1. Grundlagen

1.1. Entwicklungsgeschichte

C ist eine Programmiersprache, welche Dennis Ritchie in den frühen 1970er Jahren für das Betriebssystem Unix entwickelte. Seitdem ist es auf vielen Computer-Systemen verbreitet. Die Anwendungsbereiche von C sind universell. Es wird zur System- und Anwendungsprogrammierung eingesetzt. Die grundlegenden Programme aller Unix/Linux-Systeme sind in C programmiert. Zahlreiche Sprachen, wie C++, Java oder PHP orientieren sich an der Syntax und an anderen Eigenschaften von C.
Seit der Definition der International Standardisation Organisation (ISO) - C gibt es einen einheitlichen Standard für die Programmiersprache C. Die neuesten Standards sind ISO/IEC 9899:1999, auch als C99 bezeichnet. C99 löst den Standard ISO/IEC 9899:1990 (C90) ab. Gängige Bezeichnungen für die Standards sind auch ANSI C bzw. ISO C.

1.2. Bestandteile einer Sprache

So wie in jeder anderen Sprache, existieren auch in einer Programmiersprache fixe Regeln.

Anders ist nur das jeder sprachliche Fehler zu einem nicht funktionierenden Programm führt. Daher ist die exakte Einhaltung der sprachlichen Regeln in einer Programmiersprache extrem wichtig!

Zeichensatz in C

Es gibt nur einen begrenzten Zeichensatz der verwendet werden darf, man nennt diese auch lexikalische Elemente. Man unterscheidet hierbei sichtbare und nicht sichtbare Zeichen:

sichtbare Zeichen

Form	Zeichen
Buchstaben	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k 1 m n o p q r s t u v w x y z
Ziffern	0 1 2 3 4 5 6 7 8 9
Unterstrich	_
Sonderzeichen	! " # % & ' ( ) * + , - . / : ; < = > ? [\ ] ^ { | } ~

nicht sichtbare Zeichen

Zeichen	Bedeutung
Leertaste	Space, Leerzeichen
Warnung	BEL, Klingel, Signalton

Backspace	BS, Rückschritt
Formfeed	FF, Sprung zum nächsten Seitenanfang
Newline	NL, Zeilenende, "Line Feed", Zeilenvorschub
Return	CR, "Carriage Return", Sprung zum Anfang der aktuellen Zeile
Tab	HT, Horizontaler Tabulator

Escape Sequenzen – Steuerzeichen

Zeichen	Escape-Sequenz	Bedeutung
"	\"	Anführungszeichen
'	\'	Apostroph
?	\?	Fragezeichen
\	\\	Backslash
BEL	\a	Bell
BS	\b	Backspace
FF	\f	Formfeed
NL	\n	Newline
CR	\r	Carriage Return
HT	\t	Horizontal-Tabulator
VT	\v	Vertical-Tabulator

Grammatik in C

Die obigen Zeichen werden wie bei jeder anderen Sprache auch zu Wörtern zusammengesetzt, diesen Teil nennt man Syntax. Wie bei jeder anderen Sprache muss man auch bei einer Programmiersprache die einzelnen Wörter und Wortformen auswendig kennen, da man sonst mit dem Computer bzw. dem Betriebssystem nicht kommunizieren kann. Die einzelnen Wörter und Wortformen werden in einer bestimmten standardisierten Weise dargestellt, diese nennt man Extended Backus Naur Form (EBNF) – benannt nach den Erfindern dieser Darstellung oder man gibt diese in Syntaxdiagrammen an.

In C gibt es folgende 32 Wörter:

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned

continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

Die einzelnen Wörter bzw. Wortformen werden zu logischen Sätzen, in einer Programmiersprache spricht man von Strukturen, zusammengesetzt. Diese Strukturen ergeben eine bestimmte logische Bedeutung, man nennt dies auch Semantik. In jeder Programmiersprache unterscheidet man genau drei Grundstrukturen:

Sequenz
Verzweigung
Wiederholung

Grundbegriffe der EBNF

Die folgenden Zeichen dienen zur Darstellung der Syntax einer Programmiersprache:

Definition	=
Endezeichen	;
Logisches Oder	|
Option	[...]
Optionale Wiederholung	{ ... }
Gruppierung	(...)
Anführungszeichen, 1. Variante	" ... "
Anführungszeichen, 2. Variante	' ... '
Kommentar	(* ... *)
Spezielle Sequenz	? ... ?
Ausnahme	-

1.3. Entstehung eines Programms

Ein Computer versteht die Programme, welche in einer Hochsprache wie C geschrieben sind nicht. Diese müssen für den Computer in Maschinensprache übersetzt werden. Diese Aufgabe übernimmt ein Compiler oder ein Interpreter. Jeder Prozessortyp hat seine eigene Maschinensprache, deshalb braucht man für jeden Prozessortyp auch einen eigenen Compiler. Es gibt aber auch plattformunabhängige

Sprachen wie Java. Hier wird der Code zuerst kompiliert – entsteht daraus der plattformunabhängige Bytecode. Dieser Bytecode wird dann von einem plattformabhängigen Interpreter gelesen.

