

Einführung

- 1 Anmeldung, Verzeichnisse, Vorkenntnisse
- 2 Starten und Nutzung von MinGW
- 3 Programmieren kleinerer Beispiele
- 4 (Lösungen) später!
- 5 Laufzeitfehler
(Der Debugger GDB und Fehlersuche)
6. Ausblick

1. Anmeldung und Verzeichnisse

Sie sollten über ein Nutzerkennzeichen des IT- und Medienzentrums (ITMZ) der Universität Rostock verfügen und sich möglichst damit einloggen.

Aufbau der Nutzerkennzeichens:

(1.Buchstabe Vorname+1.Buchstabe Nachname+3 oder 4 Ziffern)

Beispiel: bk004 – Bernd Karstens

Über den Laufwerksbuchstaben R: steht Ihnen ein individueller Arbeitsbereich unabhängig vom genutzten Rechner zur Verfügung.

Legen Sie dort für den Modul Imperative Programmierung ein Verzeichnis an!

Beispiel R:/Imp2019 (Hinweis: Vermeiden Sie Leerzeichen und Umlaute!)

Falls Sie keinen individuellen Arbeitsplatzrechner im Labor heute zur Verfügung haben, aber einen eigenen Rechner mitgebracht haben.

Wählen Sie in einem Browserfenster <https://cloud.uni-rostock.de/uniComp/> und melden sich dort mit Ihren Nutzerkennzeichen des ITMZ an.

Der Laufwerksbuchstabe R: und der individuelle Arbeitsbereich steht Ihnen, wie oben zur Verfügung.

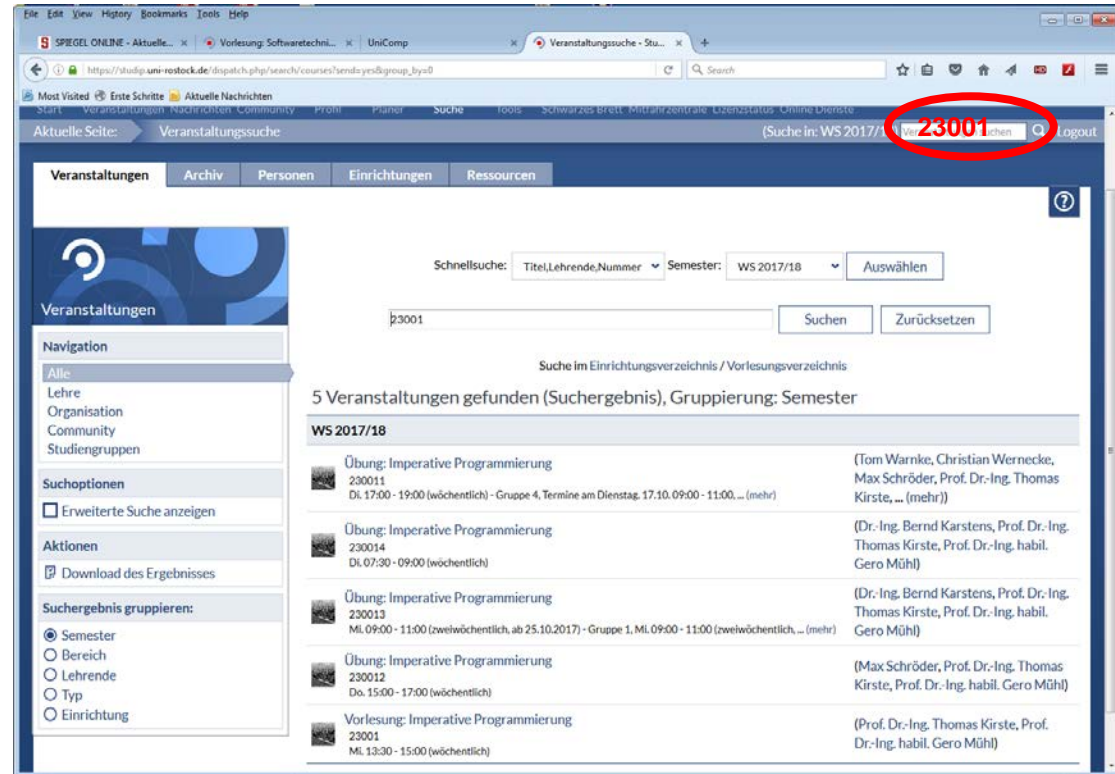
1. Übungsmaterialien im StudIP

Informationen zur Übung Imperative Programmierung finden Sie unter der Webadresse <https://studip.uni-rostock.de/>

Anmeldung mit ITMZ-Kürzel
Suche nach Veranstaltung
230013 Übung Win

(23001 –Vorlesung)

Entsprechende Veranstaltung
wählen und beitreten.

