



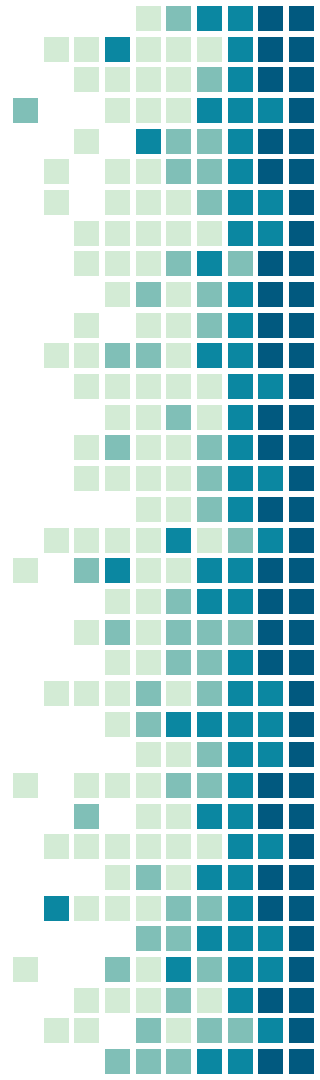
CdL in Informatica – A.A. 2024 – 2025

Programmazione 1 – Modulo 2

Lezione 2
03/10/2024

Andrea Loddo

Federico Meloni - Alessandra Perniciano - Fabio Pili



Test di Autovalutazione



Argomenti

- Cast implicito ed esplicito
- Problematiche acquisizione valori da input
- Dettagli sul tipo char
- Operatori: priorità e associatività
- Operatori composti



Cast implicito ed esplicito

```
/* Calcolo della media di tre valori interi. */  
int a, b, c;  
a = 5;  
b = 4;  
c = 2;  
float media = (a+b+c)/3;      // quanto vale media?
```

- Cast implicito: in C, la valutazione di una divisione tra interi restituisce un valore intero, anche se si assegna il risultato a una variabile float
- Soluzione: cast esplicito `float media = (float)(a+b+c)/3;`
 - si converte esplicitamente il valore di (a+b+c), temporaneamente, a float
 - quindi: l'operazione di divisione non sarà più valutata tra interi, ma tra float
 - il tipo della variabile che memorizza il risultato dev'essere float a sua volta
- Analogamente, si potrebbe scrivere: `float media = (a+b+c)/(float)3;`
- NB: L'operatore di cast si applica alla variabile o espressione immediatamente successiva

Riepilogo

- Quanto vale a dopo l'istruzione `int a = 10/4;` ?
- Quanto vale b dopo l'istruzione `float b = 10.0/4.0;` ?
- Quanto vale c dopo l'istruzione `int c = 10.0/4.0;` ?
- Quanto vale d dopo l'istruzione `float d = 10/4;` ?



REMINDER: acquisizione dei dati

Funzione della libreria stdio: `scanf()`

Legge uno o più dati da tastiera e li memorizza nelle variabili corrispondenti indicate.

Sintassi:

```
scanf(tipo di dato, indirizzo delle variabili);
```

L'**indirizzo** di una variabile si ottiene anteponendo il simbolo **&** al nome della variabile.

I **segnaposto** indicano il tipo di dato da acquisire, come nella **printf()**:

```
/* %d -> segnaposto di un valore intero  
   &num -> indica di salvare il valore acquisito nella  
   locazione di memoria occupata dalla variabile num */  
  
scanf("%d", &num);
```

REMINDER: acquisizione dei dati

È possibile acquisire più dati insieme ma è preferibile acquisire **una sola** variabile per **scanf()**.

```
scanf("%d", &a);    // legge un numero intero e lo salva in a
scanf("%f", &b);    // legge un numero float e lo salva in b
scanf("%lf", &c);   // legge un numero double (lf = long float) e lo salva in c
scanf("%c", &c);    // legge un singolo char (carattere) e lo salva in c
```

Esistono funzioni alternative a **scanf()** e **printf()**, utilizzate **SOLO** per l'inserimento di caratteri:

- **getchar()** per l'input
- **putchar()** per l'output

```
char x;
x = getchar();    // legge un carattere e lo memorizza in x
putchar(x);       // stampa a schermo il carattere contenuto in x
```

Meno versatili rispetto a **scanf()** e **printf()**: singolo valore per volta!

