GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

AUFSTIEG DURCH BILDUNG
OFFENE HOCHSCHULEN

ESF
Europäischer Sozialfonds
für Deutschland

EUROPÄISCHE UNION

Programmieren mit C

Jörg Abke

Hochschule Aschaffenburg
University of Applied Sciences

Einleitung

Programmieren ist die Erstellung von Computerprogrammen mit Hilfe einer Programmiersprache. Dazu ist es notwendig, die Programmiersprache kennenzulernen, aber sich auch ein grundlegendes Verständnis von Werkzeugen zur Programmentwicklung anzueignen.

Computer und computergesteuerte Systeme finden sich heute nicht nur auf jedem Büro-Arbeitsplatz, sondern auch in nahezu allen technischen Geräten. Moderne Kraftfahrzeuge fahren mit ca. 80 solcher Teilsysteme auf der Straße. Jedes Smartphone umfasst mindestens einen kleinen Computer. Technische Anlagen und Systeme sind heutzutage immer mit programmierbaren Computern ausgestattet, auch wenn diese nicht als Computersystem mit Tastatur und Display sofort erkennbar ist.

Daher ist es für jeden Ingenieur notwendig, Computer- und Programmierkenntnisse zu besitzen.

In diesem Modul werden Sie zum einen eine Programmiersprache C kennenlernen. Zum anderen werden Sie aber die Sprache auch zur Lösung von Problemen und Aufgabenstellungen mittels einfacher Algorithmen praktisch einsetzen.

Neben der Theorie der Sprache C wird die Herangehensweise zur Lösung von Problemstellungen mittels Computer und der Sprache C praktisch eingeï¿œbt.

Die Programmiersprache C ist eine sehr elementare Sprache, die für nahezu jedes Computersystem einsetzbar ist. Zusätzlich bildet diese Sprache die Grundlage für zahlreiche weitere Programmiersprachen, wie C++ oder Java, aber auch Sprachen für Mathematiksysteme, wie z.B. Matlab.

Dieser Lehrbrief erläutert die Grundlagen der Programmierung in der Sprache C. Er enthält zudem Übungsaufgaben. Zusäzlich werden diese Aufgaben auf der Lernplattform zu finden sein. Dort ist auch ein Diskussionsforum für Ihre Fragen und Tipps zur Lösung bereitgestellt. Lösungen zu den Aufgaben werden jeweils nach gewissen Stichtagen öffentlich gemacht, um die eigenständige Bearbeitung zu fördern und Ihre eigene Mitarbeit zu stimulieren.

Zur Vorbereitung auf die Präsenzveranstaltung werden zusätzlich einige Fragen- und Aufgaben bereitgestellt. Diese sollen Ihnen und dem Dozenten den aktuellen Wissensstand verdeutlichen und die Kenntnisse praktisch vertiefen.

Lernziele

Dieses Modul vermittelt die Grundlagen der Programmiersprache C sowie die Herangehensweise bei der Programmierung von kleinen Problemstellungen.

Nach dem Studium dieses Moduls sollten Sie:

- die Elemente der Sprache C verstehen und verwenden können
- C-Programme mit Hilfe einer Entwicklungsumgebung in Programme für Ihren Computer umsetzen können.
- kurze Programme in der Sprache C selbständig programmieren können
- Programmsequenzen in der Sprache C verstehen und deren Sinn erläutern können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

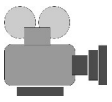
1.3.3 Programmausgabe mittels `printf`-Funktion

Im vorangegangenen Abschnitt ist bereits in dem Programm 1.1 die Ausgabefunktion `printf` verwendet worden.

Der Befehl `printf("Ausgabetext\n");` sorgt dafür, dass während des Ablaufs des Programms auf dem Bildschirm der Text `Ausgabetext` erscheint. Beim Programmaufruf aus der Entwicklungsumgebung "Eclipse" heraus erfolgt die Ausgabe in der Console von "Eclipse". Eine nachfolgende Ausgabe wird in einer neuen Zeile begonnen. Hierfür sorgt der Steuerbefehl `\n`, der im weiteren Verlauf dieses Abschnitts noch erläutert wird.

Beim Start des ausführbaren Programms auf einer Kommandozeile, wie cmd unter Windows oder xterm unter Unix, erfolgt die Ausgabe, auf der jeweiligen Kommandozeile.

