••• PASCAL TO C CONVERSIONS†

INTEGER	int
LONGINT	long
CHAR	int
BOOLEAN	char
Byte	Byte(struct), int(passed)
VAR Byte	int *
Handle	Handle
VAR Handle	Handle *
Ptr	Ptr
VAR Ptr	Ptr *
OSType, ResType	long
PACKED ARRAY[]	long
String255	Str255 or char *
VAR String255	Str255 or char *
StringPtr	StrPtr or char *
VAR StringPtr	StrPtr * or char *
Rect	Rect *
VAR Rect	Rect *
Point	Point
VAR Point	Point *

••• DATA TYPES AND SIZES (IN BYTES)†

char	1
short	2
int	2
long	4
float	4
short double	8
double	10 (12 w/68881 option)
Ptr	4
Handle	4
Byte	1
Boolean	1
Str255	256 (4 when passed)

••• OBJECT PASSING

VAR Parameter	a pointer to the object
4 bytes or smaller	the object
larger than 4 bytes	a pointer to the object

••• STRUCT AND PTR RELATIONS

```
structure.item
         == (*structure_ptr).item
== structure_ptr->item
*(character_ptr)
        == character_ptr[0]
*(character_ptr + n)
 == character_ptr[n]
```

••• OPERATOR ASSOCIATIVITY AND PRECEDENCE

```
LR
        () [] -> .
RL
        ! ~ ++ -- - * & (type) sizeof
LR
        * / %
```

LR

+ -

LR

<< >>

LR

< <= > >=

LR

== !=

LR

&

LR

^

LR

|

LR

&&

LR

||

RL

?:

RL

= += -= *= /= %= &= ^= |= <<= >>=

LR

,

••• CHARACTER CONSTANTS

newline	NL(LF)	\n
horizontal tab	HT	\t
vertical tab	VT	\v
backspace	BS	\b
carriage return	CR	\r
formfeed	FF	\f
alert	BEL	\a
backslash	\	\\
question mark	?	\?
single quote	'	\'
double quote	"	\"
octal number	ooo	\ooo
hexadecimal number	xhh	\xhh
NUL character	NUL	\0

••• PREPROCESSOR COMMANDS

#define	identifier token-sequence
#define	identifier (identifier-list) token-sequence
#undef	identifier
#include	<filename>
#include	"filename"
#include	token-sequence
#line	constant "filename"
#line	constant
#error	token-sequence(opt)
#pragma^	token-sequence(opt)
#	NULL
#if	constant-expression
#ifdef	identifier
#ifndef	identifier
#elif	constant-expression
#else	
#endif	

••• RESERVED KEYWORDS

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
asm†	pascal†		

••• STANDARD LIBRARY

<assert.h>	<float.h>	<math.h>	<stdarg.h>	<stdlib.h>
<ctype.h>	<limits.h>	<setjmp.h>	<stddef.h>	<string.h>
<errno.h>	<locale.h>	<signal.h>	<stdio.h>	<time.h>

••• SCANF: <stdio.h>

%d	decimal integer	int *
%i	integer	int *
%o	octal integer	int *
%u	unsigned decimal integer	unsigned int *
%x	hexadecimal integer	int *
%c	characters	char *
%s	string	char[] or char *
%e,f,g	floating-point number	float *
%p	pointer	void *
%%	literal %	

••• PRINTF: <stdio.h>

%d,i	decimal notation	int
%o	unsigned octal	int
%x,X	unsigned hexadecimal	int
%u	unsigned decimal	unsigned int
%c	character	int or char
%s	string	char[] char *
%f	decimal notation	double or float
%e,E	exponential notation	double or float
%g,G	use shorter of %e or %f	double or float
%p	pointer	void *
%%	literal %	

••• MATHEMATICAL FUNCTIONS: <math.h>

sin(x)	sine of x
cos(x)	cosine of x
tan(x)	tangent of x
asin(x)	sin-1(x)
acos(x)	cos-1(x)
atan(x)	tan-1(x)
atan2(y,x)	tan-1(x/y)
sinh(x)	hyperbolic sine of x
cosh(x)	hyperbolic cosine of x
tanh(x)	hyperbolic tangent of x
exp(x)	exponential function
log(x)	natural logarithm
log10(x)	base 10 logarithm
pow(x,y)	x to the power of y
sqrt(x)	square root of x

ceil(x)	smallest integer not less than x
floor(x)	largest integer not greater than x
fabs(x)	absolute value
fmod(x,y)	floating-point remainder of x/y

••• STRING FUNCTIONS: <string.h>

strcat(s,t)	concat t to the end of s
strncat(s,t,n)	concat n characters of t to end of s
strcmp(s,t)	<0 if(s<t), 0 if(s==t), >0 if(s>t)
strncmp(s,t,n)	same as strcmp but only in first n characters
strcpy(s,t)	copy t into s
strncpy(s,t,n)	copy at most n characters of t into s
strlen(s)	return length of s
strchr(s,c)	return pointer to first c in s, else NULL
strrchr(s,c)	return pointer to last c in s, else NULL

••• CHARACTER CLASS TESTS: <ctype.h>

isalpha(c)	non-zero if c is alphabetic; 0 if not
isupper(c)	non-zero if c is upper case; 0 if not
islower(c)	non-zero if c is lower case; 0 if not
isdigit(c)	non-zero if c is a digit(0..9); 0 if not
isalnum(c)	non-zero if isalpha(c) or isdigit(c); 0 if not
isspace(c)	non-zero if c is blank, \t, \n, \r, \f, \v
toupper(c)	return c converted to upper case
tolower(c)	return c converted to lower case

••• ABOUT C QUICK REFERENCE

C Quick Reference (CQR) is compiled by Stephen D. Krans. CQR is FREE so please distribute it unmodified to anyone interested in writing C Code. If you have any comments, questions, or suggestions, please contact me through one of the following electronic addresses:

GEnie: SKRANS
CIS: 76474,757
AOL: SKRANS

Many thanks to Bill Steinberg for writing DisplayDA! Click on the "About DisplayDA" box below to learn more about it.

••• EVOLUTIONS

Version: 1.0
Date: Thursday, December 6, 1990
Comments: First release

••• ENDNOTES

^ Not supported by THINK C 4.0.2.
† THINK C 4.0.2 specific; may differ on other compilers.