

Inhalt

2	C-Objekte	2-2
2.1	Das Alphabet	2-3
2.2	Variablen und Konstanten	2-4
2.2.1	C-Variablen	2-4
2.2.2	C-Standarddatentypen	2-4
2.2.3	C-Konstanten.....	2-6
2.2.4	C-Variablendeklaration	2-7

2 C-Objekte

Objekte sind i.R. **mathematische Größen** mit denen man in **Algorithmen** operieren kann. Diese müssen in der Maschine dargestellt und die **Operationen** durch die Hardware oder als **Programme** installiert werden.

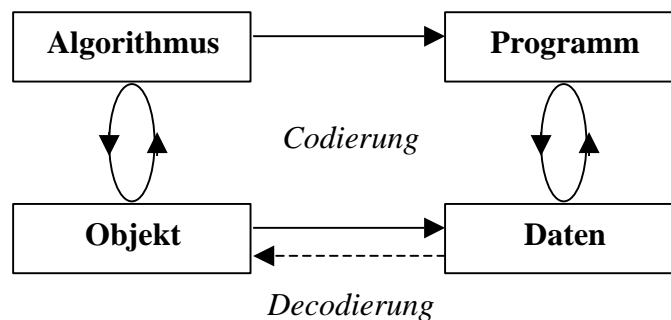
Euklidischer Algorithmus zur Berechnung des größten gemeinsamen Teilers.

$\text{ggT}(53\,667, 25\,527) = 201$

53 667	:	25 527	=	2,	Rest	2 613
25 527	:	2 613	=	9,	Rest	2 010
2 613	:	2 010	=	1,	Rest	603
2 010	:	603	=	3,	Rest	201
603	:	201	=	3,	Rest	0

Die *mathematische Größen* sind hier die **natürlichen Zahlen**, die vom Euklidischen Algorithmus benötigte *Operation* ist die **Restbildung der ganzzahligen Division**.

Umsetzung im Rechner:



ggT.c

```

/*
 * Berechnen des groessten gemeinsamen Teilers
 * der Zahlen a = 53667 und b = 25527.
 */
# include <stdio.h>

int main()
{
    /* natuerliche Zahlen */
    unsigned int a = 53667, b = 25527, c;
    do
    {
        c = a % b;    /* Rest der ganzzahligen Division */
        a = b; b = c;    /* Vertauschen der Werte */
    } while( c != 0);

    printf( "Ergebnis: %d\n", a);
    return 0;
}

```

C-Grundobjekte:

Ganze Zahlen
Gleitpunktzahlen
Zeiger (Adressen)

2.1 Das Alphabet

Der Zeichensatz von C umfasst **92 druckbare Zeichen** und **7 Steuerzeichen**.

Druckbare Zeichen:

- 26 Großbuchstaben des englischen Alphabets **A . . Z**
- 26 Kleinbuchstaben des englischen Alphabets **a . . z** (keine Umlaute, kein ß!)
- 10 Ziffern **0 . . 9**
- 30 Sonderzeichen **! " # % & ` () * + - ' . / :**
; < = > ? [\] ^ _ { | } ~ Leerzeichen

Groß- und Kleinbuchstaben sind *signifikant*.

Steuerzeichen:

Sie dienen der **Steuerung von Ausgabegeräten** wie Bildschirm und Drucker, teilweise auch für die **Eingabe** verwendbar, etwa von der Tastatur. Ihre Implementierung ist systemabhängig.

Folgende Escapesequenzen sind definiert:

<code>\a</code>	Akustischer Laut (alert)
<code>\b</code>	Versetzen um eine Position nach links (backspace)
<code>\f</code>	Seitenvorschub (formfeed)
<code>\n</code>	Zeilenvorschub (linefeed, new line)
<code>\r</code>	Positionierung am Zeilenanfang (carriage return)
<code>\t</code>	Horizontaler Tabulator (horizontal tab)
<code>\v</code>	Vertikaler Tabulator (vertical tab)

Beispiel s.o.: `printf("Hallo, Welt\n");`

Entwerter:

Vier weitere Escapesequenzen erzeugen **druckbare Zeichen** und dienen der **Entwertung von C-Metazeichen**:

<code>\'</code>	Entwerten des Abschlussymbol für Zeichenkonstante
<code>\"</code>	Entwerten des Abschlussymbol für Zeichenkettenkonstante
<code>\?</code>	Entwerten des Zeichens für Trigraphen (definierte Ersatzzeichen)
<code>\\</code>	Entwerten des Backslashes als Escapesequenz

C-Namen:

Für die Bezeichnung der Objekte und ihrer Operationen in der Maschine (Konstanten, Variablen, Funktionen usw.) werden **Namen** (*identifier*) benötigt. Diese unterliegen folgenden Bedingungen:

C-Namen sind frei wählbare, beliebig lange Wörter aus den 52 Buchstaben, 10 Ziffern und dem Unterstrich, die *nicht* mit einer Ziffer beginnen dürfen und *keines* der 32 C-Schlüsselwörter (Wörter der Sprache) sind.