Damit Sie eine Beispieldatei nicht jedesmal abtippen müssen, finden Sie alle Beispieldateien auf der Lernplattform zur Verfügung gestellt. Wie Sie eine Beispieldatei dann in Ihre Entwicklungsumgebung "Eclipse" importieren können ist am Programmbeispiel 1.2 in diesem Video 1.2 gezeigt.



Beachten Sie bitte, dass immer, wenn Sie ein neues C-Programm mit "Eclipse" erstellen oder sich ansehen wollen, ein neues C-Projekt in "Eclipse" angelegt werden muss. Mit Hilfe der Import-Funktion können Sie dann die von der Lernplattform heruntergeladene Datei in Ihr neu angelegtes Projekt importieren.

Programmbeispiel 1.2: mehrere_printf_ausgaben.c

```c
#include <stdio.h>

int main()
{
  printf("Programmieren mit C\n");
  printf("im Studiengang BBeng\n\n");
  printf("Eine Zeile Leerraum durch zwei
     Zeilenumbrueche\n");
  printf("Ein \t Tabulatorabstand ");
  printf("und kein Zeilenumbruch");

  return(0);
}
```

Programmausgabe

```
Programmieren mit C
im Studiengang BBeng

Eine Zeile Leerraum durch zwei Zeilenumbrueche
Ein   Tabulatorabstand und kein Zeilenumbruch
```

An dem Beispielprogramm 1.2 erkennen Sie,

- dass Textausgaben immer mit zwei Anführungsstrichen " beginnen und immer mit zwei Anführungsstrichen " enden. Diese Anführungsstriche markieren für den Computer eine Stringkonstante (vgl. 1.4.4)
- dass Ausgaben verteilt auf mehrere Aufrufe der `printf`-Funktion möglich sind.
- dass neben Textausgaben auch Steuerzeichen enthalten sind. Alle Steuerbefehle mit Steuerzeichen beginnen mit einem Backslash \.

Die Steuerzeichen setzen den Cursor bei der Ausgabe auf eine bestimmte Position oder veranlassen eine besondere Ausgabe, z.B. einen Piepston. Alle wichtigen Steuerbefehle sind in Tabelle 1.1 aufgelistet. Die beiden am häufigsten verwendeten sind `\n` für eine neue Zeile und `\t` für einen horizontalen Tabulator. Der Tabulator wird inbesondere für tabellarische Ausgaben verwendet.

Tabelle 1.1: Tabelle der Steuerbefehle für die Ausgabe mit printf

Steuer-zeichen	Ausgabe
`\a`	Klingelton
`\b`	Backspace (ein Zeichen zurück)
`\f`	Seitenvorschub (formfeed)
`\n`	Neue Zeile (newline)
`\r`	Wagenrücklauf (carriage return) (an den Anfang d. akt. Zeile)
`\t`	horizontaler Tabulator (i.a. 8 Zeichen weiterpositionieren)
`\v`	Vertikaler Tabulatorvorschub

Neben den Steuerbefehlen sind Anweisungen zur Ausgabe von bestimmten Zeichen notwendig. Beispielsweise müssen Sie für die Ausgabe des Zeichens \ durch `printf` zwei Backslash \\ angeben. Der Grund dafür ist die spezielle Bedeutung des Backslashes für Steuerzeichen. Andere Zeichen besitzen in der Sprache C ebenfalls besondere Bedeutungen und müssen daher für die Ausgabe mit einem Backslash vorangestellt angegeben werden. Eine Übersicht der speziellen Zeichen ist in Tabelle 1.2 dargestellt. Die Steuerbefehle und die Befehle für die speziellen Zeichen werden auch als *Escape-Sequenzen* bezeichnet.

Die Zuordnung zwischen einem Zeichen und einer Oktal- oder Hexadezimalzahl können Sie den ASCII-Tabellen A.1 und A.2 in Anhang A entnehmen. Die Abkürzung ASCII steht für *American Standard Code for Information*

Tabelle 1.2: Tabelle der speziellen Zeichen für die Ausgabe mit printf

Spezial-zeichen	Ausgabe
\"	Anführungszeichen oben "
\'	halbes Anführungszeichen bzw. Hochkomma '
\?	Fragezeichen ?
\\	Backslash \
\ooo	Zeichen mit der dreistelligen Oktalzahl ooo, z.B. \101 ergibt A
\xhh	Zeichen mit der zweistelligen Hexadezimalzahl hh z.B. \x41 ergibt A

