Variablen

Proseminar Effiziente Programmierung in C

Daniel Flat

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

28 April 2021

Gliederung

Deklaration [10] [5] [6] [7]

Deklaration [10] [5] [6] [7]

- **Alle Bezeichner müssen deklariert werden**

Deklaration [10] [5] [6] [7]

- **Alle Bezeichner müssen deklariert werden**
  - Schlüsselwörter, z.B. auto, register
  - Datentypen

Deklaration [10] [5] [6] [7]

- **Alle Bezeichner müssen deklariert werden**
  - Schlüsselwörter, z.B. auto, register
  - Datentypen
- Regeln:

Deklaration [10] [5] [6] [7]

- **Alle Bezeichner müssen deklariert werden**
  - Schlüsselwörter, z.B. auto, register
  - Datentypen
- Regeln:
  - beginnt mit Buchstaben oder Unterstrich

Deklaration [10] [5] [6] [7]

- **Alle Bezeichner müssen deklariert werden**
  - Schlüsselwörter, z.B. auto, register
  - Datentypen
- Regeln:
  - beginnt mit Buchstaben oder Unterstrich
  - nur Buchstaben des englischen Alphabets, z.B. ä, ö, ü oder ß verboten

Deklaration [10] [5] [6] [7]

- **Alle Bezeichner müssen deklariert werden**
  - Schlüsselwörter, z.B. auto, register
  - Datentypen
- Regeln:
  - beginnt mit Buchstaben oder Unterstrich
  - nur Buchstaben des englischen Alphabets, z.B. ä, ö, ü oder ß verboten
  - keine Schlüsselwörter

Deklaration [10] [5] [6] [7]

- **Alle Bezeichner müssen deklariert werden**
  - Schlüsselwörter, z.B. auto, register
  - Datentypen
- Regeln:
  - beginnt mit Buchstaben oder Unterstrich
  - nur Buchstaben des englischen Alphabets, z.B. ä, ö, ü oder ß verboten
  - keine Schlüsselwörter
  - max. 31 Zeichen

Deklaration Beispiel [10] [5] [6] [7]

```
1 (Schluesselwort) Datentyp Bezeichner; //Syntax
2 int zahl;
3 register int Zahl;
4 //akzeptiert
5
6 zahl;
7 int 8cht;
8 int long;
9 //ungueltig
```

Listing 1: Deklaration

Unterschied Definition und Deklaration [10] [5] [6] [7]

Unterschied Definition und Deklaration [10] [5] [6] [7]

- Deklaration: Vergabe eines Namens und Typs für Variable

Unterschied Definition und Deklaration [10] [5] [6] [7]

- Deklaration: Vergabe eines Namens und Typs für Variable
- Definition: Reservierung des Speicherplatzes

Unterschied Definition und Deklaration [10] [5] [6] [7]

- Deklaration: Vergabe eines Namens und Typs für Variable
- Definition: Reservierung des Speicherplatzes
- **Einmal definiert, aber mehrmals deklariert**

Initialisierung [10] [5]

Initialisierung [10] [5]

- Direkte Zuweisung eines Wertes einer Variable

Initialisierung [10] [5]

- Direkte Zuweisung eines Wertes einer Variable
- Zuweisungsoperator $=$

Initialisierung [10] [5]

- Direkte Zuweisung eines Wertes einer Variable
- Zuweisungsoperator $=$
- **Vor ersten Verwendung**

Initialisierung [10] [5]

- Direkte Zuweisung eines Wertes einer Variable
- Zuweisungsoperator =
- **Vor ersten Verwendung**
- L- und R Wert

Initialisierung Beispiel [10] [5]

```
1  Bezeichner = Wert; //Syntax
2
3  int i; //Deklaration
4  i = 5; //Initialisierung
5
6  int j = 6; //Deklaration und Initialisierung
7
8  i = 32; //L-Wert
9  j = i; //R-Wert
10 28 = 32; //ungueltig, R-Wert
```

Listing 2: Initialisierung

Arten von Variablen [11] [1] [16]

- Wertvariablen

Arten von Variablen [11] [1] [16]

- Wertvariablen
- referenzielle Variablen

Arten von Variablen [11] [1] [16]

- Wertvariablen
- referenzielle Variablen
- global, lokal und statisch

Globale Variable [11] [1]

- aus jeder Stelle des Programms zugreifbar

Globale Variable [11] [1]

