La gestione dei caratteri in C

In C i caratteri sono gestiti mediante variabili di tipo char (interi a 8 bit)
Ad ogni carattere corrisponde un codice numerico ASCII
(American Standard Code for Information Interchange)

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

Un esempio di utilizzo degli operatori (+, -, *, /, %) in un programma

```
#include <stdio.h>

int main()
{int i=7;   // inizializzo la variabile in fase di dichiarazione
 double k=4.5;
 float f;   // la variabile non viene inizializzata

printf("Ecco i tre valori: %d, %f, %lf\n",i,f,k);    // lf sta per long float; qual è il valore di f?

i=k+3.2; // posso assegnare il risultato di un'operazione tra float ad un int??
printf("Ecco il valore dell'int: %d\n",i);

f = (k + i)/35.6 - k;    // attenzione alla precedenza tra gli operatori -> uso le parentesi
printf("Ecco il valore del float: %f\n",f);

f = i*5;    // posso assegnare il risultato di un'operazione tra int ad un float?
printf("Ecco il valore del float: %f\n",f);

i=320;
/* visualizzo il valore di i e quello prodotto dall'operatore modulo (%).
    L'operatore modulo si usa solo sugli interi; se A e B sono variabili di tipo intero
    A%B restituisce il resto della divisione di A per B
printf("Ecco il risultato prodotto su %d dall'operatore modulo: %d\n", i , i%100 );

 return 0;}
```

Un esempio di utilizzo dei caratteri

```
#include <stdio.h>

int main()
{char c;
 /* inizializzo la variabile di tipo char con un intero -> viene interpretato come codice ASCII */
 c = 60;
printf("Al codice ASCII %d corrisponde il carattere %c\n",c,c);

 /* inizializzo la variabile di tipo char con un carattere */
c = '7';  /* notate gli apici! 7 senza apici è una costante numerica, 7 compreso tra apici
             diventa il carattere '7' (ovvero quello con codice ASCII 55) */

printf("Al codice ASCII %d corrisponde il carattere %c\n",c,c);

c='m';
printf("Visualizzo la lettera %c, e la corrispondente maiuscola, %c\n",c,c-32)

/* i caratteri sono gestiti mediante char, ovvero interi ad 8 bit, quindi li posso trattare come interi
    a patto di ricordarmi che, con un char, posso rappresentare SOLO valori compresi tra -127
   e 127    */

c = c+4;  // perfettamente lecito
printf("Visualizzo la lettera %c, ed il suo codice ASCII \n",c,c);
return 0;}
```

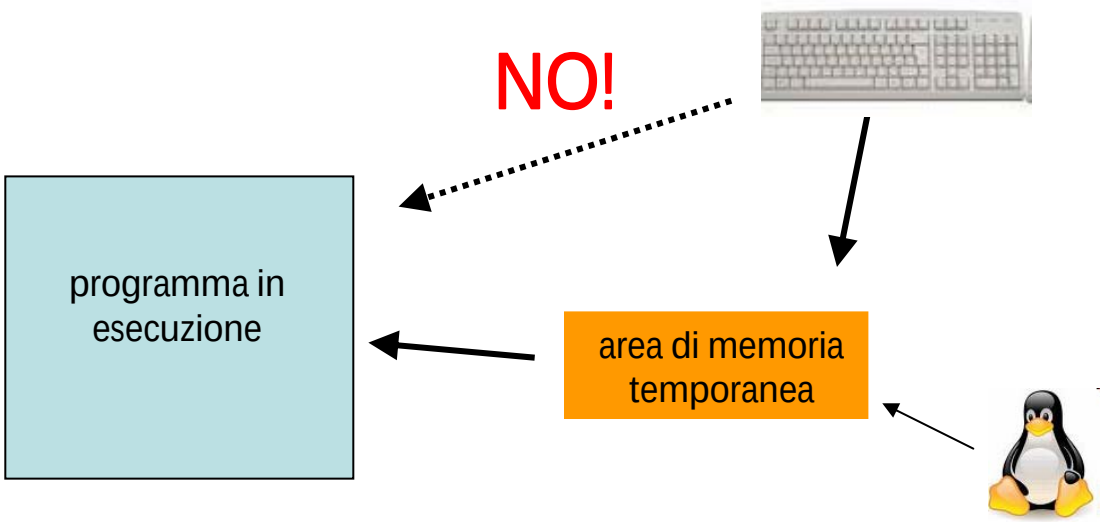
La funzione della libreria standard che ci permette di gestire l'input carattere per carattere è:

int getchar(void)

definita in stdio.h

La funzione, quando chiamata, passa il controllo del programma all'utente e rimane in attesa che quest'ultimo inserisca un carattere sulla tastiera; il controllo dell'esecuzione non viene restituito al programma fino a quando l'utente non preme il tasto ENTER.

La funzione getchar() appartiene al gruppo di funzioni che gestiscono l'input da tastiera mediante un **BUFFER di MEMORIA**.



Il contenuto dell'area di memoria (ovvero i caratteri inseriti con la tastiera) viene passato al programma quando l'utente preme ENTER

```
#include <stdio.h>

int main()
{char carattere;
 carattere=getchar()   // qui il programma si ferma in attesa che l'utente inserisca caratteri
 printf("il carattere inserito e' %c, il suo codice ASCII: %d",carattere,carattere);
 return 0;
 /* la dimensione massima del buffer è contenuta nell'etichetta (MACRO) BUFSIZ
     visualizziamone il valore! */
 printf("La dimensione del buffer di input e': %d\n",BUFSIZ);  // controlliamo l'effetto del preprocessore!
}
```

NOTA: il contenuto del buffer viene passato al programma SOLO quando l'utente preme ENTER, quindi l'utente può inserire quanti caratteri vuole, non c'è un controllo (-> potenzialmente pericoloso)
Dal buffer, ogni volta che viene chiamata **getchar()** viene prelevato **1 SOLO CARATTERE**

Esempio: supponiamo che l'utente inserisca ciao e poi prema ENTER

Il contenuto del buffer risulta essere

c	i	a	o	CR	LF

Se nel codice è presente una chiamata a getchar(), al programma viene passato il primo carattere del buffer, ovvero la lettera c, il resto rimane nel buffer in attesa di eventuali altre chiamate

i	a	o	CR	LF

Il carattere CR (carriage return) viene automaticamente tolto dal buffer, ma vi rimane il carattere LF (line feed) -> tale carattere può essere tolto dal buffer e "visualizzato"

Facciamo un esempio -> shell di Linux

```
#include <stdio.h>

int main()
{char carattere;
 printf("Inserisci un carattere ! ->");
 carattere=getchar();   // ora il programma rimane in attesa dell'intervento dell'utente
 printf("il carattere inserito e' %c, il suo codice ASCII: %d",carattere,carattere);
 carattere=getchar();  // controllo se vi siano altri caratteri rimasti nel buffer
 printf("il secondo carattere del buffer e' %c, con codice ASCII: %d",carattere,carattere);
 return 0;
}
```

Riusciamo a fare la stessa cosa mediante una funzione?
Includendo nella funzione la possibilità di capire che tipo di carattere sia?

Se con codice indichiamo il valore del codice ASCII di un carattere, allora abbiamo:

65 <= codice <= 90 LETTERE MAIUSCOLE
97 <= codice <= 122 lettere minuscole
NOTA: il codice di una lettera minuscola si ottiene sommando 32 al codice della corrispondente maiuscola
48 <= codice <= 57 numeri dallo 0 al 9

Un breve ripasso: l'istruzione di selezione

```
if(valore)
 {… codice …}
else
 {… altro codice … }
```

se valore è uguale a 0, esso viene considerato FALSO;
QUALSIASI valore diverso da 0 viene considerato VERO

