1 Escape sequences

Sequence	ASCII	Description
`\n`	10	new line
`\r`	13	carriage return
`\t`	09	horizontal tab
`\v`	11	vertical tab
`\f`	12	form feed (Curser to start of next page)
`\b`	08	backspace
`\a`	07	bell
`\'`	39	apostrophe
`\"`	34	quote
`\\`	92	backslash
`\nnn`		char value in octal (n = 0..7)
`\xhh`		char value in hex

2 Format

Sequence	Output
%d or %i	int
%c	character
%e or %E	double in format [-] d.ddd e±dd
%f	double in format [-] ddd.ddd
%ot	int as octal
%s	string
%p	As pointer address
%u	unsigned int
%x pr %X	int as hex
%%	%

`%ni` output as int / flush right / n characters wide
`%0ni` output as int / flush right / n characters wide filled with 0
`%.nf` output as float / n digits after comma
`%+i` output as int / forces plus or minus sign
`%#x` int as hex / forces `0x` before number

3 if...else...

- `{...}` is considered a code block
- `<condition> ? <true block> : <false block>`

4 switch...

- Mark when multiple cases point to one block with `//fall through` comment.

5 Syntax description

- EBNF (Erweiterte Backus-Naur-Form)
- Syntax diagram

6 For loop

```
{ init-statement ; while ( condition ) { statement expression ; } }
```

6.1 Weird operators

`i++` → Value is returned before and then incremented
`++i` → Value is first incremented and then returned

7 Casting

```
3/2 // ist 1
3.0/2.0 // ist 1.5

short a, b;
a + b // int
4 / a // int
4 / 2.0 // double
4L / 2.0 // double
4 / 2.0L // long double

int i;
double x;
i = 3 / 2.0; // i ist dann 1 (Zuweisung von 1.5 zu i)
x = 3 / 2; // x ist dann 1.0 (Zuweisung von 1 zu x)

int m=20, n=7, k;
double x;
double pi=3.14;
x = m/n; // 2.0
x = (double)20/7; // 2.8571
x = (double)(20/7); // 2.0
k = m + pi; // 20.0 + 3.14 = 23.14 => k = 23
k = m + (int)pi; // 20 + 3 = 23 => k = 23
```

8 Array

```
int a[5] = {4,7,12,77,2}; /* Alle 5 Elemente werden initialisiert */
int b[] = {4,7,12,77,2}; /* Laenge wird impliziert auf 5 gesetzt */
int c[5] = {4,3,88,5}; /* OK, letztes Element erhaelt Wert 0 */
int d[9] = {0}; /* Alle Elemente 0 */
int d[5] = {4,3,88,5,3,6}; /* Kompilierfehler */
```

8.1 Static Variablen

Static variable bleibt im Memory bestehen und wird zwischen Fkt aufrufen nicht zurück gesetzt wird nur beim ersten mal initialisiert.

9 Floating point

9.1 float

- Schreibweise: `2.0355e2f`
- Size: 4Bytes

The size of the float is 32-bit, out of which:

1. The most significant bit (MSB) is used to store the sign of the number.
2. The next 8 bits are used to store the exponent.
3. The remaining 23 bits are used to store the mantissa.

Converting to Binary form, we get:

$$\begin{aligned} 65 &= 1000001 \\ 0.125 &= 001 \\ \text{So,} & \\ 65.125 &= 1000001.001 \\ &= 1.000001001 \times 10^6 \\ \text{Normalized Mantissa} &= 000001001 \end{aligned}$$

Now, according to the standard, we will get the baised exponent by adding the exponent to 127,

$$\begin{aligned} &= 127 + 6 = 133 \\ \text{Baised exponent} &= 10000101 \end{aligned}$$

And the signed bit is 0 (positive)

So, the IEEE 754 representation of 65.125 is,

```
0 10000101 00000100100000000000000
```

9.2 double

- Literale mit Punkt und Nachkommastellen: 203.55
- Alternativ mit Zehnerpotenz multipliziert: 2.0355e2 (auch E möglich)

The size of the double is 64-bit, out of which:

1. The most significant bit (MSB) is used to store the sign of the number.
2. The next 11 bits are used to store the exponent.
3. The remaining 52 bits are used to store the mantissa.

$$\text{From above,}$$
$$65.125 = 1.000001001 \times 10^6$$
$$\text{Normalized Mantissa} = 000001001$$

Now, according to the standard, bais is 1023. So,

$$= 1023 + 6 = 1029$$
$$\text{Baised exponent} = 10000000101$$

And the signed bit is 0 (positive)
So, the IEEE 754 representation of 65.125 is,

```
0 10000000101 0000010010000000000000000000000000000000000000000000
```

9.3 long double

- Schreibweise: `2.0355e2f`
- Size: 10Bytes

Maximale und minimale Werte in der Datei float.h Ähnlich wie in limits.h für die Ganzzahltypen

10 Priorities

Priority	Symbol	Associativity	Meaning
15	(Postfix) ++, - -	L - R	Increment, Decrement
	(Postfix) - -		Decrement
	( )		Function call
	[ ]		Indexing
14	(Prefix) ++, - -	R - L	Increment, Decrement
	+, - (positiv/negativ number)		positiv/negativ number
	!		logic not
	(Type)		type conversion
	sizeof		
13	*, /, %	L - R	Rechenoperation
12	+, -	L - R	Rechenoperation
10	<	L - R	kleiner
	<=		kleiner gleich
	>		grösser
	>=		grösser gleich
9	==	L - R	gleich
	!=		ungleich
5	&&	L - R	logisches UND
4	||	L - R	logisches ODER
3	?:	R - L	Bedingung
2	=	R - L	Zuweisung
	*=, /=, %=, +=, -=,		Kombinierte Zuweisung
1	,	L - R	Komma-Operator