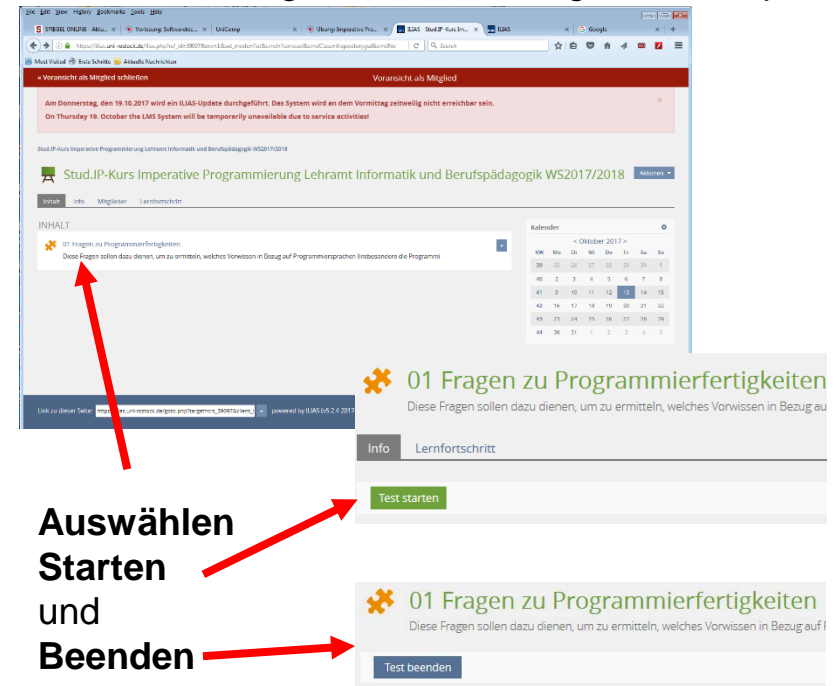
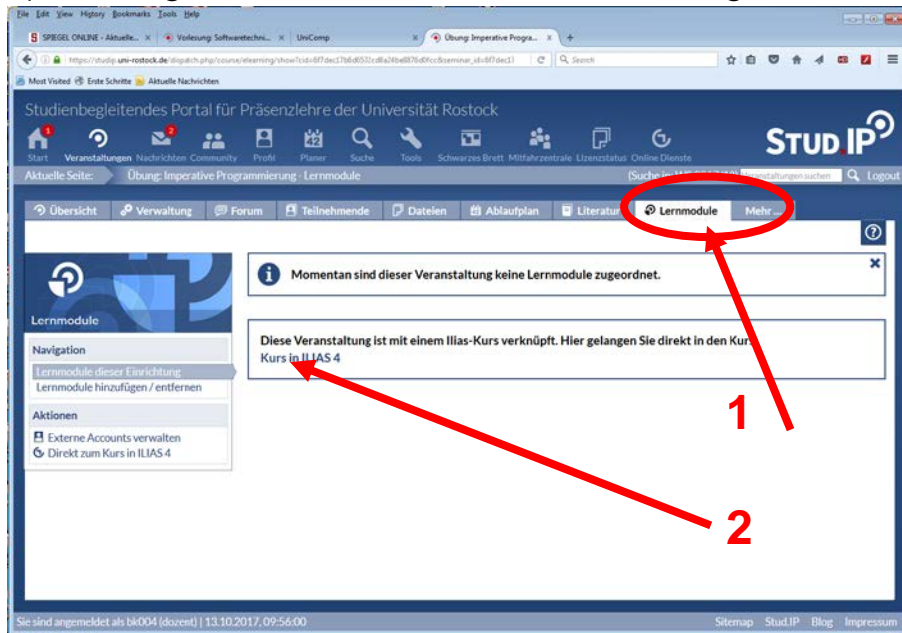


1. Fragebogen zu Vorkenntnissen (ca. 10 - 15 min)

Ich möchte eine wenig über ihre Vorkenntnisse über die Programmierung erfahren, um mich in nachfolgenden Übungen besser darauf einstellen zu können.

