Sprachdefinition

Grundsätzlicher Aufbau

- Der Aufbau eines C-Programmes gehorcht grundsätzlichen Regeln
- Erst Include-Anweisungen
- Dann folgt die main-Funktion
- Dann folgt ein Block
- Ein Block besteht immer aus
  - Eingeschlossen von geschweiften Klammern
  - Zuerst einem Satz an Deklarationen für zu benutzende Speicher
  - Dann eine Liste von Anweisungen, wobei eine Anweisung wiederum ein Block sein kann

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    // insert code here...
    printf("Hello, World!\n");
    return 0;
}

```

Beschreibung der Sprache

- Programme sind keine einfachen Sammlungen von Anweisungen
- Sie sind in **Sprachen** verfasst
- Diese Sprachen haben einzelne Worte
  - **Token**, lexikalische Bestandteile
- Für den Aufbau der Sprache gelten bestimmte Regeln
  - **Syntax** „mögliche Reihenfolge der Worte“
- Die Programme können eine Bedeutung haben
  - **Semantik**

C

- Von Brian Kernighan und Dennis Ritchie entwickelt (ca. 1970)
- Die Definition ist komplex und dient nur als Referenz
- In mehreren Runden wurde die Sprache durch Standardisierungsgremien erweitert/ verändert, um sie benutzbarer zu machen
- Compiler prüfen diese Regeln während der Übersetzung und versuchen Fehler möglichst genau zu lokalisieren

Bestandteile von C

- Jede Sprache verfügt neben den Regeln zum Aufbau über eine Reihe vordefinierter Wörter, die eine feste Bedeutung haben
- In C sind dies die folgenden:

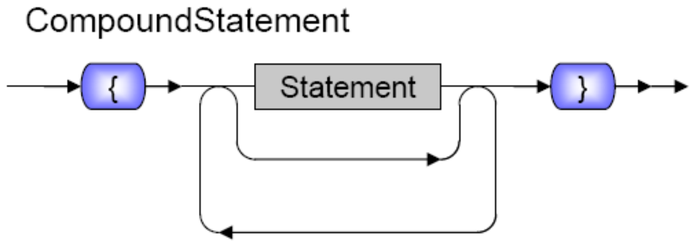
`auto`	`double`	`int`	`struct`
`break`	`else`	`long`	`switch`
`case`	`enum`	`register`	`typedef`
`char`	`extern`	`return`	`union`
`const`	`float`	`short`	`unsigned`
`continue`	`for`	`signed`	`void`
`default`	`goto`	`sizeof`	`volatile`
`do`	`if`	`static`	`while`

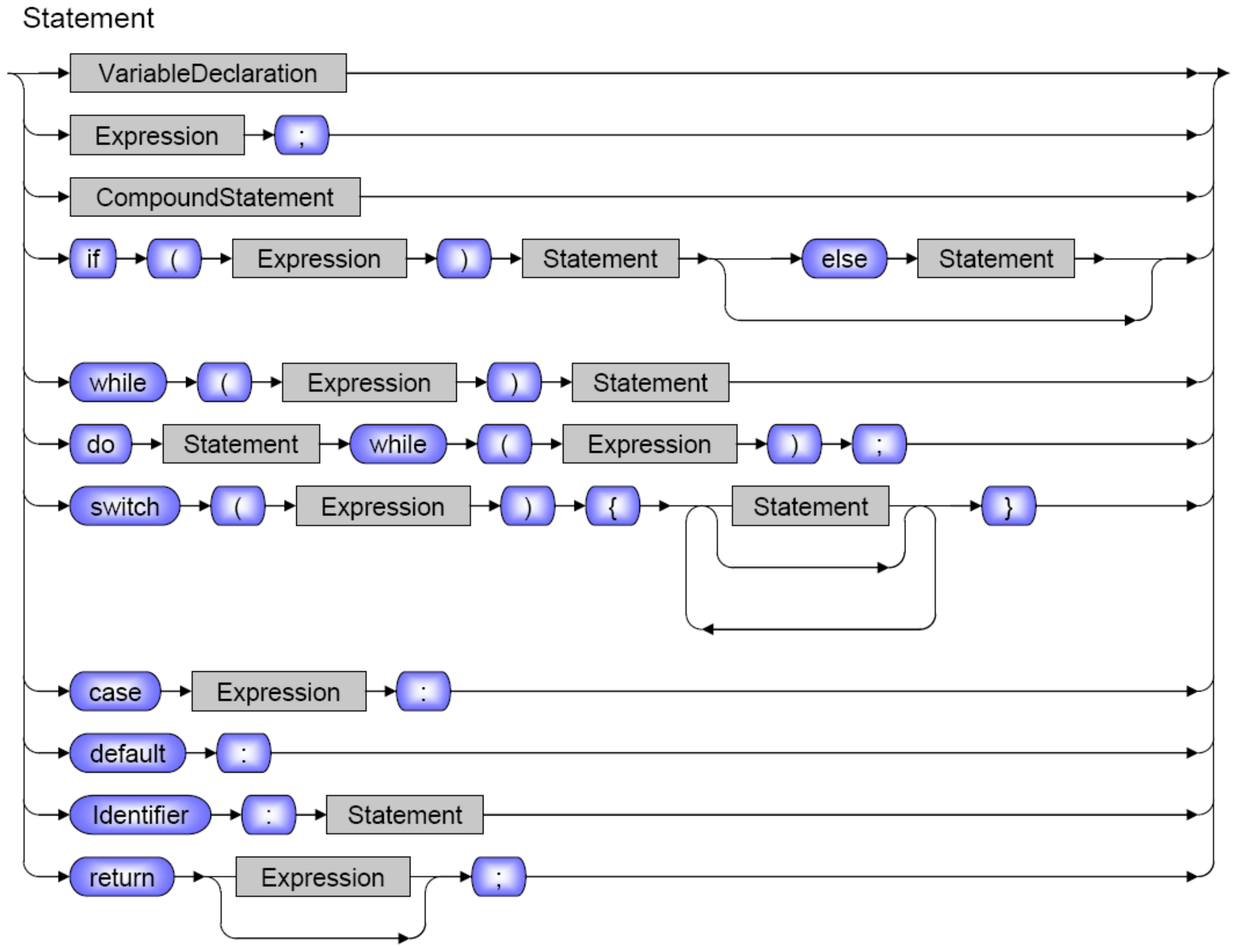
Weitere Token von C

- Literale:
  - Zahlen 12324, Zeichen 'a', Zeichenfolgen " …"
- Trennzeichen
  - ( ) { } [ ] ; .
- Operatoren
  - = > < ! ~ ?
  - : == <= >= != && || ++ --
  - + - * / & | ^ % << >>
  - += -= *= /= &= |= ^= %= <<= >>=
  - -> ,
- Vom Programmierer festgelegte Benennungen
