

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
    delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



## Dati testuali

## Caratteri

## Caratteri in C

- Ogni carattere viene rappresentato dal proprio codice ASCII
- Sono sufficienti 7 bit per rappresentare ciascun carattere
  - Il C usa variabili di 8 bit (1 byte)
- Non sono previste le lettere accentate né altri simboli diacritici
  - Richiedono estensioni speciali e librerie specifiche

# Codice ASCII

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

# Codice ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Char
0	0	000	NUL (null)	32	20	040	@
1	1	001	SOH (start of header)	33	21	041	A
2	2	002	STX (start of text)	34	22	042	B
3	3	003	ETX (end of text)	35	23	043	C
4	4	004	TX (end of transmission)	36	24	044	D

Simbolo  
corrispondente

Valore decimale  
(tra 0 e 127)

# Codice ASCII

	Dec	Hx	Oct	Html	Chr
	96	60	140	&#96;	`
	97	61	141	&#97;	a
	98	62	142	&#98;	b
	99	63	143	&#99;	c
	100	64	144	&#100;	d
	101	65	145	&#101;	e
	102	66	146	&#102;	f
	103	67	147	&#103;	g
	104	68	150	&#104;	h
	105	69	151	&#105;	i
	106	6A	152	&#106;	j
	107	6B	153	&#107;	k
	108	6C	154	&#108;	l
	109	6D	155	&#109;	m
	110	6E	156	&#110;	n
	111	6F	157	&#111;	o
	112	70	160	&#112;	p
	113	71	161	&#113;	q
	114	72	162	&#114;	r

Lettere  
minuscole

Lettere  
maiuscole



# Codice ASCII

Cifre  
numeriche

Simboli di  
punteggiatura

Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
Space	64	40	100	&#64;	@	96	60	140	&#96;	`
!	65	41	101	&#65;	A	97	61	141	&#97;	a
"	66	42	102	&#66;	B	98	62	142	&#98;	b
#	67	43	103	&#67;	C	99	63	143	&#99;	c
\$	68	44	104	&#68;	D	100	64	144	&#100;	d
%	69	45	105	&#69;	E	101	65	145	&#101;	e
&	70	46	106	&#70;	F	102	66	146	&#102;	f
'	71	47	107	&#71;	G	103	67	147	&#103;	g
(	72	48	110	&#72;	H	104	68	148	&#104;	h
)	73	49	111	&#73;	I	105	69	149	&#105;	i
*	74	50	112	&#74;	J	106	70	150	&#106;	j
+	75	51	113	&#75;	K	107	71	151	&#107;	k
,	76	52	114	&#76;	L	108	72	152	&#108;	l
-	77	53	115	&#77;	M	109	73	153	&#109;	m
.	78	54	116	&#78;	N	110	74	154	&#110;	n
/	79	55	117	&#79;	O	111	75	155	&#111;	o
0	80	50	120	&#80;	P	112	76	156	&#112;	p
1	81	51	121	&#81;	Q	113	77	157	&#113;	q
2	82	52	122	&#82;	R	114	78	158	&#114;	r
3	83	53	123	&#83;	S	115	79	159	&#115;	s
4	84	54	124	&#84;	T	116	80	160	&#116;	t
5	85	55	125	&#85;	U	117	81	161	&#117;	u
6	86	56	126	&#86;	V	118	82	162	&#118;	v
7	87	57	127	&#87;	W	119	83	163	&#119;	w
8	88	58	130	&#88;	X	120	84	164	&#120;	x
9	89	59	131	&#89;	Y	121	85	165	&#121;	y
:	90	5A	132	&#90;	Z	122	86	166	&#122;	z
;	91	5B	133	&#91;	[	123	87	167	&#123;	{
<	92	5C	134	&#92;	\	124	88	168	&#124;	
=	93	5D	135	&#93;	]	125	89	169	&#125;	}
>	94	5E	136	&#94;	^	126	90	170	&#126;	~
?	95	5F	137	&#95;	_	127	91	171	&#127;	DEL