- aus jeder Stelle des Programms zugreifbar
- stehen außerhalb der Funktionen

Globale Variable [11] [1]

- aus jeder Stelle des Programms zugreifbar
- stehen außerhalb der Funktionen
- statischen Speicher

Globale Variable [11] [1]

- aus jeder Stelle des Programms zugreifbar
- stehen außerhalb der Funktionen
- statischen Speicher
- Problematiken

Globale Variable [11] [1]

- aus jeder Stelle des Programms zugreifbar
- stehen außerhalb der Funktionen
- statischen Speicher
- Problematiken
- **mit Bedacht**

Globale Variable Beispiel [10] [6] [5] [1] [7]

```
1 int global = 10; //globale Variable, statischen
      ↪ Speicher
2
3 int main(void)
4 {
5     ...
6 }
```

Listing 3: Globale Variable

Lokale Variable [11] [1]

- abgeschlossener Block

Lokale Variable [11] [1]

- abgeschlossener Block
  - nicht von außen änderbar und zugreifbar

Lokale Variable [11] [1]

- abgeschlossener Block
  - nicht von außen änderbar und zugreifbar
  - Stack-/Stapelspeicher

Lokale Variable Beispiel [10] [6] [5] [7] [1]

```
1 int main(void)
2 {
3     {
4         int lokal = 1; //lokale Variable, Stack
5         ...
6     } //Ende des Blocks, Stack wird geleert
7     int variable = lokal; //ungueltig
8     return 0;
9 }
```

Listing 4: Lokale Variable

Statische Variable [11] [1]

- Wert erhalten?

Statische Variable [11] [1]

- Wert erhalten?
  - globale Variabeln?

Statische Variable [11] [1]

- Wert erhalten?
  - globale Variabeln?
- Schlüsselwort static

Statische Variable [11] [1]

- Wert erhalten?
  - globale Variabeln?
- Schlüsselwort static
- statischen Speicher

Statische Variable Beispiel [10] [6] [5] [7] [1]

```
 1 int funktion()
 2 {
 3     static int i = 0;
 4     return ++i;
 5 }
 6
 7 int main(void)
 8 {
 9     funktion(); //1
10     funktion(); //2
11     funktion(); //3
12     return 0;
13 }
```

Listing 5: Statische Variable

Datentypen [2] [16]

- Essenziell für Deklaration

Datentypen [2] [16]

- Essenziell für Deklaration
  - Speicherplatz

Datentypen [2] [16]

- Essenziell für Deklaration
  - Speicherplatz
  - Bitmuster

Datentypen [2] [16]

- Essenziell für Deklaration
  - Speicherplatz
  - Bitmuster
  - Operationen

Datentypen [2] [16]

- Essenziell für Deklaration
  - Speicherplatz
  - Bitmuster
  - Operationen
  - Genauigkeit

Datentypen [2] [16]

- Essenziell für Deklaration
  - Speicherplatz
  - Bitmuster
  - Operationen
  - Genauigkeit
- 3 Arten

Datentypen [2] [16]

- Essenziell für Deklaration
  - Speicherplatz
  - Bitmuster
  - Operationen
  - Genauigkeit
- 3 Arten
  - Basistypen

Datentypen [2] [16]

- Essenziell für Deklaration
  - Speicherplatz
  - Bitmuster
  - Operationen
  - Genauigkeit
- 3 Arten
  - Basistypen
  - Void

Datentypen [2] [16]

- Essenziell für Deklaration
  - Speicherplatz
  - Bitmuster
  - Operationen
  - Genauigkeit
- 3 Arten
  - Basistypen
  - Void
  - typedef

Ganzzahlen [2]

- Zahlen ohne Nachkommastellen

Ganzzahlen [2]

- Zahlen ohne Nachkommastellen
- unsigned

Ganzzahlen [2]

- n:= Anzahl der Bits

Ganzzahlen [2]

- n:= Anzahl der Bits
- unsigned: Gesamtanzahl

Ganzzahlen [2]

- n:= Anzahl der Bits
- unsigned: Gesamtanzahl

$$[0; 2^n - 1]$$

Ganzzahlen [2]

- n:= Anzahl der Bits
- unsigned: Gesamtanzahl

$$[0; 2^n - 1]$$

- signed: Höchstwertige Bit als Vorzeichen

Ganzzahlen [2]