Acquisizione dei dati da input

```
#include <stdio.h>
```

```
int main()  
{  
    int a;  
    char c;  
    scanf("%d", &a);  
    scanf("%c", &c);
```

```
    printf("Valore di a: %d.\n", a);
```

```
    printf("Valore di c: %c.\n", c);
```

```
    return 0;
```

```
}
```

Quanto vale **a**?

Quanto vale **c**?

```
5  
Valore di a: 5  
Valore di c:  
  
Process returned 0 (0x0)  
Press ENTER to continue.  
■
```

- Il linguaggio C lavora su input e output memorizzandolo in dei buffer, zone di memoria temporanee in cui transita l'informazione.
- Possono verificarsi problematiche.

Input: problema

5	\n			...
---	----	--	--	-----

Buffer di input (**stdin**)

```
#include <stdio.h>
```

```
int main()  
{
```

```
    int a;  
    char c;
```

```
    scanf("%d", &a);  
    scanf("%c", &c);
```

```
    printf("Valore di a: %d.\n", a);  
    printf("Valore di c: %c.\n", c);
```

```
    return 0;
```

```
}
```

Eseguiamo il programma insieme:

1. Supponiamo di inserire il valore 5 in input per la variabile a.
2. È "impossibile" inserire un valore per la variabile c. Il motivo è che, una volta inserito il valore 5, per confermare la scelta si deve premere il tasto **Invio**. Questo comportamento viene interpretato dal linguaggio come uno **"\n"**.
3. La prima **printf()** stamperà il valore 5.
4. La seconda **printf()**, di fatto, farà un'andata a capo, dal momento che contiene uno **"\n"**.

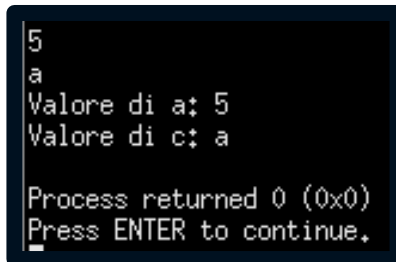
Input: soluzioni

```
...  
int a;  
char c;  
scanf("%d", &a);  
getchar();  
scanf("%c", &c);  
  
printf("Valore di a: %d.\n", a);  
printf("Valore di c: %c.\n", c);  
...
```

```
...  
scanf("%d", &a);  
scanf(" %c", &c);  
  
printf("Valore di a: %d.\n", a);  
printf("Valore di c: %c.\n", c);  
...
```

Soluzione 1: getchar()

- intercetta il tasto Invio e "consuma" \n presente nel buffer



```
5  
a  
Valore di a: 5  
Valore di c: a  
  
Process returned 0 (0x0)  
Press ENTER to continue.  
_
```

Soluzione 2: " %c"

- lo spazio prima di %c indica alla scanf che qualsiasi carattere speciale va ignorato

Tipo char e valori interi

In C, un carattere va indicato tra singoli apici (es: 'a').

Il tipo di dato **char** è legato direttamente al tipo **int**.

Nel codice ASCII a ciascun simbolo è associato un identificativo numerico:

- la lettera 'A' corrisponde al numero 65;
- la lettera 'a' corrisponde al numero 97, e così via.

