

```
#include <stdio.h>
#include <ctype.h>
#define MAXPAGINA 80
```

```
int main(int argc, char *argv[])
{
```

```
    int freq[MAXPAGINA]; /* vettore di contenuto delle frequenze delle parole */
```

```
    int i, min, max, j;
```

```
    float spazio_disco, pagine;
```

```
    FILE *fp;
```

```
    fp=fopen(argv[1], "r");
```

```
    if(fp==NULL)
```

```
        fprintf(stderr, "ERRORE, non posso aprire il file %s", argv[1]);
```

```
    else
```

```
        pagina=0;
```

```
        spazio_disco=0.0;
```

```
        for(i=0;i<MAXPAGINA;i++)
```

```
            freq[i]=0;
```

```
        while(fgets(fp, 1000, fp))
```

```
            pagine+=1.0;
```

```
            for(j=0;j<1000;j++)
```

```
                if(isalpha(fp[j]))
```

```
                    freq[fp[j]-65]++;
```

```
                else if(fp[j]=='.')
```

```
                    freq[26]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[27]++;
```

```
                else if(fp[j]==';')
```

```
                    freq[28]++;
```

```
                else if(fp[j]==':')
```

```
                    freq[29]++;
```

```
                else if(fp[j]=='-')
```

```
                    freq[30]++;
```

```
                else if(fp[j]=='+')
```

```
                    freq[31]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[32]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[33]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[34]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[35]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[36]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[37]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[38]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[39]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[40]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[41]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[42]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[43]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[44]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[45]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[46]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[47]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[48]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[49]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[50]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[51]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[52]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[53]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[54]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[55]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[56]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[57]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[58]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[59]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[60]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[61]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[62]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[63]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[64]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[65]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[66]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[67]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[68]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[69]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[70]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[71]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[72]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[73]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[74]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[75]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[76]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[77]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[78]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[79]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[80]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[81]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[82]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[83]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[84]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[85]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[86]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[87]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[88]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[89]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[90]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[91]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[92]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[93]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[94]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[95]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[96]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[97]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[98]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[99]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[100]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[101]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[102]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[103]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[104]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[105]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[106]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[107]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[108]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[109]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[110]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[111]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[112]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[113]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[114]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[115]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[116]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[117]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[118]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[119]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[120]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[121]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[122]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[123]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[124]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[125]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[126]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[127]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[128]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[129]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[130]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[131]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[132]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[133]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[134]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[135]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[136]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[137]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[138]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[139]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[140]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[141]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[142]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[143]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[144]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[145]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[146]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[147]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[148]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[149]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[150]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[151]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[152]++;
```

```
                else if(fp[j]==',')
```

```
                    freq[153]++;
```

```
                else if(fp[j]==',')
```

