C Programming Cheatsheet

String Format Specifiers

%[flags][width][.precision][length]specifier

%c	char
%hhd||%hhi	signed char
%hhu	unsigned char
%hhn	signed char*
%lc	wint_t
%ls	wchar_t*
%s	string
%d || %i	signed int
%u	unsigned int
%hi	short int
%hu	unsigned short int
%hn	short int*
%l	signed long int
%ln	long int*
%ll	signed long long int
%lln	long long int*
%llu	unsigned long long int
%f || %F	float or double (%F is uppercase)
%Lf || %Le	long double
%e || %E	scientific notation (mantissa/exponent)
%g || %G	shortest representation of %e||%E
%o	octal unsigned int
%x	lowercase hex unsigned int
%X	uppercase hex unsigned int
%a || %A	hexadecimal float-point
%ji	intmax_t
%ju	uintmax_t
%jn	intmax_t*
%zi || %zu	size_t || ssize_t
%zn	size_t*
%ti || %tu	ptrdiff_t
%tn	ptrdiff_t*
%p	pointer address
%n	NULL
%%	literal %
Width and Precision	
%.3f	float precision of 3 (like 3.141)
%4d	4 digit wide int (like 2015)
%2.2f	2 digits wide and 2 precise (19.95)
Flags	
-	Left-justify
+	Right-justify
SPACE	Blank space
#	Preceded hex & octal with "0x" || "0"
0	Left-pad with zeros

Integer from variable - printf("%d", num);
Save integer to variable - scanf("%d", &num);
Save string to variable - scanf("%s", str_var);

Character Escapes

- **\0** - NULL
- **\b** - backspace
- **\f** - form feed (new page)
- **\n** - newline
- **\r** - carriage return
- **\t** - tab
- **\v** - vertical tab

Arithmetic Operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus/Remainder
++	Increment by 1
--	Decrement by 1
++>	Pre-increment and compare
-->	Pre-decrement and compare

Equality Operators

==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Logical Operators

Operand	Meaning	Example
&&	And	(x && y)
| |	Or	(x || y)
!	Not	!(x < y)

Bitwise Operators

&	AND
|	OR
^	Exclusive OR (XOR)
~	Ones Complement (NOT)
<<	Left-shift
>>	Right-shift

Assignment Operators

Operand	Meaning	Equivalent
=	Assign	None
+=	Add	X = X + Y
-=	Subtract	X = X - Y
*=	Multiply	X = X * Y
/=	Divide	X = X / Y
%=	Modulus	X = X % Y
<<=	Left-shift	X = X << Y
>>=	Right-shift	X = X >> Y
&=	AND	X = X & Y
|=	OR	X = X | Y
^=	XOR	X = X ^ Y

Constructs

Do-While Loop

```
i=0;
do {
    printf("%d\n", i);
    ++i; } while(i<10);
```

For Loop

```
for (i=0; i<10; ++i) {
    printf("%d\n", i);
}
```

While Loop

```
i=0;
while (i<10) {
    printf("%d\n", i);
    ++i;
}
```

If, else if, else

```
if (0 == 1) {
    register signed int ZERO = 0;
} else if (8 <= 4) {
    const float PIf = 3.14F;
} else {
    static char SYM[3] = "π\0";
}
```

Macros if

```
#ifdef __linux__
#   include "custom_header.h"
#   include <system_header.h>
#endif
```

Switch-case

```
switch (INPUT) {
    case 0: break;
    default:
        break;
}
```

Ternary Operator

```
int out = (input == 7 ? 5 : 3);
```

Define Data-Type

```
typedef struct {
    int x, y;
} point_t;
typedef union {
    int i; double d;
} number_t;
```

Define Enum

```
enum cardsuit {
    CLUBS,
    DIAMONDS,
    HEARTS,
    SPADES,
};
```

Define Struct

```
struct intpair { int x, y; };
```

Define Union

```
union num_var {
    int i; double d;
};
```

Variable Aliases and Constants

```
const double PI = 3.14159;
const double *ARCHIMEDES_NUM = &PI;
extern const double PI;  // In Header
char PI_SYM[3] = "π\0";  // Unicode
```

Arrays

```
double num[2] = {3.14, 5.0};
unsigned int LargeArray[2][4] = {
    {0, 1, 2, 3}, {4, 5, 6, 7}};
char words[2][] = {
    "Linux", "Solaris"
};
```

