0.3 Programmieren in C/Werkzeuge

0.3.1 Optionen für den GCC

a) Standards

- C89-Standard beachten:

Terminal
```
schueler@debian964:~$ gcc -std=c89 -pedantic meinprog.c
```

- C99-Standard beachten:

Terminal
```
schueler@debian964:~$ gcc -std=c99 -pedantic meinprog.c
```

- C11-Standard beachten:

Terminal
```
schueler@debian964:~$ gcc -std=c11 -pedantic meinprog.c
```

- C23-Standard beachten:

Terminal
```
schueler@debian964:~$ gcc -std=c23 -pedantic meinprog.c
```

b) Gezielter Aufruf einzelner Verarbeitungs-Stufen

- Alle Stufen:

Terminal
```
schueler@debian964:~$ gcc meinprog.c
```

Die Ausgabe (in Maschinensprache) findet sich anschließend in `a.out`.

- Alle Stufen, aber passender Dateiname:

Terminal
```
schueler@debian964:~$ gcc -o meinprog.exe meinprog.c
```

Mit der zusätzlichen Option `-o` wird der Name der Ausgabedatei als `meinprog.exe` festgelegt.

- Nur Präprozessor (Stufe 1) benutzen:

Terminal
```
schueler@debian964:~$ gcc -E meinprog.c > meinprog_roh.c
```

- Nur Präprozessor (Stufe 1) und Compiler (Stufe 2):

Terminal
```
schueler@debian964:~$ gcc -S meinprog.c
```

Die Ausgabe (in Assembler-Code!) findet sich anschließend in `meinprog.s`.

- Nur Präprozessor (St.1), Compiler (St.2) und Assembler (St.3) benutzen, aber noch nicht die C-Bibliothek einbinden:

Terminal
```
schueler@debian964:~$ gcc -c meinprog.c
```

Die Ausgabe (in Maschinensprache) findet sich anschließend in `meinprog.o`.

c) Zusätzliche Warnungen

- Warnung bei allen Fehlern, auch bei Dialekt-Fehlern:

Terminal
```
schueler@debian964:~$ gcc -Wall meinprog.c
```

- Warnung bei ungewöhnlichen Konstruktionen:

Terminal
```
schueler@debian964:~$ gcc -Wextra meinprog.c
```

- Warnung bei impliziten Typkonvertierungen:

Terminal
```
schueler@debian964:~$ gcc -Wconversion meinprog.c
```

d) Spezialitäten

- Debugger-fähigen Code erzeugen (z.B. für den graphischen Debugger `ddd`):

Terminal
```
schueler@debian964:~$ gcc -g meinprog.c
```

- Profiler-fähigen Code erzeugen:

Terminal
```
schueler@debian964:~$ gcc -p meinprog.c
```

0.3.2 Fehlermeldungen des GCC und Fehlerursachen

Fehlermeldung	Mögliche Ursache	Aktion
`test1.c:6:4: error: expected ';' before 'return'`	Semikolon vergessen, Syntaxfehler	
`test2.c:5:4: error: stray '\303' in program`	Variablen- oder Funktionsnamen mit Umlaut gewählt	
`test3.c:(.text+0x11): undefined reference to 'PrInTfLaLaLa'`	Funktionsname falsch geschrieben	
`test4.c:3:4: warning: implicit declaration of function 'printf'`	Prototyp fehlt, eventuell Headerdatei nicht eingebunden	
`test5.c:5:19: error: 'i' undeclared (first use in this function)`	Variable `i` wurde nicht vereinbart	
`test6.c:7:1: error: expected declaration or statement at end of input`	Irgendwo im Programm ein `}` vergessen oder versehentlich `{` statt `}` geschrieben	
`test7.c:5:16: error: expected identifier or '(' before 'do'`	Schlüsselwort `do` als Variablenname benutzt	Anderen Variablennamen wählen
`test8.c:6:4: warning: 'return' with a value, in function returning void`	`void`-Funktion versucht einen Wert zurückzugeben	`return` ohne Argument verwenden
`test9.c:6:1: warning: control reaches end of non-void function`	Wertrückgabe vergessen	`return`-Anweisung einfügen
`test10.c:5:11: warning: missing terminating " character`	Schließendes Anführungszeichen vergessen, oder String umfasst mehrere Zeilen	Bei mehreren Zeilen: String aufteilen
`test11.c:6:8: warning: multi-character character constant`	Beim Zeichensatz UTF-8 passt ein Umlaut nicht mehr in ein `char`!	Auf Umlaute verzichten
`test12.c:5:11: warning: initialization makes integer from pointer without a cast`	Ein String steht da, wo eine Zahl oder Einzelzeichen stehen sollte (z.B. Anführungszeichen statt Hochkomma gewählt).	
`test13.c:6:5: warning: assignment makes integer from pointer without a cast`	Ein String steht da, wo eine Zahl oder Einzelzeichen stehen sollte (z.B. Anführungszeichen statt Hochkomma gewählt).	

Fortsetzung auf der nächsten Seite

Fortsetzung von der vorherigen Seite		
Fehlermeldung	Mögliche Ursache	Aktion
`test14.c:6:6: error: incompatible types when assigning to type 'char[800]' from type 'char *'`	Man kann an ein Array nichts zuweisen.	Bei Strings `strcpy()` verwenden.
`test15.c:5:4: warning: excess elements in array initializer`	In der Initialisierung sind mehr Elemente angegeben als die Array-Größe zulässt.	Array-Größe anpassen
`test16.c:5:14: warning: initializer-string for array of chars is too long`	Initialisierungs-String hat mehr Elemente als die Array-Größe zulässt.	Array-Größe anpassen
`test17.c:1:9: error: expected '=', ',', ';', 'asm' or '__attribute__' before '<' token`	Raute-Zeichen vor `include` fehlt.	
`test18.c:5:4: error: expected declaration specifiers before 'printf'`	Geschweifte Klammern bei `main()` fehlen.	
`test19.c:1:22: fatal error: gibsnich.h: Datei oder Verzeichnis nicht gefunden`	Einzubindende Datei fehlt.	Bei lokaler Datei Anführungszeichen statt spitzer Klammern verwenden.
`test20.c:5:24: warning: trigraph ??! converted to |`	`??!` ist eine 3-Zeichen-Ersatzschreibweise für `|` (ein Trigraph).	String aufteilen: `"Hello, W.?" "?!"`
`test21.c:5:12: error: invalid digit "9" in octal constant`	Jede Zahl mit 0 am Anfang wird oktal interpretiert, und 8 und 9 sind *keine* Oktalziffern.	Führende 0 entfernen