- Tipi di dato testuali
- Caratteri
- Stringhe

Caratteri e stringhe

Dati testuali

5

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

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di costante
    cioè numero massimo di caratteri nella parola */
    char rigaMAXRIGA; /* valore di costante
    cioè numero massimo di caratteri della riga */
    int i, indice, lungRiga;
    FILE *fptr;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((fptr = fopen(argv[1], "r")) == NULL)
    {
        perror("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    while(fgets(riga, MAXRIGA, fptr) != NULL)
    {
        if(strlen(riga) > lungMAXRIGA)
        {
            printf("ERRORE: la riga %s supera il limite massimo (%d)\n", riga, lungMAXRIGA);
            exit(1);
        }
        else
        {
            for(i=0; riga[i] != '\0'; i++)
            {
                if(isalpha(riga[i]))
                {
                    printf("%c", riga[i]);
                }
                else
                {
                    printf("\\%c", riga[i]);
                }
            }
            printf("\n");
        }
    }
    fclose(fptr);
}
```

Dati testuali

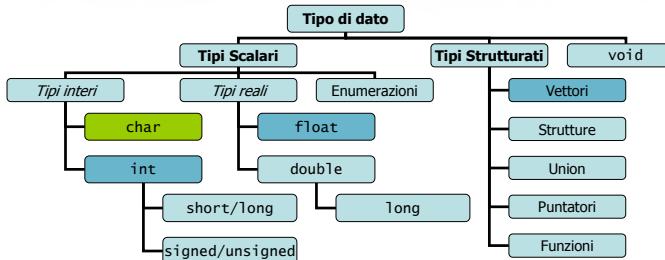
Tipi di dato testuali

7

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 C



8

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

9

Codice ASCII

Dec	Hx	Ovt	Char	Dec	Hx	Ovt	Ctrl	Dec	Hx	Ovt	Ctrl	Dec	Hx	Ovt	Ctrl
0	0 000	NUL	(null)	54	40 100	#44-#4		84	50 160	#40-#4		97	50 160	#40-#4	
1	0 001	SOH	(start of heading)	32	20 040	#32-#32		32	31 241	#33-#33		65	41 101	#65-#5	
2	0 002	STX	(start of text)	32	34 042	#34-#32		66	42 101	#66-#6		92	62 141	#88-#8	
3	0 003	ETX	(end of text)	32	35 043	#35-#35		67	43 101	#67-#7		99	64 142	#84-#8	
4	0 004	SZ	(size of transmission)	32	36 044	#36-#36		68	44 101	#68-#8		100	65 143	#85-#8	
5	0 005	ENQ	(enquiry)	32	37 045	#37-#37		69	45 101	#69-#9		101	65 145	#81-#1	
6	0 006	ACK	(acknowledge)	32	38 046	#38-#2		70	46 106	#70-#70		102	66 145	#82-#2	
7	0 007	BEL	(bell)	32	39 047	#39-#7		71	47 107	#71-#71	G	87	67 147	#103-#9	
8	0 010	BS	(backspace)	40	28 050	#40-#2		72	48 110	#72-#72		104	68 148	#104-#2	
9	0 011	TAB	(horizontal tab)	41	29 051	#41-#1		73	49 111	#73-#73		105	69 151	#105-#1	
10	0 012	LF	(line feed, new line)	42	30 052	#42-#2		74	50 112	#74-#74		106	70 152	#106-#2	
11	0 013	VT	(vertical tab)	43	32 053	#43-#3		75	48 113	#75-#5		107	68 153	#107-#2	
12	0 014	FF	(FF form feed, new page)	44	24 054	#44-#44		76	44 114	#76-#6	L	106	65 154	#108-#1	
13	0 015	CR	(carriage return)	45	20 055	#45-#5		77	44 116	#77-#7		108	60 155	#109-#9	
14	0 016	FS	(shift out)	46	26 056	#46-#6		78	46 116	#78-#8	N	110	66 156	#111-#8	
15	0 017	GS	(shift in)	47	27 057	#47-#7		79	47 117	#79-#9	P	111	67 157	#112-#9	
16	0 020	DLE	(data link escape)	48	30 060	#48-#0		80	50 120	#80-#0	R	112	70 160	#112-#2	
17	0 021	DCL	(device control 1)	49	31 061	#49-#1		81	51 121	#81-#1	Q	113	71 161	#113-#9	
18	0 022	D2C	(device control 2)	50	32 062	#50-#2		82	52 122	#82-#2	R	114	72 162	#114-#2	
19	0 023	D3C	(device control 3)	51	33 063	#51-#3		83	53 123	#83-#3	S	115	73 163	#115-#3	
20	0 024	D4C	(device control 4)	52	34 064	#52-#4		84	54 124	#84-#4	T	116	74 164	#116-#4	
21	0 025	MAY	(may, no acknowledge)	53	35 065	#53-#5		85	55 125	#85-#5	U	117	75 165	#117-#5	
22	0 026	SYN	(synchronous idle)	54	36 066	#54-#6		86	56 126	#86-#6	V	118	76 166	#118-#6	
23	0 027	FTB	(end of trans. block)	55	37 067	#55-#7		87	57 127	#87-#7	W	119	77 167	#119-#7	
24	0 030	CAN	(cancel)	56	38 070	#56-#8		88	58 128	#88-#8	X	120	78 168	#120-#8	
25	0 031	EM	(end of medium)	57	39 071	#57-#9		89	59 129	#89-#9	Y	121	79 169	#121-#9	
26	0 032	SUB	(substitute)	58	40 072	#58-#10		90	60 130	#90-#10	Z	122	80 170	#122-#10	
27	0 033	ESC	(escape)	59	58 073	#59-#11		91	58 131	#91-#11	!	123	81 171	#123-#11	
28	0 034	FS	(file separator)	60	37 074	#60-#12		92	59 132	#92-#12	,	124	72 174	#124-#12	
29	0 035	GS	(group separator)	61	38 075	#61-#13		93	59 133	#93-#13	:	125	73 175	#125-#13	
30	0 036	RS	(record separator)	62	39 076	#62-#14		94	59 134	#94-#14	_	126	74 176	#126-#14	
31	0 037	US	(unit separator)	63	37 077	#63-#15		95	59 137	#95-#15	?	127	75 177	#127-#15	

Source: www.lookuptables.com

1

- 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	

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

11

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	

1

- ▶ 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

1

Caratteri e stringhe

- Il codice ASCII permette di rappresentare un singolo **carattere**

- ▶ 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

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

11

Dati testuali

Caratteri

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

1

Codice ASCII

Dec	Hex	Oct	Char	Dec	Hex	Oct	Hmlt	Char	Dec	Hex	Oct	Hmlt	Char	
0	0000	000	(null)	32	2040	3200	Space	64	4000	1040	#48:	9	128	0000 1040 #48: #9
1	0001	001	SOH (start of heading)	33	2140	0031	#33:	65	4100	1045:	#5:	9	97	61 141 #97: #5:
2	0020	002	STX (start of text)	32	2040	0042	#54:	65	4100	1065:	#6:	9	97	61 141 #97: #6:
3	0030	003	ETX (end of text)	55	2320	0053	#75:	79	4300	1065:	#7:	9	98	61 143 #98: #7:
4	0040	004	ENQ (enquiry)	34	2140	0043	#53:	64	4000	1045:	#43:	9	97	61 140 #97: #43:
5	0050	005	ENO (enquiry)	37	2340	0037	#73:	69	4500	1059:	#E:	9	101	61 145 #101: #E:
6	0060	006	ACK (acknowledge)	36	2640	0036	#74:	70	4600	1070:	#70:	9	102	61 146 #102: #70:
7	0070	007	BEL (bell)	37	2740	0039	#75:	71	4700	1071:	#71:	9	103	61 147 #103: #71:
8	0080	008	BAC (backspace)	38	2840	0038	#76:	72	4800	1072:	#72:	9	104	61 148 #104: #72:
9	0090	009	HT (horizontal tab)	39	2940	0041	#77:	73	4900	1073:	#73:	9	105	61 149 #105: #73:
10	00A0	00A	LF (NL line feed, new line)	40	2A40	0042	#74:	74	4A00	1044:	#44:	9	106	61 150 #106: #44:
11	00B0	00B	VT (vertical tab)	39	2850	0043	#75:	75	4B00	1133:	#75:	9	107	61 153 #107: #75:
12	00C0	00C	FF (NP form feed, new page)	40	2A40	0054	#44:	76	4C00	1144:	#6:	9	108	61 154 #108: #6:
13	00D0	00D	CR (carriage return)	45	2D20	0045	#45:	77	4D00	1155:	#7:	9	109	61 155 #109: #7:
14	00E0	00E	FS (file separator)	46	2D40	0056	#46:	78	4E00	1166:	#8:	9	110	61 156 #110: #8:
15	00F0	00F	ST (shift in)	47	2F20	0057	#47:	79	4F00	1177:	#9:	9	111	61 157 #111: #9:
16	00102	002	DLE (data link escape)	48	3060	0040	#80:	80	5000	1120:	#80:	9	112	70 160 #112: #80:
17	00122	002	DCL (device control 1)	51	3160	0049	#1:	81	5120	1181:	#1:	9	113	71 161 #113: #1:
18	00124	002	DCC (device control 2)	52	3120	0050	#2:	82	5212	1182:	#2:	9	114	72 162 #114: #2:
19	00126	002	DCC (device control 3)	53	3160	0051	#3:	83	5212	1183:	#3:	9	115	72 163 #115: #3:
20	00124	002	DCC (device control 4)	52	3404	0052:	#4:	84	5414	1144:	#4:	9	116	74 164 #116: #4:
21	00125	002	NAY (negative acknowledge)	53	3565	0053:	#5:	85	5515	1155:	#5:	9	117	75 165 #117: #5:
22	00126	002	SYN (synchronous idle)	54	3666	0054:	#6:	86	5616	1166:	#6:	9	118	76 166 #118: #6:
23	00127	002	ETB (end of trans. block)	57	3767	0057:	#7:	87	5717	1177:	#7:	9	119	77 167 #119: #7:
24	00128	002	NAK (negative acknowledgement)	58	3868	0058:	#8:	88	5818	1188:	#8:	9	120	78 168 #120: #8:
25	00131	003	EM (end of medium)	59	3701	0057:	#79:	89	5913	1169:	#9:	9	121	79 171 #121: #9:
26	00132	003	SUB (substitute)	58	3702	0058:	#80:	90	5913	1152:	#1:	9	122	72 172 #122: #1:
27	00133	003	ESC (escape)	59	3873	0059:	#81:	91	5913	1153:	#1:	9	123	78 173 #123: #1:
28	00135	003	FS (file separator)	60	3874	0058:	#8:	92	5C13	1192:	#2:	9	124	78 174 #124: #2:
29	00133	003	GS (group separator)	61	3703	0057:	#81:	93	5D13	1175:	#3:	9	125	79 175 #125: #3:
30	00136	003	RE (record separator)	62	3876	0062:	#84:	94	5E13	1186:	#4:	9	126	78 176 #126: #4:
31	00137	003	US (unit separator)	63	3707	0062:	#85:	95	5F13	1195:	#5:	9	127	77 177 #127: #5:

Source: www.lookuptables.com

15

Codice ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Char
0	\000	000	NUL (null)	41	6#		
1	\001	001	SOH (start of header)	42	6#		
2	\002	002	STX (start of text)	43	6#		
3	\003	003	ETX (end of text)	44	6#		
4	\004	004	(end of transmission)	36	24	044	6#

Valore decimale
(tra 0 e 127)

Simbolo corrispondente

16

Source: www.lookuptables.com

Codice ASCII

Dec	Hx	Oct	Html	Chr
96	60	140	`	
97	61	141	a	à
98	62	142	b	à
99	63	143	c	à
100	64	144	d	à
101	65	145	e	à
102	66	146	f	à
103	67	147	g	à
104	68	150	h	à
105	69	151	i	à
106	70	152	j	à
107	71	153	k	à
108	72	154	l	à
109	73	155	m	à
110	74	156	n	à
111	75	157	o	à
112	76	160	p	à
113	77	161	q	à
114	78	162	r	à

Lettere minuscole

Lettere maiuscole

17

Source: www.lookuptables.com

Codice ASCII

Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
Space	64	40	100	@	à	96	60	140	`	à
0	65	41	101	A	à	97	61	141	a	à
1	66	42	102	B	à	98	62	142	b	à
2	67	43	103	C	à	99	63	143	c	à
3	68	44	104	D	à	100	64	144	d	à
4	69	45	105	E	à	101	65	145	e	à
5	70	46	106	B	à	102	66	146	f	à
6	71	47	107	C	à	103	67	147	g	à
7	72	48	108	D	à	104	68	148	h	à
8	73	49	109	E	à	105	69	149	i	à
9	74	50	110	<	à	106	70	150	j	à
;	75	51	111	=	à	107	71	151	k	à
,	76	52	112	>	à	108	72	152	l	à
:	77	53	113	?	à	109	73	153	m	à
;	78	54	114	@	à	110	74	154	n	à
;	79	55	115	A	à	111	75	155	o	à
;	80	56	116	B	à	112	76	156	p	à
;	81	57	117	C	à	113	77	157	q	à
;	82	58	118	D	à	114	78	158	r	à
;	83	59	119	E	à	115	79	159	s	à
;	84	60	120	<	à	116	70	160	t	à
;	85	61	121	=	à	117	71	161	u	à
;	86	62	122	>	à	118	72	162	v	à
;	87	63	123	?	à	119	73	163	w	à
;	88	64	124	@	à	120	74	164	n	à
;	89	65	125	A	à	121	75	165	o	à
;	90	66	126	B	à	122	76	166	p	à
;	91	67	127	C	à	123	77	167	q	à
;	92	68	128	D	à	124	78	170	r	à
;	93	69	129	E	à	125	79	171	s	à
;	94	70	130	<	à	126	70	172	t	à
;	95	71	131	=	à	127	71	173	u	à
;	96	72	132	>	à	128	72	174	v	à
;	97	73	133	?	à	129	73	175	w	à
;	98	74	134	@	à	130	74	176	n	à
;	99	75	135	A	à	131	75	177	o	à
;	100	76	136	B	à	132	76	178	p	à
;	101	77	137	C	à	133	77	179	q	à
;	102	78	138	D	à	134	78	180	r	à
;	103	79	139	E	à	135	79	181	s	à
;	104	80	140	<	à	136	70	182	t	à
;	105	81	141	=	à	137	71	183	u	à
;	106	82	142	>	à	138	72	184	v	à
;	107	83	143	?	à	139	73	185	w	à
;	108	84	144	@	à	140	74	186	n	à
;	109	85	145	A	à	141	75	187	o	à
;	110	86	146	B	à	142	76	188	p	à
;	111	87	147	C	à	143	77	189	q	à
;	112	88	148	D	à	144	78	190	r	à
;	113	89	149	E	à	145	79	191	s	à
;	114	90	150	<	à	146	70	192	t	à
;	115	91	151	=	à	147	71	193	u	à
;	116	92	152	>	à	148	72	194	v	à
;	117	93	153	?	à	149	73	195	w	à
;	118	94	154	@	à	150	74	196	n	à
;	119	95	155	A	à	151	75	197	o	à
;	120	96	156	B	à	152	76	198	p	à
;	121	97	157	C	à	153	77	199	q	à
;	122	98	158	D	à	154	78	200	r	à
;	123	99	159	E	à	155	79	201	s	à
;	124	100	160	<	à	156	70	202	t	à
;	125	101	161	=	à	157	71	203	u	à
;	126	102	162	>	à	158	72	204	v	à
;	127	103	163	?	à	159	73	205	w	à
;	128	104	164	@	à	160	74	206	n	à
;	129	105	165	A	à	161	75	207	o	à
;	130	106	166	B	à	162	76	208	p	à
;	131	107	167	C	à	163	77	209	q	à
;	132	108	168	D	à	164	78	210	r	à
;	133	109	169	E	à	165	79	211	s	à
;	134	110	170	<	à	166	70	212	t	à
;	135	111	171	=	à	167	71	213	u	à
;	136	112	172	>	à	168	72	214	v	à
;	137	113	173	?	à	169	73	215	w	à
;	138	114	174	@	à	170	74	216	n	à
;	139	115	175	A	à	171	75	217	o	à
;	140	116	176	B	à	172	76	218	p	à
;	141	117	177	C	à	173	77	219	q	à
;	142	118	178	D	à	174	78	220	r	à
;	143	119	179	E	à	175	79	221	s	à
;	144	120	180	<	à	176	70	222	t	à
;	145	121	181	=	à	177	71	223	u	à
;	146	122	182	>	à	178	72	224	v	à
;	147	123	183	?	à	179	73	225	w	à
;	148	124	184	@	à	180	74	226	n	à
;	149	125	185	A	à	181	75	227	o	à
;	150	126	186	B	à	182	76	228	p	à
;	151	127	187	C	à	183	77	229	q	à
;	152	128	188	D	à	184	78	230	r	à
;	153	129	189	E	à	185	79	231	s	à
;	154	130	190	<	à	186	70	232	t	à
;	155	131	191	=	à	187	71	233	u	à
;	156	132	192	>	à	188	72	234	v	à
;	157	133	193	?	à	189	73	235	w	à
;	158	134	194	@	à	190	74	236	n	à
;	159	135	195	A	à	191	75	237	o	à
;	160	136	196	B	à	192	76	238	p	à
;	161	137	197	C						



Errore frequente

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

int
7

char
7
55

- Per chiarezza useremo gli apici per indicare i caratteri

char
'7'
55

22



Errore frequente

- Pensare che un singolo carattere possa memorizzare più simboli

char
Fulvio

char F
55
char u
117
char l
108

23

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

#define MAXPAROLA 30
#define MAXCIGA 80

int main(int argc, char *argv[])
{
    int lunghezzaParola; /* /> valore di controllo
    char testuale[MAXCIGA]; /* /> stringa testuale inserita dall'utente */
    char risposta[MAXCIGA];
    int i, lunghezza;
    i = 0;

    for(i=0; i<MAXCIGA; i++)
        testuale[i] = '\0';

    if(argc > 1)
    {
        /* legge la parola inserita dall'utente */
        strcpy(testuale, argv[1]);
        /* legge la risposta inserita dall'utente */
        strcpy(risposta, argv[2]);
        /* stampa la stringa inserita dall'utente */
        printf("Inserisci: %s", testuale);
        /* stampa la risposta inserita dall'utente */
        printf("Risposta: %s", risposta);
    }
    else
        /* stampa la stringa inserita dall'utente */
        printf("Inserisci: %s", testuale);

    /* stampa la lunghezza della stringa inserita dall'utente */
    printf("\nLunghezza: %d", lunghezzaParola);
}
```

