

Einführung in C - Introduction to C

2. Data types, variables, operators

Prof. Dr. Eckhard Kruse

DHBW Mannheim

Data types, variables ... example (1)

```
int main( ) {
```

```
    int a, b, c, res;
```

Explicit, static typing: Variable definition

```
    a=5;  b=7;  c=10;
```

Initialization (no default values)

```
    res=a/2+b/4+c;
```

```
    printf("result is %d - wow!\n", res);
```

placeholder int

```
}
```

Datatype

Variables (identified by names store values of a given type)

Arithmetic operators

```
>calc.exe
```

```
result is 13 - wow!
```


```
>
```

Data types, variables (2)

Example

```
int main( ) {  
    float a, b, c, res;  
    a=5.0;  b=7.0;  c=10.0;  
    res=a/2+b/4+c;  
    printf("result is %f - wow!\n", res);  
}
```

placeholder float



Datatype

Variables (identified by names store values of a given type)

Arithmetic operators

```
>calc.exe
```

```
result is 14.250000 - wow!
```

```
>
```

Variables: Concepts

Definition

Scope

```
int main( ) {  
    float a, b, c, res;  
    a=5.0;  b=7.0;  c=10.0;  
    res=a/2+b/4+c;  
    printf("result is %f \n", res);  
}
```

← Explicit, static typing: Variable definition

← Initialization (no default values)

Variables:

- Have a **name**
- are of some **type**
- represent memory storing a value of that **type**
- need to be defined before being used (this reserves the memory)
- need to be assigned a value before being used (initialization)
- have a **scope**, i.e. they are visible in certain portions of the program.

Data types

Integer (Ganzzahl)

int standard integer: maximum range depends on CPU

-32768 to +32767 (16 Bit)

-2147483648 to +2147483647 (32Bit)

short, long, long long (C99)

unsigned int, unsigned short, unsigned long

0 to 65535

0 to +4294967295

Floating point number (Gleitkommazahl)

float

double

long double (C99)

Float/double Notation with e or E, e.g.:

$2.308E3 = 2.308 \cdot 10^3 = 2308$

$5.12E-1 = 5.12 \cdot 10^{-1} = 0.512$

Character (Byte)

char

-128 to 127

0 to 255

Typically used for characters, e.g. 'A' = 65, 'B'=66

unsigned char

Boolean Values: true/false are represented as integers (1 / 0)

(C99: `_Bool`, `<stdbool.h>` → `bool`)

Control flow – some basics...

Example



```
#include <stdio.h>

int main()
{
    char c, old_c;
    c=0;
    while(1) { // comments... (see file)
        old_c=c;
        c=c+1;
        if(old_c>c) {
            printf("overflow %d -> %d \n", old_c, c);
        }
    }
}
```

`int_overflow.c`

Code snippet
201

`floating_point.c`

Code snippet
202

for-loop – some basics...

Example



```
#include <stdio.h>

int main()
{
    int i, j;
    char c;
    for(i=0; i<256; i=i+16)
    {
        printf("%3d ", i);
        for(j=0; j<16; j=j+1) {
            c=i+j;
            printf("%c", c);
        }
        printf("\n");
    }
}
```

`ASCII_table.c`

Code snippet
203

Characters - ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Source: www.LookupTables.com

```
int main( ) {  
    float a, b, c, res;  
    a = 5.0;  b = 7.0;  c = 10.0;  
    res = a/2+b/4+c;  
    printf("result is %f", res);  
}
```

Operators (Basics)

Definition

Arithmetic

+ - * /

Basic arithmetic operations

%

Modulo operator

a % b calculates the remainder of **a** divided by **b**
e.g. **13 % 5 == 3**

Comparison

< > <= >=

Less, greater, less or equal, greater or equal

== !=

Equal, not equal

Comparison operators return (int) 1 if the result is true, otherwise 0.

Note: **==** and **=** are very different things!

Logical operators (combining boolean values)

&& ||

!

`ASCII_art.c`

Code snippet
204

Bit-wise operators

& | ^

Bitwise AND, OR, XOR

~

Complement: invert every bit

<<

Shift left

>>

Shift right

Selection

? :

b ? val1 : val2
returns **val2** if **b** is 0,
otherwise **val1**

Example: **max** = **a > b ? a : b ;**

`bitwise_operators.c`

Code snippet
205

Operator priorities

Operators have different priorities and directions of evaluation.
(To change the order use round brackets.)

Priority	Operator	Direction
first	<code>() [] -> . ++ --</code>	from left
	<code>! ~ ++ -- - (type) * & sizeof</code>	from right
	<code>* / %</code>	from left
	<code>+ -</code>	from left
	<code><< >></code>	from left
	<code>< <= > >=</code>	from left
	<code>== !=</code>	from left
	<code>&</code>	from left
	<code>^</code>	from left
	<code> </code>	from left
	<code>&&</code>	from left
	<code> </code>	from left
	<code>? :</code>	from right
	<code>= += -= ...</code>	from right
	<code>,</code>	from left
last		

`operator_priorities.c`

Code snippet
206