Questa corrispondenza rende possibile lavorare sui caratteri come sugli interi

- è possibile usare operazioni aritmetiche o di confronto sui char!

```
...  
char var = 'a';  
var = var + 3;  
  
printf("var: %c", var);  
...
```

```
var: d
```

Tabella ASCII (estesa)

ASCII control characters			ASCII printable characters			Extended ASCII characters										
00	NULL	(Null character)	32	space	64	@	96	`	128	Ç	160	á	192	Ł	224	Ó
01	SOH	(Start of Header)	33	!	65	A	97	a	129	ü	161	í	193	ł	225	ô
02	STX	(Start of Text)	34	"	66	B	98	b	130	é	162	ó	194	Ł	226	Ô
03	ETX	(End of Text)	35	#	67	C	99	c	131	â	163	ú	195	ł	227	Ò
04	EOT	(End of Trans.)	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	ō
05	ENQ	(Enquiry)	37	%	69	E	101	e	133	à	165	Ñ	197	†	229	Õ
06	ACK	(Acknowledgement)	38	&	70	F	102	f	134	â	166	ª	198	ä	230	μ
07	BEL	(Bell)	39	'	71	G	103	g	135	ç	167	º	199	Ä	231	þ
08	BS	(Backspace)	40	(72	H	104	h	136	ê	168	¿	200	Ł	232	Þ
09	HT	(Horizontal Tab)	41)	73	I	105	i	137	ë	169	®	201	ƒ	233	Ú
10	LF	(Line feed)	42	*	74	J	106	j	138	è	170	™	202	ƒ	234	Û
11	VT	(Vertical Tab)	43	+	75	K	107	k	139	ï	171	½	203	ƒ	235	Ü
12	FF	(Form feed)	44	,	76	L	108	l	140	î	172	¼	204	ƒ	236	Ý
13	CR	(Carriage return)	45	-	77	M	109	m	141	ì	173	»	205	=	237	Ÿ
14	SO	(Shift Out)	46	.	78	N	110	n	142	Ë	174	«	206	ƒ	238	—
15	SI	(Shift In)	47	/	79	O	111	o	143	À	175	»	207	ƒ	239	,
16	DLE	(Data link escape)	48	0	80	P	112	p	144	É	176		208	ð	240	≡
17	DC1	(Device control 1)	49	1	81	Q	113	q	145	æ	177		209	Ð	241	±
18	DC2	(Device control 2)	50	2	82	R	114	r	146	Æ	178		210	È	242	¾
19	DC3	(Device control 3)	51	3	83	S	115	s	147	ô	179		211	Ê	243	¾
20	DC4	(Device control 4)	52	4	84	T	116	t	148	ö	180		212	Ë	244	ŋ
21	NAK	(Negative acknowl.)	53	5	85	U	117	u	149	ò	181	À	213	Ì	245	§
22	SYN	(Synchronous idle)	54	6	86	V	118	v	150	û	182	Á	214	Í	246	÷
23	ETB	(End of trans. block)	55	7	87	W	119	w	151	ù	183	Â	215	Î	247	°
24	CAN	(Cancel)	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	EM	(End of medium)	57	9	89	Y	121	y	153	Ö	185		217		249	°
26	SUB	(Substitute)	58	:	90	Z	122	z	154	Ü	186		218		250	°
27	ESC	(Escape)	59	;	91	[123	{	155	ø	187		219		251	°
28	FS	(File separator)	60	<	92	\	124		156	£	188		220		252	°
29	GS	(Group separator)	61	=	93]	125	}	157	Ø	189	¢	221		253	°
30	RS	(Record separator)	62	>	94	^	126	~	158	×	190	¥	222		254	°
31	US	(Unit separator)	63	?	95	_			159	f	191		223		255	nbsp
127	DEL	(Delete)														

Errori semantici sui tipi

```
#include <stdio.h>

int main()
{
    int a = 75;
    char c1 = 'a';
    char c2 = a;

    printf("c1: %c.\n", c1);
    printf("c2: %c.\n", c2);
    return 0;
}
```

```
#include <stdio.h>

int main()
{
    char c1 = '5';
    char c2 = 2;

    printf("c1: %c.\n", c1);
    printf("c2: %c.\n", c2);
    return 0;
}
```

Quanto valgono c1 e c2 in questi esempi?

