C Reference Cheat Sheet

by chris23balln via cheatography.com/69951/cs/17730/

Main Function

int main(int argc, char *argv[]){return int;}

Function Arguments	Description
int argc	Number of command line args
char *argv[]	Command line args in an array *name of program*

Data Types

Type	Bytes	Range
char	1	U:(0 to 2^8-1) S:(-2^7 to 2^7-1)
short	2	U:(0 to 2^{16}-1) S:(-2^{15} to 2^{15}-1)
int	4	U:(0 to 2^{31}-1) S:(-2^{31} to 2^{32}-1)
long	4	U:(0 to 2^{31}-1) S:(-2^{31} to 2^{32}-1)
float	4	6 decimal places
long long	8	U:(0 to 2^{64}-1) S:(-2^{31} to 2^{32}-1)
double	8	15 decimal places
long double	10	19 decimal places

Math

#include <math.h>

Function	Description
sin(x) sinh(x) asin(x)	Sine of x Hyperbolic sine of x Arc sine of x
cos(x) cosh(x) acos(x)	Cosine of x Hyperbolic cosine of x Arc cosine of x
tan(x) tanh(x) atan(x)	Tangent of x Hyperbolic tangent of x Arc tangent of x
exp(x)	Returns value of e raised to the xth power
log(x) log10(x)	Natural logarithm (base-e) of x Base-10 of x

Math (cont)

pow(x,y)	Returns x raised to the power of y
sqrt(x)	Square root of x
abs(x)	Absolute value of x (<stdlib.h>)

Standard Library

#include <stdlib.h>

Function	Description
rand()	Returns a random long
qsort(array, length, size, compr)	Quick Sort "array" of array "length" with elements of "size" by function "compr"

Compr Function

int cmpfunc(const void * a, const void * b); {return (*(int*)a - *(int*)b);}

Char Library

#include <ctype.h>

tolower(char)	Lowercase char
toupper(char)	Uppercase char
islower(char)	Checks if char is lowercase
isupper(char)	checks if char is uppercase
isnumber(char)	Checks if char is 0-9
isalpha(char)	Checks if char is a letter
isblank(char)	Checks if char is whitespace

Strings

#include <string.h>

Function	Description
strlen(str)	Return length of string "str"
strcat(dest,src)	Appends "src" to end of the string "dest"
strncat(dest,src,n)	Appends "src" to end of the string "dest" by upto "n" characters long
strcpy(dest,str)	Copy string "str" to "dest" and return "dest"

Strings (cont)

strncpy(dest,src,n)	Copy "n" characters from "src" to "dest", return "dest"
strcmp(str1,str2)	Compares "str1" to "str2"
strncmp(str1,str2,n)	Compares the first "n" bytes of "str1" to "str2"
strtok(str,delim)	Breaks string "str" into a series of tokens separated by "delim"
memcpy(dest,src,n)	Copies "n" characters from "src" to "dest"
memset(str,c,n)	Copies the char "c" to the first "n" characters of the string "str"

Arrays

Declaration	
type arrayName[size]; type arrayName[size][size];	1-Dimensional 2-Dimensional
Initialization	
int myArray[2] = {3,4}; int myArray[2] = {3}	Each element All elements 3
Accessing	
int a = myArray[c];	Value at "c" position
&myArray[c]	Memory address at "c" position
Array Size	
sizeof(myArray);	Returns length of array
Passing to Function	
void foo(int *myArray)	Passed as pointer to array
void foo(int myArray[])	Passed array

Pointers

A pointer is a variable who's value is an address of another variable

In Arrays

Keywords (Non-trivial)

Keyword	Definition
auto	Defines variables as having a local lifetime
default	Declaration of a default case
extern	Extend the visiblity of the variables and functions
register	Hints to compiler that a given variable can be put in a register
union	Allows storage of different data types in same memory location
volatile	Used when a variable can change unexpectedly
typedef	Assigns alternative names to existing types
static	Preserves value even after out of scope
enum	Used to assign names to integral constants
continue	Forces the next iteration in a loop

Memory

Function	Description
calloc(n,size)	Alloc array of "n" elements each of size in bytes "size" p=(T*)calloc(n,sizeof(t))
malloc(n)	Alloc array of "n" bytes p=(T*)malloc(sizeof(t))
realloc(addr,size)	Re-allocates memory extending it upto "size"
free(addr)	Releases block of memory specified by "addr"

File Input / Output

#include <stdio.h>

Function	Description
fopen(filename,mode)	Open file/create new (Mode: r,w,a,r+,w+,a+)
fclose(FILE *fp)	Closes file(Retuns 0 for success, EOF for failure)
fputc(int c, FILE *fp)	Writes character vaue to output stream pointed by fp
fputs(const char *s, FILE* fp)	Writes string to output stream pointed by fp
fgetc(FILE * fp)	Reads a character from input stream fp
fgets(char *buf, int n, FILE* fp)	Reads n characters from input stream fp into buf

Standard Input / Output

Function	Descrition
getchar()	Reads next available character from screen
putchar(int c)	Puts character "c" on screen
gets(char *s)	Reads line from stdin to buffer "s"
puts(const char *s)	Writes string "s" to stdout
scanf(const char *format,..)	Scans input from stdin into format
printf(const char *format,..)	Writes output to stdout according to format