

LABORATORIO DI PROGRAMMAZIONE
CORSO DI LAUREA IN SICUREZZA INFORMATICA
UNIVERSITÀ DEGLI STUDI DI MILANO

2025–2026

INDICE

Parte 1. Esperimenti con l'output formattato di printf	3
Esercizio 1	3
Il primo programma in C: Ciao Mondo	3
Tempo: 10 min.	3
Esercizio 2	3
Ritorno a capo	3
Tempo: 10 min.	3
Esercizio 3	4
Tabulazione orizzontale	4
Tempo: 10 min.	4
Esercizio 4	5
Codice ASCII	5
Tempo: 10 min.	5
Parte 2. Overflow & Underflow	5
Esercizio 5	5
Range dei tipi integrali	5
Tempo: 10 min.	5
Esercizio 6	5
Range dei tipi reali	5
Tempo: 10 min.	5
Esercizio 7	6
Overflow (con gli interi)	6
Tempo: 10 min.	6
Esercizio 8	6
Underflow (con i numeri in virgola mobile a precisione singola)	6
Tempo: 10 min.	6
Parte 3. Primi passi coi tipi primitivi	6
Esercizio 9	7
Eco di <code>int</code> e <code>double</code> dalla console	7
Tempo: 20 min.	7
Esercizio 10	7
Le quattro operazioni con <code>int</code>	7

Tratti dagli esercizi del corso del prof. Vincenzo Marra.
Ultima revisione: 3 ottobre 2025.

Tempo: 15 min.	7
Esercizio 11	8
Le quattro operazioni con <code>double</code>	8
Tempo: 15 min.	8

Parte 1. Esperimenti con l'output formattato di printf

Sequenze di escape

Negli esercizi di questa prima parte sperimenterete con l'output formattato di `printf`. Fate riferimento alla seguente tabella di sequenze escape del linguaggio C.

Escape	Nome	Effetto
\n	newline	va a capo
\r	carriage return	sposta il cursore a inizio riga
\t	horizontal tab	tabulazione orizzontale
\v	vertical tab	tabulazione verticale
\b	backspace	sposta il cursore indietro di un carattere
\'	single quote	visualizza un apice
\"	double quote	visualizza doppio apice
\?	question mark	visualizza punto interrogativo
\ooo	octal code	carattere in notazione ottale
\xhh	hexadecimal code	carattere in notazione esadecimale
\uhhhh	Unicode	carattere in notazione esadecimale Unicode

Alcune sequenze di escape saranno illustrate negli esercizi di questa dispensa.

ESERCIZIO 1

Il primo programma in C: Ciao Mondo.

Tempo: 10 min.

Si scriva un programma che produca in uscita la scritta **Ciao Mondo**.

ESERCIZIO 2

Ritorno a capo.

Tempo: 10 min.

Si scriva un programma che produca in uscita:

```
Ciao
Mondo
```

Si scriva una prima versione usando due chiamate a `printf`, e una seconda versione che usa una sola chiamata a `printf`.

Cosa succede se si tenta di scrivere il programma con una sola chiamata a `printf`, spezzandola però su due righe nel modo seguente?

```
printf("Ciao
Mondo")
```

Il programma compila senza errori? In caso di risposta negativa, che errore è segnalato?

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	,	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	-
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	:	87	57	1010111	127	W					
40	28	101000	50	{	88	58	1011000	130	X					
41	29	101001	51	}	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	.	93	5D	1011101	135	J					
46	2E	101110	56	,	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	-					

FIGURA 1. Il codice ASCII.

ESERCIZIO 3

Tabulazione orizzontale.

Tempo: 10 min.

Si scriva un programma che visualizzi sul terminale tre colonne di dati i cui campi siano allineati a sinistra, come in questo esempio:

Lun	10.30	Programmazione 1
Mar	14.00	Algebra
Mer	8.30	Geometria

Si usi la sequenza di escape \t per incolonnare i dati. Si provi poi a raddoppiare ciascuna occorrenza di \t per ottenere una maggiore spaziatura fra le colonne.

Il codice ASCII

I caratteri che si digitano tramite terminale hanno una rappresentazione *numerica* interna al calcolatore. Una delle codifiche storiche è nota come *codice ASCII*, acronimo di *American Standard Code for Information Interchange*. Sebbene oggi sia comune l'uso di codifiche più estese, come lo standard *Unicode*, il codice ASCII è compatibile con i nuovi standard. La Figura 1 riporta il codice ASCII.

ESERCIZIO 4

Codice ASCII.**Tempo:** 10 min.

Si scriva un programma che produca in uscita la scritta **Ciao** usando una sola chiamata a `printf`, e usando solo sequenze di escape della forma `\ooo`. (Si veda l'Avvertenza all'inizio della sezione.) Si ripeta l'esercizio usando solo sequenze di escape della forma `\xhh`.

Cosa succede se si tenta di compilare un programma la cui funzione `main` contiene solo l'istruzione seguente?

```
printf("\xFFFF");
```

Il programma compila senza errori? In caso di risposta negativa, che errore è segnalato? Sperimentate con le sequenze di escape `\ooo` e `\xhh`, provando a generare altre situazioni anomale.

Parte 2. Overflow & Underflow

ESERCIZIO 5

Range dei tipi integrali.**Tempo:** 10 min.