Errori semantici sui tipi

```
#include <stdio.h>

int main()
{
    int a = 75;
    char c1 = 'a';
    char c2 = a;

    printf("c1: %c.\n", c1);
    printf("c2: %c.\n", c2);
    return 0;
}
```

c1: a

c2: K

```
#include <stdio.h>

int main()
{
    char c1 = '5';
    char c2 = 2;

    printf("c1: %c.\n", c1);
    printf("c2: %c.\n", c2);
    return 0;
}
```

c1: 5

c2: 😊

MASSIMA attenzione: assegnare a un char un **nome di variabile**, o un **valore costante** al posto di un carattere, è errore semantico non rilevabili dal compilatore.

Domanda

Cosa stamperà la seguente operazione:

```
printf("%c", 5 + 'b');
```

?

Assegnamento composto

Rendono la scrittura del codice più compatta

IMPORTANTE: hanno precedenza più bassa rispetto agli operatori aritmetici

```
a += 2;    // a = a + 2;  
a -= 2;    // a = a - 2;  
a *= 2;    // a = a * 2;  
a /= 2;    // a = a / 2;  
a %= 2;    // a = a % 2;
```

Esempi:

```
int i, j, k;  
...  
i = 2, j = 2, k = 3;  
i *= j + k;           // Quanto vale i ?  
  
i = 2, j = 2, k = 3;  
i = i*j + k;          // Quanto vale i ?  
  
i = 2, j = 2, k = 3;  
i = i*(j + k);        // Quanto vale i ?
```


Assegnamento composto

Rendono la scrittura del codice più compatta

IMPORTANTE: hanno precedenza più bassa rispetto agli operatori aritmetici

```
a += 2;    // a = a + 2;  
a -= 2;    // a = a - 2;  
a *= 2;    // a = a * 2;  
a /= 2;    // a = a / 2;  
a %= 2;    // a = a % 2;
```

Esempi:

```
int i, j, k;  
  
i = 2, j = 2, k = 3;  
i *= j + k;           // valuta prima + e poi *=           -> i vale 10.  
  
i = 2, j = 2, k = 3;  
i = i*j + k;          // valuta prima *, poi +, infine = -> i vale 7.  
  
i = 2, j = 2, k = 3;  
i = i*(j+k);          // valuta prima +, poi *, infine = -> i vale 10.
```

Autovalutazione



Autovalutazione

1. Per quale motivo si potrebbe creare un problema nella gestione dell'input nel caso in cui acquisisca un carattere?
2. Come viene utilizzata la `getchar()` per gestire al meglio l'input di caratteri?
3. Ci sono soluzioni alternative alla `getchar()` per gestire al meglio l'input di caratteri?
4. È necessario usare la `getchar()` per l'input di valori numeri? Perché?
5. Che differenza c'è tra cast implicito ed esplicito?
6. L'operatore di casting su quale o quali operandi ha effetto?
7. Che cosa si ottiene assegnando a una variabile float il risultato della divisione tra i due interi 2 e 7?
8. Che tipo di cast viene effettuato se dividiamo un float per un intero ?
9. Come viene assegnato un carattere ad una variabile char?
10. Quali sono gli operatori composti?

Esercizi

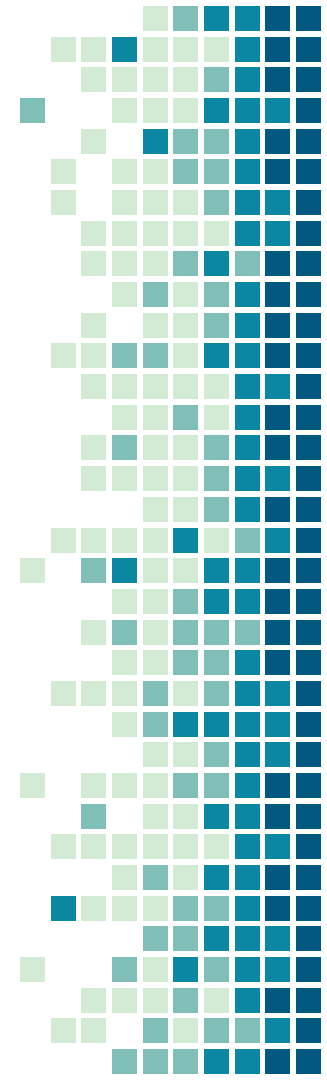


Esercizio 1

Scrivere un programma che, ricevuto un numero di secondi in input dall'utente, determini la quantità di ore, minuti e secondi corrispondenti.

