C Programming Cheat Sheet

By Jim Hall

C is a straightforward compiled programming language. Other programming languages borrow concepts from C, which makes C a great starting point if you want to learn programming languages such as Lua, C++, Java, or Go.

Basics

Include header files first, then define your global variables, then write your program.

```
/* comment to describe the program */
#include <stdio.h>
/* definitions */
int main(int argc, char **argv) {
   /* variable declarations */
   /* program statements */
}
```

Variables

Variable names can contain uppercase or lowercase letters (A to Z, or a to z), or numbers (0 to 9), or an underscore (_). Cannot start with a number.

`int`	Integer values (-1, 0, 1, 2, …)
`char`	Character values, such as letters
`float`	Floating point numbers (0.0, 1.1, 4.5, or 3.141)
`double`	Double precision numbers, like float but bigger

Functions

Indicate the function type and name followed by variables inside parentheses. Put your function statements inside curly braces.

```
int celsius(int fahr) {
  int cel;
  cel = (fahr – 32) * 5 / 9;
  return cel;
}
```

Allocate memory with **malloc**. Resize with **realloc**. Use **free** to release.

```
int *array;
int *newarray;

arr = (int *) malloc(sizeof(int) * 10);
if (arr == NULL) {
   /* fail */
}

newarray = (int *) realloc(array,
sizeof(int) * 20);

if (newarray == NULL) {
   /* fail */
}
arr = newarray;

free(arr);
```

C Programming Cheat Sheet

By Jim Hall

Binary operators

`a & b`	Bitwise AND (1 if both bits are 1)
`a | b`	Bitwise OR (1 if either bits are 1)
`a ^ b`	Bitwise XOR (1 if bits differ)
`a<<n`	Shift bits to the left
`a>>n`	Shift bits to the right

Assignment shortcuts

`a += b;`	Addition	`a = a + b;`
`a -= b;`	Subtraction	`a = a - b;`
`a *= b;`	Multiplication	`a = a * b;`
`a /= b;`	Division	`a = a / b;`
`a %= b;`	Modulo	`a = a % b;`

Useful functions <stdio.h>

`stdin`	Standard input (from user or another program)
`stdout`	Standard output (print)
`stderr`	Dedicated error output

```
FILE *fopen(char *filename, char *mode);

size_t fread(void *ptr, size_t size,
size_t nitems, FILE *stream);
int fclose(FILE *stream);

int puts(char *string);
int printf(char *format, …);
int fprintf(FILE *stream, char *format);
int sprintf(char *string, char *format);

int getc(FILE *stream);
int putc(int ch, FILE *stream);

int getchar();
int putchar(int ch);
```

Useful functions <stdlib.h>

```
void *malloc(size_t size);
void *realloc(void *ptr, size_t newsize);

void free(void *ptr);

void qsort(void *array, size_t nitems,
size_t size, int (*compar)(void *a, void
*b));

void *bsearch(void *key, void *array,
size_t nitems, size_t size, int (*compar)
(void *a, void *b));

void srand(unsigned int seed);

void rand();
```

Always test for **NULL** when allocating memory with ***malloc*** or ***realloc***.

If you ***malloc*** or ***realloc***, you should also ***free***. But only free memory once.