Ich habe einen interaktiven Fragebogen mit Fragen erstellt, den Sie Bitte jetzt ausfüllen. (Wer die Folien vor der Übung heruntergeladen hat, Bitte nicht vorher beantworten)

(Warnung 100% zu erreichen, ist kaum möglich oder Sie sind ein außergewöhnlicher Programmierexperte!)

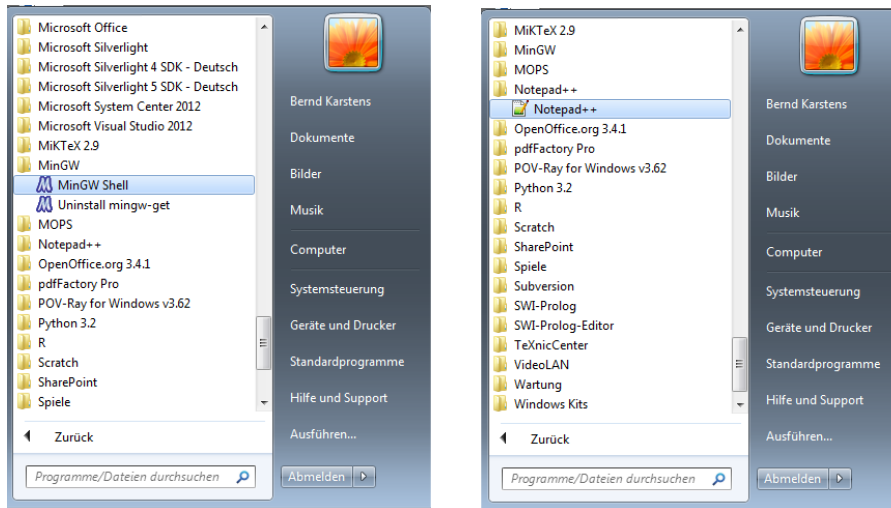


2. MinGW

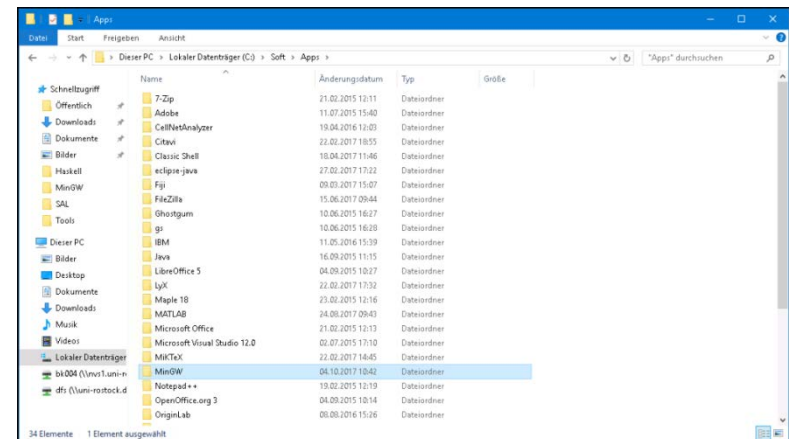
MinGW - Minimalist GNU for Windows

Compiler und Tools zu den GNU-Compilern (GNU General Public License) Im Labor 310 des ITMZ ist die eine Version installiert und soll im Modul Imperative Programmierung genutzt werden.