C Programming Cheatsheet

Data Types

NULL	void
_Bool	bool - <stdbool.h>
char16_t - <uchar.h>	char32_t - <uchar.h>
char	double
enum	EOF - <stdio.h>
FILE - <stdio.h>	fpos_t - <stdio.h>
float	imaxdiv_t - <inttypes.h>
int	long
long double	long int
long long	long long int
nullptr_t - <stddef.h>	ptrdiff_t - <stddef.h>
sig_atomic_t - <signal,h>	short
short char	short int
size_t - <stddef.h>	ssize_t - <stddef.h>
struct	union
wctrans_t - <wchar.h>	wctype_t - <wctype.h>
wchar_t - <wchar.h>	__ibm128
WEOF - <wchar.h>	wint_t - <wchar.h>
signed	unsigned
signed char	unsigned char
signed int	unsigned int
signed long	unsigned long
signed long int	unsigned long int
signed long long	unsigned long long
signed long long int	unsigned long long int
signed short	unsigned short
signed short int	unsigned short int
__float80	__float128
<complex.h>	
complex	_Complex
float complex	float _Complex
double complex	double _Complex
long double complex	long double _Complex
imaginary	_Imaginary
float imaginary	float _Imaginary
double imaginary	double _Imaginary
long double imaginary	long double _Imaginary
_Complex80	_Complex128
<stdint.h>	
intmax_t	uintmax_t
int8_t	uint8_t
int16_t	uint16_t
int32_t	uint32_t
int64_t	uint64_t
int_least8_t	uint_least8_t
int_least16_t	uint_least16_t
int_least32_t	uint_least32_t
int_least64_t	uint_least64_t
int_fast8_t	uint_fast8_t
int_fast16_t	uint_fast16_t
int_fast32_t	uint_fast32_t
int_fast64_t	uint_fast64_t
intptr_t	uintptr_t

Literal Constant Suffixes

unsigned	U || u
unsigned long long	ULL
long	L
float	F
double	D
long double	L
__float80	W || w
__float128	Q || q
__ibm128	W
_Imaginary	i
_Complex128	KC
exponent	E
__Decimal32	df || DF
__Decimal64	dd || DD
__Decimal128	dl || DL
short _Fract || _Sat short _Fract	HR || hr
_Fract || _Sat _Fract	R || r
long _Fract || _Sat long _Fract	lr || LR
long long _Fract || _Sat long long _Fract	llr || LLR
unsigned short _Fract || _Sat unsigned short _Fract	uhr || UHR
unsigned _Fract || _Sat unsigned _Fract	ur || UR
unsigned long _Fract and _Sat unsigned long _Fract	ulr || ULR
unsigned long long _Fract || _Sat unsigned long long _Fract	ullr || ULLR
short _Accum || _Sat short _Accum	hk || HK
Accum || _Sat _Accum	k || K
long _Accum || _Sat long _Accum	lk || LK
long long _Accum || _Sat long long _Accum	llk || LLK
unsigned short _Accum || _Sat unsigned short _Accum	uhk || UHK
unsigned _Accum || _Sat unsigned _Accum	uk || UK
unsigned long _Accum || _Sat unsigned long _Accum	ulk || ULK
unsigned long long _Accum || _Sat unsigned long long _Accum	ullk || ULLK

https://gcc.gnu.org/onlinedocs/gcc/Fixed-Point.html

Literal Constant Prefixes

Octal	0
Binary	0b
Hexadecimal	0x
char	\u
wchar_t string	L
UTF-8 string	u8
UTF-16 string	u
Unicode string	U
Raw literal string	R"delimiter(STRING)delimiter"

Storage Classes

- **auto** - Stored in stack during the code-block
- **extern** - Lasts the whole program, block, or compilation unit; globally visible
- **register** - Stored in stack or CPU-register during the code block
- **static** - Lasts the whole program, block, or compilation unit; private in program
- **typedef** - The data specifies a new datatype
- **__thread** - Thread-local-storage; one instance per thread
- **_Thread_local** - Thread-local data

Type Qualifiers

- **const** - Value does not change; read-only
- **restrict** - For the lifetime of the pointer, the object can only be accessed via the pointer
- **volatile** - Optimizing-compilers must not change
- **_Atomic** - Map a variable to one of the 5 basic built-in types

Function Specifiers

- **inline** - Inline the function when compiling
- **__inline__** - Same as "inline"
- **_Noreturn** - The function does not return

Function Attributes (__attribute__(()))