Interchange. Dieser ASCII-Code ordnet einzelnen Zeichen, egal ob druckbare Buchstaben oder Steuerzeichen, einen Zahlwert im Bereich von 0 bis 127 dezimal zu. Dieser Wertebereich kann mit 7 Bit dargestellt werden. Es fällt dabei auf, dass z.B. deutsche Umlaute in der Tabelle fehlen. Neben dem englischen Alphabet sind nur wenige griechische Buchstaben, jedoch keine japanischen, chinesischen oder arabischen Zeichen enthalten. Diese Einschränkungen sind in heutigen Programmiersprachen und Textverarbeitungssystemen mit der UTF-Codierung behoben. UTF steht für *Universal Character Set Transformation Format*. Zur Verdeutlichung der speziellen Zeichen kann Ihnen das Beispielprogramm 1.3 dienen.

Programmbeispiel 1.3: spezialzeichen_printf_ausgaben.c

```
#include <stdio.h>

int main()
{
  printf("Ausgabe \"in Anfuehrungszeichen\"\n");
  printf("Ausgabe \'in halben Anfuehrungszeichen\'\n");
  printf("\\ heisst Backslash\n");
  printf("Sie lernen gerade das \101 und \x4F der
     C-Programmierung\n");
  printf("Haben Sie den Satz lesen koennen\?");

  return(0);
}
```

Programmausgabe

```
Ausgabe "in Anfuehrungszeichen"
Ausgabe 'in halben Anfuehrungszeichen'
\ heisst Backslash
Sie lernen gerade das A und O der C-Programmierung
Haben Sie den Satz lesen koennen?
```

Weitere Details zum Thema Ausgabe mit der `printf`-Funktion werden Sie in Abschnitt 2.1.2 kennenlernen.

Neben der Ausgabe aus Programmen ist natürlich eine Eingabe möglich. Sie werden das Thema Eingabe im späteren Abschnitt 2.7.1 wieder aufgreifen.

1.3.4 Übungsaufgaben

1.1 **Einfache Zeilenausgabe**
Implementieren Sie in einem neuen C-Projekt ein Programm, um den Satz

```
''Richtig,     hier gibt es einen Zwischenraum''
```

und zwei Ausgabezeilen später:

```
''Nun eine Ausgabe mit einer Zeile Abstand''
```

in der Ausgabe erscheinen zu lassen.

1.2 **Formatierte Textausgabe**
Schreiben Sie ein Programm, das nachfolgendes Gedichtfragment wie gezeigt auf dem Bildschirm ausgibt. Für die Einrückung (horizontaler Tabulator) verwenden Sie bitte einen geeignete Befehl aus Tabelle 1.2

```
Das Lied von der Glocke

Fest gemauert in der Erden
Steht die Form, aus Lehm gebrannt.
    Heute muss die Glocke werden.
    Frisch Gesellen, seid zur Hand.
        Von der Stirne heiß
```

```
          Rinnen muss der Schweiß
              Soll das Werk den Meister loben,
              Doch der Segen kommt von oben.
[...]
Friedrich Schiller
```

1.4 Allgemeine Begriffe der Sprache C

Die allgemeinen Begriffe stellen die Basis für das Verständnis der weiteren Lehrmaterialien, Fragen und die praktischen Übungen dar.

In der Sprache C wird Groß- und Kleinschreibung unterschieden. Die Schreibweise in der Sprache C ist standardmäßig klein, wie Sie evtl. schon an den vorstehenden Beispielen bemerkt haben. Groß geschriebene Worte sind für die Verwendung von Bezeichnern für den Präprozessor vorgesehen (vgl. Abschnitt 6.2).

1.4.1 Schlüsselwörter

In der Programmiersprache C besitzen Schlüsselwörter festgelegte Bedeutungen. In Tabelle B.1 in Anhang B sind die Schlüsselwörter aufgelistet. Der Compiler versteht diese Schlüsselworte nur im vereinbarten Sinn. Somit können und dürfen Schlüsselwörter in keinem anderen Zusammenhang, z.B. als Bezeichner, verwendet werden (vgl. 1.4.2). Viele Schlüsselwörter und deren Bedeutung werden Sie im weiteren Verlauf des Lehrmoduls kennenlernen.

