C

AINF/FI

Schuljahr 2014/15

DI Thomas Helml

Inhalt

- Geschichte von C
- Hello World
- Variablen und Datentypen

C

- C wurde von Dennis Ritchie Anfang '70 an den Bell Laboratories (Teil von Alcatel-Lucent) entwickelt

C

Dennis Ritchie (1941-2011)

C

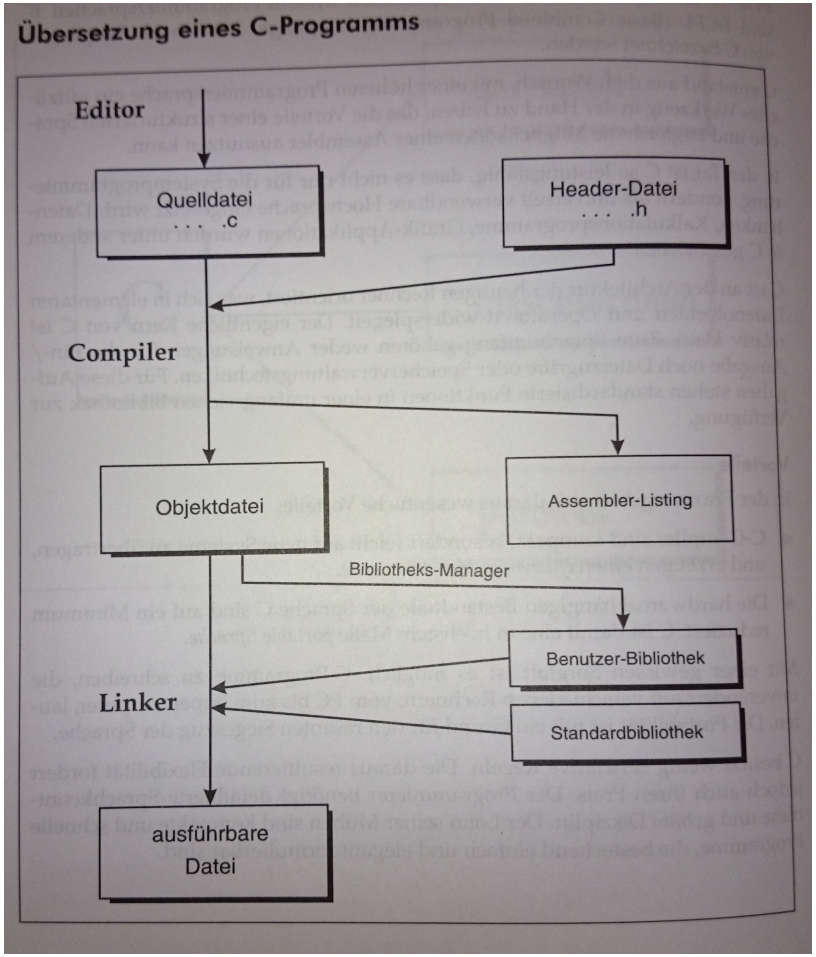
- ab 1972 wurde UNIX, sowie die dazugehörigen Dienstprogramme in C entwickelt
- Einsatz
  - Anwendungs- und Systemprogrammierung

Linus Torvalds,
Entwickler von Linux

Eigenschaften von C

- **Universell**
  - Einsetzbar für verschiedene Einsatzgebiete: Systemprogrammierung, Datenbanken, Kalkulationen, Grafik-Applikationen
- **Maschinennah**
  - Schnell
  - Hardwarenahe Systemprogrammierung möglich (UNIX-Systemaufrufe, direkte Integration von Assembler)
- **Portabel**
  - Leicht auf andere Systeme portierbar, kaum hardwareabhängige Bestandteile
  - Gut Standardisiert (ANSI 99 – American National Standards Institute)

Kompiliervorgang in C



Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int main ()
{
   printf („Hello World!\n“);
   return 0;
}
```

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int main ()
{
   printf („Hello World!\n");
   retu
}
```

`/* */` definiert ein (mehrzeiliges) Kommentar. Alles innerhalb dieser Markierung wird vom Compiler ignoriert.

`//` definiert ein (einzeiliges) Kommentar. Alles nach dieser Markierung wird vom Compiler ignoriert.

Kommentare dienen der Dokumentation des Quellcodes.