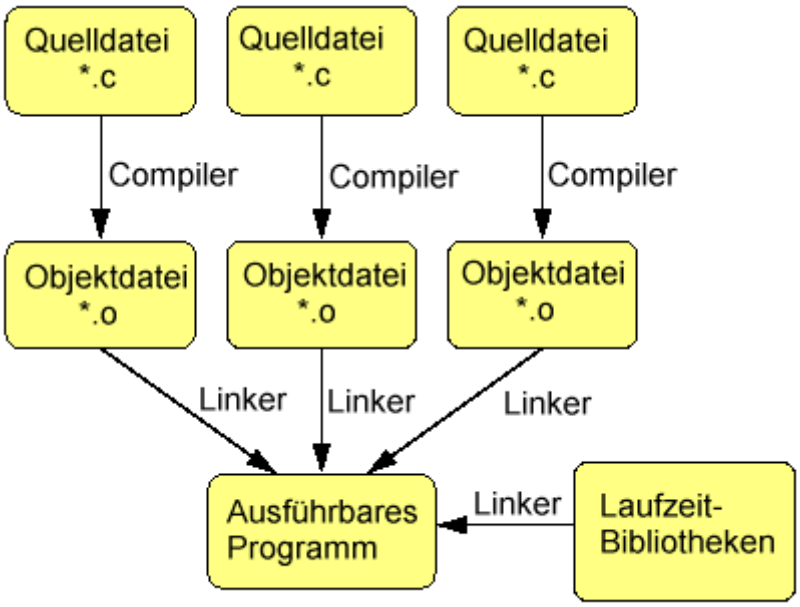

Abbildung 1 Compiler und Linker Vorgang

Der Programmcode wird in einem Editor geschrieben, dies kann auch ein normaler Texteditor sein komfortabler ist eine sogenannte Integrated Development Enviroment, kurz IDE. Als Beispiel zu einer IDE soll hier Eclipse genannt werden. Diese Datei wird mit der Endung „.c“ wird gespeichert, man nennt diese auch Quell- oder Sourcecodedatei. Danach kann diese Datei kompiliert werden. Ein Compiler liest das gesamte Programm ein und übersetzt dieses als Ganzes und erstellt daraus die so genannte Objektdatei mit der Endung „.o“. Ein Interpreter übersetzt nicht das gesamte Programm auf einmal, sondern nur zeilenweise.
Nach dem kompilieren muss noch der so genannte Linker die Laufzeitbibliotheken einbinden und daraus ein funktioniertes Programm machen. Das funktionierende Programm hat die Endung „.exe“.

Was macht der Compiler und Linker genau?

Ein Compiler und Linker sind jeweils zwei eigenständige ausführbare Hilfsprogramme.

Ein Compiler analysiert den Sourcecode in einzelnen Phasen und macht daraus in der Synthesephase Assemblercode bzw. Maschinensprache. Ein Computer kann nur Programme ablaufen lassen, welche in Maschinensprache des jeweiligen Computers vorliegen.