N.B.: i valori visualizzati dovranno essere interi.

Esempio di output: 4 ore, 3 minuti e 12 secondi.



Esercizi 2

Scrivere un programma che chieda cinque numeri in input e ne visualizzi somma e media.

NB: gestire eventuali problematiche legate ai tipi.



Esercizio 3

- Scrivere un programma per eseguire il prodotto di tre interi.
- Dichiarare le variabili x, y, z e result di tipo int.
- Impostare 3 valori rispettivamente uno per x, uno per y e uno per z.
- Calcolare il prodotto delle tre variabili e assegnare il risultato alla variabile result.
- Infine, visualizzare: "Il prodotto è __.", dove __ dovrà essere sostituito dal valore di result.
- Non devono essere presenti magic numbers e si deve commentare opportunatamente tutto il codice.



Esercizio 4

- Scrivere un programma C in cui verranno dichiarati due interi.
- Assegnare un valore a entrambi gli interi.
- Stampare, quindi:
 - La loro somma
 - Il precedente e il successivo di entrambi
 - La loro media
- Non devono essere presenti magic numbers e si deve commentare opportunatamente tutto il codice.



Esercizio 5

- Calcolare la media aritmetica di 3 voti di uno studente e stampare il risultato a schermo.
- Specifiche: i 3 voti devono essere interi, mentre la media è un numero reale.
- Suggerimento: utilizzare opportunamente il cast esplicito per risolvere eventuali anomalie.

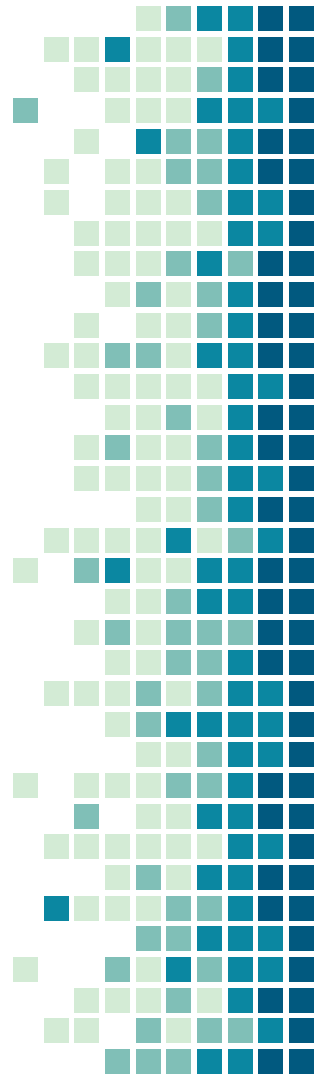


Esercizio 6

Cosa sarà visualizzato (se lo sarà), quando ognuna delle seguenti istruzioni verrà eseguita?

Nel caso in cui non venga visualizzato niente, rispondere "niente". Assumete che: `int x = 2` e `int y = 3`.

- a) `printf("%d", x);`
- b) `printf("%d", x+x);`
- c) `printf("x=");`
- d) `printf("x=%d", x);`
- e) `printf("%d = %d", x+y, y+x);`
- f) `z = x + y;`
- g) `/* printf("x + y = %d", x+y); */`
- h) `printf("\n");`