Dati testuali

Stringhe

25

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

26

Manipolazione 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

27



Errore frequente

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

int	char
137	1 3 7
49 51 55	

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

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di controllo delle frequenze delle lunghezze delle parole */
    char freq[MAXSIGA]; /* frequenze delle lunghezze */
    int i, indice, lunghezza;
    FILE *fptr;
    char riga[100];
    int nlinee;
    int spazio;
    int c;
    int posizione;
    int maxlunghezza;
    int maxfreq;
    int i1, i2, i3;
    int i4, i5, i6;
    int i7, i8, i9;
    int i10, i11, i12;
    int i13, i14, i15;
    int i16, i17, i18;
    int i19, i20, i21;
    int i22, i23, i24;
    int i25, i26, i27;
    int i28, i29, i30;
    int i31, i32, i33;
    int i34, i35, i36;
    int i37, i38, i39;
    int i39, i40, i41;
    int i42, i43, i44;
    int i45, i46, i47;
    int i47, i48, i49;
    int i49, i50, i51;
    int i51, i52, i53;
    int i53, i54, i55;
    int i55, i56, i57;
    int i57, i58, i59;
    int i59, i60, i61;
    int i61, i62, i63;
    int i63, i64, i65;
    int i65, i66, i67;
    int i67, i68, i69;
    int i69, i70, i71;
    int i71, i72, i73;
    int i73, i74, i75;
    int i75, i76, i77;
    int i77, i78, i79;
    int i79, i80, i81;
    int i81, i82, i83;
    int i83, i84, i85;
    int i85, i86, i87;
    int i87, i88, i89;
    int i89, i90, i91;
    int i91, i92, i93;
    int i93, i94, i95;
    int i95, i96, i97;
    int i97, i98, i99;
    int i99, i100, i101;
    int i101, i102, i103;
    int i103, i104, i105;
    int i105, i106, i107;
    int i107, i108, i109;
    int i109, i110, i111;
    int i111, i112, i113;
    int i113, i114, i115;
    int i115, i116, i117;
    int i117, i118, i119;
    int i119, i120, i121;
    int i121, i122, i123;
    int i123, i124, i125;
    int i125, i126, i127;
    int i127, i128, i129;
    int i129, i130, i131;
    int i131, i132, i133;
    int i133, i134, i135;
    int i135, i136, i137;
    int i137, i138, i139;
    int i139, i140, i141;
    int i141, i142, i143;
    int i143, i144, i145;
    int i145, i146, i147;
    int i147, i148, i149;
    int i149, i150, i151;
    int i151, i152, i153;
    int i153, i154, i155;
    int i155, i156, i157;
    int i157, i158, i159;
    int i159, i160, i161;
    int i161, i162, i163;
    int i163, i164, i165;
    int i165, i166, i167;
    int i167, i168, i169;
    int i169, i170, i171;
    int i171, i172, i173;
    int i173, i174, i175;
    int i175, i176, i177;
    int i177, i178, i179;
    int i179, i180, i181;
    int i181, i182, i183;
    int i183, i184, i185;
    int i185, i186, i187;
    int i187, i188, i189;
    int i189, i190, i191;
    int i191, i192, i193;
    int i193, i194, i195;
    int i195, i196, i197;
    int i197, i198, i199;
    int i199, i200, i201;
    int i201, i202, i203;
    int i203, i204, i205;
    int i205, i206, i207;
    int i207, i208, i209;
    int i209, i210, i211;
    int i211, i212, i213;
    int i213, i214, i215;
    int i215, i216, i217;
    int i217, i218, i219;
    int i219, i220, i221;
    int i221, i222, i223;
    int i223, i224, i225;
    int i225, i226, i227;
    int i227, i228, i229;
    int i229, i230, i231;
    int i231, i232, i233;
    int i233, i234, i235;
    int i235, i236, i237;
    int i237, i238, i239;
    int i239, i240, i241;
    int i241, i242, i243;
    int i243, i244, i245;
    int i245, i246, i247;
    int i247, i248, i249;
    int i249, i250, i251;
    int i251, i252, i253;
    int i253, i254, i255;
    int i255, i256, i257;
    int i257, i258, i259;
    int i259, i260, i261;
    int i261, i262, i263;
    int i263, i264, i265;
    int i265, i266, i267;
    int i267, i268, i269;
    int i269, i270, i271;
    int i271, i272, i273;
    int i273, i274, i275;
    int i275, i276, i277;
    int i277, i278, i279;
    int i279, i280, i281;
    int i281, i282, i283;
    int i283, i284, i285;
    int i285, i286, i287;
    int i287, i288, i289;
    int i289, i290, i291;
    int i291, i292, i293;
    int i293, i294, i295;
    int i295, i296, i297;
    int i297, i298, i299;
    int i299, i300, i301;
    int i301, i302, i303;
    int i303, i304, i305;
    int i305, i306, i307;
    int i307, i308, i309;
    int i309, i310, i311;
    int i311, i312, i313;
    int i313, i314, i315;
    int i315, i316, i317;
    int i317, i318, i319;
    int i319, i320, i321;
    int i321, i322, i323;
    int i323, i324, i325;
    int i325, i326, i327;
    int i327, i328, i329;
    int i329, i330, i331;
    int i331, i332, i333;
    int i333, i334, i335;
    int i335, i336, i337;
    int i337, i338, i339;
    int i339, i340, i341;
    int i341, i342, i343;
    int i343, i344, i345;
    int i345, i346, i347;
    int i347, i348, i349;
    int i349, i350, i351;
    int i351, i352, i353;
    int i353, i354, i355;
    int i355, i356, i357;
    int i357, i358, i359;
    int i359, i360, i361;
    int i361, i362, i363;
    int i363, i364, i365;
    int i365, i366, i367;
    int i367, i368, i369;
    int i369, i370, i371;
    int i371, i372, i373;
    int i373, i374, i375;
    int i375, i376, i377;
    int i377, i378, i379;
    int i379, i380, i381;
    int i381, i382, i383;
    int i383, i384, i385;
    int i385, i386, i387;
    int i387, i388, i389;
    int i389, i390, i391;
    int i391, i392, i393;
    int i393, i394, i395;
    int i395, i396, i397;
    int i397, i398, i399;
    int i399, i400, i401;
    int i401, i402, i403;
    int i403, i404, i405;
    int i405, i406, i407;
    int i407, i408, i409;
    int i409, i410, i411;
    int i411, i412, i413;
    int i413, i414, i415;
    int i415, i416, i417;
    int i417, i418, i419;
    int i419, i420, i421;
    int i421, i422, i423;
    int i423, i424, i425;
    int i425, i426, i427;
    int i427, i428, i429;
    int i429, i430, i431;
    int i431, i432, i433;
    int i433, i434, i435;
    int i435, i436, i437;
    int i437, i438, i439;
    int i439, i440, i441;
    int i441, i442, i443;
    int i443, i444, i445;
    int i445, i446, i447;
    int i447, i448, i449;
    int i449, i450, i451;
    int i451, i452, i453;
    int i453, i454, i455;
    int i455, i456, i457;
    int i457, i458, i459;
    int i459, i460, i461;
    int i461, i462, i463;
    int i463, i464, i465;
    int i465, i466, i467;
    int i467, i468, i469;
    int i469, i470, i471;
    int i471, i472, i473;
    int i473, i474, i475;
    int i475, i476, i477;
    int i477, i478, i479;
    int i479, i480, i481;
    int i481, i482, i483;
    int i483, i484, i485;
    int i485, i486, i487;
    int i487, i488, i489;
    int i489, i490, i491;
    int i491, i492, i493;
    int i493, i494, i495;
    int i495, i496, i497;
    int i497, i498, i499;
    int i499, i500, i501;
    int i501, i502, i503;
    int i503, i504, i505;
    int i505, i506, i507;
    int i507, i508, i509;
    int i509, i510, i511;
    int i511, i512, i513;
    int i513, i514, i515;
    int i515, i516, i517;
    int i517, i518, i519;
    int i519, i520, i521;
    int i521, i522, i523;
    int i523, i524, i525;
    int i525, i526, i527;
    int i527, i528, i529;
    int i529, i530, i531;
    int i531, i532, i533;
    int i533, i534, i535;
    int i535, i536, i537;
    int i537, i538, i539;
    int i539, i540, i541;
    int i541, i542, i543;
    int i543, i544, i545;
    int i545, i546, i547;
    int i547, i548, i549;
    int i549, i550, i551;
    int i551, i552, i553;
    int i553, i554, i555;
    int i555, i556, i557;
    int i557, i558, i559;
    int i559, i560, i561;
    int i561, i562, i563;
    int i563, i564, i565;
    int i565, i566, i567;
    int i567, i568, i569;
    int i569, i570, i571;
    int i571, i572, i573;
    int i573, i574, i575;
    int i575, i576, i577;
    int i577, i578, i579;
    int i579, i580, i581;
    int i581, i582, i583;
    int i583, i584, i585;
    int i585, i586, i587;
    int i587, i588, i589;
    int i589, i590, i591;
    int i591, i592, i593;
    int i593, i594, i595;
    int i595, i596, i597;
    int i597, i598, i599;
    int i599, i600, i601;
    int i601, i602, i603;
    int i603, i604, i605;
    int i605, i606, i607;
    int i607, i608, i609;
    int i609, i610, i611;
    int i611, i612, i613;
    int i613, i614, i615;
    int i615, i616, i617;
    int i617, i618, i619;
    int i619, i620, i621;
    int i621, i622, i623;
    int i623, i624, i625;
    int i625, i626, i627;
    int i627, i628, i629;
    int i629, i630, i631;
    int i631, i632, i633;
    int i633, i634, i635;
    int i635, i636, i637;
    int i637, i638, i639;
    int i639, i640, i641;
    int i641, i642, i643;
    int i643, i644, i645;
    int i645, i646, i647;
    int i647, i648, i649;
    int i649, i650, i651;
    int i651, i652, i653;
