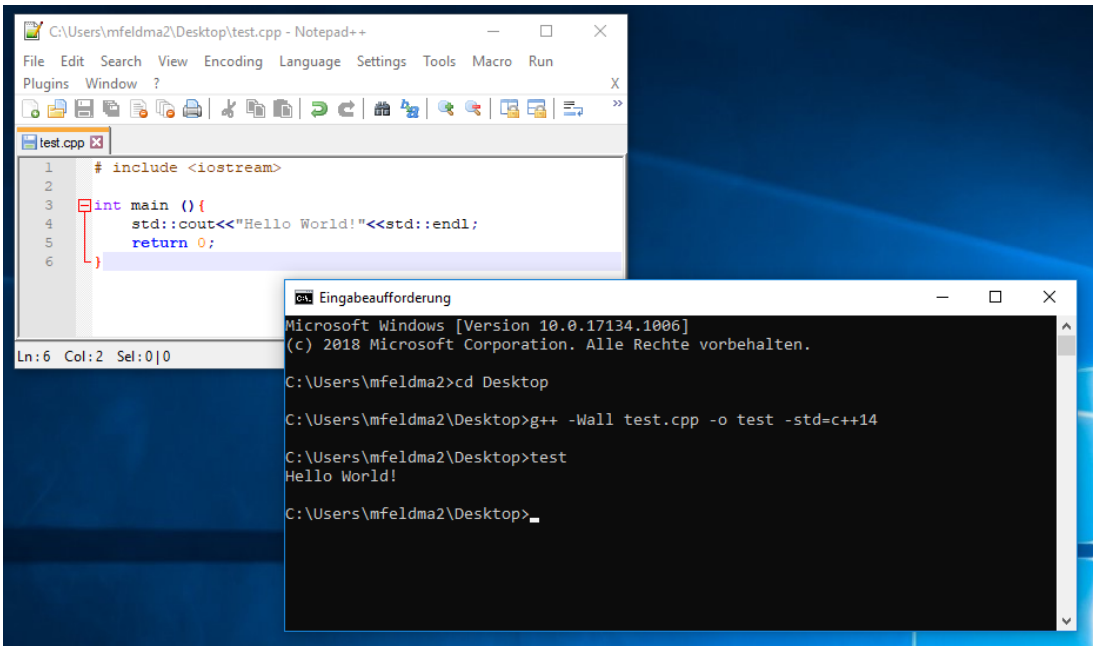
1. Öffne die Kommandozeile (Windows-Taste + r und `cmd` eingeben) und navigiere zu deinem angelegten Ordner (z.B. `U:` eingeben und dann `cd data\gping\uebung-0`,
   bzw. `cd ~/data/gping/uebung-0` für Linux).

2. Kompiliere das Program: `g++ -Wall test.cpp -o test -std=c++14`

   Wenn du alles richtig gemacht hast, sollte der Befehl ohne weitere Ausgaben ausgeführt worden sein und du findest eine neue ausführbare Datei (z.B. `test.exe`) in dem Order.

3. Führe das Program aus: `test` (`./test` in Linux)

   Nun sollte der Text `Hello World!` in der Konsole angezeigt werden.



0.6 Aufgabe

Ändere das Programm so, dass es `Hallo Paderborn!` ausgibt.

0.7 Makefiles

Wir haben bereits kennengelernt, wie man über die Kommandozeile eine einfache C++ Datei in ein ausführbares Programm übersetzt. Natürlich möchte man nicht immer lange Kommandos eingeben, wie `g++ -Wall test.cpp -o test -std=c++14`. Spätestens wenn man mit mehr als einer C++ Datei arbeitet, wird dies auch schnell unübersichtlich, da man jede Datei getrennt kompilieren muss und anschließend nochmal die kompilierten Dateien verlinken muss.

Abhilfe verschaffen hier sogenannte *Makefiles*. Makefiles sind im Grunde einfache Textdateien, die für jede zu kompilierende Datei die nötigen Kommandos und Abhängigkeiten zwischen mehreren Dateien beschreiben. Mit so einer Makefile muss man nur noch einen Befehl eingeben (`make` (Linux) bzw. `mingw32-make` (Windows)) und das *Make-Tool* übernimmt das Erstellen des Programmes.

Ein *Makefile* ist immer wie folgt aufgebaut:

```
1  <ZielA>: <AbhängigkeitA1> <AbhängigkeitA2> ...
2      <AktionA1>
3      <AktionA2>
4      ...
5
6  <ZielB>: <AbhängigkeitB1> <AbhängigkeitB2> ...
7      <AktionB1>
8      <AktionB2>
9      ...
10 ...
```

Es besteht aus einer beliebig langen Aufzählung von Zielen `<Ziel>:` gefolgt von einem Doppelpunkt. Ein Ziel kann entweder eine zu erstellende Datei sein, oder ein sogenanntes symbolisches Ziel, welches ein beliebiger Name sein kann. Hinter dem Doppelpunkt stehen für jedes Ziel Abhängigkeiten zu Dateien oder anderen Zielen. D.h. die nötigen Aktionen für ein Ziel können erst ausgeführt werden wenn die Aktionen für alle abhängigigen Ziele ausgeführt wurden. Unter dem Ziel stehen eine Reihe von Aktionen (eingerückt durch das Tabulator-Zeichen) die für dieses Ziel ausgeführt werden sollen. Eine Aktion kann entweder das Kompilieren oder Verlinken einer oder mehrerer Dateien sein (`g++ ...`), kann aber auch ein beliebiger andere Konsolenbefehle wie das Kopieren von Dateien (`copy ...`) oder eine einfache Statusmeldung (`@echo ...`) sein.

Im Moment reicht uns aber ein einfaches *Makefile* für unser erstes Beispielprogramm. Im Verlauf der nächsten Übungen werden wir kompliziertere Makefiles schreiben und sie werden auch kurz in der Vorlesung behandelt.

Erzeugen wir nun eine neue Datei mit Namen `Makefile` in unserem aktuellen Ordner (in dem auch die Datei `test.cpp` ist). In die Datei schreiben wir folgenden Text:

```
1  test: test.cpp
2      g++ -Wall test.cpp -o test -std=c++14
```

In der Kommandozeile geben wir nun den Befehl `make` (Linux) oder `mingw32-make` (Windows) ein und bestätigen mit Enter (Stellt vorher sicher, dass ihr euch im richtigen Ordner (`U:\data\gping\uebung-0`) befindet). In der Kommandozeile sollte nun entweder das auszuführende Kommando erscheinen (`g++ -Wall test.cpp -o test -std=c++14`), oder soetwas wie: `make: 'test' is up to date.`

Das *Make-Tool* erkennt auch ob es irgendwelche Änderungen in deinem Quellcode gab und kompiliert die Datei nur, wenn es auch Änderungen gab.

Mache eine kleine Änderung in deinem Quellcode und probiere den `make`-Befehl nochmal aus (oder lösche die Datei `test.exe`).

0.8 Zugang zu den Lehrmaterialien über PANDA

Die Materialien zu der Lehrveranstaltung werden online über das PANDA-System zur Verfügung gestellt. Das System erreichst du unter https://panda.uni-paderborn.de. Damit du in PANDA zu der Lehrveranstaltung registriert werden kannst, melde dich auf der PANDA-Seite mit deinem IMT-Login an.