Cheatography

C program Cheat Sheet

by Arnezzi (genta) via cheatography.com/164651/cs/34496/

Console Input/Output

#include <stdio.h>

Character

getchars{}	Returns a single character's ANSI code from the input stream buffer as an integer. (safe)
putchar(int)	Prints a single character from an ANSI code integer to the output stream buffer.

Strings

gets(strName)	Reads a line from the input stream into a string variable. (Unsafe, removed in C11.)
Alternative	
fgets(strName, length, stdin);	Reads a line from the input stream into a string variable. (Safe)
puts("string")	Prints a string to the output stream.

Formatted Data

scanf("%d", &x)	Read value/s (type defined by format string) into variable/s (type must match) from the input stream. Stops reading at the first whitespace. & prefix not required for arrays (including strings.) (unsafe)
printf("I love %c %d!", 'C', 99)	Prints data (formats defined by the format string) as a string to the output stream.
Alternative	

Console Input/Output (cont)

fgets(strName, length, stdin); sscanf(strName, "%d", &x);	Uses fgets to limit the input length, then uses sscanf to read the resulting string in place of scanf. (safe)

Dynamic Memory

Remember to #include <stdio.h>

Allocate

malloc	ptr = malloc(n *sizeof* ptr);
calloc	ptr = calloc(n, sizeof *ptr);

Change Size

realloc	newsize = n *sizeof* ptr; tmp = realloc(ptr, newsize); if (tmp) ptr = tmp; else / *ptr is still valid* /;

Release

free	free(ptr);

Escape Character

\a	alarm (bell/beep)	\b	backspace
\f	formfeed	\n	newline
\r	carriage return	\t	horizontal tab
\v	vertical tab	\\	backslash
\'	single quote	\"	double quote
\?	question mark		
\nnn	Any octal ANSI character code.		
\xhh	Any hexadecimal ANSI character code.		

Data type

Character	char	1 byte	-128 to 127
	unsigned char	1 byte	0 to 255
Integer	int	4 byte	-32,767 to 32,767
	unsigned int	4 byte	0 to 65,535

Data type (cont)

	short int	2 byte	-32,767 to 32,767
	unsigned short int	2 byte	0 to 65,535
	long int	4 byte	-2,147,483,647 to 2,147,483,647
	unsigned long int	4 byte	0 to 4,294,967,295
	long long int	8 byte	-(263 – 1) to 263 – 1 (It will be added by the C99 standard)
	unsigned long long int	8 byte	264 – 1 (It will be added by the C99 standard)
Float	float	4 byte	1E-37 to 1E+37 along with six digits of the precisions here
	double	8 byte	1E-37 to 1E+37 along with six digits of the precisions here
	long double	8 byte	1E-37 to 1E+37 along with six digits of the precisions here

File Input/Output

#include <stdio.h>	
Opening	
FILE *fptr = fopen(filename, mode);	
FILE *fptr	Declares fptr as a FILE type pointer (stores stream location instead of memory location.)
fopen()	Returns a stream location pointer if successful, 0 otherwise.
filename	String containing file's directory path & name.

File Input/Output (cont)

mode	String specifying the file access mode.
Modes	
"r" / "rb"	Read existing text/binary file.
"w" / "wb"	Write new/over existing text/binary file.
"a" / "ab"	Write new/append to existing text/binary file.
"r+" / "r+b" / "rb+"	Read and write existing text/binary file.
"w+" / "w+b" / "wb+"	Read and write new/over existing text/binary file.
"a+" / "a+b" / "ab+"	Read and write new/append to existing text/binary file.
Closing	
fclose(fptr);	Flushes buffers and closes stream. Returns 0 if successful, EOF otherwise.
Random Access	
ftell(fptr)	Return current file position as a long integer.
fseek(fptr, offset, origin);	Sets current file position. Returns false is successful, true otherwise. The offset is a long integer type.
Origins	
SEEK_SET	Beginning of file.
SEEK_CUR	Current position in file.
SEEK_END	End of file.
Utilities	
feof(fptr)	Tests end-of-file indicator.
rename(strOldName, strNewName)	Renames a file.
remove(strName)	Deletes a file.