    int i653, i654, i655;
    int i655, i656, i657;
    int i657, i658, i659;
    int i659, i660, i661;
    int i661, i662, i663;
    int i663, i664, i665;
    int i665, i666, i667;
    int i667, i668, i669;
    int i669, i670, i671;
    int i671, i672, i673;
    int i673, i674, i675;
    int i675, i676, i677;
    int i677, i678, i679;
    int i679, i680, i681;
    int i681, i682, i683;
    int i683, i684, i685;
    int i685, i686, i687;
    int i687, i688, i689;
    int i689, i690, i691;
    int i691, i692, i693;
    int i693, i694, i695;
    int i695, i696, i697;
    int i697, i698, i699;
    int i699, i700, i701;
    int i701, i702, i703;
    int i703, i704, i705;
    int i705, i706, i707;
    int i707, i708, i709;
    int i709, i710, i711;
    int i711, i712, i713;
    int i713, i714, i715;
    int i715, i716, i717;
    int i717, i718, i719;
    int i719, i720, i721;
    int i721, i722, i723;
    int i723, i724, i725;
    int i725, i726, i727;
    int i727, i728, i729;
    int i729, i730, i731;
    int i731, i732, i733;
    int i733, i734, i735;
    int i735, i736, i737;
    int i737, i738, i739;
    int i739, i740, i741;
    int i741, i742, i743;
    int i743, i744, i745;
    int i745, i746, i747;
    int i747, i748, i749;
    int i749, i750, i751;
    int i751, i752, i753;
    int i753, i754, i755;
    int i755, i756, i757;
    int i757, i758, i759;
    int i759, i760, i761;
    int i761, i762, i763;
    int i763, i764, i765;
    int i765, i766, i767;
    int i767, i768, i769;
    int i769, i770, i771;
    int i771, i772, i773;
    int i773, i774, i775;
    int i775, i776, i777;
    int i777, i778, i779;
    int i779, i780, i781;
    int i781, i782, i783;
    int i783, i784, i785;
    int i785, i786, i787;
    int i787, i788, i789;
    int i789, i790, i791;
    int i791, i792, i793;
    int i793, i794, i795;
    int i795, i796, i797;
    int i797, i798, i799;
    int i799, i800, i801;
    int i801, i802, i803;
    int i803, i804, i805;
    int i805, i806, i807;
    int i807, i808, i809;
    int i809, i810, i811;
    int i811, i812, i813;
    int i813, i814, i815;
    int i815, i816, i817;
    int i817, i818, i819;
    int i819, i820, i821;
    int i821, i822, i823;
    int i823, i824, i825;
    int i825, i826, i827;
    int i827, i828, i829;
    int i829, i830, i831;
    int i831, i832, i833;
    int i833, i834, i835;
    int i835, i836, i837;
    int i837, i838, i839;
    int i839, i840, i841;
    int i841, i842, i843;
    int i843, i844, i845;
    int i845, i846, i847;
    int i847, i848, i849;
    int i849, i850, i851;
    int i851, i852, i853;
    int i853, i854, i855;
    int i855, i856, i857;
    int i857, i858, i859;
    int i859, i860, i861;
    int i861, i862, i863;
    int i863, i864, i865;
    int i865, i866, i867;
    int i867, i868, i869;
    int i869, i870, i871;
    int i871, i872, i873;
    int i873, i874, i875;
    int i875, i876, i877;
    int i877, i878, i879;
    int i879, i880, i881;
    int i881, i882, i883;
    int i883, i884, i885;
    int i885, i886, i887;
    int i887, i888, i889;
    int i889, i890, i891;
    int i891, i892, i893;
    int i893, i894, i895;
    int i895, i896, i897;
    int i897, i898, i899;
    int i899, i900, i901;
    int i901, i902, i903;
    int i903, i904, i905;
    int i905, i906, i907;
    int i907, i908, i909;
    int i909, i910, i911;
    int i911, i912, i913;
    int i913, i914, i915;
    int i915, i916, i917;
    int i917, i918, i919;
    int i919, i920, i921;
    int i921, i922, i923;
    int i923, i924, i925;
    int i925, i926, i927;
    int i927, i928, i929;
    int i929, i930, i931;
    int i931, i932, i933;
    int i933, i934, i935;
    int i935, i936, i937;
    int i937, i938, i939;
    int i939, i940, i941;
    int i941, i942, i943;
    int i943, i944, i945;
    int i945, i946, i947;
    int i947, i948, i949;
    int i949, i950, i951;
    int i951, i952, i953;
    int i953, i954, i955;
    int i955, i956, i957;
    int i957, i958, i959;
    int i959, i960, i961;
    int i961, i962, i963;
    int i963, i964, i965;
    int i965, i966, i967;
    int i967, i968, i969;
    int i969, i970, i971;
    int i971, i972, i973;
    int i973, i974, i975;
    int i975, i976, i977;
    int i977, i978, i979;
    int i979, i980, i981;
    int i981, i982, i983;
    int i983, i984, i985;
    int i985, i986, i987;
    int i987, i988, i989;
    int i989, i990, i991;
    int i991, i992, i993;
    int i993, i994, i995;
    int i995, i996, i997;
    int i997, i998, i999;
    int i999, i1000, i1001;
    int i1001, i1002, i1003;
    int i1003, i1004, i1005;
    int i1005, i1006, i1007;
    int i1007, i1008, i1009;
    int i1009, i1010, i1011;
    int i1011, i1012, i1013;
    int i1013, i1014, i1015;
    int i1015, i1016, i1017;
    int i1017, i1018, i1019;
    int i1019, i1020, i1021;
    int i1021, i1022, i1023;
    int i1023, i1024, i1025;
    int i1025, i1026, i1027;
    int i1027, i1028, i1029;
    int i1029, i1030, i1031;
    int i1031, i1032, i1033;
    int i1033, i1034, i1035;
    int i1035, i1036, i1037;
    int i1037, i1038, i1039;
    int i1039, i1040, i1041;
    int i1041, i1042, i1043;
    int i1043, i1044, i1045;
    int i1045, i1046, i1047;
    int i1047, i1048, i1049;
    int i1049, i1050, i1051;
    int i1051, i1052, i1053;
    int i1053, i1054, i1055;
    int i1055, i1056, i1057;
    int i1057, i1058, i1059;
    int i1059, i1060, i1061;
    int i1061, i1062, i1063;
    int i1063, i1064, i1065;
    int i1065, i1066, i1067;
    int i1067, i1068, i1069;
    int i1069, i1070, i1071;
    int i1071, i1072, i1073;
    int i1073, i1074, i1075;
    int i1075, i1076, i1077;
    int i1077, i1078, i1079;
    int i1079, i1080, i1081;
    int i1081, i1082, i1083;
    int i1083, i1084, i1085;
    int i1085, i1086, i1087;
    int i1087, i1088, i1089;
    int i1089, i1090, i1091;
    int i1091, i1092, i1093;
    int i1093, i1094, i1095;
    int i1095, i1096, i1097;
    int i1097, i1098, i1099;
    int i1099, i1100, i1101;
    int i1101, i1102, i1103;
    int i1103, i1104, i1105;
    int i1105, i1106, i1107;
    int i1107, i1108, i1109;
    int i1109, i1110, i1111;
    int i1111, i1112, i1113;
    int i1113, i1114, i1115;
    int i1115, i1116, i1117;
    int i1117, i1118, i1119;
    int i1119, i1120, i1121;
    int i1121, i1122, i1123;
    int i1123, i1124, i1125;
    int i1125, i1126, i1127;
    int i1127, i1128, i1129;
    int i1129, i1130, i1131;
    int i1131, i1132, i1133;
    int i1133, i1134, i1135;
    int i1135, i1136, i1137;
    int i1137, i1138, i1139;
    int i1139, i1140, i1141;
    int i1141, i1142, i1143;
    int i1143, i1144, i1145;
    int i1145, i1146, i1147;
    int i1147, i1148, i1149;
    int i1149, i1150, i1151;
    int i1151, i1152, i1153;
    int i1153, i1154, i1155;
    int i1155, i1156, i1157;
    int i1157, i1158, i1159;
    int i1159, i1160, i1161;
    int i1161, i1162, i1163;
    int i1163, i1164, i1165;
    int i1165, i1166, i1167;
    int i1167, i1168, i1169;
    int i1169, i1170, i1171;
    int i1171, i1172, i1173;
    int i1173, i1174, i1175;
    int i1175, i1176, i1177;
    int i1177, i1178, i1179;
    int i1179, i1180, i1181;
    int i1181, i1182, i1183;
    int i1183, i1184, i1185;
    int i1185, i1186, i1187;
    int i1187, i1188, i1189;
    int i1189, i1190, i1191;
    int i1191, i1192, i1193;
    int i1193, i1194, i1195;
    int i1195, i1196, i1197;
    int i1197, i1198, i1199;
    int i1199, i1200, i1201;
    int i1201, i1202, i1203;
    int i1203, i1204, i1205;
    int i1205, i1206, i1207;
    int i1207, i1208, i1209;
    int i1209, i1210, i1211;
    int i1211, i1212, i1213;
    int i1213, i1214, i1215;
    int i1215, i1216, i1217;
    int i1217, i1218, i1219;
    int i1219, i1220, i1221;
    int i1221, i1222, i1223;
    int i1223, i1224, i1225;
    int i1225, i1226, i1227;
    int i1227, i1228, i1229;
    int i1229, i1230, i1231;
    int i1231, i1232, i1233;
    int i1233, i1234, i1235;
    int i1235, i1236, i1237;
    int i1237, i1238, i1239;
    int i1239, i1240, i1241;
    int i1241, i1242, i1243;
    int i1243, i1244, i1245;
    int i1245, i1246, i1247;
    int i1247, i1248, i1249;
    int i1249, i1250, i1251;
    int i1251, i1252, i1253;
    int i1253, i1254, i1255;
    int i1255, i1256, i1257;
    int i1257, i1258, i1259;
    int i1259, i1260, i1261;
    int i1261, i1262, i1263;
    int i1263, i1264, i1265;
    int i1265, i1266, i1267;
    int i1267, i1268, i1269;
    int i1269, i1270, i1271;
    int i1271, i1272, i1273;
    int i1273, i1
```

Esempi

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

7

Esempi

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

8

Esempi

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

9

Esempi

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

10

Esempi

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

11

Esempi

