



Programmazione I

Il Linguaggio C

Dati testuali

Daniel Riccio

Università di Napoli, Federico II

25 ottobre 2021



Sommario



- Argomenti
 - Tipi di dato testuali
 - Caratteri
 - Operazioni sui caratteri

Tipi di dato testuali

I programmi visti finora erano in grado di elaborare esclusivamente informazioni numeriche

- Numeri interi (int), numeri reali (float)

- Variabili singole o vettori

In molti casi è necessario elaborare informazioni di tipo testuale

- Vuoi continuare (s/n)?

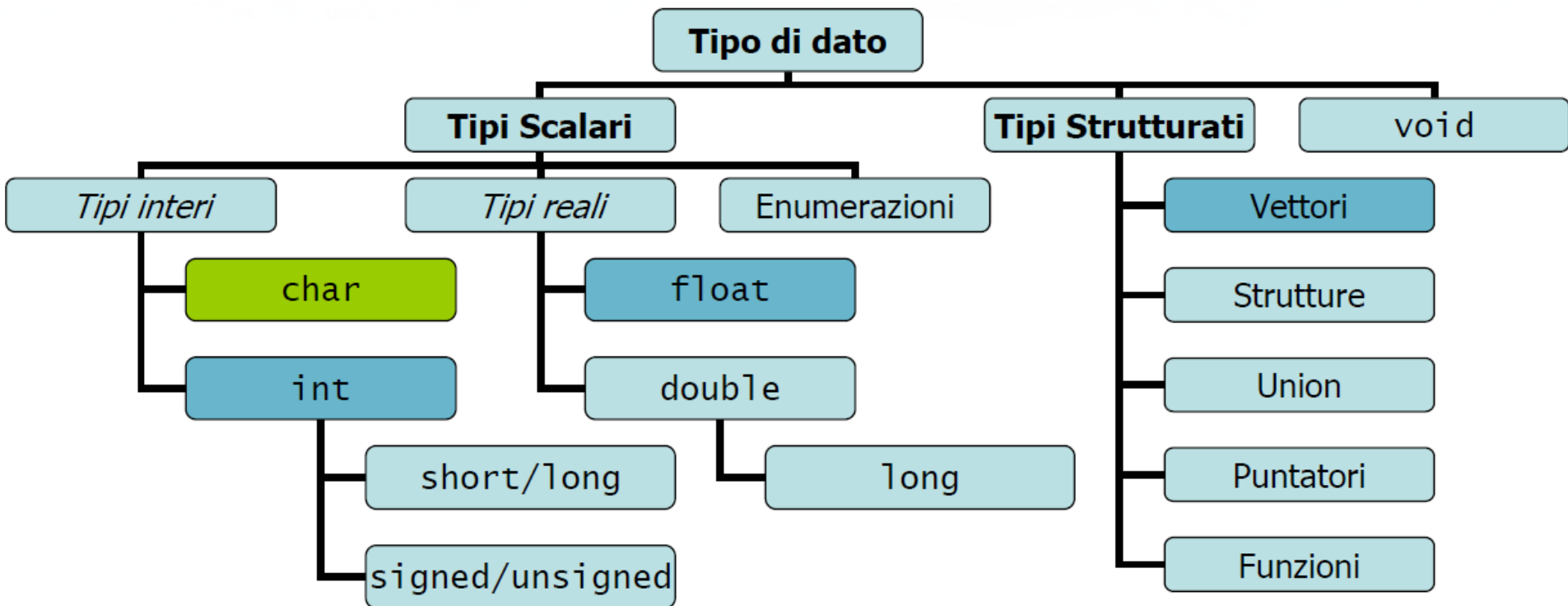
- Conta le parole di un testo scritto

- Gestisci una rubrica di nomi e numeri di telefono

- ...



Il sistema dei tipi in C



Rappresentazione dei testi

Il calcolatore è in grado di rappresentare i caratteri alfabetici, numerici ed i simboli speciali di punteggiatura

Ad ogni diverso carattere viene assegnato, convenzionalmente, un codice numerico corrispondente

Il programma in C lavora sempre con i codici numerici

Le funzioni di input/output sono in grado di accettare e mostrare i caratteri corrispondenti





Codice ASCII

La tabella ASCII (American Standard Code for Information Interchange) è un codice convenzionale usato per la rappresentazione dei caratteri di testo attraverso i byte

Ad ogni byte viene fatto corrispondere un diverso carattere della tastiera (lettere, numeri, segni).