Als Editor wollen wir einen einfachen Texteditor nutzen: Notepad++

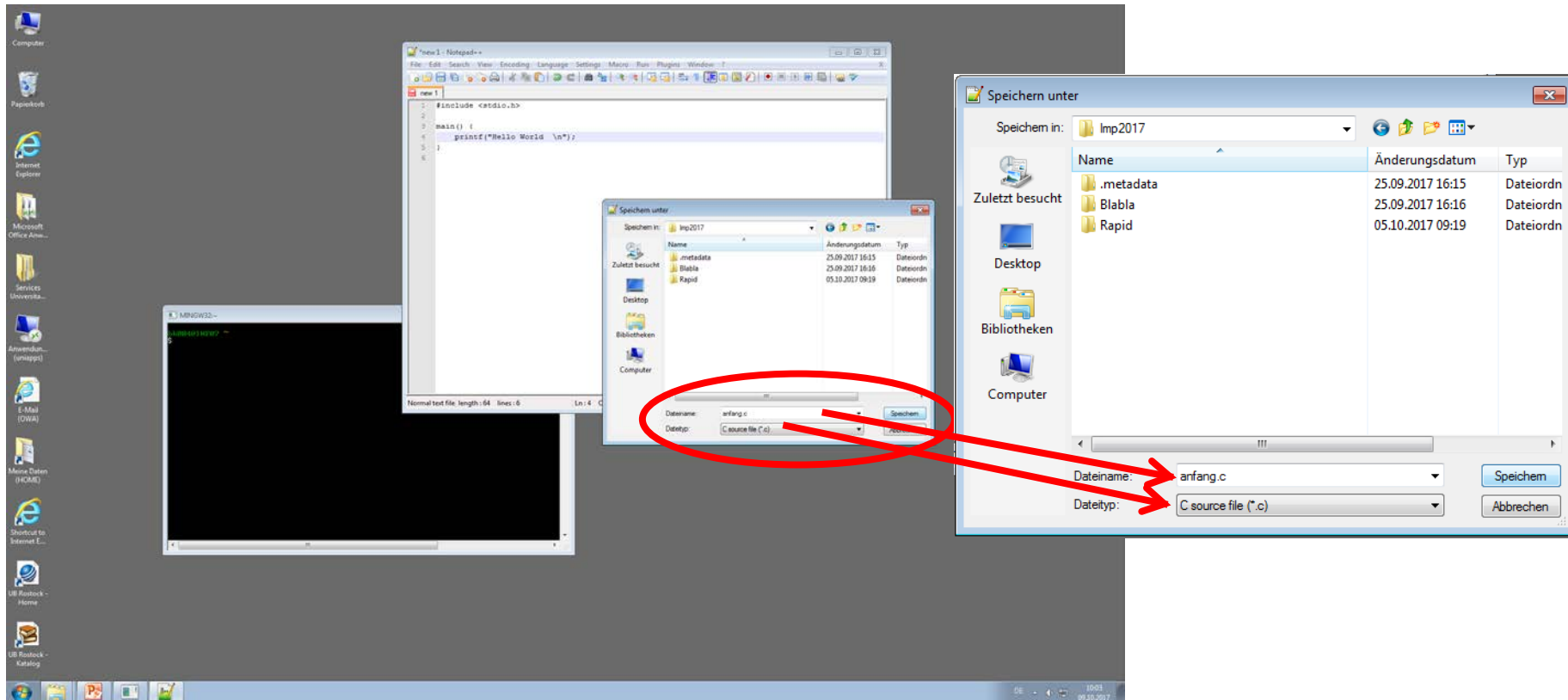


Die MinGW-Shell auf uniconp ist im Menü zu finden, Notepad++ auf C:\Soft\Apps



2. Arbeit im Labor

1. Nach Start von MinGW Shell,
2. Start von Notepad++
3. Eingabe eines Quelltextes in Notepad++ und sichern als anfang.c



2. Arbeit im Labor

Geben sie in Notepad ++ folgendes Programm ein:

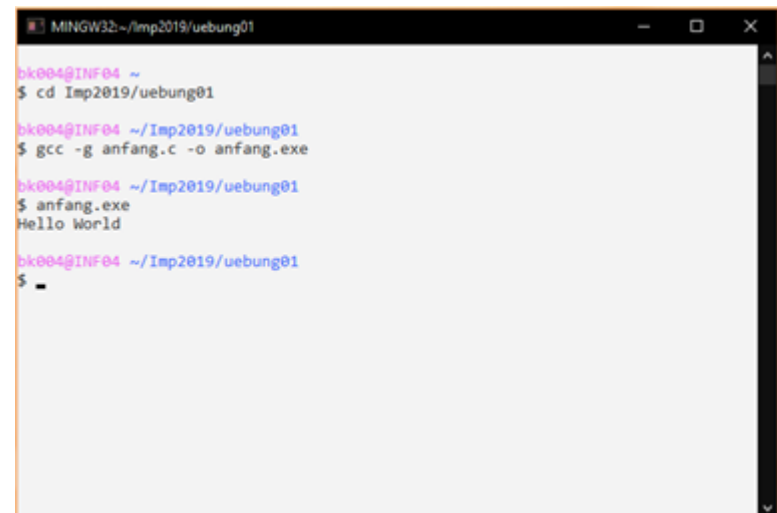
```
#include <stdio.h>
main() {
    printf("Hello World \n");
}
```

