C Preprocessor

line continuation char

• •	•
name defined?	$\mathtt{defined}(name)$
is name defined, not defined?	#ifdef, #ifndef
conditional execution	#if, #else, #elif, #endif
concatenate args and rescan	##
Example. #define msg(A)	printf("%s = %d", #A, (A))
quoted string in replace	#
undefine	#undef $name$
Example. #define max(A,1	B) ((A)>(B) ? (A) : (B))
replacement macro	#define name(var) text
replacement text	#define $name\ text$
include user file	#include "filename"
include library file	<pre>#include <filename></filename></pre>

${\bf Data\ Types/Declarations}$

_ , , _, ,	
size of a data type (type is size_t)	${ t sizeof}(type)$
size of an object (type is size_t)	$\verb"sizeof" object"$
create new name for data type	typedef type name;
structure	<pre>struct tag {};</pre>
no value	void
local persistent between calls	static
internal to source file	static
declare external variable	extern
constant (read-only) value	type const $name;$
enumeration constant enum tag	$\{name_1 = value_1, \dots\};$
pointer to int, float,	int*, float*,
non-negative modulo 2^m	unsigned
positive or negative	signed
double long (64 bit integer)	long long
long (32 bit integer)	long
short (16 bit integer)	short
real number (single, double precision)	float, double
integer	int
character (1 byte)	char

Initialization

10 Oct 2008 – 2 Mar 2021

initialize char string	<pre>char name[]="string";</pre>
initialize array	$type name[]=\{value_1, \dots\};$
initialize variable	$type \ name = value;$

© 2007 Joseph H. Silverman Permissions on back. v2.2

```
unsigned member: b;
bit field with b bits
single object, multiple possible types
                                        union
    Example. (*p).x and p->x are the same
member of pointed-to structure
                                        pointer -> member
member of structure from template
                                        name.member
                                        {\tt struct}\ tag\ name
create structure
                           declaration of members
       declarations \\
    struct tag {
                           structure template
Structures
multi-dim array
                                     name [dim_1] [dim_2] \dots
                                        name [dim]
array
address of object name
                                        &name
```

Operators (grouped by precedence)

expression evaluation separator	,
assignment operators	+=, -=, *=,
conditional expression	$expr_1$? $expr_2$: $expr_3$
logical or	11
logical and	&&
or (inclusive) [bit op]	<u> </u>
exclusive or [bit op]	•
and [bit op]	&
equality comparisons	==, !=
relational comparisons	>, >=, <, <=
left, right shift [bit ops]	<<, >>
add, subtract	+, -
multiply, divide, modulus (remainder)	*, /, %
size of an object	sizeof
cast expression to type	(type) $expr$
indirection via pointer, address of obje	•
plus, minus, logical not, bitwise not	+, -, !, ~
increment, decrement	++,
struct member through pointer	pointer->member
struct member operator	name . $member$
oborgania (Sroabog p)	procedence)

ators group right to left; all others group left to right.

Unary operators, conditional expression and assignment oper-

ANSI Standard Libraries

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

Character Class Tests <ctype.h>

	_
convert to upper case	toupper(c)
convert to lower case	tolower(c)
nexadecimal digit?	<pre>isxdigit(c)</pre>
upper case letter?	<pre>isupper(c)</pre>
space, formfeed, newline, cr, tab, vtab?	isspace(c)
printing char except space, letter, digit?	<pre>ispunct(c)</pre>
printing character (incl space)?	<pre>isprint(c)</pre>
lower case letter?	islower(c)
printing character (not incl space)?	isgraph(c)
decimal digit?	isdigit(c)
control character?	<pre>iscntrl(c)</pre>
alphabetic?	isalpha(c)
alphanumeric?	isalnum(c)

String Operations <string.h>

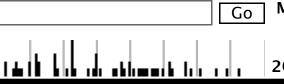
s is a string; cs, ct are constant strings

out c into first n chars of s	memset(s,c,n)
pointer to first c in first n chars of cs	memchr(cs,c,n)
ompare n chars of cs with ct	<pre>memcmp(cs,ct,n)</pre>
opy n chars from ct to s (may overlap)	memmove(s,ct,n)
opy n chars from ct to s	<pre>memcpy(s,ct,n)</pre>
pointer to last c in cs	strrchr(cs,c)
pointer to first c in cs	<pre>strchr(cs,c)</pre>
only first n chars	strncmp(cs,ct,n)
ompare cs to ct	strcmp(cs,ct)
oncatenate ct after s	<pre>strcat(s,ct)</pre>
opy ct to s	<pre>strcpy(s,ct)</pre>
$\text{ength of } \mathbf{s}$	strlen(s)



http://www.digilife.be/quickreferences/QRC/C Reference Card (ANSI) 2.2.pdf

86 captures







▼ About this capture