Es gelten allgemeine *Regeln* für die Wahl von Namen:

- Grundsätzlich sollten Namen den Inhalt beschreiben (*sprechende Namen*).
- *Variablen- und Funktionsnamen* beginnen mit einem **Kleinbuchstaben**. Ist dieser aus mehreren Worten zusammengesetzt, so beginnt jedes neue Wort mit einem Großbuchstaben (**readInt**).
- *Konstantennamen* bestehen aus nur **Grossbuchstaben**. Ist dieser aus mehreren Worten zusammengesetzt, so beginnt jedes neue Wort mit einem Unterstrich (**MAX**, **MAX_ANZAHL**).

2.2 Variablen und Konstanten

2.2.1 C-Variablen

Euklidischer Algorithmus: Die mathematischen Objekte sind natürliche Zahlen. Diese müssen im Rechner abgespeichert werden und auch wieder aufgefunden werden. Drei Fragen sind zu klären:

Wo befindet sich im Speicher die Zahl?

Wie viel Speicherplatz benötigt sie?

Wie wird sie abgespeichert?

Daten werden als **Bitfolgen** abgespeichert, d.h. Folgen aus 0 und 1. **Variablen** dienen als Platzhalter im Speicher für die Daten und besitzen **drei Grundbestandteile**:

Name	Anfangsadresse der Bitfolge
Typ	Länge der Bitfolge und ihre Interpretation
Wert	die interpretierte Bitfolge

Name

C-Name, symbolischer Bezeichner für die Speicheradresse.

Typ

Es gibt Standardtypen für ganze Zahlen und Gleitpunktzahlen, deren Grenzen sind in *<limits.h>* und *<float.h>* deklariert. Des weiteren gibt es Zeigertypen und aus den Standardtypen und den Zeigern zusammengesetzte Typen.

Wert

Die Interpretation einer Bitfolge wird in einem späteren Kapitel abgehandelt.

2.2.2 C-Standarddatentypen

Die Datentypen müssen nach ANSI-C folgende Mindestanforderungen erfüllen.

Mindestschranken für die Wertebereiche (s. auch Anhang A):

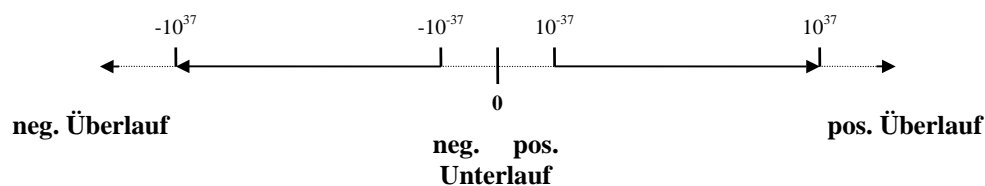
1. Ein Byte muss 8 Bit lang sein.
2. Wertebereiche der ganzzahligen Typen:

Typ	kleinster Wert (MIN)	größter Wert (MAX)	Byte
signed char	$-127 (= -2^7 + 1)$	$127 (= 2^7 - 1)$	1
unsigned char	0	$255 (= 2^8 - 1)$	1
signed short int	$-32'767 (= -2^{15} + 1)$	$32'767 (= 2^{15} - 1)$	2
unsigned short int	0	$65'535 (= 2^{16} - 1)$	2
signed int	$-32'767 (= -2^{15} + 1)$	$32'767 (= 2^{15} - 1)$	2
unsigned int	0	$65'535 (= 2^{16} - 1)$	2
signed long int	$-2'147'483'647 (= -2^{31} + 1)$	$2'147'483'647 (= 2^{31} - 1)$	4
unsigned long int	0	$4'294'967'295 (= 2^{32} - 1)$	4

3. Wertebereich für Gleitpunkttypen: $[-10^{37}, -10^{-37}] \cup \{0\} \cup [10^{-37}, 10^{37}]$
Genauigkeit (Dichte) für Gleitpunkttypen:

Typ	Genauigkeit	Stellen	Byte
float	10^{-5}	6	4
double	10^{-9}	10	6
long double	10^{-9}	10	6

Überlauf → Laufzeitfehler
Unterlauf → Rundungsfehler



Auf verschiedenen Rechnern sind die Datentypen durchaus unterschiedlich implementiert. Mit dem Operator `sizeof(Typ)` kann man die aktuellen Größen in Byte (`sizeof`) des Typs `Typ` ermitteln.