In C le espressioni possiedono un valore, quindi all'interno delle parentesi ci può andare una qualsiasi espressione che restituisca un valore int,char,double o float

Esempi:

Verifichiamo che le espressioni contenute tra le parentesi seguono il criterio indicato sopra

```
int i=4;
if(i<6)                VERA
 {… }
```

```
int i=4;
printf("ecco il valore della prima espressione: %d",i<6);
```

```
if(i+65)               VERA
 { … }
```

```
printf("ecco il valore della seconda espressione: %d",i+65);
```

```
if(i=i-4)              FALSA
 {… }
```

```
printf("ecco il valore della terza espressione: %d",i=i-4);
```

```
if(getchar())          VERA
```

```
printf("ecco il valore della quarta espressione: %d",getchar());
```

Nel caso ci siano più condizioni da valutare contemporaneamente -> OPERATORI LOGICI

AND (operatore binario) restituisce vero se entrambe le condizioni sono vere
(condizione 1) && (condizione 2)

OR (operatore binario) restituisce vero se una o l'altra condizione sono vere
(condizione 1) || (condizione 2)

NOT (operatore unario) restituisce vero se la condizione è falsa
!(condizione 1)

Un breve ripasso: le funzioni

Una funzione racchiude un blocco di codice che esegue un compito specifico. In generale una funzione può ricevere parametri in ingresso e restituire valori.

Per la definizione di una funzione: **PROTOTIPO** & **IMPLEMENTAZIONE**
PROTOTIPO: indica il nome della funzione, il tipo degli eventuali parametri, il tipo eventualmente restituito
IMPLEMENTAZIONE: contiene le istruzioni che vengono svolte dalla funzione

Supponiamo di voler preparare una funzione che calcola il triplo di un valore di tipo intero

La funzione deve poter ricevere il valore dell'intero -> deve avere un parametro di tipo int
la funzione deve restituire un valore intero -> la funzione produce un valore di tipo int

```
int triplo(int);    // nel PROTOTIPO basta indicare il tipo del parametro, non serve anche il nome

int triplo(int numero)    // numero è il nome del parametro
{int nuovo_numero;
 nuovo_numero=numero*3;
 return nuovo_numero;
}
```

Un breve ripasso: le funzioni

```
int triplo(int);    // nel PROTOTIPO basta indicare il tipo del parametro, non serve anche il nome

int triplo(int numero)    // numero è il nome del parametro
{int nuovo_numero;
 nuovo_numero=numero*3;
 return nuovo_numero;
}
```

Posso fare le cose in modo più rapido??

```
int triplo(int);    // nel PROTOTIPO basta indicare il tipo del parametro, non serve anche il nome

int triplo(int valore)    // valore è il nome del parametro
{valore=valore*3;
 return valore;
}
```

Aver modificato il parametro crea problemi? NO! il parametro è solo la COPIA del valore inserito, quindi la modifica NON altera il valore del parametro stesso al di fuori della funzione

VERIFICHIAMOLO!

```
#include <stdio.h>

int triplo(int);    // nel PROTOTIPO basta indicare il tipo del parametro, non serve anche il nome

int main()
{ int var,var1;
  var = 8;
  // passo var alla funzione e memorizzo il valore restituito nella variabile var1
  var1 = triplo(var);
 printf("Ecco il valore della prima variabile: %d, e quello della seconda: %d\n",var,var1);
 return 0;}

int triplo(int numero)    // valore è il nome del parametro
{numero=numero*3;
 return numero;}
```

```
#include <stdio.h>   // Anche così va bene!

int triplo(int);

int main()
{ int var,var1;
  var = 8;
  var1 = triplo(var);
 printf("Ecco il valore della prima variabile: %d, e quello della seconda: %d\n",var,var1);
 return 0;}

int triplo(int var)
 {var = 3*var;
 return var}
```

```c
#include <stdio.h>

int main()
{char carattere;
 printf("Inserisci un carattere ! ->");
 carattere=getchar();   // ora il programma rimane in attesa dell'intervento dell'utente
 printf("il carattere inserito e' %c, il suo codice ASCII: %d",carattere,carattere);
 carattere=getchar();  // controllo se vi siano altri caratteri rimasti nel buffer
 printf("il secondo carattere del buffer e' %c, con codice ASCII: %d",carattere,carattere);
 return 0;
}
```

