

Console Input/Output	Console Input/Output (cont)	Data Types (cont)	Derived Data Types								
<pre>#include &lt;stdio.h&gt;</pre> <p><b>Formatted Data</b></p> <p><code>scanf()</code> Read value/s (type defined by format string) into variable/s (type must match) from the input stream. Stops reading at the first whitespace. <code>&amp;</code> prefix not required for arrays (including strings.) (unsafe)</p> <p><code>printf()</code> Prints data (formats defined by the format string) as a string to the output stream</p> <p><b>Alternative</b></p> <p><code>fgets( - str Name, length, stdin);</code> Uses <code>fgets</code> to limit the input length, then uses <code>sscanf</code> to read the resulting string in place of <code>scanf</code>. (safe)</p>	<p><b>Characters</b></p> <p><code>getchar()</code> Returns a single character's ANSI code from the input stream buffer as an integer. (safe)</p> <p><code>putchar( r(int))</code> Prints a single character from an ANSI code <i>integer</i> to the output stream buffer.</p> <p><b>Strings</b></p> <p><code>fgets( - str Name, length, stdin);</code> Reads a line from the input stream into a string variable. (Safe)</p> <p><code>puts("string")</code> Prints a string to the output stream.</p> <p><b>Data Types</b></p> <p>Basic Data Type: Floating-point, integer, double, character</p> <p>Derived Data Type: Union, structure, array, etc</p>	<p>Enumerated Data Type: Enums</p> <p>Void Data Type: Empty Value</p> <p>Boolean Type: True or False</p> <p><b>Primary Data Types</b></p> <p>Integer: <code>int</code> Storing Whole Numbers</p> <p>Character: <code>char</code> Refers to all ASCII character sets and single alphabets</p> <p>Long: <code>long</code> Long integer</p> <p>Floating point: <code>float</code> Refer to all the real number value or decimal point</p> <p>Double (Long float): <code>double</code> Include all large type of numeric that do not come under floating point or integer</p> <p>Void: <code>void</code> No value</p>	<p><b>Data Type Description</b></p> <p>Arrays: A Sequence of a finite number of data items</p> <p>Function: A self-contained block of single or multiple statements.</p> <p>Pointers: Special form of variables for holding other variables' addresses.</p> <p>Unions: The memory that we allocate to the largest data type gets reused for all the other types present in the group.</p> <p>Structures: A collection of various different types of data type items</p> <p><b>Standard Library Functions</b></p> <pre>#include&lt;...&gt;</pre> <table border="1"> <thead> <tr> <th>Header File</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td><code>stdio.h</code></td> <td>Standard input/output header file</td> </tr> <tr> <td><code>conio.h</code></td> <td>Console input/output header file</td> </tr> <tr> <td><code>string.h</code></td> <td>String related functions are defined in this header file</td> </tr> </tbody> </table>	Header File	Description	<code>stdio.h</code>	Standard input/output header file	<code>conio.h</code>	Console input/output header file	<code>string.h</code>	String related functions are defined in this header file
Header File	Description										
<code>stdio.h</code>	Standard input/output header file										
<code>conio.h</code>	Console input/output header file										
<code>string.h</code>	String related functions are defined in this header file										



### Standard Library Functions (cont)

string.h	Contains general functions used in C
math.h	All math related functions are defined here
time.h	Contains time and clock related functions
ctype.h	All character handling functions are defined here
stdarg.h	Variable argument functions are declared here
signal.h	Signal handling functions are declared here
setjmp.h	Includes all jump functions
locale.h	Includes locale functions
errno.h	Includes error handling functions
assert.h	Includes diagnostic functions

### Commenting

//	Insert single-line comment
example:	
//This is a single line comment	

### Commenting (cont)

/*	Insert multiple-line comment
example:	
/*This is a multiple line comment*/	

### Conditional Statements

if	used to specify a block of code to be executed, if a specified condition is true
else	used to specify a block of code to be executed, if the same condition is false
else if	used to specify a new condition to test, if the first condition is false
switch	Used instead of writing many if...else statements
break	Used to stop the execution of more code and case testing inside the block

### Looping

while	Loops through a block of code as long as a specified condition is true
-------	--