In realtà lo standard ASCII copre solo i primi 128 byte (da 00000000 a 01111111), i successivi byte fino al 256° costituiscono la *tabella ASCII estesa* che presenta varie versioni a carattere nazionale.

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		DEL

Caratteri e stringhe

Il codice ASCII permette di rappresentare un singolo carattere

y	7	w	!	%
---	---	---	---	---

Nelle applicazioni pratiche spesso serve rappresentare sequenze di caratteri: **stringhe**

F	u	l	v	i	o
---	---	---	---	---	---

0	6	A	Z	N
---	---	---	---	---

0	1	1	-	5	6	4	6	3	3	2
---	---	---	---	---	---	---	---	---	---	---

Dualità caratteri - numeri

Ogni carattere è rappresentato dal suo codice ASCII

y	7	w	!	%
121	55	87	33	37

Ogni stringa è rappresentata dai codici ASCII dei caratteri di cui è composta

F	u	l	v	i	o	0	6	A	Z	N
70	117	108	118	105	111	48	54	65	90	78
0	1	1	-	5	6	4	6	3	3	2
48	49	49	45	53	54	52	54	51	51	50

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

L'alfabeto latino, usato nella scrittura di molte lingue nel mondo, presenta una grande quantità di varianti grafiche (es. vocali accentate, simboli di valuta, etc.)

Le varianti sono talmente numerose che i 128 byte della tabella estesa non sono purtroppo sufficienti a rappresentarle tutte, per questo motivo esistono diverse estensioni della tabella ASCII

Per cercare di ovviare al problema è stato creato un nuovo standard internazionale detto **Unicode**, definito dalla Unicode Consortium e dalla International Organization for Standardization (ISO 10646), che rappresenta i caratteri usando 2 byte (16 bit).

Caratteri di controllo

Un **carattere di controllo** o **carattere non visualizzabile**, è un codice (un numero) in un set di caratteri che non rappresenta in sé un simbolo scritto

Tutti i caratteri nella tabella ASCII al di sotto della posizione 32 fanno parte di questa categoria

I caratteri di controllo nella tabella ASCII ancora d'uso comune comprendono

7 (bell) – provoca l'emissione di un segnale sonoro da parte del terminale ricevente

8 (backspace) – utilizzato per cancellare l'ultimo carattere visualizzato, di solito quello immediatamente a sinistra del cursore

9 (horizontal tab) – tabulatore orizzontale

10 (line feed) – utilizzato per terminare le linee di testo

12 (form feed) – per terminare la pagina sulla stampante e avanzare al modulo successivo

13 (carriage return) – ritorno a capo, utilizzato per terminare le linee di testo **27 (escape)**.

Dec	Hx	Oct	Char
0	0	000	NUL (null)
1	1	001	SOH (start of heading)
2	2	002	STX (start of text)
3	3	003	ETX (end of text)
4	4	004	EOT (end of transmission)
5	5	005	ENQ (enquiry)
6	6	006	ACK (acknowledge)
7	7	007	BEL (bell)
8	8	010	BS (backspace)
9	9	011	TAB (horizontal tab)
10	A	012	LF (NL line feed, new line)
11	B	013	VT (vertical tab)
12	C	014	FF (NP form feed, new page)
13	D	015	CR (carriage return)
14	E	016	SO (shift out)
15	F	017	SI (shift in)
16	10	020	DLE (data link escape)
17	11	021	DC1 (device control 1)
18	12	022	DC2 (device control 2)
19	13	023	DC3 (device control 3)
20	14	024	DC4 (device control 4)
21	15	025	NAK (negative acknowledge)
22	16	026	SYN (synchronous idle)
23	17	027	ETB (end of trans. block)
24	18	030	CAN (cancel)
25	19	031	EM (end of medium)
26	1A	032	SUB (substitute)
27	1B	033	ESC (escape)
28	1C	034	FS (file separator)
29	1D	035	GS (group separator)
30	1E	036	RS (record separator)
31	1F	037	US (unit separator)



Maiuscole, minuscole e numeri

Lettere **MAIUSCOLE**:

codici ASCII compresi tra 65 e 90, estremi inclusi

Lettere **minuscole**:

codici ASCII compresi tra 97 e 122, estremi inclusi

Caratteri **numerici**:

codici ASCII compresi tra 48 e 57, estremi inclusi

Tra il codice ASCII di una lettera MAIUSCOLA e quello della corrispondente lettera minuscola esiste una differenza di 32 unità.