# Codice ASCII

Hx Oct	Char	Dec Hx Oct
0 000	<b>NU</b> (null)	32 20 040
1 001	<b>SO</b> (start of heading)	33 21 041
2 002	<b>ST</b> (start of text)	34 22 042
3 003	<b>ET</b> (end of text)	35 23 043
4 004	<b>EOT</b> (end of transmission)	36 24 044
5 005	<b>ENQ</b> (enquiry)	37 25 045
6 006	<b>ACK</b> (acknowledge)	38 26 046
7 007	<b>BEL</b> (bell)	39 27 047
8 010	<b>BS</b> (backspace)	40 28 050
9 011	<b>TAB</b> (horizontal tab)	41 29 051
A 012	<b>LF</b> (NL line feed, new line)	42 2A 052
B 013	<b>VT</b> (vertical tab)	43 2B 053
C 014	<b>FF</b> (NP form feed, new page)	44 2C 054
D 015	<b>CR</b> (carriage return)	45 2D 055
E 016	<b>SO</b> (shift out)	46 2E 056
F 017	<b>SI</b> (shift in)	47 2F 057
10 020	<b>DLE</b> (data link escape)	48 30 060
11 021	<b>DC1</b> (device control 1)	49 31 061
12 022	<b>DC2</b> (device control 2)	50 32 062
13 023	<b>DC3</b> (device control 3)	51 33 063
14 024	<b>DC4</b> (device control 4)	52 34 064
15 025	<b>NAK</b> (negative acknowledge)	53 35 065
16 026	<b>SYN</b> (synchronous idle)	54 36 066
17 027	<b>ETB</b> (end of transmission block)	55 37 067
18 030	<b>CAN</b> (cancel)	56 38 070
19 031	<b>EM</b> (end of medium)	57 39 071
1A 032	<b>SUB</b> (substitute)	58 3A 072
1B 033	<b>ESC</b> (escape)	59 3B 073
1C 034	<b>FS</b> (file separator)	60 3C 074
1D 035	<b>GS</b> (group separator)	61 3D 075
1E 036	<b>RS</b> (record separator)	62 3E 076
1F 037	<b>US</b> (unit separator)	63 3F 077

Caratteri di controllo

Spazio bianco

## Caratteristiche del codice ASCII

- Le lettere maiuscole sono tutte consecutive, ed in ordine alfabetico
- Le lettere minuscole sono tutte consecutive, ed in ordine alfabetico
- Le lettere maiuscole vengono “prima” delle minuscole
- Le cifre numeriche sono tutte consecutive, in ordine dallo 0 al 9
- I simboli di punteggiatura sono sparsi



## Caratteri di controllo

- Caratteri speciali, non visualizzabili
- Rappresentano comandi di stampa, e non simboli da stampare
- Esempi:
  - 7 – BEL: emetti un “bip”
  - 8 – BS: cancella l’ultimo carattere
  - 10 – LF: avanza di una riga
  - 13 – CR: torna alla prima colonna
  - 27 – ESC: tasto “Esc”
- Per alcuni esiste una sequenza di escape in C: `\n`



## Errore frequente

- Non confondere il carattere ASCII che rappresenta una cifra numerica con il valore decimale associato a tale cifra

int  
7

char  
7  
55

- Per chiarezza useremo gli apici per indicare i caratteri

char  
'7'  
55



## Errore frequente

- Pensare che un singolo carattere possa memorizzare più simboli

~~char~~  
~~Fu~~  
~~lvio~~

char	char	char
F	u	l
55	117	108

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
for(i=0; i<MAXPAROLA; i++)
    freq[i]=0;
```

```
if(argc != 2)
{
    fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
    exit(1);
}
```

```
f = fopen(argv[1], "r");
if(f==NULL)
{
    fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
    exit(1);
}
```

```
while( fgets( riga, MAXRIGA, f ) != NULL )
```



## Il tipo char

## Variabili char

## Variabili char

- I caratteri in C si memorizzano in variabili di tipo char

```
char lettera ;
```

- Le costanti di tipo char si indicano ponendo il simbolo corrispondente tra singoli apici