Speichern Sie dieses in Ihrem Verzeichnis (Imp2019) unter dem Namen anfang.c

Anschließend geben Sie im MinGW Shell - Fenster ein:

```
cd Imp2019
gcc -g anfang.c -o anfang.exe
anfang.exe
```

(wechsele ins Verzeichnis Imp2019 und führe dort C-Compiler aus und starte das Programm anfang.exe)

A screenshot of a MinGW32 shell window titled "MINGW32:~/imp2019/uebung01". The window shows a series of commands and their outputs. The user is at the prompt "bk004@INF04 ~". They enter "\$ cd Imp2019/uebung01", then "\$ gcc -g anfang.c -o anfang.exe", and finally "\$ anfang.exe". The output of the last command is "Hello World". The prompt then changes to "\$ -".

```
MINGW32:~/imp2019/uebung01
bk004@INF04 ~
$ cd Imp2019/uebung01
bk004@INF04 ~/Imp2019/uebung01
$ gcc -g anfang.c -o anfang.exe
bk004@INF04 ~/Imp2019/uebung01
$ anfang.exe
Hello World
bk004@INF04 ~/Imp2019/uebung01
$ -
```

2. Arbeit im Labor

Programm:

```
#include <stdio.h>
main() {
    printf("Hello World \n");
}
```

Falls ihr Programm fehlerfrei abgearbeitet wurde, herzlichen Glückwunsch, sie können C programmieren.

Versuchen sie sich durch Fehler (Weglassen von einzelnen Zeichen (Semikolon, Klammern, Anführungszeichen) und Groß- und Kleinschreibungen von Buchstaben im Programm sich mit Fehlerausschriften vertraut zu machen.

2. Warum ist folgendes Programm fehlerhaft

fehlerhaft	korrekt
<pre>#Include <Std.h> Main() { Printf("Hello World \n"); }</pre>	<pre>#include <stdio.h> main() { printf("Hello World \n"); }</pre>

anfang.c:1:2 error: invalid preprocessing directive #Include

anfang.c:1:17 fatal error: Std.h : No such file or directory

C:\Temp\bk004\ccSJUrA.o: In function 'Main':

R:\Imp2017/anfang.c: 3: undefined reference to '_Printf'

C:/soft/apps/mingw/...(Zeichen gelöscht)..undefined refernce to _WinMain@16

C:\Temp\bk004\ccSJUrA.o: In function 'main':

R:\Imp2017/anfang.c: 3: undefined reference to '_Printf'

Auf Groß- und Kleinschreibungen achten!

Es gibt Bezeichnungen (Schlüsselworte), die zu beachten sind.