GNU-GCC only

Use in function declaration (header)

https://gcc.gnu.org/onlinedocs/gcc/Function-Attributes.html

https://gcc.gnu.org/onlinedocs/gcc/Common-Function-Attributes.html

- **alias** - The function is an alias for another; Example: void f () __attribute__ ((weak, alias ("__f")));
- **aligned** - Set alignment
- **always_inline** - Inline the function despite optimization options
- **cold** - Unlikely to execute; used for optimizations
- **constructor** - Call function before main()
- **destructor** - Call function after main()
- **deprecated** - Emit warning msg when called
- **error** - Emit error message when called
- **flatten** - Inline all functions in the function; __attribute__((flatten))
- **hot** - Very likely to execute; used for optimizations
- **nonnull** - None of the input pointers are NULL
- **nothrow** - The function is guaranteed not to throw an exception
- **optimize** - Set specific optimization options for the function
- **pure** - The function accepts arguments, has single return, and has no other effects
- **returns_twice** - The function returns two values separately
- **simd** - Create multiple functions that can process arguments using SIMD instructions
- **warning** - Emit warning message when called

Created by Devyn Collier Johnson <DevynCJohnson@Gmail.com> (2016v2) More cheatsheets at DCJTech.info

C Programming Cheatsheet

Type Attributes

GNU-GCC only
https://gcc.gnu.org/onlinedocs/gcc/Type-Attributes.html

- **aligned** - Set alignment
- **deprecated** - Emit warning msg when called
- **mode** - Set type mode. Example: `typedef _Complex float __attribute((mode(TC))) _Complex128;`
- **packed** - Each member of the struct or union is placed to minimize the memory required; for enums, use the smallest integral type
- **unused** - Inform the compiler that members of a struct or union may appear unused, but that is fine; i.e. the compiler will not issue warnings

Variable Attributes

GNU-GCC only
https://gcc.gnu.org/onlinedocs/gcc/Variable-Attributes.html

- **aligned** - Set alignment
- **common** - Place variable in "common" storage; the common section of an object-file
- **deprecated** - Emit warning msg when called
- **nocommon** - Allocate space for the variable directly
- **unused** - Inform the compiler that members of a struct or union may appear unused, but that is fine; i.e. the compiler will not issue warnings
- **vector_size** - Set the variable's size in bytes and then divide it into parts (based on the datatype); A size of 4 and a type of "char" would make the variable contain four "char" values

Special Macros and Keywords

- **__asm__** - Inline assembly code
- **__attribute__** - Function attribute
- **__auto_type** - Duck typing
- **__extension__** - Inform compiler that the following code is a GCC extension
- **__GNUC__** - GNU-GCC compiler
- **__label__** - Create a local label by declaring it in the beginning of the scope (__label__ label;); then, place the actual label where needed (label:;)
- **__restrict__** - There is only one pointer to the referenced object; Example: `int FUNC(char *__restrict__ DATA) {}`

https://gcc.gnu.org/onlinedocs/gcc/C-Extensions.html
&&label - Address of *label*
typeof(*x) y - Declare y with x's type

Machine Modes

- **BI** - 1 Bit
- **QI** - Quarter Integer; 1 byte
- **HI** - Half Integer; 2 bytes
- **PSI** - Partial Single Integer; 4 bytes; not all bits used
- **SI** - Single Integer; 4 bytes
- **PDI** - Partial Double Integer; 8 bytes; not all bits used
- **DI** - Double Integer; 8 bytes
- **TI** - Tetra Integer; 16 bytes
- **OI** - Octa Integer; 32 bytes
- **QF** - Quarter Floating; 1 byte quarter-precision float-point
- **HF** - Half Floating; 2 byte half-precision float-point
- **TQF** - Three Quarter Floating; 3 byte three-quarter-precision float-point
- **SF** - Single Floating; 4 byte single-precision float-point
- **DF** - Double Floating; 8 byte double-precision float-point
- **XF** - Extended Floating; 12 byte extended-precision float-point
- **TF** - Tetra Floating; 16 byte tetra-precision float-point
- **CQI** - Complex Quarter Integer; 1 byte
- **CHI** - Complex Half Integer; 2 bytes
- **CSI** - Complex Single Integer; 4 bytes
- **CDI** - Complex Double Integer; 8 bytes
- **CTI** - Complex Tetra Integer; 16 bytes
- **COI** - Complex Octa Integer; 32 bytes
- **QC** - Quarter Complex; 1 byte quarter-precision complex float-point
- **HC** - Half Complex; 2 byte half-precision complex float-point
- **SC** - Single Complex; 4 byte single-precision complex float-point
- **DC** - Double Complex; 8 byte double-precision complex float-point
- **XC** - Extended Complex; 12 byte extended-precision complex float-point
- **TC** - Tetra Complex; 16 byte tetra-precision complex float-point
- **CC** - Condition Code
- **BLK** - Block
- **VOID** - Void
- **P** - Address mode
- **V4SI** - Vector; 4 single integers
- **V8QI** - Vector; 8 single-byte integers
- **BND32** - 32-bit pointer bound
- **BND64** - 32-bit pointer bound

https://gcc.gnu.org/onlinedocs/gccint/Machine-Modes.html

Printing Width-based Integrals

Datatype	Print Macros
int8_t	PRId8
uint8_t	PRIu8
int16_t	PRId16
uint16_t	PRIu16
uint64_t	PRIu64
intmax_t	PRIdMAX
int_least32_t	PRIdLEAST32
u int_fast32_t	PRIuFAST32
intptr_t	PRIdPTR