- n:= Anzahl der Bits
- unsigned: Gesamtanzahl

$$[0; 2^n - 1]$$

- signed: Höchstwertige Bit als Vorzeichen

$$[-2^{n-1}; 2^{n-1} - 1]$$

Ganzzahlen [2]

Typ	Speicherplatz	Wertebereich	Format
char	1 byte	-128 - 127, 0 - 255	%c, %d

Ganzzahlen [2]

Typ	Speicherplatz	Wertebereich	Format
char	1 byte	-128 - 127, 0 - 255	%c, %d
unsigned char	1 byte	0 - 255	%c, %hhu
signed char	1 byte	-128 - 127	%c, %hhi

Ganzzahlen [2]

Typ	Speicherplatz	Wertebereich	Format
char	1 byte	-128 - 127, 0 - 255	%c, %d
unsigned char	1 byte	0 - 255	%c, %hhu
signed char	1 byte	-128 - 127	%c, %hhi
int	2 oder 4 bytes	-32.768 - 32.767, -2.147.483.648 - 2.147.483.647	%d, %i
unsigned int	2 oder 4 bytes	0 - 64.535, 0 - 4.294.967.295	%u

Ganzzahlen [2]

Typ	Speicherplatz	Wertebereich	Format
char	1 byte	-128 - 127, 0 - 255	%c, %d
unsigned char	1 byte	0 - 255	%c, %hhu
signed char	1 byte	-128 - 127	%c, %hhi
int	2 oder 4 bytes	-32.768 - 32.767, -2.147.483.648 - 2.147.483.647	%d, %i
unsigned int	2 oder 4 bytes	0 - 64.535, 0 - 4.294.967.295	%u
short	2 bytes	-32.768 - 32.767	%hi, %hd
unsigned short	2 bytes	0 - 65.535	%hu

Ganzzahlen [2]

Typ	Speicherplatz	Wertebereich	Format
char	1 byte	-128 - 127, 0 - 255	%c, %d
unsigned char	1 byte	0 - 255	%c, %hhu
signed char	1 byte	-128 - 127	%c, %hhi
int	2 oder 4 bytes	-32.768 - 32.767, -2.147.483.648 - 2.147.483.647	%d, %i
unsigned int	2 oder 4 bytes	0 - 64.535, 0 - 4.294.967.295	%u
short	2 bytes	-32.768 - 32.767	%hi, %hd
unsigned short	2 bytes	0 - 65.535	%hu
long	4 oder 8 bytes	-9.223.372.036.854.775.808 - 9.223.372.036.854.775.807	%hi, %hd
unsigned long	8 bytes	0 - 18.446.744.073.709.551.615	%lu

Fließkommazahlen [2]

- Zahlen mit Nachkommastellen

Fließkommazahlen [2]

- Zahlen mit Nachkommastellen
- kein unsigned möglich

Fließkommazahlen [2]

Typ	Speicherplatz	Wertebereich	Genauigkeit	Format
float	4 byte	1,2E-38 - 3,4E38	6 Stellen	%f

Fließkommazahlen [2]

Typ	Speicherplatz	Wertebereich	Genauigkeit	Format
float	4 byte	1,2E-38 - 3,4E38	6 Stellen	%f
double	8 byte	2,3E-308 - 1.7E308	15 Stellen	%lf

Fließkommazahlen [2]

Typ	Speicherplatz	Wertebereich	Genauigkeit	Format
float	4 byte	1,2E-38 - 3,4E38	6 Stellen	%f
double	8 byte	2,3E-308 - 1.7E308	15 Stellen	%lf
long double	10 byte	3.4E-4932 - 1.1E4932	19 Stellen	%Lf

Void [12]

Void [12]

- Kein Wert

Void [12]

- Kein Wert
- 3 Situationen

Void [12]

- Kein Wert
- 3 Situationen
  - Void Funktion
    - void funktion(int argument)

Void [12]

- Kein Wert
- 3 Situationen
  - Void Funktion
    - void funktion(int argument)
  - Void Argument
    - int funktion(void)

Void [12]

- Kein Wert
- 3 Situationen
  - Void Funktion
    - void funktion(int argument)
  - Void Argument
    - int funktion(void)
  - Void Zeiger
    - void *zeiger

typedef [4] [17]

typedef [4] [17]

- Schlüsselwort

typedef [4] [17]