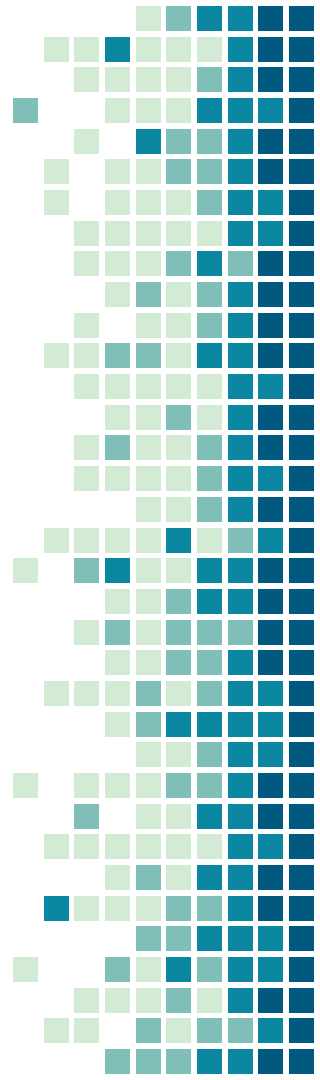
Esercizio 7

- Scrivere un programma che permetta di gestire la conversione mph – kmh (miglia orarie in km orari) sapendo che il tasso di conversione mph - kmh è pari a 1.61 (1 mph = 1,61 kmh). Associare tale tasso ad una macro chiamata **T_CONV_MPH_TO_KMH**. Assegnare alla variabile **velocitaMph** un valore a piacere e stampare il seguente messaggio in output:

Velocita in mph: ____.__ mph.

Velocita in kmh: ____.__ kmh.

- Dove ____.__ dovrà essere sostituito dal valore corrispondente e dovrà essere stampata solo una cifra decimale.
- Non devono essere presenti magic numbers e si deve commentare opportunatamente tutto il codice.



Esercizio 8

- Scrivere un programma in cui, dopo aver impostato due numeri, visualizzi la loro somma, prodotto, differenza, quoziente e resto. Assumete che il secondo numero sia diverso da zero.
- Scrivere un programma che visualizzi i numeri da 1 a 4 sulla stessa riga. Scrivere il programma utilizzando i seguenti metodi:
 - Usando un'istruzione printf senza segnaposto.
 - Usando un'istruzione printf con 4 segnaposto.
 - Usando 4 istruzioni printf.

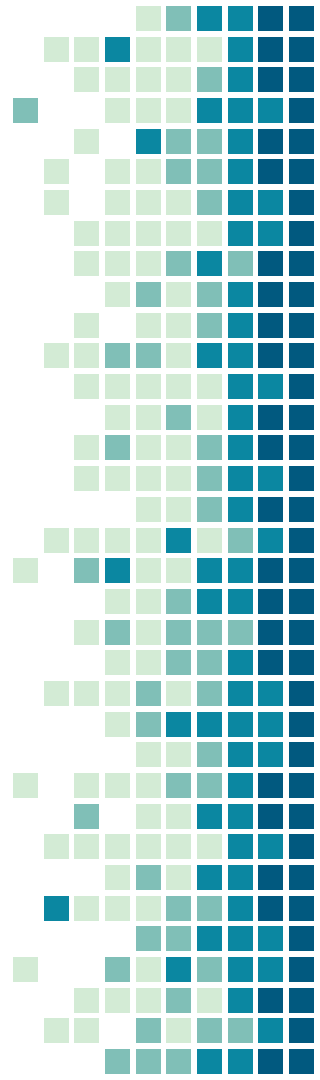


Esercizio 9

Scrivere un programma che esegua la somma tra un intero minore di 100 e un carattere a piacere.

Stampare il risultato usando i segnaposti **%c** e **%d**.

- Perché ciò è possibile?
- Cosa viene stampato?



Esercizio 10

- Scrivere un programma che esegua le 4 operazioni di base tra due variabili:
 - intere e intere
 - float e float
 - intere e float.
- Stampare poi il risultato.
- In particolare: effettuare una divisione tra interi assegnando il risultato
 - a una variabile **intera** prima
 - e **float** dopo
- Utilizzare opportunatamente il cast esplicito per risolvere eventuali anomalie.



Esercizi 11 e 12

11. Scrivere un programma che dato un numero intero assegnato a una variabile **numero** di 4 cifre stampi il numero di unità, decine, centinaia e migliaia.
12. Modificare il programma precedente permettendo la stampa di decimi e centesimi di un numero float.