```
int i ;  
char c ;  
  
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') ...
```

12

```

int i ;
char c ;

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++) ...

```

13

- 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

14

Punteggiatura speciale in C

- 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/sccanf, 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

17

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* variezza di parola */
    char s[MAXPAROLA]; /* stringa contenente la parola */
    char riga[MAXIGRA];
    int i, j, lungriga;
    i=0; j=0;

    for(i=0; i<MAXPAROLA; i++)
        riga[i]=0;

    if(argc > 1)
    {
        strcpy(riga, argv[1]);
        lungriga = strlen(argv[1]);
        if(lungriga > MAXIGRA)
            lungriga = MAXIGRA;
        i=0;
        while((riga[i] != '\0') && (i < lungriga))
            s[i] = riga[i];
        s[i] = '\0';
    }
    else
        s[0] = '\0';

    printf("La parola inserita è: %s\n", s);
}

```

Il tipo char

Input/output di char

```
char ch ;
```

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

Stampa di caratteri

```

char ch ;
putchar(ch) ;

```

18

Lettura di caratteri

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

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

19

Suggerimenti (1/2)

- 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) ;

20

Suggerimenti (2/2)

- La funzione getchar è generalmente più comoda in tutti i casi
 - printf("\vuoi continuare (s/n)? ");
ch = getchar() ;