- Schlüsselwort
- Neuer Name für Datentyp

typedef [4] [17]

- Schlüsselwort
- Neuer Name für Datentyp
- kein eigener Datentyp

typedef [4] [17]

- Schlüsselwort
- Neuer Name für Datentyp
- kein eigener Datentyp
- Zeiger

typedef Beispiel [4] [17]

```
1 typedef Definition Bezeichner; //Syntax
2
3 typedef unsigned int UINT;
4
5 UINT a = 5;
```

Listing 6: typedef

Typkonvertierung [15] [13] [9]

Typkonvertierung [15] [13] [9]

- Umwandeln des Datentyps

Typkonvertierung [15] [13] [9]

- Umwandeln des Datentyps
  - Viele Gründe

Typkonvertierung [15] [13] [9]

- Umwandeln des Datentyps
  - Viele Gründe
- implizit und explizit

Implizite Typkonvertierung [15] [13] [9]

Implizite Typkonvertierung [15] [13] [9]

- Durch Compiler durchgeführt

Implizite Typkonvertierung [15] [13] [9]

- Durch Compiler durchgeführt
- Erweiternd

Implizite Typkonvertierung [3]

bool → char → short → int → unsigned int → long → unsigned long →

long long → unsigned long long → float → double → long double

Implicit Type Conversion

Implizite Typkonvertierung [15] [13] [9]

```
1  int i = 2; // 2
2  float f = i; // 2.000000
3  char c = f; //ungueltig
4
5  char a = 'A'; //A oder 65
6  float b = 12.0f;
7  int c = 10;
8  char d = a + b + c; //ungueltig
9  int e = a + b + c; //ungueltig
10 float f = a + b + c; //gueltig
```

Listing 7: Implizierte Typumwandlung

Implizite Typkonvertierung [15] [13] [9]

- ist kompatibel?

Implizite Typkonvertierung [15] [13] [9]

- ist kompatibel?
  - ja: Ausführung

Implizite Typkonvertierung [15] [13] [9]

- ist kompatibel?
  - ja: Ausführung
  - nein: Abbruch

Implizite Typkonvertierung [15] [13] [9]

- ist kompatibel?
  - ja: Ausführung
  - nein: Abbruch
- float zu int: Kommastellenverlust

Implizite Typkonvertierung [15] [13] [9]

- ist kompatibel?
  - ja: Ausführung
  - nein: Abbruch
- float zu int: Kommastellenverlust
- long zu char: Genauigkeitsverlust

Explizite Typkonvertierung [15] [13] [9]

Explizite Typkonvertierung [15] [13] [9]

- Explizit im Code

Explizite Typkonvertierung [15] [13] [9]

- Explizit im Code
- Einschränkend

Explizite Typkonvertierung [15] [13] [9]

- Explizit im Code
- Einschränkend

```
1 (Zieltyp)Variable
```

Listing 11: Explizite Typumwandlung Syntax

Explizite Typkonvertierung [15] [13] [9]

- Explizit im Code
- Einschränkend

```
1 (Zieltyp)Variable
```

Listing 12: Explizite Typumwandlung Syntax

- unabhänig von Hierarchie

Explizite Typkonvertierung Beispiel [15] [13] [9]

```
1 int a = 2.718281f; //Fehler
2 char b = 65L;      //Fehler
3 float c = 4.1;     //Fehler
```

Listing 13: Explizite Typumwandlung 1

Explizite Typkonvertierung Beispiel [15] [13] [9]

```
1 int a = (int)2.718281f; //2
2 char b = (char)65L;     //'A'
3 float c = (float)4.1;   //4.1f
```

Listing 14: Explizite Typumwandlung 2

Typkonvertierung [15] [13] [9]

- Bitmuster: 1101 1100

Typkonvertierung [15] [13] [9]

- Bitmuster: 1101 1100
  - signed short: -92

Typkonvertierung [15] [13] [9]

- Bitmuster: 1101 1100
  - signed short: -92
  - unsigned int: 220

Typkonvertierung [15] [13] [9]

- Bitmuster: 1101 1100
  - signed short: -92
  - unsigned int: 220
  - char: 'Ü'

Typkonvertierung [15] [13] [9]

- Bitmuster: 1101 1100
  - signed short: -92
  - unsigned int: 220
  - char: 'Ü'
- **Einzigartige Interpretation**

printf [14] [8]

printf [14] [8]