2. Kommandos zur Arbeit mit der MinGW Shell

Change Directory - cd - Verzeichniswechsel

`cd Imp2019`

List - ls - Liste Inhalt des Verzeichnisses

`ls`

`ls -lasi`

Print Working Directory pwd - Zeige den aktuellen Pfad

`pwd`

Gnu C-Compiler aufrufen – gcc

Programm mit Namen „anfang.exe“ bilden aus C-Quelle „anfang.c“

`gcc -g anfang.c -o anfang.exe`

`gcc -c anfang.c //Übersetze .c in .o, Resultat anfang.o`

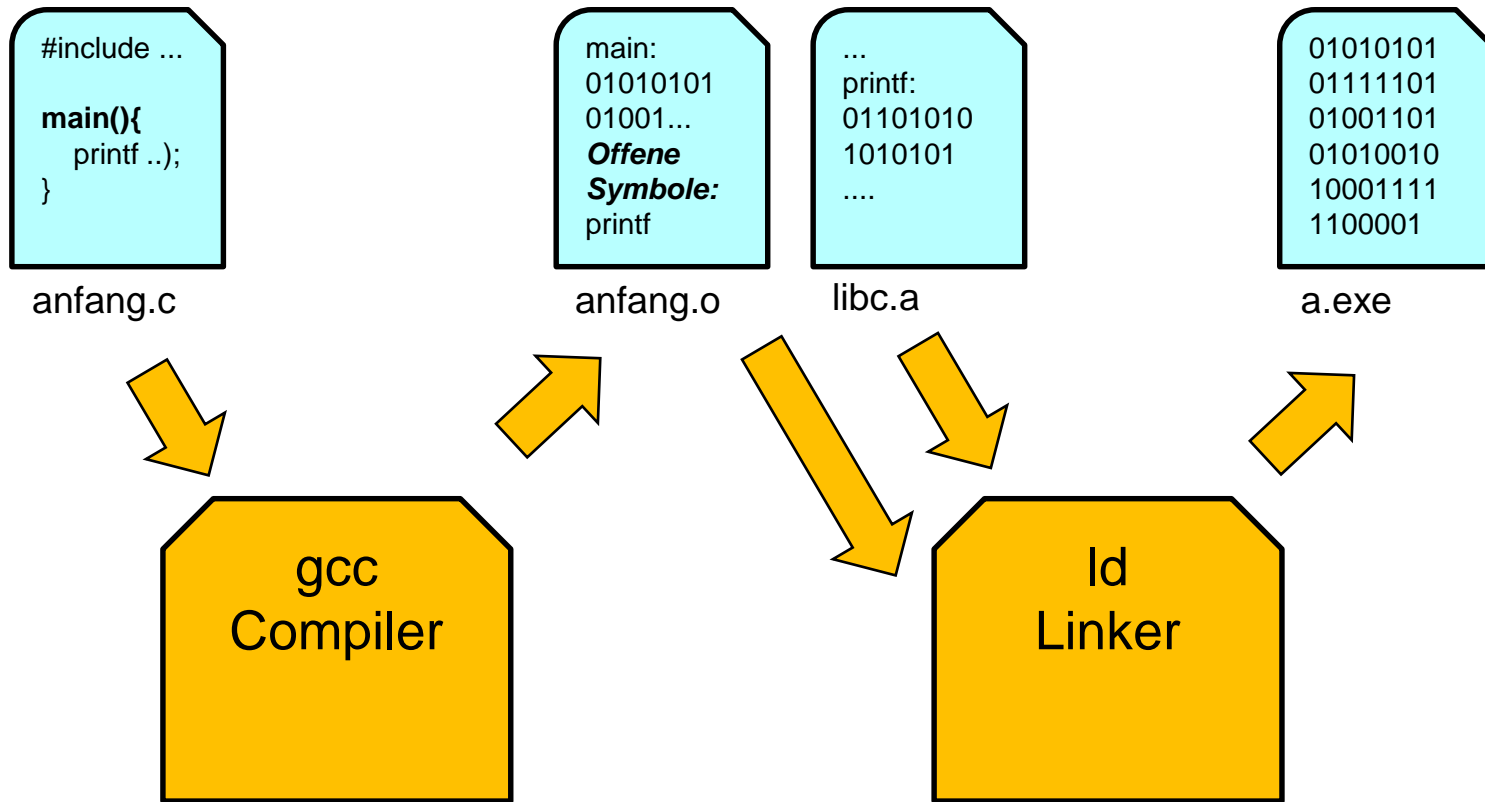
`gcc -g anfang.c //Übersetze .c in .o mit Debuginfo`

`gcc -o anfang.exe anfang.o // Linke ausführbares Programm`

`gcc -v // Zeige Version des Gnu C-Compilers`

`gcc anfang.c //übersetze Programm und bilde a.exe`

2. Übersetzung eines Programms



Aus einem Quelltext (erstellt durch den Programmierer) wird eine Objektdatetei für das jeweilige Betriebssystem gebildet und mit Hilfe von vordefinierten Programmbausteinen zu einem abarbeitbarem Programm zusammengefügt.