byte.c

```
/*
 * Bestimmen der Speicherplatzgrösse in Byte
 * der Elementardatentypen
 */
#include <stdio.h>

int main()
{
    printf( "char=%d short=%d int=%d long=%d\n",
           sizeof( char), sizeof( short),
           sizeof( int), sizeof( long));
    printf( "float=%d double=%d long double=%d\n",
           sizeof( float), sizeof( double),
           sizeof( long double));
    return 0;
}
```

byte.out

```
char=1 short=2 int=4 long=4
float=4 double=8 long double=16
```

"%d" dient als **Formatbeschreiber** für die Ausgabe **ganzzahliger** Werte mit der Funktion `int printf(const char *, . . .)`. Der Formatbeschreiber wird als Platzhalter in die Ausgabezeichenkette eingetragen, die Werte werden aus den Argumenten nach der Ausgabezeichenkette berechnet und in den Typ des Platzhalters konvertiert.

2.2.3 C-Konstanten

Konstanten sind **explizit** angegebene Objekte im Programm. Diese werden durch bloßes **Hinschreiben** definiert, besitzen einen Typ, der sich aus der Schreibweise der Konstanten ergibt.

⇒ **Konstanten haben einen Typ und einen unveränderbaren Wert, aber i. R. keinen Namen.**

Ganzzahlige Konstanten (*integer-constant*)

decimal-constant { 1, 2, 3, 4, 5, 6, 7, 8, 9 } { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }^{*1}

Typ: int ⇒ long ⇒ unsigned long

Beispiel: **1234**

octal-constant 0 { 0, 1, 2, 3, 4, 5, 6, 7 }^{*}

Typ: int ⇒ unsigned int ⇒ long ⇒ unsigned long

Beispiel: **0123** = (123)₈ = 1*8²+2*8+3 = 64+16+3 = 83

hexa-decimal-constant 0 { x | X } { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, a, B, b, C, c, D, d, E, e, F, f }⁺²

Typ: int ⇒ unsigned int ⇒ long ⇒ unsigned long

Beispiel: **0x10Fa** = (10fa)₁₆ = 1*16³+15*16+10 = 4096+240+10 = 4346

Gleitpunktkonstanten (*floating-constant*)

fractional-constant Z . { 0 }^{*} Z oder . { 0 }^{*} Z oder Z .
(Festpunktschreibweise) (Dezimalpunkt! Z *decimal-constant*)

Typ: double

Beispiel: **1.01234 .1234 1.**

floating-constant { F | Z } { e | E } { | + | - } Z
(Gleitpunktschreibweise) (F *fractional-constant*)

Typ: double

Beispiel: **1.e1** = 1.0*10¹ = 10.
.12e+3 = 0.12*10⁺³ = 120.
5E-4 = 5*10⁻⁴ = 0.0005

^{1*} Iteration über eine Menge: Menge aller Wörter dieser Menge, **einschließlich** dem leeren Wort.

²⁺ Iteration über eine Menge: Menge aller Wörter dieser Menge, **ausschließlich** dem leeren Wort.

Zeichenkonstanten (*character-constant*)

Zeichenkonstanten sind ganze Zahlen vom Typ *char*, durch ihren Code verschlüsselt, i. R. ist es der ASCII-Code³.

Alphabetzeichen werden in Apostrophe eingeschlossen.

'A'	'\n'
65	10

Explizite Angabe des Codes eines Zeichens:

<i>octal</i>	'\ooo'	ooo ein bis dreistellige Oktalzahl	'\100'	'@'
<i>hexadecimal</i>	'\xhh'	hh ein bis zweistellige Hexadezimalzahl	'\x07'	Bell

Zeichenkettenkonstanten (*string-literal*)

Zeichenkettenkonstanten sind **unbenannte C-Konstanten** und stehen im Zusammenhang mit Feldern. Sie sind als Abkürzung einer Folge von Zeichenkonstanten mit dem letzten Folgenglied '\0' zu verstehen. Das Abschlusszeichen '\0' wird dabei automatisch gesetzt.

Zeichenkettenkonstanten werden in Ausführungszeichen eingeschlossen.

" *Zeichenkette* "

2.2.4 C-Variablendeklaration

Alle Variablen müssen vor dem ersten Zugriff deklariert werden, d. h. dem Compiler muss der *Name* und der *Typ* bekannt gegeben werden, damit er den entsprechenden Speicherplatz zur Verfügung stellt und die Interpretationsvorschrift kennt.

Typ Variablenname [= Konstante], Variablenname [= Konstante], ... ;

```
int anzahl = 0;
float zahl, summe = 0;
float a = 1e10, b = 1e-10;
```

Beispiel

Euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers:

Zur Berechnung benötigt man drei Variablen für natürliche Zahlen: zwei für den Dividenden und den Divisor, anfangs mit Eingabewerten belegt, und einen zum Speichern für den Rest der ganzzahligen Division.

```
unsigned int a = 53667, b = 25527, c;
```

³ASCII American Standard Code for Information Interchange