- Konsole schreiben

printf [14] [8]

- Konsole schreiben
  - selbstständig Funktion

printf [14] [8]

- Konsole schreiben
  - selbstständig Funktion
  - stdio.h;

printf [14] [8]

- Konsole schreiben
  - selbstständig Funktion
  - stdio.h;
  - Format

printf [14] [8]

- Konsole schreiben
  - selbstständig Funktion
  - stdio.h;
  - Format

```
1 printf(const char *format, ...);
```

Listing 20: printf Syntax

printf [14] [8]

- Konsole schreiben
  - selbstständig Funktion
  - stdio.h;
  - Format

```
1 printf(const char *format, ...);
```

Listing 21: printf Syntax

- **Anfang muss ein String sein**

printf [2]

Typ	Speicherplatz	Wertebereich	Format
char	1 byte	-128 - 127, 0 - 255	%c, %d
unsigned char	1 byte	0 - 255	%c, %hhu
signed char	1 byte	-128 - 127	%c, %hhi
int	2 oder 4 bytes	-32.768 - 32.767, -2.147.483.648 - 2.147.483.647	%d, %i
unsigned int	2 oder 4 bytes	0 - 64.535, 0 - 4.294.967.295	%u
short	2 bytes	-32.768 - 32.767	%hi, %hd
unsigned short	2 bytes	0 - 65.535	%hu
long	4 oder 8 bytes	-9.223.372.036.854.775.808 - 9.223.372.036.854.775.807	%hi, %hd
unsigned long	8 bytes	0 - 18.446.744.073.709.551.615	%lu

printf [14] [8]

```
1  #include <stdio.h>
2  void main()
3  {
4  char a = 'A';
5  printf(%d, a); //ungueltig
6  printf("%d", a); //65
7  printf("%c", a); //A
8  ...
9  printf("%c", (char)a); //A
10 } //explizit
```

Listing 22: C printf

Umwandlungen [14] [8]

Umwandlungen [14] [8]

- Formatierung zu ändern

Umwandlungen [14] [8]

- Formatierung zu ändern
- benutzerdefiniert

Umwandlungen [14] [8]

- Formatierung zu ändern
- benutzerdefiniert
  - Flags/Kennzeichen

Umwandlungen [14] [8]

- Formatierung zu ändern
- benutzerdefiniert
  - Flags/Kennzeichen
  - Feldbreite

Umwandlungen [14] [8]

- Formatierung zu ändern
- benutzerdefiniert
  - Flags/Kennzeichen
  - Feldbreite
  - Nachkommastellen

Kennzeichen [14] [8]

Kennzeichen [14] [8]

- optional nach dem %-Zeichen

Kennzeichen [14] [8]

- optional nach dem %-Zeichen

Symbol	Bedeutung
-	Linksbündig
+	Vorzeichen
Leerzeichen	Leerzeichen vor dem Vorzeichen eines positiven Wertes
#	Umwandeln in andere Zahlensysteme
0	Auffüllen von 0

Feldbreite [14] [8]

- optional nach dem Kennzeichen

Feldbreite [14] [8]

- optional nach dem Kennzeichen
- Auffüllen der Ausgabe mit Zeichen

Nachkommastellen [14]

- Angeben der auszugebenen Nachkommastellen

Nachkommastellen [14]

- Angeben der auszugebenen Nachkommastellen
- nur bei Fließkommazahlen

Formatierung Beispiele [14] [8]

```
1 #include <stdio.h>
2 void main(void)
3 {
4 int i = 25;
5 float f = 25.1f;
6 printf("%#o", i);     //031
7 printf("%5f", f);     //     25.100000
8 printf("%0.3f", f);   //25.100
9 }
```

Listing 23: Formatierung Beispiele

Zusammenfassung

Zusammenfassung

- benannte Speicherzelle

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert
  - global, lokal oder statisch

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert
  - global, lokal oder statisch
- Arten

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert
  - global, lokal oder statisch
- Arten
  - Sichtbarkeit und Lebensdauer

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert
  - global, lokal oder statisch
- Arten
  - Sichtbarkeit und Lebensdauer
  - Speicherort

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert
  - global, lokal oder statisch
- Arten
  - Sichtbarkeit und Lebensdauer
  - Speicherort
- Datentyp = Herz

