C PROGRAMMING LANGUAGE REFERENCE CARD

STATEMENT FORMATS

```
/* comments in C are enclosed by slash-star and star-slash */
i = 1;                                  simple statements are terminated with a semicolon
;                                       statements may have null body
{ tmp = a; a = b; b = tmp; }            compound statements are within braces
                                        and used wherever a simple statement is allowed
if (a < 0) a = -a;                      perform statement if condition is true
else printf ("was plus \n");            optional else after if
while (i < MAX) a [i++] = 0;            perform statement while condition is true
for (i=0; i < MAX; i++) a[i] = 0;       perform initialization once, then
                                        statement and increment while condition is true
do c = getchar( ); while (c == ' ');    perform statement until condition
                                        false, test done at bottom of loop
switch (getchar()) {                    evaluate expression and goto
      case 'X': exit (0);               appropriate case statement
      case 'H': help ( ); break;        if no break would fall into next case
      case 'A': case 'B': arg++; break; multiple cases allowed
      default: printf("try again \n");  default if no case matched
}                                       end switch
break;                                  terminate smallest enclosing while, do, for, or switch
continue;                               goto bottom of loop in while, do, or for
return A;                               exit function and return optional expression to caller
goto error;                             unconditional jump to statement preceeded with label
error : printf("INVALID FRAMUS/n"); exit(1);   label marks statement
```

PREPROCESSOR COMMANDS

```
#define TRUE 1                          substitute optional string for identifier
#define NEG(x) (-(x))                   substitute expanded macro for identifier
#undef DEBUG                            forget previous define
#if MODE == 1                           compile if constant expression is true
#ifdef DEBUG                            compile if identifier is defined
#ifndef TEST                            compile if identifier is not defined
#else                                   compile if previous if condition false
#endif                                  terminates conditional compile
#include "local.h"                      replace this line with contents of file
#include <stdio.h>                      replace this line with contents of system file
#line 100 test3                         renumber and optional rename for diagnostic printouts
```

CONSTANTS

1234	decimal number	1234L	long decimal number
0xaa55	hexadecimal number	0xaa55L	long hexadecimal number
0177	octal number	0177L	long octal number
32.5	float number	1.2e-5	scientific notation
'a'	character	"abcd"	null terminated string

SPECIAL CHARACTERS

'\n'	newline	'\r'	carriage return
'\t'	tab	'\f'	form feed
'\b'	backspace	'\\'	backslash
'\''	single quote	'\ddd'	octal constant

VARIABLE DECLARATIONS

```
char   a;                               signed, one byte
int    i, j, k;                         signed integers
long   sum;                             signed large integer
short  x, y;                            signed small integers
unsigned limit = 0xffff;                unsigned integer, initialized
float  matrix[10] [50];                 two dimensional array of floating points
double big;                             large floating point
```

Note: **short int, long int, unsigned int, long float** are valid, some compilers accept other combinations such as **unsigned char**.

```
char  *ptr;                             variable ptr points to data of type char
register  short quick;                  advises that variable is often used
extern  int  flag,  open ( );           variable and function in other modules
static  char  here_to_stay;             local permanent storage
auto  long  amnesia;                    dynamic storage, default for function variables
char  msg[ ] = "HELP \n";               initialized array
struc  name {                           definition of complex data type, name
      char  first[10];                  with members, employee.first,
      char  last[20];                   employee.last,
      unsigned  sex  :  1;              and the bit field employee.sex
} employee;                             declaration of variable employee of type struct name
union  kludge {                         defines an overlay of different data types
      char  c;                          the member mixed.c shares its storage area
      float  f;                         with the longer member mixed.f
} mixed;                                declaration of a variable mixed
typedef  char  *string;                 creates a new variable type name, string
```


OPERATOR PRECEDENCE

PRIMARY EXPRESSION — LEFT TO RIGHT

( )	[ ]	.	–>
function	array element	structure member	structure pointer

UNARY OPERATORS — RIGHT TO LEFT

*	&	–	!	~	++	––	sizeof ( )
indirect	address	minus	negate	1's comp	inc	dec	cast

BINARY OPERATORS — decreasing precedence — LEFT TO RIGHT

```
* multiply   / divide   % modulus
+ add   – subtract
>> shift right      << shift left
< less than   > greater than   <= less or equal   >= greater or equal
== equals   != not equals
& bitwise and
^ bitwise exclusive or
| bitwise or
&& logical and
|| logical or
```

CONDITIONAL EXPRESSION — RIGHT TO LEFT

condition ? true : false

ASSIGNMENT OPERATORS — RIGHT TO LEFT

= += -= *= /= %= >>= <<= &= ^= |= see BINARY OPERATORS

COMMA OPERATOR — LEFT TO RIGHT

, discards value of left expression

FORMATTED I/O

printf (format, exp1, exp2, ...)	to standard output
fprintf (stream, format, exp1, exp2, ...)	to specified output
sprintf (buffer, format, exp1, exp2, ...)	to string buffer
scanf (format, addr1, addr2, ...)	from standard input
fscanf (stream, format, addr1, addr2, ...)	from specified input
sscanf (buffer, format, addr1, addr2, ...)	from string buffer

Note that destination addresses are required by **scanf, fscanf,** and **sscanf**

Format string consists of text to be printed or matched containing format specifiers.
A format specifier has the form:

% [–] [*] [W] [.M] [l] <conversion character>

where:

–	forces left justification (**printf** only)
*	assignment suppression (**scanf** only)
W	width in characters (leading 0 means zero pad)
M	precision (**printf** only)
l	letter l - specifies long integer or double

conversion characters:

d	signed decimal integer
u	unsigned decimal integer (**printf** only)
x	unsigned hexadecimal integer
h	unsigned short integer (**scanf** only)
o	unsigned octal integer
c	single character
s	null terminated string
f	fixed point notation for float or double
e	scientific notation for float or double (**printf** only)
g	use **%e** or **%f**, whichever is shorter (**printf** only)

UNIX* I/O CALLS

Note: unless specified below, arguments and return values are **int**'s.

```
char buffer[], ch, *name, *ptr, *s_mode;
long offset;
struct stat *stat_buf;
FILE *stream;
```

```
open (name, mode);   0: read,  1: write,  2: both
read (filedes, buffer, count);                  write (filedes, buffer, count);
long lseek (filedes, offset, from);   0: begin,  1: current,  2: end
creat (name, mode);                             close (filedes);
FILE * fopen (name, s_mode);   "r": read,  "w": write,  "a": append
FILE * freopen (name, s_mode, stream);          FILE * fdopen (filedes, s_mode);
fread (ptr, item_size, count, stream);          char * gets (buffer);
fwrite (ptr, item_size, count, stream);         puts (buffer);
getc (stream);                                  getchar ( );
putc (ch, stream);                              putchar (ch);
fseek (stream, offset, from);                   fclose (stream);
```

* UNIX is a Trademark of Bell Laboratories

The C PROGRAMMING LANGUAGE REFERENCE CARD is available
from SSC, PO Box 806, Mercer Island, WA 98040
for $2.50 each or 2 for $4.00 postage paid.
Copyright 1983, Cscapes, PO Box 2233, Everett, WA 98203. All rights reserved.