### Looping (cont)

do/while	This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true
for	Loops through a block of code for a specified amount of repetitions
continue	Breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop
break	Skips rest of loop contents and exits loop

### Keywords

auto	double	int
break	else	long
case	enum	register
char	extern	return
const	float	short
continue	for	signed
default	goto	sizeof
do	if	static

### Control characters (Escape sequences)

\a	Alert (bell) character
\b	Backspace
\f	Formfeed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\\	Backslash
\?	Question mark
\'	Single quote
\"	Double quote
\0	null
\nnn	Any octal ANSI character code
\xhh	Any hexadecimal ANSI character code

### Arithmetic Operators

Operator	Name	Description
+	Addition	Adds together two values
-	Subtraction	Subtracts one value from another
*	Multiplication	Multiplies two values
/	Division	Divides one value by another
%	Modulus	Returns the division remainder



Arithmetic Operators (cont)		
++	Increment	Increases the value of a variable by 1
-	Decrement	Decreases the value of a variable by 1
compute, calculate, or add statements can also be used		

Format Specifiers	
%a	Signed hexadecimal float
%c	A character
%d or %i	Signed decimal integer
%6d	Print as a decimal integer, at least 6 characters wide
%e	Signed decimal with scientific notation
%f	Signed decimal float
%6f	Print as floating point, at least 6 characters wide
%.2f	Print as floating point, 2 characters after decimal point
%6.2f	Print as floating point, at least 6 wide and 2 after decimal point
%g	Shortest representation of %f or %e
%o	Unsigned octal integer
%s	Character string

Format Specifiers (cont)	
%u	Unsigned decimal integer
%x	Unsigned hexadecimal integer
%p	Display a pointer
%%	Print a %

**Primitive Variable Types**

*applicable but not limited to most ARM, AVR, x86 & x64 installations*

[class] [qualifier]  
[unsigned] type/void name;

*by ascending arithmetic conversion*

Integers			
Type	Bytes	Value Range	
char	1	unsigned or signed	
unsigned char	1	0 to 255	
signed char	1	-128 to 127	
int	2 / 4	unsigned or signed	
unsigned int	2 / 4	0 to 65,535 or $2^{31}-1$	
signed int	2 / 4	-32,768 to 32,767 or $-2^{31}$ to $2^{32}-1$	
short	2	unsigned or signed	

Primitive Variable Types (cont)		
unsigned short	2	0 to 65,535
signed short	2	-32,768 to 32,767
long	4	unsigned or signed
unsigned long	4	0 to $2^{32}-1$ or $2^{64}-1$
signed long	4	$-2^{31}$ to $2^{31}-1$ or $-2^{63}$ to $2^{63}-1$
long long	8	unsigned or signed
unsigned long long	8	0 to $2^{64}-1$
signed long long	8	$-2^{63}$ to $2^{63}-1$

// signed is the default modifier of int and char data type (allows + or - value)

// unsigned only stores positive values

**Floats**

Primitive Variable Types (cont)		
Type	Bytes	Value Range (Normalized)
float	4	$\pm 1.2 \times 10^{-38}$ to $\pm 3.4 \times 10^{38}$
double	8 / 4	$\pm 2.3 \times 10^{-308}$ to $\pm 1.7 \times 10^{308}$ or alias to float for AVR
long double	ARM: 8, AVR: 4, x86: 10, x64: 16	

**Qualifiers**

const type	Flags variable as read-only (compiler can optimise)
volatile type	Flags variable as unpredictable (compiler cannot optimise)



### Primitive Variable Types (cont)

#### Storage Classes

`register` Quick access required. May be stored in RAM or a register. Maximum size is register size.

`static` Retained when out of scope. `static` global variables are confined to the scope of the compiled object file they were declared in.

`extern` Variable is declared by another file

### Primitive Variable Types (cont)

#### Typecasting

`(type) a` Returns a as data type

```
char x = 1, y = 2; float  
z = (float) x / y;
```

Some types (denoted with `or`) are architecture dependant.

There is no primitive boolean type, only zero (`false`, `0`) and non-zero (`true`, usually `1`).