```
lettera = 'Q' ;
```



- Non confondere i 3 tipi di apici presenti sulla tastiera:

Apice singolo (apostrofo)	'	In C, delimita singoli caratteri
Apice doppio (virgolette)	"	In C, delimita stringhe di caratteri
Apice rovesciato (accento grave)	`	Non utilizzato in C

## Dualità dei char

- Sintatticamente, i char non sono altro che degli `int` di piccola dimensione
  - Ogni operazione possibile su un `int`, è anche possibile su un `char`
  - Ovviamente solo alcune di tali operazioni avranno senso sull'interpretazione testuale (ASCII) del valore numerico

## Esempi

```
int i ;  
char c ;  
  
c = 'A' ;
```

## Esempi

```
int i ;  
char c ;
```

```
c = 'A' ;
```

```
c = 65 ; /* equivalente! */
```

## Esempi

```
int i ;  
char c ;
```

```
c = 'A' ;
```

```
c = 65 ; /* equivalente! */
```

```
i = c ; /* i sarà 65 */
```



## Esempi

```
int i ;  
char c ;
```

```
c = 'A' ;
```

```
c = 65 ; /* equivalente! */
```

```
i = c ; /* i sarà 65 */
```

```
c = c + 1 ; /* c sarà 66 = 'B' */
```

## Esempi

```
int i ;  
char c ;
```

```
c = 'A' ;
```

```
c = 65 ; /* equivalente! */
```

```
i = c ; /* i sarà 65 */
```

```
c = c + 1 ; /* c sarà 66 = 'B' */
```

```
c = c * 2 ; /* non ha senso... */
```

## Esempi

```
int i ;  
char c ;
```

```
c = 'A' ;
```

```
c = 65 ; /* equivalente! */
```

```
i = c ; /* i sarà 65 */
```

```
c = c + 1 ; /* c sarà 66 = 'B' */
```

```
c = c * 2 ; /* non ha senso... */
```

```
if (c == 'Z') ...
```

## Esempi

```
int i ;  
char c ;
```

```
c = 'A' ;
```

```
c = 65 ; /* equivalente! */
```

```
i = c ; /* i sarà 65 */
```

```
c = c + 1 ; /* c sarà 66 = 'B' */
```

```
c = c * 2 ; /* non ha senso... */
```

```
if (c == 'Z') ...
```

```
for( c='A'; c<='Z'; c++) ...
```

## Caratteri speciali

- Per alcuni caratteri di controllo il linguaggio C definisce una particolare **sequenza di escape** per poterli rappresentare

C	ASCII	Significato
'\n'	LF – 10	A capo
'\t'	TAB – 9	Tabulazione
'\b'	BS – 8	Backspace – cancella ultimo car.
'\a'	BEL – 7	Emette un “bip”
'\r'	CR – 13	Torna alla prima colonna



## Punteggiatura speciale in C

- Alcuni caratteri hanno un significato particolare dentro gli apici. Per poterli inserire come carattere esistono apposite sequenze di escape

C	ASCII	Significato
'\\'	\	Immette un backslash
'\''	'	Immette un apice singolo
'\"'	"	Immette un apice doppio
'\ooo'	ooo	Immette in carattere ASCII con codice (ottale) ooo
'\xhh'	hh	Immette in carattere ASCII con codice (esadecimale) hh

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
```

```
{
    int freq[MAXPAROLA]; /* vettore di contatori
    delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
```

```
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
```

```
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



## Il tipo char

## Input/output di char

## Input/output di char

- Esistono due insiemi di funzioni che permettono di leggere e stampare variabili di tipo char:
  - Le funzioni `printf/scanf`, usando lo specificatore di formato `"%c"`
  - Le funzioni `putchar` e `getchar`
- In entrambi i casi è sufficiente includere la libreria `<stdio.h>`
- È possibile mescolare liberamente le due famiglie di funzioni

## Stampa di caratteri

```
char ch ;
```

```
printf("%c", ch) ;
```

```
char ch ;
```

```
putchar(ch) ;
```

## Lettura di caratteri

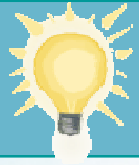
```
char ch ;
```