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert
  - global, lokal oder statisch
- Arten
  - Sichtbarkeit und Lebensdauer
  - Speicherort
- Datentyp = Herz
  - unterschiedliche Interpretationen

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert
  - global, lokal oder statisch
- Arten
  - Sichtbarkeit und Lebensdauer
  - Speicherort
- Datentyp = Herz
  - unterschiedliche Interpretationen
- Typkonvertierung ermöglicht Ändern des Datentyps

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert
  - global, lokal oder statisch
- Arten
  - Sichtbarkeit und Lebensdauer
  - Speicherort
- Datentyp = Herz
  - unterschiedliche Interpretationen
- Typkonvertierung ermöglicht Ändern des Datentyps
- printf ersten Funktionen für die Konsole

Zusammenfassung

- benannte Speicherzelle
  - zuvor deklariert, definitiert und initialisiert
  - global, lokal oder statisch
- Arten
  - Sichtbarkeit und Lebensdauer
  - Speicherort
- Datentyp = Herz
  - unterschiedliche Interpretationen
- Typkonvertierung ermöglicht Ändern des Datentyps
- printf ersten Funktionen für die Konsole
- **Ein Muss für jedes Programm**

Literatur

[1] 10.3. lebensdauer von variablen. *C-buch.sommergut*. eingesehen am 22.05.2021. URL: http://www.c-buch.sommergut.de/Kapitel10/Lebensdauer-von-Variablen.shtml.

[2] C - Data Types. *Tutorialspoint*. eingesehen am 22.05.2021. URL: https://www.tutorialspoint.com/cprogramming/c_data_types.htm.

[3] C++ - type casting. *AlphaCodingSkills*. eingesehen am 22.05.2021. URL: https://www.alphacodingskills.com/cpp/cpp-type-casting.php.

[4] C - typedef. *Tutorialspoint*. eingesehen am 22.05.2021. URL: https://www.tutorialspoint.com/cprogramming/c_typedef.htm.

[5] C - Variables. *Tutorialspoint*. eingesehen am 22.05.2021. URL: https://www.tutorialspoint.com/cprogramming/c_variables.htm.

[6] C Variables, Constants and Literals. *Programiz*. eingesehen am 22.05.2021. URL: https://www.programiz.com/c-programming/c-variables-constants.

[7] Definition von Variablen. *Grund-wissen*. eingesehen am 22.05.2021. URL: https://www.grund-wissen.de/informatik/c/variablen-und-datentypen.html.

[8] Output formatting using printf function in c. *Know Program*. eingesehen am 22.05.2021. URL: https://www.knowprogram.com/c-programming/printf-in-c/.

[9] Typecasting in c: Implicit, explicit with example. *Guru99*. eingesehen am 22.05.2021. URL: https://www.guru99.com/c-type-casting.html.

[10] Variablen in C und C++, Deklaration und Definition. *Coder-Welten*. eingesehen am 22.05.2021. URL: http://www.coder-welten.de/einstieg/variablen-in-c-3.html.

[11] Variablen und Schlüsselwörter in C. *GeeksforGeeks*. eingesehen am 22.05.2021. URL: https://www.geeksforgeeks.org/variables-and-keywords-in-c/.

[12] What is void in C. *C-Programming-Simple-Steps*. eingesehen am 22.05.2021. URL: https://www.c-programming-simple-steps.com/what-is-void.html.

[13] Type casting in c language. *Improgrammer*, 05 2019. eingesehen am 22.05.2021. URL: https://www.improgrammer.net/type-casting-c-language/.

[14] C-Programmierung: Einfache Ein- und Ausgabe. *Wikibooks*, 10 2020. eingesehen am 22.05.2021. URL: https://de.wikibooks.org/wiki/C-Programmierung:_Einfache_Ein-_und_Ausgabe#printf.

[15] C-Programmierung: Typumwandlung. *Wikibooks*, 06 2020. eingesehen am 22.05.2021. URL: https://de.wikibooks.org/wiki/C-Programmierung:_Typumwandlung.

[16] Variable (Programmierung). *Wikipedia*, 06 2020. eingesehen am 22.05.2021. URL: https://de.wikipedia.org/wiki/Variable_(Programmierung).

[17] typedef. *Wikipedia*, 03 2021. eingesehen am 22.05.2021. URL: https://en.wikipedia.org/wiki/Typedef.