

## 1. Basic template

```
#include <stdio.h>

int main() {
}
```

## 2. Input template

```
#include <stdio.h>

int main() {
    char c;

    while ((c = getchar()) != EOF) {
        //do something
    }

    return 0;
}
```

## 3. Pointers

```
#include <stdio.h>

int main() {
    int values[] = {1, 2, 3, 4, 5};
    int * index; //pointer

    //read beginning of values array
    index = values;

    //incr pointer, go to 2nd value
    index +=1;

    printf("%d \n", * index); //print: 2
    return 0;
}
```

### 4. Structures

```
#include <stdio.h>

typedef struct point { float x, y; } point;

point makepoint(double x, double y) {
    point p;
    p.x = x;
    p.y = y;
    return p;
}

int main() {

    point p;

    p = makepoint(5, 7);

    printf("x: %f, y: %f \n", p.x, p.y);

    return 0;
}
```

### 5. Go ASCII go!

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	##32;	Space	64	40	100	##64;	␣	96	60	140	##96;	`
1	1	001	<b>SOH</b> (start of heading)	33	21	041	##33;	!	65	41	101	##65;	A	97	61	141	##97;	a
2	2	002	<b>STX</b> (start of text)	34	22	042	##34;	"	66	42	102	##66;	B	98	62	142	##98;	b
3	3	003	<b>ETX</b> (end of text)	35	23	043	##35;	#	67	43	103	##67;	C	99	63	143	##99;	c
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	##36;	\$	68	44	104	##68;	D	100	64	144	##100;	d
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	##37;	%	69	45	105	##69;	E	101	65	145	##101;	e
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	##38;	&	70	46	106	##70;	F	102	66	146	##102;	f
7	7	007	<b>BEL</b> (bell)	39	27	047	##39;	'	71	47	107	##71;	G	103	67	147	##103;	g
8	8	010	<b>BS</b> (backspace)	40	28	050	##40;	(	72	48	110	##72;	H	104	68	150	##104;	h
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	##41;	)	73	49	111	##73;	I	105	69	151	##105;	i
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	##42;	*	74	4A	112	##74;	J	106	6A	152	##106;	j
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	##43;	+	75	4B	113	##75;	K	107	6B	153	##107;	k
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	##44;	,	76	4C	114	##76;	L	108	6C	154	##108;	l
13	D	015	<b>CR</b> (carriage return)	45	2D	055	##45;	-	77	4D	115	##77;	M	109	6D	155	##109;	m
14	E	016	<b>SO</b> (shift out)	46	2E	056	##46;	.	78	4E	116	##78;	N	110	6E	156	##110;	n
15	F	017	<b>SI</b> (shift in)	47	2F	057	##47;	/	79	4F	117	##79;	O	111	6F	157	##111;	o
16	10	020	<b>DLE</b> (data link escape)	48	30	060	##48;	0	80	50	120	##80;	P	112	70	160	##112;	p
17	11	021	<b>DC1</b> (device control 1)	49	31	061	##49;	1	81	51	121	##81;	Q	113	71	161	##113;	q
18	12	022	<b>DC2</b> (device control 2)	50	32	062	##50;	2	82	52	122	##82;	R	114	72	162	##114;	r
19	13	023	<b>DC3</b> (device control 3)	51	33	063	##51;	3	83	53	123	##83;	S	115	73	163	##115;	s
20	14	024	<b>DC4</b> (device control 4)	52	34	064	##52;	4	84	54	124	##84;	T	116	74	164	##116;	t
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	##53;	5	85	55	125	##85;	U	117	75	165	##117;	u
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	##54;	6	86	56	126	##86;	V	118	76	166	##118;	v
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	##55;	7	87	57	127	##87;	W	119	77	167	##119;	w
24	18	030	<b>CAN</b> (cancel)	56	38	070	##56;	8	88	58	130	##88;	X	120	78	170	##120;	x
25	19	031	<b>EM</b> (end of medium)	57	39	071	##57;	9	89	59	131	##89;	Y	121	79	171	##121;	y
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	##58;	:	90	5A	132	##90;	Z	122	7A	172	##122;	z
27	1B	033	<b>ESC</b> (escape)	59	3B	073	##59;	;	91	5B	133	##91;	[	123	7B	173	##123;	{
28	1C	034	<b>FS</b> (file separator)	60	3C	074	##60;	<	92	5C	134	##92;	\	124	7C	174	##124;	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	##61;	=	93	5D	135	##93;	]	125	7D	175	##125;	}
30	1E	036	<b>RS</b> (record separator)	62	3E	076	##62;	>	94	5E	136	##94;	^	126	7E	176	##126;	~
31	1F	037	<b>US</b> (unit separator)	63	3F	077	##63;	?	95	5F	137	##95;	_	127	7F	177	##127;	DEL

Source: [www.LookUpTables.com](http://www.LookUpTables.com)

## 6. Types

Format	Type
%d or %i	int
%c	char
%f	float
%lf (letters: el ef)	double
%s	string
%x	hexadecimal

## 7. Binary operations

&	binary bitwise AND
^	binary bitwise OR (XOR)
	binary bitwise inclusive OR
~	unary bitwise complement (NOT)
<<	Shift left
>>	Shift right

## 8. Switch

```
switch( Grade )
{
    case 'A' : printf( "Excellent" );
    case 'B' : printf( "Good" );
    case 'C' : printf( "OK" );
    case 'D' : printf( "Mmmmm...." );
    case 'F' : printf( "You must do better than this" );
    default : printf( "What is your grade anyway?" );
}
```

## 9. Parameters from command line (when the program is launched):

```
int main(int argc, char *argv[]) {
    int i;
    printf("You gave me %d parameters:\n", argc);
    for (i = 0; i < argc; ++i)
        printf("argv[%d] = %s\n", i, argv[i]);
    return 0;
}
```

## 10. Unions:

A union is used to allocate a space in the memory as big as the biggest variable of that union.

```
union number
{
    short little_number;
    long long_number;
    double big_number;
} numbers;
```

## 11. Enums

The enum type is useful to define a special structure where each field inside it can be accessed by a number. For example, defining this enum:

```
enum days { mon, tue, wed, thu, fri, sat, sun } week;
```

(if not otherwise specified) **mon** has an int value 0, **tue** is 1, **wed** is 2, ...

Other index (non numeric or non-zero based):

```
// non numerical
enum seq_escape { alert = 'a', back = 'b', tab = 't', return = 'r' };

// index different from 0
enum months { jan = 1, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dic };
```

## 12. Multidimensional arrays

```
//what's the problem?
int matrix[n][m];
```

Reference used to write this:

🌐 Slides from the course “Introduction to C programming” (USI Lugano)

🌐 <http://html.it>

🌐 <http://www.asciitable.com/>

🌐 <http://www.somacn.com/p125.php>

🌐 <http://www.cs.cf.ac.uk/Dave/C/node13.html>

🌐 [http://www.space.unibe.ch/comp\\_doc/c\\_manual/C/SYNTAX/switch.html](http://www.space.unibe.ch/comp_doc/c_manual/C/SYNTAX/switch.html)