```
scanf("%c", &ch) ;
```

```
char ch ;
```

```
ch = getchar() ;
```





## Suggerimenti (1/2)

- La funzione `printf` è più comoda quando occorre stampare altri caratteri insieme a quello desiderato
  - `printf("La risposta e': %c\n", ch) ;`
  - `printf("Codice: %c%d\n", ch, num ) ;`
- La funzione `putchar` è più comoda quando occorre stampare semplicemente il carattere
  - `for(ch='a'; ch<='z'; ch++)  
    putchar(ch) ;`



## Suggerimenti (2/2)

- La funzione `getchar` è generalmente più comoda in tutti i casi
  - `printf("Vuoi continuare (s/n)? ");`  
`ch = getchar() ;`

# Bufferizzazione dell'input-output

- Tutte le funzioni della libreria `<stdio.h>` gestiscono l'input-output in modo **bufferizzato**
  - Per maggior efficienza, i caratteri non vengono trasferiti immediatamente dal programma al terminale (o viceversa), ma solo a gruppi
  - È quindi possibile che dopo una `putchar`, il carattere **non** compaia **immediatamente** sullo schermo
  - Analogamente, la `getchar` **non** restituisce il carattere finché l'utente non preme **invio**

## Conseguenza pratica

```
char ch, ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```

Dato: \_

Il programma stampa l'invito ad inserire un dato

## Conseguenza pratica

```
char ch, ch2 ;  
printf("Dato: ");  
→ ch = getchar() ;  
ch2 = getchar() ;
```

Dato: \_

getchar blocca il programma in attesa del dato

## Conseguenza pratica

```
char ch, ch2 ;  
printf("Dato: ");  
→ ch = getchar() ;  
ch2 = getchar() ;
```

Dato: a\_

L'utente immette 'a', il programma non lo riceve

## Conseguenza pratica

```
char ch, ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
→ ch2 = getchar() ;
```

Dato: a

—

L'utente immette Invio, il programma prosegue



## Conseguenza pratica

```
char ch, ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
→ ch2 = getchar() ;
```


Dato: a

—

Ora `ch='a'`, il programma fa un'altra `getchar()`

## Conseguenza pratica

```
char ch, ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```




Dato: a

—

Il programma **non** si blocca in attesa dell'utente

## Conseguenza pratica

```
char ch, ch2 ;  
printf("Dato: ");  
ch = getchar() ;  
ch2 = getchar() ;
```



Dato: a

—

C'era già un carattere pronto: Invio! `ch2 = '\n'`

## Consigli pratici

- Ricordare che l'utente deve sempre premere Invio, anche se il programma richiede un singolo carattere
- Ricordare che, se l'utente inserisce più di un carattere, questi verranno restituiti uno ad uno nelle `getchar` successive
- Ricordare che l'Invio viene letto come tutti gli altri caratteri

## Soluzione proposta

```
char ch, temp ;

printf("Dato: ");

ch = getchar() ; /* leggi il dato */

/* elimina eventuali caratteri successivi
ed il \n che sicuramente ci sarà */
do {
    temp = getchar() ;
} while (temp != '\n') ;
```

## Soluzione proposta

```
char ch, temp ;
```

```
printf("Dato: ");
```

```
ch = getchar() ;
```

```
/* elimina eventuali caratteri non validi  
ed il \n che sicuramente c'è */
```

```
do {  
    temp = getchar() ;  
} while (temp != '\n') ;
```

```
/* forma più compatta */  
while ( getchar() != '\n' )  
    /*niente*/ ;
```

```
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
```

```
#define MAXPAROLA 30
#define MAXRIGA 80
```

```
int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori
    delle frequenze delle lunghezze delle parole */
    char riga[MAXRIGA];
    int i, inizio, lunghezza;
    FILE *f;
```

```
    for(i=0; i<MAXPAROLA; i++)
        freq[i]=0;
```

```
    if(argc != 2)
    {
        fprintf(stderr, "ERRORE, serve un parametro con il nome del file\n");
        exit(1);
    }
```

```
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        fprintf(stderr, "ERRORE, impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }
```

