LECTURE 8: VARIABLE NAMES (K&R § 2.1)

VARIABLE NAMES

1.	Made up of letters and digits.	
2.	Underscore (_) counts as a letter.	
	This is useful for improving readability of long names.	
	Note:	Do not begin a variable name with underscore. _names are reserved for library routines.
3.	Variable names are case-sensitive (uppercase and lowercase letters are different)	
4.	The first character in the name must be a letter.	

C keywords	reserved by C language cannot be used as variable names must be in lowercase		
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

significant characters	
external variables	function names / global variables (variables defined outside of { })
internal variables	variables defined within { }
At the time of our text's publication, the C standard guaranteed: first 31 characters of an internal variable are significant.	
internal	first 31 characters are significant meaning that the compiler was only guaranteed to treat two variable names as different if they were distinct in the first 31 characters
external	first 6 monocase characters are significant these variables are handled by the linker and were more limited: abcdefg and Abcdef5 would be treated as the same variable.
The current C standard guarantees more. (see C11, § 5.2.4.1)	

LECTURE 8: DATA TYPES AND SIZES (K&R § 2.2)

char (character)

A single byte (8 bits).

Capable of holding 1 character in the local character set.

unsigned	0	≤	char	≤	$2^8 - 1$	
	00000000	≤	char	≤	11111111	
	overflow	255	(255 + 1 = 0)			
	underflow	0	(0 − 1 = 255)			
signed (if supported)	-2^7	≤	char	≤	$2^7 - 1$	
	10000000	≤	char	≤	01111111	
	overflow	127	(127 + 1 = −128)			
	underflow	−128	(−128 − 1 = 127)			

int (integer)

Size is implementation-dependent.

On our machines: 32 bits (stored in 4 sequential bytes)

Reminder:	An integer is a number that can be written without a fractional component (i.e., 0 and the natural numbers, or whole numbers).

unsigned	0	≤	int	≤	$2^{32} - 1$
	0x00000000	≤	int	≤	0xffffffff
	overflow	4294967295	(4294967295 + 1 = 0)		
	underflow	0	(0 − 1 = 4294967295)		
signed	-2^{31}	≤	int	≤	$2^{31} - 1$
	0x80000000	≤	int	≤	0x7fffffff
	overflow	2147483647 (2147483647 + 1 = −2147483648)			
	underflow	−2147483648 (−2147483648 − 1 = 2147483647)			

short (qualifier for ints, means short integer)

On our machines: 16 bits (stored in 2 sequential bytes)

On any machine :	must be at least 16 bits
	cannot be longer than an int

Used for smaller integers.

unsigned	0	≤	short	≤	$2^{16} - 1$
	0x0000	≤	short	≤	0xffff
	overflow	65535	(65535 + 1 = 0)		
	underflow	0	(0 − 1 = 65535)		
signed	-2^{15}	≤	short	≤	$2^{15} - 1$
	0x8000	≤	short	≤	0x7fff
	overflow	32767	(32767 + 1 = −32768)		
	underflow	−32768	(−32768 − 1 = 32767)		

long (qualifier for ints, means long integer)

On our machines: 32 bits (same as int)

On any machine :	must be at least 32 bits
	must be at least as long as an int

float (floating point number)

Represent numbers with fractional parts.

Size is implementation-dependent.

On our machines: 32 bits (stored in 4 sequential bytes)

1 bit	8 bits	23 bits
sign	exponent	coefficient

Based on the IEEE 754 floating point standard.		
Each finite number (may be binary or decimal) is described by three integers:		
1.	s	a sign (zero or one)
2.	c	a significand (coefficient)
3.	q	an exponent
The numerical value of a finite number = $(-1)^s * c * b^q$ where b is the base (2 or 10)		
example: base = 10, sign = 1, significand = 12345, exponent = −3: $-1^1 * 12345 * 10^{-3} = -1 * 12345 * .001 = -12.345$		

1 bit	8 bits	23 bits
1	10000010	10001011000010100011111
	3	12345

double (double-precision floating point number)

A float with more precision.

On our machines: 64 bits (stored in 8 sequential bytes)

signed / unsigned

May be applied to any char or integer.

unsigned	always positive or 0	
	obey the laws of arithmetic modulo 2^n where n is the number of bits in the type (see overflow/underflow)	
	use the most significant bit	
signed	have negative values and positive values	
	treat MSB as bit flag for +/- sign.	
	number is positive	number is negative
sign bit is:	0	1

LECTURE 8: MORE ON CONSTANTS, NUMBERING SYSTEMS (K&R § 2.3)

CONSTANTS

An identifier with an associated value that cannot be altered by the program during normal execution, e.g.,

integer constant	12345
long constant	123456789l or 123456789L
unsigned long int constant	12345UL
double constant	12345. (b/c of decimal point)
double constant	1.2345e2 (b/c of exponent)
float constant	1234.5F (b/c of suffix)

BINARY

Base 2. Binary place values:

N^7	N^6	N^5	N^4	N^3	N^2	N^1	N^0
128	64	32	16	8	4	2	1

C does not have a way to specify binary literals.

OCTAL

Base 8. Octal place values:

N^7	2,097,152	Specify the value of an integer in octal with a leading 0: 037 = 31 in decimal (7 * 1) + (3 * 8) = 31
N^6	262,144	
N^5	32,768	
N^4	4,096	
N^3	512	
N^2	64	
N^1	8	
N^0	1	

Caution:	You cannot write a decimal number with a leading 0. It will be interpreted as octal.

HEXADECIMAL (HEX)

Base 16. Uses letters for digits > 9:

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
										10	11	12	13	14	15

Hexadecimal place values:

N^7	268,435,456	Specify the value of an integer in hexadecimal with a leading 0x or 0X: 0x1f = 31 in decimal 0X1F = 31 in decimal (15 * 1) + (1 * 16) = 31
N^6	16,777,216	
N^5	1,048,576	
N^4	65,536	
N^3	4,096	
N^2	256	
N^1	16	
N^0	1	

USEFUL CONVERSIONS TO KNOW

decimal	binary	hexadecimal	octal
0	0000	0X0	00
1	0001	0X1	01
2	0010	0X2	02
3	0011	0X3	03
4	0100	0X4	04
5	0101	0X5	05
6	0110	0X6	06
7	0111	0X7	07
8	1000	0X8	010
9	1001	0X9	011
10	1010	0XA	012
11	1011	0XB	013
12	1100	0XC	014
13	1101	0XD	015
14	1110	0XE	016
15	1111	0XF	017

CHARACTER CONSTANT

An integer, written as one character within single quotes.

The value of a character constant is the numeric value of the character in the machine's character set.

'a'	integer value in ASCII code for letter a
'0'	integer value in ASCII code for number 0
'\b'	integer value in ASCII code for backspace
'\ooo'	octal value 000 – 377 (0 – 255 decimal)
'\xhh'	hex value 0x00 – 0xff (0 – 255 decimal)

STRING CONSTANT

Remember distinction between character constant and string constant. 'x' ≠ "x"

'x'	"x"
An integer used to produce the numeric value of the letter x in the machine's character set.	An array of characters that contains one character (the letter x) and a '\0' (the null terminator).

ENUMERATION CONSTANT

enumeration	a list of constant integer values e.g., enum boolean {NO, YES}; The names have values 0, 1, and so on.

Explicit values can also be specified, e.g.:
enum escapes = {BELL = '\a', [...], RETURN = '\r'};