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)
I I	Or	(x || y)
!	Not	!(x < y)

Bitwise Operators

&	AND
I	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
I=	OR	X = X I 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

```
register int i=0;
while (i<10) { ++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);
```

Goto

```
label:
goto label;
```

Define Datatype

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

Define Enum

```
enum cardsuit {
    CLUBS = 0,
    DIAMONDS, HEARTS, SPADES
};
```

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
char PI_UTF8[] = u8"π\0";
char16_t PI_UTF16[] = u"π\0";
char32_t PI_UTF32[] = U"π\0";
```

Arrays

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

Order of Operations

() [] -> . ::	! ~ - + * & ++ --
* / %	+ -
<< >>	< <= > >=
!= ==	& (Bitwise)
^ (Bitwise)	| (Bitwise)
&& || (Logical)	Ternary operator
Assignment	Comma Operator

Created by Devyn Collier Johnson <DevynCJohnson@Gmail.com> (2017) More cheatsheets at DCJTech.info