3. Aufgaben

1. Geben Sie die Zeichenkette Hallo Welt im nachfolgenden Schema aus!
Nutzen Sie Formatsteueranweisungen \n und \t !

```
H       a       l       l       o
W
e
l
t
```

2. Wie können die Zahlen 9 , -9 und 16.25 linksbündig, rechtsbündig mit Mindestanzahlen von Stellen ausgegeben werden?

3. (Erster Blick in die Zukunft: Ablaufkonstrukte (Sequenz und Schleifen))
Berechne die Summe und das Maximum von 3 Zahlen und gebe es aus.
(Wir wollen die Zahlen 9, 3 und 7 nutzen, später beliebige 3 Zahlen)

4. (C – Experten):
Gebe die Summe und das Maximum von 5 Zahlen aus,
lasse beliebige Zahlen zur Eingabe zu.)

3. Hinweise

Formatsteuerzeichen

<code>\n</code>	Newline – Neue Zeile
<code>\t</code>	Tabulator – nächste Tabposition (mind 8 Leerzeichen)
<code>\“</code>	Ausgabe Hochkomma
<code>\\</code>	Ausgabe \
<code>\0</code>	Später noch wichtig Zeichenkettenendezeichen

3. Hinweise

Formatierungen für Zahlen in printf

Lückentextausgabe:

```
printf("Die Zahlen %d und %f \n", 9, 16.25);
```

%d oder %i	Für Integerwerte
%X %x	Für Ausgabe in Hexadezimalsystem
%u	Für Integerwerte ohne Vorzeichen
%f	Floatwerte
%e	Floatwerte in Exponentialdarstellung
%c	Ausgabe von Buchstaben
%s	Ausgabe von Zeichenketten

3. Hinweise

Formatierungen links-und rechtsbündig, Mindestanzahl von Stellen

Durch Zeichen zwischen den % und der Formatwahl kann die Anzahl und Ausrichtung der Ausgabe gesteuert werden.

Linksbündig „-“(Minus) , Rechtsbündig(Standard ohne Formatierungszeichen)

Ganzzahl – Anzahl der Zeichen in der Ausgabe

Folgt der Ganzzahl ein Punkt und eine weitere Zahl, so sind es Mindestziffern

Folgt dem % ein „+“, so wird das Vorzeichen auf jeden Fall mit ausgegeben

%10d	Reserviere 10 Zeichen Platz für Ganzzahl, gebe die Zahl rechtsbündig aus.
%-10d	Reserviere 10 Zeichen Platz für Ganzzahl, gebe die Zahl linksbündig aus.
%10.4d	Reserviere 10 Zeichen Platz für Ganzzahl, gebe die Zahl mit mindestens 4 Ziffern (ggf führende Nullen) aus.
%10f	Reserviere 10 Zeichen Platz für Gleitpunktzahl, gebe die Zahl rechtsbündig aus
%-10f	Reserviere 10 Zeichen Platz für Gleitpunktzahl, gebe die Zahl linksbündig aus
%10.3f	Reserviere 10 Zeichen Platz für Gleitpunktzahl, gebe die Zahl mit mindestens 3 Nachkommaziffern aus
%+d	Gebe das Vorzeichen mit aus

5. Laufzeitfehler und Debugging

Folgendes Programm: (anfangDivision.c) hat einen Fehler!

Wir dividieren durch 0!

```
#include <stdio.h>
main() {
    int a;
    int b;
    a = 4;
    b = 2;
    printf("Hello World \n");
    printf("a =%d, b= %d, a/b=%d \n", a, b, a/b);
    b = 0;
    printf("a =%d, b= %d, a/b=%d \n", a, b, a/b);
}
```

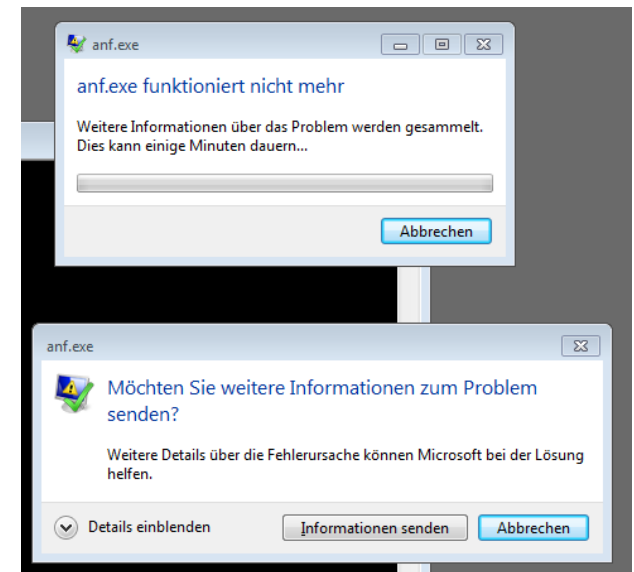

5. Laufzeitfehler und Debugging

Folgendes Programm: (anfangDivision.c) hat einen Fehler!

