

Prova d'Esame

Programmazione Cl. B (Parte 2)

02 Febbraio 2023

Ingegneria e Scienze Informatiche A.A. 2022-2023

Si vuole implementare in linguaggio C una libreria di funzioni per una versione semplificata del gioco del paroliere.

Si deve consegnare un file **paroliere.c** contenente l'implementazione delle funzioni che si trovano nel file header **paroliere.h**.

Regole del gioco

Si ha a disposizione una board con delle "tessere". Ciascuna tessera evidenzia una lettera ed un numero, che ne determina il valore in termini di punteggio.

Lo scopo del giocatore è quello di comporre delle parole con le lettere della board, massimizzando il punteggio.

Per semplicità, la posizione delle tessere non conta, è possibile comporre parole anche usando lettere che non sono vicine.

Parole valide:

- Fieno (9 punti)
- Rana (5 punti)
- Laringe (12 punti)
- Pieni (8 punti)
- Falena (11 punti)
- Fagiano (14 punti)
- Grano (9 punti)

board		letter	
A (1)	R (1)	L (2)	E (1)
O (1)	N (2)	F (4)	I (1)
P (3)	I (1)	G (4)	A (1)

Parole non valide:

- Rene (C'è un'unica E)
- Finte (La T non c'è)
- Argo (Non è presente nel dizionario)

value

tile

A (1)	R (1)	L (2)	E (1)
O (1)	N (2)	F (4)	I (1)
P (3)	I (1)	G (4)	A (1)

Per rappresentare i concetti di board e di tessera si definiscono le seguenti strutture:

```
/*Definisce una tessera*/
struct tile {
    char letter;
    int value;
};

/*Definisce la board*/
struct board {
    struct tile tiles[NROWS][NCOLS];
    int width;
    int height;
};
```

Le funzioni da implementare sono le seguenti:

- **board_gen (4 punti):** crea una board, la inizializza e la restituisce. L'inizializzazione della board viene fatta impostando i valori corretti di altezza e larghezza e assegnando a ciascuna tile una lettera **casuale** e il relativo valore.

Il numero di righe e colonne dell'array bidimensionale "tiles" è definito nel file paroliere.h

Si prende come riferimento l'alfabeto italiano, escludendo quindi le lettere J,K,W,X,Y,Z.

Di seguito la tabella dei punteggi per ciascuna lettera:

1 punto	A, C, E, I, O, R, S, T
2 punti	L, M, N
3 punti	P
4 punti	B,D,F,G,U,V
8 punti	H,Z
9 punti	Q

La signature della funzione è:

```
struct board board_gen();
```

- **check_letters (4 punti):** prende in input una parola e una board.
Restituisce:
 - **1** se è possibile comporre la parola usando le tessere della board
 - **0** altrimenti

NB: la funzione non deve fare distinzione tra maiuscole e minuscole.

La signature della funzione è la seguente:

```
int check_letters(char* word, struct board b);
```

- **check_dictionary (4 punti):** prende in input una parola e il nome del file di testo contenente il dizionario di riferimento.

Restituisce:

- **1** se la parola è contenuta nel dizionario
- **0** altrimenti

NB: la funzione non deve fare distinzione tra maiuscole e minuscole.

La signature della funzione è:

```
int check_dictionary(char* word, char* filename);
```

- **compute_score (4 punti):** funzione **ricorsiva** che prende in input una parola e la sua lunghezza. Computa ricorsivamente la somma dei punteggi di ogni lettera che compone la parola. Ritorna il punteggio complessivo.

