

Cheat Sheet C

Preprocessor

#include <stdio.h>	// Insert standard header file
#include "my file.h"	// Insert file in current directory
#define add(a,b) a+b	// define a Makro
#ifndef file_name	// conditional compilation
#define file_name	// for header files
#endif	//

Characters

%c	// single char
%s	// string
%d	// decimal integer
%f	// floating point value
%ld	// long
%o	// octal integer
%x	// hexadecimal
%%	// percent char

Variable Types

char	// 1 Byte
short	// 2 Byte
Int	// 4 Byte
float	// 4 Byte float
long	// 8 Byte int
double	// 8 Byte float

Qualifier

const type	// read only
static type	// function keeps values between
invocation	

Typecasting

(type)a	// return s as datatype
typedef old new	// changing name
atoi(string)	// s -> int
atof(string)	// s -> double
atol(string)	// s -> long

Struct

struct name{	// struct with member a
type a	
};	
struct structName varName	// initialize
varName.a	// accessing
ptrName->a	// accessing if pointer

Enum

enum name {	// creating own datatype
name,name2	
};	
enum name varName;	// accessing

Pointer

type *ptr	// creating
ptr	// memory address
*ptr	// value
&varName	// memory address normal variable

Array

```
type name[int]           // set array length
name[pos]                // accessing arr at pos
*(name+pos)              // accessing arr at pos
sizeof(array) / sizeof(array[0]) // return size
```

Strings

```
c[4] = "Ash";             //strings are char arrays
c[4] = { `A`, `s`, `h` }; // equivalent
\0                        //Null-terminated == false
```

Functions

```
type funcName(type var){ //declaration

}
by Value                 // passing Value
by reference             // passing memAddress
```

Main

```
Int main(int argc, char *argv[]){ //main method call
    return 0;
}
argc                          // Number of command line args
*argv[pos]                    // actual arguments
*c = *(arg+pos)                // get input at pos
```

Consol I/O

```
printf("Hello World");       // output
getc(stream)                  // return single char from consol
putc(int,stream)              // print asci char to consol
scanf("%s" buffer)            // write input to buffer
```

File I/O

EOF // end of file
fgets(buffer,MAX,Stream) // read line from stream
sscanf(str,"%c",&i); // scanning stream

File Opening

FILE *ptr = fopen(fileName,mode); // set ptr to start of file
fclose(ptr) // close after usage
ftell(ptr) // return position
fseek(ptr,offset,origin) // sets current file pos

Allocation

type *ptr = (type*)malloc(size) // allocation
realloc(ptr,size) // resize
free(ptr) // release memory