- Kommentare:
  - // … : Alles bis zum Ende der Zeile ignorieren
  - /* … */ : Alles zwischen den Begrenzungen ignorieren

Syntax von C

- Die Syntax einer Sprache wird über eine Grammatik beschrieben:





Semantik einer Sprache

- Die Semantik einer imperativen Programmiersprache wird bei der Übersetzung vom Compiler definiert:
  - Syntax bedeutet, wie Programme auszusehen haben
  - Semantik erklärt, was der Effekt bestimmter Ausdrücke auf der Maschine ist
- Bestimmte Aspekte der Semantik sind aber bereits auf Sprachebene angegeben, da der Benutzer sie kennen muss
  - Vorrangbeziehungen und Auswertungsreihenfolgen („Punkt- vor Strich-Rechnung“)
  - Gültigkeit von Namensräumen

Auswertungsreihenfolgen

	Operatoren	Typ	Auswertung
16	() [] . -> ++ -- ***(postfix-Varianten)*** (type)	postfix	→
15	++ -- ***(prefix-Varianten)*** ! ~ + - & * sizeof	unär	←
14	(type)	unär	←
13	* / %	binär	→
12	+ -	binär	→
11	<< >>	binär	→
10	< <= > >=	binär	→
9	== !=	binär	→
8	& (bitweises und)	binär	→
7	^ (bitweises exklusives oder)	binär	→
6	| (bitweises oder)	binär	→
5	&& (logisches und)	binär	→
4	|| (logisches oder)	binär	→
3	?: (Vergleich und Auswahl)	ternär	←
2	= += -= *= /= %=^= &= |= <<= >>=	binär	←
1	,	binär	→

Namen und ihre Gültigkeit

- Alle in einem Programm deklarierten Dinge müssen benannt werden:
  - int i;
- Namen beginnen mit einem Buchstaben. Danach folgen weitere Buchstaben bzw. Ziffern oder das Zeichen „_“
- Reservierte Worte dürfen nicht zur Benennung benutzt werden
- Im Programm definierte Dinge müssen über ihre Namen angesprochen werden:
  - i = 3;
- Jede Deklaration erfolgt innerhalb von Klammern der Art { }
- Diese Klammern können geschachtelt werden. Die hierdurch bedingten Regeln für die Gültigkeit von Namen innerhalb der Schachtelung sind zu beachten

Namensräume

- Ein durch { } begrenzter Bereich wird auch als **Namensraum** bezeichnet
- Namen gelten in dem durch { } begrenzten Namensraum, **in dem sie definiert wurden**
- Sie gelten auch in **allen darin enthaltenen** Namensräumen
- Weiter innen definierte Namen verdecken außen definierte Namen.