21

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**

22

Conseguenza pratica

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

Dato: _

Conseguenza pratica

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

Dato: _

getchar blocca il programma in attesa del dato

23

24

Il programma stampa l'invito ad inserire un dato

Conseguenza pratica

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

Dato: a_

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

25

Conseguenza pratica

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

Dato: a

L'utente immette Invio, il programma prosegue

26

Conseguenza pratica

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

Dato: a

_

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

27

Conseguenza pratica

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

Dato: a

_

C'era già un carattere pronto: Invio! ch2='\n'

29

Conseguenza pratica

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

Dato: a

_

Il programma **non** si blocca in attesa dell'utente

28

Consigli pratici

- Ricordare che l'utente deve sempre premere Invio, anche se il programma richiede un singolo carattere
- Ricordare che, se l'utente inserisce più di un carattere, questi verranno restituiti uno ad uno nelle getchar successive
- Ricordare che l'Invio viene letto come tutti gli altri caratteri

30

Soluzione proposta

```
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') ;
```

31

Soluzione proposta

```
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') ;
```

32



Il tipo char

Operazioni sui char

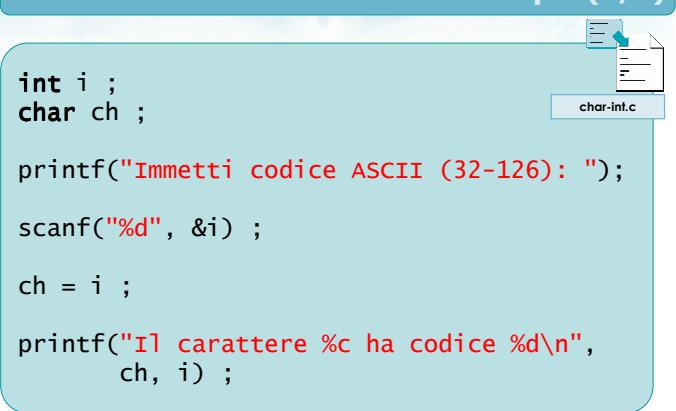
34



Conversione ASCII-Carattere

- 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' ;

35



Esempio (1/3)

```
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) ;
```

36

Esempio (2/3)

```
printf("Immetti un carattere: ") ;
ch = getchar() ;

while( getchar() != '\n' )
/**/ ;

i = ch ;

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



char-int.c

37

Esempio (3/3)

es 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
```



char-int.c

38

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') ;
```

39

Verifica se è una lettera

- ▶ 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 se è 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) ;
```

41

Valore di una cifra

- ▶ 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) ;
}
```

42

Da minuscolo a maiuscolo (1/2)

- I codici ASCII delle lettere maiuscole e delle minuscole differiscono solamente per una costante:
 - 'A' = 65 ... 'z' = 90
 - 'a' = 97 ... 'Z' = 122
- Se ch è una lettera minuscola
 - ch - 'a' è la sua posizione nell'alfabeto
 - (ch - 'a') + 'A' è la corrispondente lettera maiuscola

43

Da minuscolo a maiuscolo (2/2)

- Possiamo interpretare la conversione come una traslazione della quantità ('A' - 'a')

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

44

Confronto alfabetico

- Se due caratteri sono **entrambi maiuscoli** (o entrambi minuscoli) 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) ;
```

45

Esercizio "Quadrati di lettere"

- 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

47

```
#include <stdio.h>  
#include <string.h>  
#include <ctype.h>  
  
#define MAXPAROLA 30  
#define MAXRIGA 80  
  
int main(int argc, char *argv[]){  
    int lungMAXPAROLA; /* variezza di parola */  
    char maxParola[MAXPAROLA]; /* stringa massima */  
    char riga[MAXRIGA];  
    int i, j, k, lungRiga;  
    k=0 ;  
  
    for(i=0; i<MAXPAROLA; i++)  
        maxParola[i]=0 ;  
  
    if(argc > 1){  
        /* se c'è un argomento, lo leggono e lo inseriscono nella parola */  
        strcpy(maxParola, argv[1]);  
        if(strlen(argv[1])>MAXPAROLA){  
            printf("ERRORE, imposta un nome di file da 1 a %d caratteri", MAXPAROLA);  
            exit(1);  
        }  
    }  
  
    while(1){  
        if(k>lungRiga) k=0 ;  
        riga[k]=maxParola[k];  
        k++;  
    }  
}
```

Il tipo char

Esercizio "Quadrati di lettere"

Analisi

```
PS: Prompt dei comandi  
Quanti quadrati vuoi stampare? 4  
A  
BB  
BB  
CCC  
CCC  
CCC  
DDDD  
DDDD  
DDDD  
DDDD
```

48

Soluzione (1/2)

```
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) ;
}
```

quadrati.c

49

Soluzione (2/2)

```
for( i=0; i<N; i++ )
{
    /* stampa un quadrato
       di dimensione (i+1) */

    ch = i + 'A' ;

    for(riga=0; riga<i+1; riga++)
    {
        for(col=0; col<i+1; col++)
            putchar(ch);
        putchar('\n') ;
    }
    putchar('\n') ;
}
```

quadrati.c

50

Caratteri e stringhe

Vettori di carattere

2

```

MAXBIGA.hd
MAXBIGA.hc
MAXBIGA.hn
MAXBIGA.hcp

#define MAXBIGA_30
#define MAXBIGA_60

/* funzione di inserimento dati */
int main(int argc, char *argv[])
{
    int freq[MAXBIGA_30]; // vettore di componenti
    int i;
    FILE *f;
    char chp[MAXBIGA_30];
    int l;
    long int, longlong;
    int k;

    for(i=0; i<MAXBIGA_30; i++)
        freq[i] = 0;

    if(argc<2)
    {
        printf("uso: %s <file>\n", argv[0]);
        exit(1);
    }
    f = fopen(argv[1], "r");
    if(f==NULL)
    {
        printf("impossibile aprire il file %s\n", argv[1]);
        exit(1);
    }

    /* legge i dati da file */
    for(i=0; i<MAXBIGA_30; i++)
    {
        fscanf(f, "%s", chp);
        if(chp[0]=='.')
            freq[i] = 1;
        else
            freq[i] = 0;
    }
    /* chiude il file */
    fclose(f);
}

```

Vettori di carattere

Il tipo stringa

Vettori di caratteri

- Il tipo stringa
 - Terminatore nullo
 - Input/output di stringhe

Cori di caratteri

Original file is located at <http://www.industrydocuments.ucsf.edu/docs/lnan0001>

- Nel linguaggio C non è supportato esplicitamente alcun tipo di dato "stringa"
 - Le informazioni di tipo stringa vengono memorizzate ed elaborate ricorrendo a semplici **vettori di caratteri**

```
char saluto[10]
```

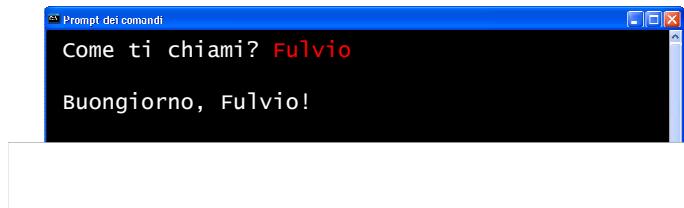
Buongiorno

4

Il logo (n. 2) è composto da un cerchio con la scritta "TOM" al centro e intorno il nome del film.