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

Dec	Hx	Oct	Html	Chr
65	41	101	A	A
66	42	102	B	B
67	43	103	C	C
68	44	104	D	D
69	45	105	E	E
70	46	106	F	F
71	47	107	G	G
72	48	110	H	H
73	49	111	I	I
74	4A	112	J	J
75	4B	113	K	K
76	4C	114	L	L
77	4D	115	M	M
78	4E	116	N	N
79	4F	117	O	O
80	50	120	P	P
81	51	121	Q	Q
82	52	122	R	R
83	53	123	S	S
84	54	124	T	T
85	55	125	U	U
86	56	126	V	V
87	57	127	W	W
88	58	130	X	X
89	59	131	Y	Y
90	5A	132	Z	Z

Dec	Hx	Oct	Html	Chr
97	61	141	a	a
98	62	142	b	b
99	63	143	c	c
100	64	144	d	d
101	65	145	e	e
102	66	146	f	f
103	67	147	g	g
104	68	150	h	h
105	69	151	i	i
106	6A	152	j	j
107	6B	153	k	k
108	6C	154	l	l
109	6D	155	m	m
110	6E	156	n	n
111	6F	157	o	o
112	70	160	p	p
113	71	161	q	q
114	72	162	r	r
115	73	163	s	s
116	74	164	t	t
117	75	165	u	u
118	76	166	v	v
119	77	167	w	w
120	78	170	x	x
121	79	171	y	y
122	7A	172	z	z

Dec	Hx	Oct	Html	Chr
48	30	060	0	0
49	31	061	1	1
50	32	062	2	2
51	33	063	3	3
52	34	064	4	4
53	35	065	5	5
54	36	066	6	6
55	37	067	7	7
56	38	070	8	8
57	39	071	9	9

Errori frequenti

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

int	char
7	7
	55

Per chiarezza useremo gli apici per indicare i caratteri

char
'7'
55

Pensare che un singolo carattere possa memorizzare più simboli

char	char	char	char
Fu1vio	F	u	1
	55	117	108

Vettori di caratteri – le stringhe



Una stringa è una struttura dati capace di memorizzare sequenze di caratteri

In C non esiste un tipo di dato specifico

Si usano vettori di caratteri

La lunghezza di una stringa è tipicamente variabile durante l'esecuzione del programma

Occorrerà gestire l'occupazione variabile dei vettori di caratteri



Caratteristiche delle stringhe

Memorizzate come singoli caratteri, ma il loro significato è dato dall'intera sequenza di caratteri

Lunghezza variabile

Mix di lettere/cifre/punteggiatura/spazi

Solitamente non contengono caratteri di controllo

F	u	l	v	i	o
70	117	108	118	105	111

0	6	A	Z	N
48	54	65	90	78

0	1	1	-	5	6	4	6	3	3	2
48	49	49	45	53	54	52	54	51	51	50

Caratteristiche delle stringhe

Occorre trattare l'insieme di caratteri memorizzato nel vettore come un'unica "variabile"

Ogni operazione elementare sulle stringhe coinvolgerà tipicamente dei cicli che scandiscono il vettore

Molte funzioni di libreria sono già disponibili per compiere le operazioni più frequenti ed utili

Non confondere una stringa composta da cifre numeriche con il valore decimale associato a tale sequenza

int
137

char
1 3 7
49 51 55

Il tipo 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

Esempio:

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

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

```
c = 'A';  
c = 65; /* equivalente! */  
i = c; /* i sarà 65 */
```

```
c = 'A';  
c = 65; /* equivalente! */  
i = c; /* i sarà 65 */  
c = c + 1; /* c sarà 66 = 'B' */
```

```
c = 'A';  
c = 65; /* equivalente! */  
i = c; /* i sarà 65 */  
c = c + 1; /* c sarà 66 = 'B' */  
c = c * 2; /* non ha senso... */
```

```
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

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

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

Esempio:

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

```
char ch;  
putchar(ch);
```

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

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

Osservazioni

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) ;
```

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

Il programma stampa l'invito ad inserire un dato

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

Dato: _

getchar blocca il programma in attesa del dato

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

Dato: _

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

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

Dato: a_

L'utente immette Invio, il programma prosegue

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

Dato: a
_

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

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

Dato: a
_

Il programma non si blocca in attesa dell'utente, C'era già Invio! ch2='\n'