```
    while( fgets( riga, MAXRIGA, f ) != NULL )
```



## Il tipo char

## Operazioni sui char

## Operazioni sui char

- Le operazioni lecite sui char derivano direttamente dalla combinazione tra
  - Le operazioni permesse sugli `int`
  - La disposizione dei caratteri nella tabella ASCII
  - Le convenzioni lessicali della nostra lingua scritta



## Conversione ASCII-Carattere

- Una variabile di tipo char è allo stesso tempo
  - Il valore numerico del codice ASCII del carattere
    - `printf("%d", ch) ;`
    - `i = ch ;`
    - `ch = j ;`
    - `ch = 48 ;`
  - Il simbolo corrispondente al carattere ASCII
    - `printf("%c", ch) ;`
    - `putchar(ch) ;`
    - `ch = 'Z' ;`
    - `ch = '4' ;`

## Esempio (1/3)

```
int i ;  
char ch ;
```

```
printf("Immetti codice ASCII (32-126): ");
```

```
scanf("%d", &i) ;
```

```
ch = i ;
```

```
printf("Il carattere %c ha codice %d\n",  
      ch, i) ;
```



char-int.c

## Esempio (2/3)

```
printf("Immetti un carattere: ") ;  
ch = getchar() ;
```

```
while( getchar() != '\n' )  
    /**/ ;
```

```
i = ch ;
```

```
printf("Il carattere %c ha codice %d\n",  
       ch, i) ;
```



char-int.c

## Esempio (3/3)

C:\ Prompt dei comandi

Immetti un codice ASCII (32-126): 44  
Il carattere , ha codice ASCII 44

Immetti un carattere: \$  
Il carattere \$ ha codice ASCII 36

char-int.c

## Scansione dell'alfabeto

- È possibile generare tutte le lettere dell'alfabeto, in ordine, grazie al fatto che nella tabella ASCII esse compaiono consecutive e ordinate

```
char ch ;
```

```
for( ch = 'A' ; ch <= 'Z' ; ch++ )  
    putchar(ch) ;
```

```
putchar('\n') ;
```

## Verifica se è una lettera

- Per sapere se un carattere è alfabetico, è sufficiente verificare se cade nell'intervallo delle lettere (maiuscole o minuscole)

```
if( ch>='A' && ch<='Z' )  
    printf("%c lettera maiuscola\n", ch) ;
```

```
if( ch>='a' && ch<='z' )  
    printf("%c lettera minuscola\n", ch) ;
```

```
if( (ch>='A' && ch<='Z') ||  
    (ch>='a' && ch<='z') )  
    printf("%c lettera\n", ch) ;
```

## Verifica se è una cifra

- Per sapere se un carattere è numerico ('0' - '9'), è sufficiente verificare se cade nell'intervallo delle cifre

```
if( ch>='0' && ch<='9' )  
    printf("%c cifra numerica\n", ch) ;
```

## Valore di una cifra

- Conoscere il valore decimale di un carattere numerico ( '0' - '9' ), è sufficiente calcolare la "distanza" dalla cifra '0'

```
if( ch>='0' && ch<='9' )
{
    printf("%c cifra numerica\n", ch) ;
    val = ch - '0' ;
    printf("Il suo valore e': %d", val ) ;
}
```



## Da minuscolo a maiuscolo (1/2)

- I codici ASCII delle lettere maiuscole e delle minuscole differiscono solamente per una costante:
  - 'A' = 65 ... 'Z' = 90
  - 'a' = 97 ... 'z' = 122
- Se ch è una lettera minuscola
  - $ch - 'a'$  è la sua posizione nell'alfabeto
  - $(ch - 'a') + 'A'$  è la corrispondente lettera maiuscola

## Da minuscolo a maiuscolo (2/2)

- Possiamo interpretare la conversione come una traslazione della quantità ( 'A' - 'a' )

```
if( ch>='a' && ch<='z' )
{
    printf("%c lettera minuscola\n", ch) ;
    ch2 = ch + ('A'-'a') ;
    printf("La maiuscola e': %c\n", ch2) ;
}
```

## Confronto alfabetico

- Se due caratteri sono **entrambi maiuscoli** (o entrambi minuscoli) è sufficiente confrontare i rispettivi codici ASCII

```
if( ch < ch2 )  
    printf("%c viene prima di %c", ch, ch2) ;  
else  
    printf("%c viene prima di %c", ch2, ch) ;
```