File Input/Output (cont)

Characters	
fgetc(fptr)	Returns character read or EOF if unsuccessful. (safe)
fputc(int c, fptr)	Returns character written or EOF if unsuccessful.
Strings	
fgets(char *s, int n, fptr)	Reads n-1 characters from file fptr into string s. Stops at EOF and \n. (safe)
fputs(char *s, fptr)	Writes string s to file fptr. Returns non-negative on success, EOF otherwise.
Formatted Data	
fscanf(fptr, format, [...])	Same as scanf with additional file pointer parameter. (unsafe)
fprintf(fptr, format, [...])	Same as printf with additional file pointer parameter.
Alternative	
fgets(strName, length, fptr); sscanf(strName, "%d", &x);	Uses fgets to limit the input length, then uses sscanf to read the resulting string in place of scanf. (safe)
Binary	
fread(void *ptr, sizeof(element), number, fptr)	Reads a number of elements from fptr to array *ptr. (safe)
fwrite(void *ptr, sizeof(element), number, fptr)	Writes a number of elements to file fptr from array *ptr.

Variabels

Declaring	
int x;	A variable.
char x = 'C';	A variable & initialising it.
float x, y, z;	Multiple variables of the same type.
const int x = 88;	A constant variable: can't assign to after declaration (compiler enforced.)
Naming	
johnny5IsAlive ✔	Alphanumeric, not a keyword, begins with a letter.
2001ASpaceOddysey X	Doesn't begin with a letter.
while X	Reserved keyword.
how exciting! X	Non-alphanumeric.
iamaverylongvariablenameohmygoshyesiam X	Longer than 31 characters (C89 & C90 only)

Input Format

Specifier	Input Text is a	Destination type
%c	Character	Char
%d	Decimal	Int, short
%x	Hexadecimal	Int, short, long
%ld	Long Decimal	Long
%lld	Very Long Decimal	Long Long
%f	Floating-point	Float
%lf	Floating-point	Double
%le	Exponential	Double

Read file line-by-line

```
#include <stdio.h>

FILE h;
char line[100];
h = fopen("filename", "rb");
/error checking missing /
while (fgets(line, sizeof line, h)) {
/deal with line /
}
/if needed test why last read failed /
if (feof(h) || ferror(h)) /whatever */;
fclose(h);
```

C program Cheat Sheet

by Arnezzi (genta) via cheatography.com/164651/cs/34496/

main()

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

Anatomy

int main	Program entry point.
int arcg	# of command line arguments.
char *argv[]	Command line arguments in an array of strings. #1 is always the program filename.
return int;	Exit status (integer) returned to the OS upon program exit.

Command Line Arguments

app two 3	Three arguments, "app", "two" and "3".
app "two 3"	Two arguments, "app" and "two 3".

main is the first function called when the program executes.

Placeholder Types (f/printf And f/scanf)

printf("%d%d...", arg1, arg2...);

Type	Example	Description
%d or %I	-42	Signed decimal integer.
%u	42	Unsigned decimal integer.
%o	52	Unsigned octal integer.
%x or %X	2a or 2A	Unsigned hexadecimal integer.
%f or %F	1.21	Signed decimal float.
%e or %E	1.21e+9 or 1.21E+9	Signed decimal w/ scientific notation.
%g or %G	1.21e+9 or 1.21E+9	Shortest representation of %f/%F or %e/%E.
%a or %A	0x1.207c8ap+30 or 0X1.207C8AP+30	Signed hexadecimal float.
%c	a	A character.
%s	A string.	A character string.
%p		A pointer
%%	&	A percent character.

Placeholder Types (f/printf And f/scanf) (cont)

%n	No output, saves # of characters printed so far. Respective printf argument must be an integer pointer.

Comments

```
// We're single-line comments!
//Nothing compiled after // on these lines.

/* I'm a multi-line comment!
Nothing compiled between
these delimiters. */
```

Escape Characters

\a	alarm (bell/beep)
\f	formfeed
\r	carriage return
\v	vertical tab
\'	single quote
\?	question mark
\nnn	Any octal ANSI character code.
\xhh	Any hexadecimal ANSI character code.
\b	backspace
\n	newline
\t	horizontal tab
\\	backslash
\"	double quote

Strings

`'A'` character	Single quotes.
`" AB"`string	Double quotes.
`\0`	Null terminator.

Strings are `char` arrays.

`char name[4] = " Ash ";`

is equivalent to

`char name[4] = {'A', 's', 'h', '\0'};`

`int i; for(i = 0; name[i]; i++){}`

`\0` ***evaluates as false.***

Strings must include a `char` element for `\0`.