1.4.2 Bezeichner

Mit einem Bezeichner werden Objekte der Sprache C eindeutig identifiziert. Ein Bezeichner in der C-Programmierung darf

- **nicht** mit einer Ziffer beginnen
- **kein** Schlüsselwort sein (vgl. Tabelle B.1)
- **nicht** mit deutschen Umlauten oder ß versehen sein, da diese nicht im ASCII-Zeichensatz vorkommen (vgl. Anhang A).

- aus einer Folge von Buchstaben (aus dem ASCII-Zeichensatz), Ziffern (mit obiger Einschränkung) und dem Unterstrich-Zeichen _ bestehen.
- beliebig lang sein. Wie viele Stellen jedoch zum Vergleich genommen werden, d.h. signifikant sind, ist durch den verwendeten C-Compiler festgelegt.

Beispiel 1.4.1
Gültige Bezeichner sind:

programmiersprache, zahl, zaehler, __teiler, h_ab

Nicht gültige Bezeichner sind:

0815_variable, auto, schlüssel, groß, h-ab

1.4.3 Konstanten

Konstanten haben in der Sprache C einen einmal im Programm festgelegten Wert. Sie werden vom Compiler mit einem Datentyp versehen. Datentypen lernen Sie in Abschnitt 2 kennen.

Zahlkonstanten

Die Programmiersprache C unterscheidet Zahlen im Programm in unterschiedlichen Kategorien:

- Ganzzahlkonstanten
- Fließkommakonstanten
- Zeichenkonstanten

Ganzzahlkonstanten

In der Programmiersprache C können ganze Zahlen zu folgenden Stellenwertsystemen angegeben werden:

Dezimalzahlen werden mit den Ziffern $0, 1, \ldots, 9$ angegeben, wobei *nie* die 0, Null, als führende Ziffer verwendet wird. Es können $-$ und $+$ für negative bzw. positive Zahlen vorangestellt werden.

Oktalzahlen werden mit den Ziffern $0, 1, \ldots, 7$ angegeben. Dazu müssen Sie eine Oktalzahl mit einer 0 beginnen lassen, damit der Compiler versteht, dass die Ziffernfolge zur Basis 8, also oktal, zu verstehen ist.

Hexadezimalzahlen werden mit den Ziffern $0, 1, \ldots, 9$ und den Buchstaben a, b, c, d, e, f angegeben. Um dem Compiler eine Hexadezimalzahl erkennbar zu machen, müssen Sie `0x` voranstellen. Für Angaben mit Großbuchstaben A, B, C, D, E, F stellen Sie bitte `0X` der Ziffern- und Buchstabenfolge voran.

Beispiel 1.4.2

Zahlkonstanten im Dezimalsystem: 3, 95, 354, -12, +65
Oktalzahlkonstanten: 017, 05, 0815
Konstante Hexadezimalzahlen: 0x1f, 0x46, 0X7C6B, 0X18D

Ganzzahlen speichert C üblicherweise im Datentyp **int**, d.h. Integer. Um dem Compiler mitzuteilen, dass eine vorzeichenlose Zahl, also eine Zahl mit einem Wert 0 oder größer als 0, zu verstehen ist, wird ein `u` oder `U` für *unsigned* an die Ziffernfolge hinten angestellt. Damit verwendet der Compiler den Datentyp **unsigned int**.

Beispiel 1.4.3

Zahlkonstanten als **unsigned**: 0x45u, 07U, 1634u

Falls Sie große Zahlwerte angeben wollen, sollten Sie bitte beachten, dass Sie durch Anfügen von `l` oder `L` den Compiler dazu bewegen können, den Datentyp **long** bzw. **long int** , also Long Integer, für Ihre Zahl zu verwenden.

Beispiel 1.4.4

Zahlkonstanten als **long**: 1436323535L, 121456731

Fließkommakonstanten

Um die Zahlen aus der Menge der reellen Zahlen anzugeben, ist die Fließkommazahl in der Sprache C vereinbart. Grundsätzlich beachten Sie bitte, dass Sie statt eines Kommas immer die amerikanische Schreibweise mit Punkt verwenden. Daher wird im Englischen diese als *floating point number* bezeichnet. Es gibt mehrere Schreibweisen für Fließkommazahlen

$\pm ffff.fffffff$ Fließkommazahl mit Vor- und Nachkommastellen

$\pm fffff.ffffffff\mathbf{e}\pm eee$ Zahl mit Exponentenschreibweise. Der Exponent aus ganzen Zahlen *eee* hat dabei immer die Basis 10

$\pm fffff.ffffffff\mathbf{E}\pm eee$ wie oben als Exponentenschreibweise, es macht also keinen Unterschied, ob e oder E angegeben wird

Beispiel 1.4.5

Fließkommazahlen in C: 1.25, 412e−3, −56421.3245e+124

Standardmäßig wird der C-Compiler eine Fließkommakonstante mit dem Datentyp **double** in Verbindung bringen. Sie können jedoch durch Anhängen eines `f` oder `F` dem Compiler den einfach genauen Datentyp **float** nahelegen.