Wir dividieren durch 0!

```
#include <stdio.h>
main() {
    int a;
    int b;
    a = 4;
    b = 2;
    printf("Hello World \n");
    printf("a =%d, b= %d, a/b=%d \n", a, b, a/b);
    b = 0;
    printf("a =%d, b= %d, a/b=%d \n", a, b, a/b);
}
```

```
bk004@INF08 ~/Imp2017
$ gcc -g anfangDivision.c -o anf.exe
bk004@INF08 ~/Imp2017
$ anf.exe
Hello World
a =4, b= 2, a/b=2
```



5. Laufzeitfehler und Debugging

Das Programm `anf.c` wird übersetzt mit:

`gcc -g anf.exe` ----- Anschließend wird der Debugger `gdb` gerufen:

`gdb anf.exe`

(gdb) `b main` Breakpoint Setze Haltepunkt auf Funktion `main`

(gdb) `r` run lasse das Programm laufen

(gdb) `l` list Liste den Kontext in der aktuellen Umgebung auf

(gdb) `l 10` list Liste Programm ab Quelltextzeile 10

(gdb) `bt` backtrace Zeige die Aufrufstruktur bis zur aktuellen Stelle

(gdb) `p x` print variable Zeige den Inhalt von Variable `x`

(gdb) `b main` Breakpoint Setze Haltepunkt auf Funktion `main`

(gdb) `b 9` Breakpoint Setze Haltepunkt in Zeile 9

(gdb) `clear 9` Breakpoint Lösche Haltepunkt in Zeile 9

(gdb) `n` next Nächste Zeile abarbeiten (geht in Ergebnis der Funktion)

(gdb) `s` step Führe nächsten Befehl aus (geht in Funktionen)

(gdb) `u` until Führe alle Befehle in der Schleifen aus

(gdb) `f` finish Führe bis Ende der Funktion Befehle aus.

(gdb) `q` quit Beende Debugger

5. Laufzeitfehler und Debugging

Das Programm an.c wird übersetzt mit:

gcc -g anf.exe ----- Anschließend wird der Debugger gdb gerufen:

gdb anf.exe

(gdb) b main

(gdb) r

(gdb) l

(gdb) l 10

(gdb) bt

(gdb) p x

(gdb) b main

(gdb) b 9

(gdb) clear 9

(gdb) n

(gdb) s

(gdb) u

(gdb) f

(gdb) q

```
MINGW32:~/Imp2019/uebung01
bk004@INF04 ~/Imp2019/uebung01
$ gdb anfangDivision.exe
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from r:\Imp2019\uebung01\anfangDivision.exe...done.
(gdb) b main
Breakpoint 1 at 0x4013ce: file anfangDivision.c, line 5.
(gdb) r
Starting program: r:\Imp2019\uebung01\anfangDivision.exe
[New Thread 7136.0x7d0]
[New Thread 7136.0x1e50]
Breakpoint 1, main () at anfangDivision.c:5
5       a = 4;
(gdb) l
1       #include <stdio.h>
2       main() {
3           int a;
4           int b;
5           a = 4;
6           b = 2;
7           printf("Hello World \n");
8           printf("a =%d, b= %d, a/b=%d \n", a, b, a/b);
9           b = 0;
10          printf("a =%d, b= %d, a/b=%d \n", a, b, a/b);
(gdb) s
6           b = 2;
(gdb) s
7           printf("Hello World \n");
(gdb) s
Hello World
8           printf("a =%d, b= %d, a/b=%d \n", a, b, a/b);
(gdb) s
a =4, b= 2, a/b=2
9           b = 0;
(gdb) s
10          printf("a =%d, b= %d, a/b=%d \n", a, b, a/b);
(gdb) s
```

6. Ausblick

Nächste Woche:

weitere Beispiele entsprechend Vorlesung

schrittweise Einführung der Programmierkonstrukte in C

Arbeit mit Subversion (nicht vergessen: Eintrag in eine Hausaufgabengruppe!)