Esercizio 13

- Cosa visualizzerà il seguente codice?

```
printf("*\n**\n***\n****\n*****\n");
```

- Cosa visualizzerà il seguente codice?

```
printf("%%\n%%%\n\t%%%\n%%\n\t%%\n");
```

- Cosa visualizzerà il seguente codice?

```
printf("\\ \n\\ \\ \"\n\\ \\ \"\\ \\ \n\\ \\ \n\" \n");
```

- Cosa visualizzerà il seguente codice?

```
printf("%%\n%%%\n%%%\n%%%\n%%%\n%%%\n%%%\n");
```

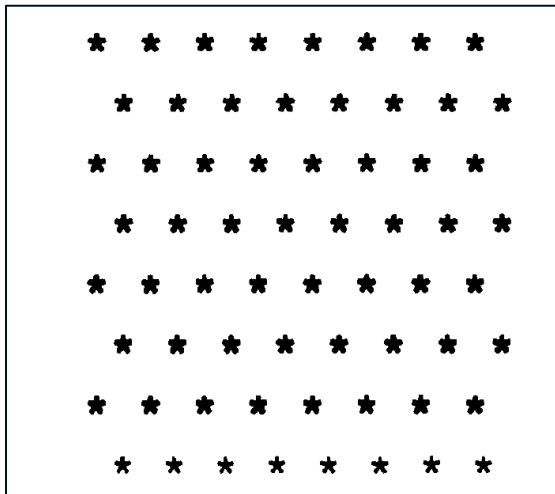

Esercizio 14

Scrivere un programma che visualizzi una scatola, un ovale, una freccia e un diamante come i seguenti:

```
*****          ***          *          *
*      *      *      *      ***      *  *
*      *      *      *      *      *  *
*      *      *      *      *      *  *
*      *      *      *      *      *  *
*      *      *      *      *      *  *
*      *      *      *      *      *  *
*****          ***          *          *
```

Esercizio 15

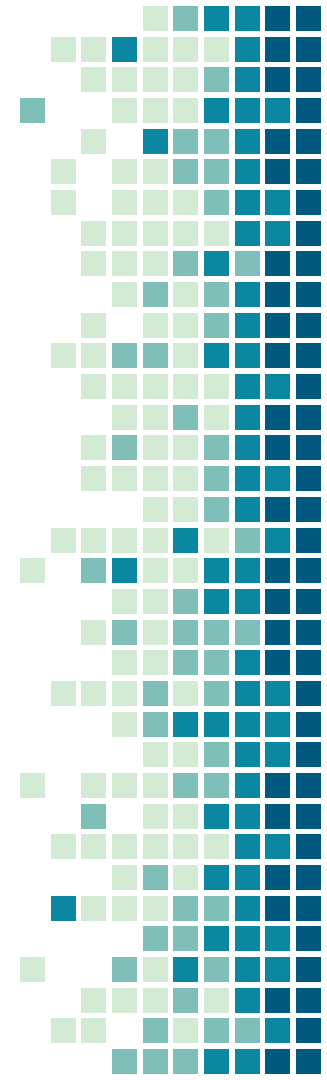
Scrivete un programma che visualizzi il disegno di una scacchiera utilizzando otto istruzioni `printf()` e quindi stampate lo stesso disegno con il minor numero possibile d'istruzioni `printf()`.



Esercizio 16

Scrivere un programma per calcolare i quadrati e i cubi dei numeri da 0 a 10, utilizzando le tabulazioni (`\t`) per visualizzare la tabella in questo modo:

numero	quadrato	cubo
0	0	0
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000



Esercizio 17

Finora abbiamo visto il tipo di dato carattere: **char**. Ogni carattere è rappresentato, in linguaggio C, mediante il codice ASCII. Quindi, è ammessa una stampa del genere:

```
printf("%d", 'A'); // stampa il codice ASCII di 'A'
```

che restituisce il valore intero corrispondente alla rappresentazione di A in codice ASCII.

Determinare, a questo punto, il codice intero corrispondente a: A B C a b c 0 1 2 \$ + / e del carattere spazio.