Replace "PRI" with "SCN" in scanf()

C POSIX Library

- <aio.h> - Asynchronous I/O
- <arpa/inet.h> - Functions for manipulating numeric IP addresses (part of Berkeley sockets)
- <assert.h> - Macros assertions
- <complex.h> - Arithmetic with complex numbers
- <cpio.h> - Magic numbers for the cpio archive format
- <dirent.h> - Functions for opening and listing directories
- <dlfcn.h> - Dynamic linking
- <errno.h> - Retrieving Error Number
- <fcntl.h> - File opening, locking, and other file operations
- <fenv.h> - Floating-Point environment
- <float.h> - Floating Types
- <fmtmsg.h> - Message display structures
- <fnmatch.h> - Filename matching
- <ftw.h> - File tree traversal
- <glob.h> - Pathname pattern-matching (globbing)
- <grp.h> - User group information and control
- <iconv.h> - Codeset conversion facility
- <inttypes.h> - Fixed-size integer data-types
- <iso646.h> - Alternative spellings
- <langinfo.h> - Language information constants
- <libgen.h> - Pathname manipulation
- <limits.h> - Implementation-defined constants
- <locale.h> - Category macros
- <math.h> - Mathematical and trigonometric functions
- <monetary.h> - Monetary unit string formatting
- <mqueue.h> - Message queue
- <ndbm.h> - NDBM database operations
- <net/if.h> - List local network interfaces
- <netdb.h> - Translating protocol and hostnames into numeric addresses
- <netinet/in.h> - Internet protocol and address family definitions
- <netinet/tcp.h> - Additional TCP control options
- <nl_types.h> - Localization message catalog functions
- <poll.h> - Asynchronous file descriptor multiplexing
- <pthread.h> - API for creating and manipulating POSIX threads
- <pwd.h> - passwd and user information access and control
- <regex.h> - Regular expression matching
- <sched.h> - Execution scheduling
- <search.h> - Search tables
- <semaphore.h> - POSIX semaphores
- <setjmp.h> - Stack environment declarations
- <signal.h> - Signals
- <spawn.h> - Process spawning
- <stdarg.h> - Handle Variable Argument List
- <stdbool.h> - Boolean type and values
- <stddef.h> - Standard Type Definitions
- <stdint.h> - Integer Types
- <stdio.h> - Standard Buffered I/O
- <stdlib.h> - Standard Library Definitions
- <string.h> - Several String Operations
- <strings.h> - Case-insensitive string comparisons
- <stropts.h> - Stream manipulation and ioctl
- <sys/ipc.h> - Inter-process communication (IPC)
- <sys/mman.h> - Memory management, POSIX Shared Memory, and Memory-mapped files
- <sys/msg.h> - POSIX message queues
- <sys/resource.h> - Resource usage, priorities, and limiting
- <sys/select.h> - Synchronous I/O multiplexing
- <sys/sem.h> - XSI (SysV style) semaphores
- <sys/shm.h> - XSI (SysV style) Shared Memory
- <sys/socket.h> - Main Berkley sockets header
- <sys/stat.h> - File information
- <sys/statvfs.h> - Filesystem information
- <sys/time.h> - Time and date functions and structures
- <sys/times.h> - File access and modification times
- <sys/types.h> - Various data-types
- <sys/uio.h> - Vectored I/O operations
- <sys/un.h> - Unix domain sockets
- <sys/utsname.h> - Operating system information and uname
- <sys/wait.h> - Status of terminated child processes
- <syslog.h> - System error logging
- <tar.h> - Magic numbers for the tar archive format
- <termios.h> - Terminal I/O interfaces
- <tgmath.h> - Type-Generic math macros
- <time.h> - Type-Generic time macros
- <trace.h> - Tracing of runtime behavior
- <ulimit.h> - Resource limiting (DEPRECATED; use <sys/resource.h>)
- <unistd.h> - Various essential POSIX functions and constants
- <utime.h> - Inode access and modification times
- <utmpx.h> - User accounting database functions
- <wchar.h> - Wide-Character handling
- <wctype.h> - Wide-Character classification and mapping utilities
- <wordexp.h> - Word-expansion like the shell would perform

Created by Devyn Collier Johnson <DevynCJohnson@Gmail.com> (2016v2) More cheatsheets at DCJTech.info