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int main ()
{
   printf („Hello World!\n“);
   return 0;
}
```

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int
{
   p
   r
}
```

Präprozessoranweisungen beginnen mit #
Sie werden VOR dem compilieren ausgeführt.
`#include <stdio.h>` lädt die „Bibliothek“ für die Ein-/und Ausgabe und ersetzt diese Anweisung durch den Inhalt der Datei `stdio.h`

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int main ()
{
   printf („Hello World!\n“);
   return 0;
}
```

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int main ()
{
   prin
   retu
}
```

`main ()` ist das sogenannte Hauptprogramm. Hier beginnt die Programmausführung

jedes C Programm **muss eine** Funktion `main ()` haben

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int main ()
{
   printf („Hello World!\n“);
   return 0;
}
```

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int main ()
{

}
```

{ } definiert einen Anweisungsblock. Hier stehen die eigentlichen Befehle

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int main ()
{
   printf („Hello World!\n“);
   return 0;
}
```

Hello World

```
/* Unser erstes C-Programm */
#inc
int
{
   printf („Hello World!\n“);
   return 0;
}
```

`printf(“...“);` gibt Text am Bildschirm aus

`printf` ist eine Anweisung

Anweisungen müssen mit Semikolon abgeschlossen werden.

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int main ()
{
   printf („Hello World!\n“);
   return 0;
}
```

Hello World

```
/* Unser erstes C-Programm */
#include <stdio.h>

int
{
   printf („Hello World!\n");
   return 0;
}
```

return 0;
beendet das Programm und gibt den Fehlercode 0 (= alles ok) zurück

printf

- Dient zur Ausgabe am Bildschirm
  - Ausgabe von Sonder-und Steuerzeichen mittels sog. Escape-Sequenzen (beginnen mit Backslash) innerhalb der Anführungszeichen

Steuerzeichen	Bedeutung
\n	Linefeed
\t	Horizontal Tab
\v	Vertical Tab
\“	“
\'	'
\?	?
\\	\

Variablen

- für Verarbeitung von Informationen muss Speicher reserviert werden => Variable(n)
- vor Verwendung einer Variable muss diese deklariert werden
- Jede Variable benötigt einen Variablennamen
  - Buchstaben, Ziffern, '_' – keine Sonderzeichen
  - 1. Zeichen Buchstabe oder '_'
  - klein geschrieben

Variablen

- Folgende reservierte Schlüsselwörter dürfen nicht als Variablennamen verwendet werden:

auto	const	double	float	int	short	struct	unsigned
break	continue	else	for	long	signed	switch	void
case	default	enum	goto	register	sizeof	typedef	volatile
char	do	extern	if	return	static	union	while

Datentypen

Datentyp	Bytes	Range
char	1	-128 ... 127
unsigned char	1	0 ... 255
short	2	-32768 .. 32767
unsigned short	2	0 ... 65535
int	2/4	
unsigned int	2/4	
long	4	-2 147 483 648 ... 2 147 483 647
unsigned long	4	0 .. 4 294 967 295
float	4	1.17E-38 ... 3.4E38
double	8	2.2E-308 ... 1.8E308

Deklaration

- (Variablen-)Deklaration geschieht zu Beginn des Programms
- Deklaration setzt sich zusammen aus
  - Datentyp (Größe der Daten)
  - Variablenname(n)
- Bsp.:

```
int a;
float x, y;
```

Initialisierung

- Bei der Deklaration einer Variable kann/soll gleichzeitig ein Startwert (=Initialwert) zugewiesen werden.
- Beispiel

```
int x = 0;
float f = 0.5;
```

Ausgabe von Variablen

- Ausgabe mittels `printf`
- notwendig: Platzhalter
  - beginnen mit `%`
  - z.B. `%d` für `int`
  - Beispiel:

```
printf ("Die Zahl lautet: %d", x);
```

Grundrechnungsarten

- Zuweisungen von Berechnungen mit =
- Variable steht immer links!
- Grundrechnungsarten:
  - Addition: +
  - Subtraktion: -
  - Multiplikation: *
  - Division: /
  - Modulo („Restrechnung“): %

Grundrechnungsarten

- Vorrangregeln wie in der Mathematik
- Klammern sind zulässig
- Bsp.:
  - `int x = 7;`
  - `x = 2*(a + b) + c/3;`
  - `x = x + 1;  // x wird um 1 erhöht`
  - `x = 7 / 3;  // x = 2`
  - `x = 7 % 3;  // 7:3=2, Rest 1 => x = 1`

Eingabe

- Befehl: `scanf`
  - ähnlich zu verwenden wie `printf`
  - gleiche Platzhalter für Variablen wie `printf`
  - Eingabe wird erst mit ENTER übernommen
  - KEINE Ausgabe mit `scanf` möglich!!!
  - vor Variable MUSS Ampersand ‘&‘ (=Adressoperator) stehen