Zeichenkonstanten

Einzelne Zeichen werden in halben oder einfachen Anführungszeichen angegeben, auch einfache Hochkommata genannt. Als Zeichen kann jedes Zeichen der ASCII-Tabelle verwendet werden, wobei die Steuerzeichen durch spezielle Sequenzen anzugeben sind (vgl. Tabelle 1.1) Diese Angabe veranlasst den C-Compiler, den Datentyp **char** zum Abspeichern des Zeichens zu verwenden. Im Detail speichert der Computer das Zeichen als die entsprechende Zahl, die die ASCII-Codierung dem Zeichen zuordnet (vgl. Anhang A).

Beispiel 1.4.6

Zeichenkonstanten: `'A'`, `'\n'`, `'o'`, `'C'`

1.4.4 Zeichenkettenkonstante

Die Zeichenkettenkonstante wird auch als *String* bezeichnet. Häufig wird auch nur von einer Zeichenkette gesprochen. In einer Zeichenkette können Sie ein oder mehrere einzelne Zeichen hintereinander angeben. Die Zeichen werden zwischen zwei Anführungsstrichen " angegeben. Im Abschnitt 1.3.3 haben Sie bereits eine Zeichenkettenkonstante kennengelernt, die bei der Verwendung der `printf`-Funktion anzugeben ist. Der C-Compiler legt dafür einen speziellen Datentyp Feld von Einzelzeichen an. Dieses Thema vertiefen Sie im späteren Abschnitt 4.2.5.

Beispiel 1.4.7

Strings: `"Hallo Welt"`, `"Ein String besteht aus mehreren Einzelzeichen in einer Kette"`, `"o\n Steuerzeichen duerfen auch mit hinein"`, `"Progr \t in \t C"`

Zusammenfassung

- Zeichen in der Sprache C sind im ASCII-Zeichensatz dargestellt, siehe ASCII-Tabelle in Anhang A.
- Bezeichner in C können Buchstaben, Zahlen, und das Unterstrich-Zeichen _ enhalten.
- Bezeichner dürfen jedoch nicht mit einer Zahl beginnen.
- Bezeichner in C dürfen keine deutschen Umlaute und Sonderzeichen enthalten.
- Bezeichner dürfen keine Schlüsselworte sein, vgl. Tabelle B.1
- Folgende Konstanten können in C-Programmen verwendet werden:
  - Ganzzahlkonstanten
  - Fließkommakonstanten
  - Zeichenkonstanten
  - Zeichenkettenkonstanten, Strings
- Ganzzahlkonstanten können in folgenden Stellenwertsystemen angegeben sein:
  - Dezimal, z.B. 1234
  - Oktal, z.B. 015, 072
  - Hexadezimal, z.B. `0xFC`, `0x9AC`

1.5 Formatierte Ausgabe von Konstanten

Um Konstanten in einer ersten Anwendung verwenden zu können, lernen Sie an dieser Stelle die Ausgabeformierung von Werten kennen.

Programmbeispiel 1.4: einfache_dezimalformatierungsausgabe.c

```
#include <stdio.h>

int main()
```

```
{
  printf("In Deutschland wird man seit %d mit %d
     volljährig\n",1975, 0x12);

  return(0);
}
```

Programmausgabe

Ausgabe am Bildschirm:

```
In Deutschland wird man seit 1975 mit 18 volljährig
```

In dem Programmbeispiel 1.4 befinden sich im Ausgabestring die Anweisungen %d. Diese Formatanweisungen lassen die `printf`-Funktion nach dem String (hier `"In Deutschland wird man seit %d mit %d volljährig\n"`) zwei weitere Zahlen erwarten, die im Dezimalsystem ausgegeben werden, obwohl die zweite Zahlkonstante hexadezimal angegeben ist. Der String und die beiden Zahlen werden als Argumente der Funktion bezeichnet.

