LABORATORIO DI PROGRAMMAZIONE 1
CORSO DI LAUREA IN MATEMATICA
UNIVERSITÀ DEGLI STUDI DI MILANO
2018–2019
13.III.2019, PARTE 2

VINCENZO MARRA

Indice

Ultima revisione: 13 marzo 2019.

Parte 1. Esperimenti con l'output formattato di `printf`

Avvertenza

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
`\x`*hh*	hexadecimal code	carattere in notazione esadecimale
`\u`*hhhh*	Unicode	carattere in notazione esadecimale Unicode

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

Esercizio 1

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`.

Domanda

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
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

FIGURA 1. Il codice ASCII.

ESERCIZIO 2

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 3

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 `\x`*hh*.

Domanda

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 negativo, che errore è segnalato? **Sperimentate** con le sequenze di escape `\`*ooo* e `\x`*hh*, provando a generare altre situazioni anomale.

Parte 2. Overflow & Underflow

Esercizio 4

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 5

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.

Suggerimento

Nell'Esercizio 5, 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.

Avvertenza

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`. Dichiarazione 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 6

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 7

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 la costante reale grande[1] `1e20`. Visualizzate di nuovo il valore di `num`. **Datevi una spiegazione del risultato**.

Parte 3. Primi passi coi tipi primitivi

Avvertenza

Negli esercizi di questa terza parte userete la libreria didattica `Prog1` per acquisire dati inseriti dall'utente. Le istruzioni per l'uso della libreria si trovano alla pagina web del laboratorio. L'uso della libreria didattica sarà brevemente illustrato dai docenti durante questa lezione di laboratorio.

[1] Vi 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.

Esercizio 8

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`. (*Suggerimento.* Si usino le funzioni

```
int leggi_int(String msg)
double leggi_double(String msg)
```

della libreria `Prog1`. Occorre includere il file di intestazione `IO.h` della libreria.) **Sperimentate** con l'uso delle funzioni di `Prog1` utilizzate in questo esercizio. Cosa succede se, alla richiesta di un intero, l'utente immette `J.S. Bach`? E se invece immette `1685J.S. Bach`? *Idem* per il caso `double`. **Provate** l'uso delle due funzioni della libreria fino a che non vi sentite ragionevolmente a vostro agio con la loro semantica.

Domanda

Nel risolvere l'Esercizio 8, avete utilizzato variabili? Se sì, riscrivete il programma *senza* dichiarare alcuna variabile. Si può sostenere che questa seconda codifica sia "migliore" della prima? Se sì, in che senso?

Esercizio 9

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`.

Domanda

In relazione al programma che avete scritto per risolvere l'Esercizio 9, cosa succede se il secondo valore intero inserito dall'utente vale zero?

Esercizio 10

Le quattro operazioni con `double`.

Tempo: 15 min.

Si ripeta l'Esercizio 9 sostituendo `double` a `int`, *mutatis mutandis*. (Cosa succede se si tenta di eseguire `a%b` con `a` o `b` di tipo double?)

(V. Marra) Dipartimento di Matematica *Federigo Enriques*, Università degli Studi di Milano, via Cesare Saldini, 50, I-20133 Milano
Email address: vincenzo.marra@unimi.it