Eingabe

- Beispiel:

```
int x = 0;

printf (“Geben Sie eine Zahl ein: “);
fflush(stdout);
scanf (“%d“, &x);

printf (“Sie haben %d eingegeben!“, x);
```

<math.h>

- (Software)Bibliotheken werden mit `#include <...>` hinzugefügt
- z.B. `#include <stdio.h>` für `printf, scanf, …`
- Bibliothek für Mathematische Funktionen:
  - <math.h>

<math.h>

- Folgende Funktionen stehen uns zur Verfügung (Ausschnitt):
  - sin(double x), cos(double x), tan(double x)
  - asin(double x), acos(double x), atan(double x)
  - ...
- Konstante M_PI (für PI)

<math.h>

```
double ergebnis = 0.0;
double x = 2.14;

ergebnis = sin(x);
printf ("%lf", ergebnis);

printf ("%lf", sin(x));
```

<math.h>

- Wurzelberechnung: sqrt(double x)
  - berechnet die Wurzel von x.
  - Ergebnis ist double-Wert
- Potenzen: pow (double x, double y)
  - x ... Basis
  - y ... Exponent
  - berechnet „x hoch y“
  - Ergebnis ist double Wert

Vergleichsoperatoren

Operator	Bedeutung
==	Vergleich
!=	Ungleichheit
>	Größer
>=	Größer gleich
<	Kleiner
<=	Kleiner gleich

Aussage ist falsch (=0) oder wahr (!=0)

Boolsche Algebra

Aussage	Ergebnis
4 >= 5	**0 (falsch)**
2.7 < 3.8	**1 (wahr)**
(4+2) == 5	**0 (falsch)**
2*4 != 5	**1 (wahr)**

Boolsche Algebra

- Wahrheitstabellen

NOT,NICHT, !

x	!x
0	**1**
1	**0**

AND, UND, &&

x	y	x && y
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

OR, ODER, ||

x	y	x || y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Boolsche Algebra

x	y	Logischer Ausdruck	Ergebnis
1	-1	x < y || y >= 0	0
0	0	x > -5 && !y	1
1	0	x == 1 && !y	1
0	1	!(x+1) || y-2 > 2	0
5	5	17 && (x-4) > 5	0
7		x < 9 && x >= -5	1
7		!x && x >= 3	0

Boolsche Algebra

- Formuliere einen logischen Ausdruck, der WAHR ergibt für:
  - alle x, die zwischen 8 und 18 liegen
  - alle x, die größer als 7, aber kleiner als 19 sind
  - alle x, die kleiner als 7 oder größer als 19 sind
  - alle x, die ungerade sind

Verzweigungen

- if und else
  - Eine Verzweigung innerhalb eines Programmes wird durch eine Bedingung entschieden
  - Bsp: Wenn Benutzer X eingibt, mach A.
- Syntax:

```
if (Bedingung){          // wenn Bedingung wahr ist
   mach was              // wird das gemacht
}
else{                    // sonst
   mach was anderes      // wird das gemacht
}
```

Verzweigungen

```
int zahl=5;

if(zahl==5) {
    printf("fuenf\n");
}
```

fuenf

```
int zahl=6;

if(zahl==5) {
    printf("fuenf\n");
}else {
    printf("nicht fuenf\n");
}
```

nicht fuenf

Verzweigungen

- Verzweigungen können ineinander geschachtelt werden!

```
int zahl=6;

if(zahl==5) {
    printf("fuenf\n");
}else {
     if(zahl==6) {
         printf("sechs\n");
     }else {
        printf("nicht fuenf und nicht sechs\n");
    }
}
```

sechs

Verzweigungen

- Kommt nur eine Anweisung in den if Block {}, so kann man die Klammern weglassen

```
int zahl=6;

if(zahl==5) printf("fuenf\n");
else if(zahl==6) printf("sechs\n");
else printf("nicht fuenf und nicht sechs\n");
```

Schleifen

- Schleifen (Wiederholungen)
  - dienen zur mehrfachen Ausführung von Codeblöcken
  - die Anzahl der Wiederholungen hängt von der SCHLEIFENBEDINGUNG ab
  - die nie enden nennt man ENDLOSSCHLEIFEN

Schleifen

- Syntax:

```
while(Schleifenbedingung){
   mach was
}
```

- Sprich:
  - solange Schleifenbedingung gilt (= wahr), wird der Anweisungsblock {} ausgeführt

Schleifen

- Beispiel: zählen von ____ bis ____

```
int i = 0;

while (i < 9)          // solange i < 9
{
   printf ("%d", i);  // gib i aus
   i=i+1;             // erhöhe i um 1
}
```