Tabelle 1.3: Formatzeichen (Fz) für die Ausgabe mit `printf`

Fz	Argument-datentyp	Erläuterung	Beispiel
c	char oder int	einzelnes Zeichen	`printf("%c", 'A');` druckt A
d	int	Dezimalzahl mit Vorzeichen (Vz.)	`printf("%d",−45);` druckt -45
i	int	Dezimalzahl mit Vz.	`printf("%i", −45);` druckt -45
u	unsigned int	Dezimalzahl ohne Vz.	`printf("%u", 3u);` druckt 3
o	unsigned int	Oktalzahl ohne Vz.	`printf("%o", 8);` druckt 10
x	unsigned int	Hex. 0-9abcdef	`printf("%x",11);` druckt b
X	unsigned int	Hex. 0-9ABCDEF	`printf("%X", 11);` druckt B
f	float	Fließkommazahl	`printf("%f", 5.234);` druckt 5.234000
e,E	float	Exponenten-schreibweise	`printf("%e", 0.05);` druckt 5.000000e-02
g,G	float	das kürzere von f und e	`printf("%g", 0.05);` druckt 0.05 `printf("%g", 0.0000005);` druckt 5e-07
s	char*	Ausgabe einer Zeichenkette	`printf("%s","string");` druckt string
ld	long int	dezimal	`printf("%ld", −454264321);` druckt −45426432
lf	double	Gleitpunktzahl doppelt genau	`printf("%lf", 1.4e2);` druckt 140.000000
le, lE	double	Exponentenschreib-weise, doppelt genau	`printf("%le",1.4e2);` druckt 1.400000e+02
hd	short int	dezimal, Short-Int	`printf("%hd", 812);` druckt 812
hx	short int	hex., Short-Int	`printf("%hx", 812);` druckt 32c
p	void*	Adresse in hex.	`printf("%p",&variable);` druckt die Adresse der variable
%	–	Ausgabe von %	`printf("%%");` druckt %

In Tabelle 1.3 ist eine Übersicht über alle Formatierungszeichen mit kurzen Erläuterung gegeben.

Jede Ausgabeformatierung besteht aus dem Prozentzeichen und einem folgenden Formatzeichen. Das Formatzeichen läßt die Ausgabefunktion einen bestimmten Datentyp als weiteres Argument beim Funktionsaufruf erwarten. Um das Prozentzeichen selber auszugeben, ist die Zeichenfolge `"%%"` notwendig. Die am häufigsten verwendeten Formatierungszeichen sind die ersten acht Zeilen der Tabelle. Prägen Sie sich also diese besonders ein. Neben den hier gezeigten Formatierungen der Ausgaben gibt es eine Reihe weiterer Angaben, um Zahlen und Zeichen eingerückt, rechts- oder linksbündig, mit und ohne führende Nullen oder Leerzeichen mittels `printf` auszugeben. Für diese Fälle informieren Sie sich bitte in weitergehender Literatur zur C-Programmierung [2, 7, 8].

Für eigene Versuche mit der Ausgabeformatierung nehmen Sie sich das Beispielprogramm `uebersicht_printf_formatierung.c` als Basis. Aus Platzgründen ist es an dieser Stelle nicht abgedruckt.

Zusammenfassung

- Die Ausgabe mit der Funktion `printf` kann formatiert werden.
- Jede Ausgabeformatierungsanweisung beginnt mit einem einzelnen %-Zeichen.
- Jede Formatierungsanweisung hat ein weiteres Argument in der Funktion `printf` zur Folge. Einzige Ausnahme bildet %% nach der kein weiteres Argument folgt.
- Jede Ausgabeformatierung erwartet einen bestimmten Datentyp des Arguments.

1.6 Übungsaufgaben

1.3 **Ausgabe der Zahlen von 1 bis 10**
Schreiben Sie ein Programm, das die Zahlen 1 bis 10 dezimal am Bildschirm jeweils in einer neuen Zeile ausgibt.
Hinweis: Verwenden Sie für jede neue Ausgabe eine neue `printf`-Funktion.

1.4 **Ausgabe der Zahlen von 8 bis 16 in Oktal- und Hexadezimal-Format**
Schreiben Sie ein C-Programm, das die Zahlen 8 bis 16 jeweils oktal

und hexadezimal in jeweils einer neuen Zeile ausgibt.
Hinweis: Verwenden Sie für jede neue Ausgabe eine neue `printf`-Funktion.