Includendo il file di intestazione `limits.h` della libreria standard, e usando le costanti `CHAR_MAX`, `CHAR_MIN`, `INT_MAX`, e `INT_MIN` lì definite, appurate il range dei tipi integrali `char` e `int` sul vostro sistema. Sulla base del risultato, quanti bit ritenete che siano usati, sul vostro sistema, per codificare un valore `int`?

ESERCIZIO 6

Range dei tipi reali.**Tempo:** 10 min.

Includendo il file di intestazione `float.h` della libreria standard, e usando le costanti `FLT_MAX` e `DBL_MAX` lì definite, appurate il massimo valore dei tipi `float` e `double` rappresentabile sul vostro sistema. Usate poi le costanti `FLT_MIN` e `DBL_MIN` per appurare il minimo valore positivo dei tipi `float` e `double` rappresentabile sul vostro sistema.

Negli Esercizi 6 e 8, usate `%g` invece di `%f` nella chiamata a `printf` per visualizzare i valori `float` e `double`. D'ufficio, `%f` visualizza solo 6 cifre decimali — troppo poco per distinguere `FLT_MIN` da 0, per esempio.

Tipi primitivi e variabili

A partire dal prossimo esercizio farete i primi passi nell'uso delle variabili e dei tipi primitivi. Ricordate che, per esempio, l'istruzione

```
int x;
```

dichiara la variabile di nome `x` e di tipo `int`; e che l'istruzione

```
x=-2;
```

assegna alla variabile (già dichiarata!) di nome `x` il valore intero `-2`. Dichiaraione e assegnamento si possono mettere assieme, volendo, in questo modo:

```
int x=-1;
```

Idem per i tipi primitivi `float`, `double` e `char`, *mutatis mutandis*.

ESERCIZIO 7

Overflow (con gli interi).

Tempo: 10 min.

Scrivete un programma che assegni alla variabile intera `num` il valore `INT_MAX` definito in `limits.h`. Visualizzate il valore di `num`. Sommate la costante 1 a `num`. Visualizzate di nuovo il valore di `num`. Datevi una spiegazione del risultato.

ESERCIZIO 8

Underflow (con i numeri in virgola mobile a precisione singola).

Tempo: 10 min.

Scrivete un programma che assegni alla variabile `num` di tipo `float` il valore `FLT_MIN` definito in `float.h`. Visualizzate il valore di `float`. Dividete `num` per una costante reale grande,¹ per esempio `FLT_MAX`. Visualizzate di nuovo il valore di `num`.

Parte 3. Primi passi coi tipi primitivi

Lettura di numeri con `scanf`

Negli esercizi di questa terza parte avrete bisogno di leggere numeri inseriti dall'utente. Per ora vi basterà usare in modo elementare la funzione `scanf` di `stdio.h`. La Figura 2 riporta un esempio che potrete usare come guida.

¹Vi è un motivo per cui non basta qui dividere per 2, diciamo, per osservare in modo ovvio il fenomeno dell'underflow. Non approfondiremo l'argomento.

```

1  /* Leggere numeri dal terminale con scanf */ scannum.c
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int i; float f; double d;
8      printf("Digita un int, un float, un double: ");
9      /* & e' necessario. */
10     scanf("%d%f%lf",&i,&f,&d);
11
12     printf("Hai digitato: %d,%g,%g.\n",i,f,d);
13
14     return 0;
15 }
```

FIGURA 2. Uso di `scanf` per leggere `int`, `float` e `double`.

ESERCIZIO 9

Eco di int e double dalla console.**Tempo:** 20 min.

Si scriva un programma che chieda all’utente di inserire un valore di tipo `int`, e lo riscriva subito dopo sul terminale. Il programma prosegue facendo la stessa cosa per un valore di tipo `double`.

Sperimentate con la funzione `scanf` usata in questo esercizio. Cosa succede se l’utente digita degli spazi o delle tabulazioni prima o dopo il numero? Cosa succede se, alla richiesta di un numero, l’utente immette J.S. Bach? E se invece immette 1685J.S. Bach? Fate altri esperimenti, fino a quando non vi sentiate relativamente a vostro agio con la semantica di `scanf` riguardo alla lettura di numeri.

ESERCIZIO 10

Le quattro operazioni con int.**Tempo:** 15 min.

Si scriva un programma che chieda all’utente di inserire due valori di tipo `int`, e visualizzi poi la loro somma, il valore del primo meno il secondo, il loro prodotto, il valore del primo diviso il secondo (divisione intera `/`), e il valore del resto della divisione del primo per il secondo. Per quel che riguarda l’ultimo punto, usate l’operatore binario di *modulo* `%`: l’espressione `a%b`, con `a` e `b` espressioni di tipo intero, vale il resto della divisione di `a` per `b`.

In relazione al programma che avete scritto per risolvere l'Esercizio 10, cosa succede se il secondo valore intero inserito dall'utente è zero?

ESERCIZIO 11

Le quattro operazioni con double.

Tempo: 15 min.

Si ripeta l'Esercizio 10 sostituendo **double** a **int**, *mutatis mutandis*.

Cosa succede se si tenta di eseguire $a \% b$ con a o b di tipo **double**?