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

Dato: a
_

Soluzione

```
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') ;
```

```
char ch;  
printf("Dato: ");  
ch = getchar(); /* leggi il dato */  
/* elimina eventuali caratteri successivi  
ed il \n che sicuramente ci sarà */  
  
while(getchar() != '\n' )
```

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

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' ;
```

Operazioni sui char



Esempio:

```
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);

printf("Immetti un carattere: ");
ch = getchar();
while(getchar() != '\n');

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

CA 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
```


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') ;
```

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 e valore di 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);
```

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



I codici ASCII delle lettere maiuscole e delle minuscole differiscono solamente per una costante:

'A' = 65 ... 'Z' = 90

'a' = 97 ... 'z' = 122

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

Se **ch** è una lettera minuscola

ch - 'a' è la sua posizione nell'alfabeto

(**ch** - 'a') + 'A' è la corrispondente lettera maiuscola

Se due caratteri sono entrambi maiuscoli (o entrambi minuscoli), per effettuare il confronto alfabetico 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);
```

Esercizio

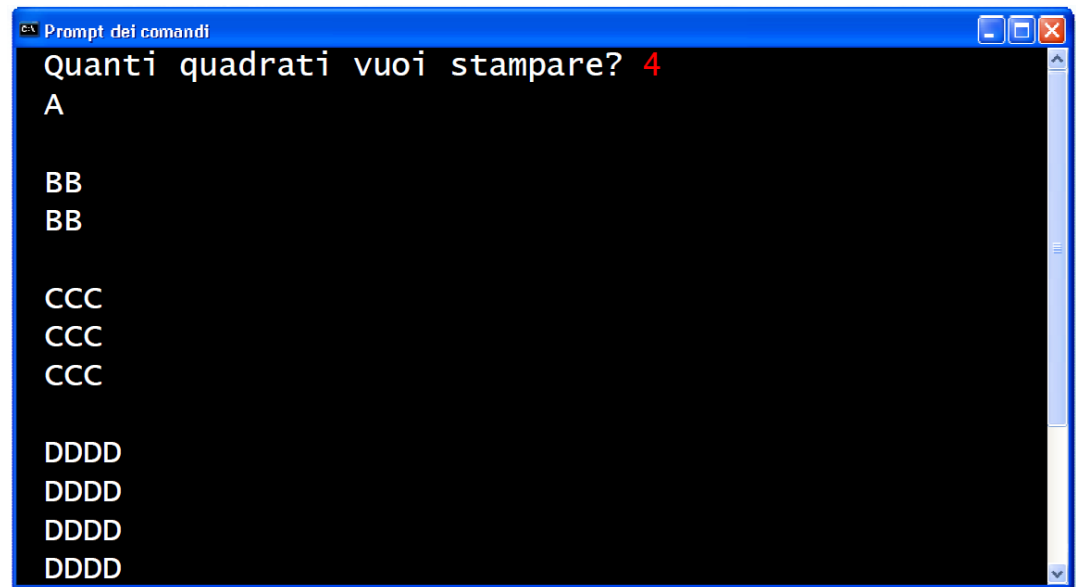
Si scriva un programma in linguaggio C che stampi su video una serie di quadrati, composti dalle successive lettere dell'alfabeto, di dimensioni sempre crescenti:

Un quadrato 1x1 di lettere A

Un quadrato 2x2 di lettere B

Un quadrato 3x3 di lettere C

...eccetera



```

c:\ Prompt dei comandi
Quanti quadrati vuoi stampare? 4
A

BB
BB

CCC
CCC
CCC

DDDD
DDDD
DDDD
DDDD

```

Esercizio



```
int i, N;
int riga, col;
char ch;
printf("Quanti quadrati? ");
scanf("%d", &N);
while (N<1 || N>26) {
    printf("Deve essere tra 1 e 26\n");
    printf("Quanti quadrati? ");
    scanf("%d", &N);
}

/* stampa un quadrato di dimensione (i+1) */
for(i=0; i<N; i++){
    ch = i + 'A';
    for(riga=0; riga<i+1; riga++){
        for(col=0; col<i+1; col++){
            putchar(ch);
            putchar('\n');
        }
        putchar('\n');
    }
}
```