Riusciamo a costruire una funzione che ci permetta di capire che tipo di carattere sia stato inserito?

```c
#include <stdio.h>

void analizza_carattere(void);    // prototipo

int main()
{printf("Inserisci un carattere! ->");
 analizza_carattere();
 return 0;}

void analizza_carattere(void)
{char carattere;
 carattere=getchar();  // il programma ora rimane in attesa
 if((97<=carattere)&&(carattere<=122))
   printf("il carattere appartiene al gruppo delle lettere minuscole\n");
}
```

Esercizio:

Basandosi sull'esempio precedente scrivere una funzione che non accetti alcun parametro e che restituisca un valore di tipo int.

int analizza_numero(void)

La funzione deve:

1) dichiarare una variabile di tipo int ed una di tipo char
2) servirsi della funzione getchar() per assegnare il carattere digitato sulla tastiera dall'utente alla variabile di tipo char
3) controllare il codice ASCII contenuto nella variabile di tipo char per assicurarsi che sia quello di un numero
   (a) se il codice ASCII è quello di un numero, allora la funzione deve assegnare alla variabile di tipo int in numero corrispondente al codice ASCII contenuto nella variabile di tipo char (se la variabile di tipo char contiene il codice ASCII 49, la variabile di tipo int deve assumere il valore 1)
   (b) se il codice ASCII non è quello di un numero, allora la funzione deve assegnare alla variabile di tipo int il valore -1

Fatto questo, richiamare la funzione dall'interno di main() e verificarne il comportamento

```
#include <stdio.h>

int analizza_numero(void);    // prototipo

int main()
{int valore;
 printf("Inserisci un carattere! ->");
 valore = analizza_numero();
 printf("il valore restituito dalla funzione di analisi e': %d",valore);
 return 0;}

int analizza_numero(void)
{char carattere;
 int numero;
 carattere=getchar();  // il programma ora rimane in attesa
 if((48<=carattere)&&(carattere<=57))   // analizzo il codice ASCII
   {printf("il carattere inserito e' di tipo numerico\n");
    numero = carattere – 48;  /* i codici ASCII di 1,2,...9 sono 48,49, ...,57
                                 quindi per passare dal codice ASCII al numero corrispondente devo
                                 togliere 48 al codice ASCII    */
    }
 else
  {printf("il carattere inserito non e' di tipo numerico\n");
   numero=-1;
  }
 return numero;  // sono consistente con il prototipo
}
```

Un altro esercizio:

Scrivere un programma in cui:

1) si dichiara una variabile di tipo int ed una di tipo char
2) si chiede all'utente di inserire un carattere
3) si usa la funzione getchar() per assegnare il carattere digitato sulla tastiera dall'utente alla variabile di tipo char
4) si controlla il codice ASCII contenuto nella variabile di tipo char
   (a) se il codice ASCII è quello di una lettera minuscola, allora bisogna mostrare sul video la corrispondente lettera maiuscola
   (b) se il codice ASCII è quello di una lettera maiuscola, allora bisogna mostrare sul video la corrispondente lettera minuscola
   (c) se il codice ASCII è quello di un carattere numerico, allora bisogna mostrare sul video un valore che risulta essere pari a 5 volte quello del carattere (NON del suo codice ASCII)

5) assegnare alla variabile di tipo int un valore pari a 10 volte quello contenuto nella variabile di tipo char; mostrare a video il valore della variabile di tipo int
6) mostrare a video il valore che si ottiene dividendo per 7 il valore della variabile di tipo int