La signature della funzione è la seguente:

```
int compute_score(char* word, int length);
```

Strumenti utili

Tabella ASCII

0		32		64	@	96	`	128	Ç	160	á	192	Ł	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	⌂	225	ß
2	☼	34	"	66	B	98	b	130	é	162	ó	194	└	226	Ë
3	♥	35	#	67	C	99	c	131	â	163	ú	195	┌	227	Ò
4	♦	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	ô
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	+	229	Õ
6	♠	38	&	70	F	102	f	134	á	166	ª	198	â	230	μ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
8	◼	40	(72	H	104	h	136	ê	168	¿	200	⌋	232	ƒ
9	◊	41)	73	I	105	i	137	ë	169	®	201	┐	233	Ú
10	◼	42	*	74	J	106	j	138	è	170	¬	202	⌋	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	└	235	Ü
12	♀	44	,	76	L	108	l	140	î	172	¾	204	┐	236	Ý
13	🎵	45	-	77	M	109	m	141	ï	173	ı	205	=	237	Ÿ
14	🎵	46	.	78	N	110	n	142	Ä	174	«	206	┐	238	—
15	☼	47	/	79	O	111	o	143	Å	175	»	207	◻	239	˙
16	▶	48	0	80	P	112	p	144	É	176	◼	208	◻	240	-
17	◀	49	1	81	Q	113	q	145	æ	177	◼	209	◻	241	±
18	↑	50	2	82	R	114	r	146	Æ	178	◼	210	◻	242	—
19	!!	51	3	83	S	115	s	147	ø	179		211	◻	243	¼
20	¶	52	4	84	T	116	t	148	ö	180	└	212	◻	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	—	54	6	86	V	118	v	150	û	182	Â	214	ı	246	÷
23	↑	55	7	87	W	119	w	151	ù	183	Ã	215	ı	247	,
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	ı	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	¶	217	└	249	ˆ
26	→	58	:	90	Z	122	z	154	Ü	186		218	┐	250	˙
27	←	59	;	91	[123	{	155	ø	187	¶	219	◼	251	¹
28	└	60	<	92	\	124		156	£	188	┐	220	◼	252	º
29	↔	61	=	93]	125	}	157	Ø	189	φ	221	ı	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	ı	254	■
31	▼	63	?	95	_	127	◻	159	f	191	└	223	◼	255	

Funzioni principali della libreria **string.h**

Funzione	Descrizione
char * strcat (char *dest, const char *src);	Concatena la stringa src alla stringa dest
char * strncat (char *dest, const char *src, size_t n);	Concatena n caratteri della stringa src alla stringa dest
char *strchr(const char *str, int c);	Restituisce il puntatore alla prima occorrenza del carattere c all'interno della stringa str
char *strrchr(const char *str, int c);	Restituisce il puntatore all'ultima occorrenza del carattere c all'interno della stringa str
int strcmp (const char *str1, const char *str2);	Confronta la stringa str1 con la stringa str2
int strncmp (const char *str1, const char *str2, size_t n);	Confronta al massimo i primi n caratteri della stringa str1 con la stringa str2
int strcoll(const char *str1, const char *str2);	Confronta le due stringhe str1 e str2 utilizzando l'ordine lessicografico
char * strcpy (char *dest, const char *src);	Copia la stringa src nella stringa dest
char * strncpy (char *dest, const char *src, size_t n);	Copia al massimo n caratteri della stringa src nella stringa dest
size_t strlen (const char *str);	Restituisce la lunghezza della stringa str
size_t strspn(const char *str1, const char *str2);	Restituisce la lunghezza iniziale della stringa str1 che consiste interamente da caratteri presenti nella stringa str2
size_t strcspn(const char *str1, const char *str2);	Restituisce la lunghezza iniziale della stringa str1 che consiste interamente da caratteri non presenti nella stringa str2
char *strpbrk(const char *str1, const char *str2);	Restituisce il puntatore al primo carattere nella stringa str1 che corrisponde a qualsiasi carattere presente nella stringa str2
char *strstr(const char *haystack, const char *needle);	Trova la prima occorrenza della stringa needle all'interno della stringa haystack
char *strtok(char *str, const char *delimiters);	Spezza la stringa str in una serie di token separati dal carattere delimiters