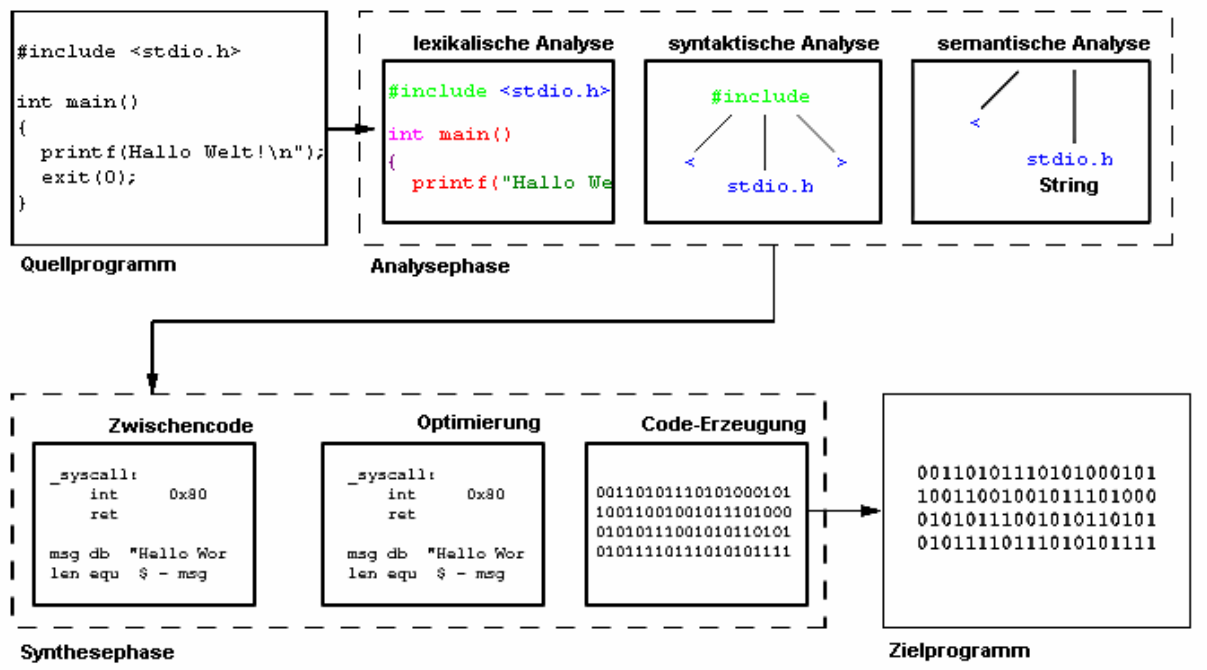

Abbildung 2 Compiler Phasen

Da die meisten Programme Bestandteile enthalten, die auch in anderen Programmen Verwendung finden können, müssen Programme mit dem Linker verlinkt werden. Mehrere kompilierte Prozeduren und Funktionen können zu Laufzeitbibliotheken zusammengefasst werden. Man unterscheidet hierbei so genannte in Linux shared- und static bzw. unter Windows dynamische und statische Bibliotheken. Sie unterscheiden sich nach der Art ihrer Einbindung Der Code wird vom Linker zum Hauptprogramm hinzugefügt, falls die entsprechende Funktion benötigt wird.

1.4. Erstes C Programm

```
# include <stdio.h>

int main(void)
{
     printf("Hallo Welt!");
}
```

Dieses Programm soll den Text „Hallo Welt!“ auf dem Bildschirm (Konsole) ausgeben. Dazu wird zuerst die Funktionsbibliothek „stdio.h“ eingebunden, dies geschieht mit dem Befehl include. Das # gibt an, das es sich dabei nicht um einen C – Befehl, sondern um eine sogennate Präprozessor Anweisung handelt. Der Präprozessor ersetzt gewisse Programmteile, in unserem Fall wird hier die Datei mit dem Namen „stdio.h“ in das Programm eingefügt.

Jedes C Programm verfügt genau über eine Funktion mit dem Namen „main“ – diese beinhaltet den Hauptteil des Programms. Der Bereich der Funktion wird durch das geschwungenen Klammernpaar definiert, man so einen Bereich auch Block.

Ausgabe auf die Konsole mittels der *printf( )* Funktion.

printf(“formatstring“, [argument_1, argument_2, ….., argument_n]);

Der formatstring ist eine Zeichenkette, welche angibt wie die Ausgabe erscheinen soll. Er kann Formatangaben und alle sichtbaren Zeichen enthalten. Formatangaben steuern die Formatierung und Ausgabe der Argumente argument_1 bis argument_n. Achte dass jede Anweisung mit einem Semikolon „ ; “ abgeschlossen wird.

1.5. Programmbibliotheken

Es gibt in jeder Programmiersprachen fertige Laufzeitbibliotheken, welche man in seinen eigenen Programmen verwenden kann. In C sind die wichtigsten in der folgenden Liste angeführt:

Bibliothek	Aufgabenbereiche
assert.h	Überprüfung von Bedingungen
ctype.h	Typkonvertierungen und Typtests
errno.h	Behandlung von Systemfehlern
float.h	Fließkomma-Bibliothek
limits.h	Grenzen für Datentypen
locale.h	Verwaltung der lokalen Struktur
math.h	mathematische Funktionen
signal.h	Prozeßsteuerung
stddef.h	Standardkonstanten
stdio.h	Standardeingabe und -ausgabe
stdlib.h	Standardbibliotheksfunktionen
string.h	Funktionen zur Stringverarbeitung
time.h	Zeitmanagement

Um diese verwenden zu können muss die Headerdatei wie unter Punkt 4 gezeigt eingefügt werden. Diese Bibliotheken enthalten eine Unzahl von vorgefertigten Funktionen.

1.6. Kommentare

Einzeilige Kommentare werden in C mittel *//* eingefügt. Kommentarblöcke oder mehrzeilige Kommentare beginnen mit */** und enden mit **/*.

```
// Dies ist ein einzeiliger Kommentar

/* Dies ist ein Kommentar
   block */
```