Esempi

- ▶ Si realizzi un programma in linguaggio C che acquisisca da tastiera il nome dell'utente (una stringa di max 20 caratteri), e stampi a video un saluto per l'utente stesso



Soluzione (1 / 3)

Soluzione (1/3)

```
const int MAX = 20 ;
char nome[MAX] ;
int N ;
char ch ;
int i ;

printf("Come ti chiami? ")

N = 0 ;
```

A small blue icon representing a file or document, located at the bottom right of the slide.

saluti.

6

Soluzione (2/3)

```
ch = getchar() ;
while( ch != '\n' && N<MAX )
{
    nome[N] = ch ;
    N++ ;
    ch = getchar() ;
}
```

7

Soluzione (3/3)

```
printf("Buongiorno, ") ;
for(i=0; i<N; i++)
    putchar( nome[i] ) ;
printf("!\n") ;
```

8

Commenti (1/2)

- » Qualsiasi operazione sulle stringhe si può realizzare agendo opportunamente su vettori di caratteri, gestiti con occupazione variabile
- » Così facendo, però vi sono alcuni svantaggi
 - Per ogni vettore di caratteri, occorre definire un'opportuna variabile che ne indichi la lunghezza
 - Ogni operazione, anche elementare, richiede l'uso di cicli `for/while`

9

Commenti (2/2)

- » Alcune convenzioni ci possono aiutare
 - Gestire in modo standard i vettori di caratteri usati per memorizzare stringhe
 - Apprendere le tecniche solitamente utilizzate per compiere le operazioni più frequenti
- » Molte funzioni di libreria seguono queste convenzioni
 - Conoscere le funzioni di libreria ed utilizzarle per accelerare la scrittura del programma

10

```
#include "stdlib.h"
#include "string.h"
#include "ctype.h"

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* vettore di dimensione
        lunghezza massima della parola */
    char nomeMAXRIGA[ ]; /* vettore di dimensione
        lunghezza massima della riga */
    int i, indice, lunghezza;
    FILE *f;

    if(argc < 2)
    {
        printf("Usage: %s file\n", argv[0]);
        exit(1);
    }

    f = fopen(argv[1], "r");
    if(f == NULL)
    {
        perror("Error opening file");
        exit(1);
    }

    fgets(nomeMAXRIGA, MAXRIGA, f);
    lungMAXPAROLA = strlen(nomeMAXRIGA);
    nomeMAXRIGA[lungMAXPAROLA] = '\0';
    nomeMAXRIGA[lungMAXPAROLA+1] = '\0';

    for(i=0; i<lungMAXPAROLA; i++)
    {
        if(isalpha(nomeMAXRIGA[i]))
            indice++;
        else
            break;
    }

    if(indice > lungMAXPAROLA)
    {
        printf("Error: too many characters in the word\n");
        exit(1);
    }
}
```

Vettori di caratteri

Terminatore nullo

Lunghezza di una stringa

- » Vi sono due tecniche per determinare la lunghezza di una stringa
 1. utilizzare una variabile intera che memorizzi il numero di caratteri validi

```
char nome[10] ;
int lungh_nome ;
```

6

12

Lunghezza di una stringa

- Vi sono due tecniche per determinare la lunghezza di una stringa
 - utilizzare una variabile intera che memorizzi il numero di caratteri validi

```
char nome[10] ; F u l v i o z ! $ .  
int lungh_nome ; 6
```

- utilizzare un carattere "speciale", con funzione di terminatore, dopo l'ultimo carattere valido

```
char nome[10] ; F u l v i o 0 ! $ .
```

13

Carattere terminatore

- Il carattere "terminatore" deve avere le seguenti caratteristiche
 - Fare parte della tabella dei codici ASCII
 - Deve essere rappresentabile in un char
 - Non comparire mai nelle stringhe utilizzate dal programma
 - Non deve confondersi con i caratteri "normali"
- Inoltre il vettore di caratteri deve avere una posizione libera in più, per memorizzare il terminatore stesso

14

Terminatore standard in C

- Per convenzione, in C si sceglie che tutte le stringhe siano rappresentate mediante un carattere terminatore
- Il terminatore corrisponde al carattere di codice ASCII pari a zero
 - nome[6] = 0 ;
 - nome[6] = '\0' ;

```
F u l v i o 0 ! $ .
```

15

Vantaggi

- Non è necessaria un'ulteriore variabile intera per ciascuna stringa
- L'informazione sulla lunghezza della stringa è interna al vettore stesso
- Tutte le funzioni della libreria standard C rispettano questa convenzione
 - Si aspettano che la stringa sia terminata
 - Restituiscono sempre stringhe terminate

16

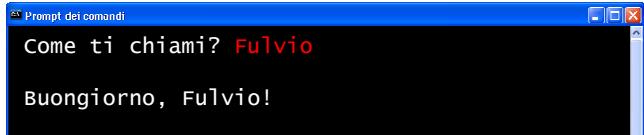
Svantaggi

- Necessario 1 byte in più
 - Per una stringa di N caratteri, serve un vettore di N+1 elementi
- Necessario ricordare di aggiungere sempre il terminatore
- Impossibile rappresentare stringhe contenenti il carattere ASCII 0

17

Esempio

- Si realizzi un programma in linguaggio C che acquisisca da tastiera il nome dell'utente (una stringa di max 20 caratteri), e stampi a video un saluto per l'utente stesso



```
Prompt dei comandi  
Come ti chiami? Fulvio  
Buongiorno, Fulvio!
```

18

Soluzione (1/3)

```
const int MAX = 20 ;
char nome[MAX+1] ;
char ch ;
int i ;

printf("Come ti chiami? ") ;

i = 0 ;
```

salutIO.c

19

Soluzione (2/3)

```
i = 0 ;

ch = getchar() ;

while( ch != '\n' && i<MAX )
{
    nome[i] = ch ;
    i++ ;
    ch = getchar() ;
}
/* aggiunge terminatore nullo */
nome[i] = '\0' ;
```

salutIO.c

20

Soluzione (3/3)

```
printf("Buongiorno, ") ;

for(i=0; nome[i]!='\0'; i++)
    putchar( nome[i] ) ;

printf("\n") ;
```

salutIO.c

21

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

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* variezza di parola */
    char nome[MAXRIGA]; /* stringa che contiene il nome */
    char singlMAXRIGA; /* singola riga */
    int i, j, lunghezza;
    FILE *f;

    f=fopen(argv[1], "r");
    if(f==NULL)
    {
        printf("ERRORE: non è possibile aprire il file %s\n", argv[1]);
        exit(1);
    }

    /* legge singlRIGA */
    singlMAXRIGA=fgets(singlMAXRIGA, MAXRIGA, f);
    if(singlMAXRIGA==NULL)
    {
        printf("ERRORE: impossibile leggere il file %s\n", argv[1]);
        exit(1);
    }

    while(fgets(nome, MAXRIGA, f)!=NULL)
```

Vettori di caratteri

Input/output di stringhe

I/O di stringhe

- Diamo per scontato di utilizzare la convenzione del terminatore nullo
- Si possono utilizzare
 - Funzioni di lettura e scrittura carattere per carattere
 - Come nell'esercizio precedente
 - Funzioni di lettura e scrittura di stringhe intere
 - scanf e printf
 - gets e puts

23

Lettura di stringhe con scanf

- Utilizzare la funzione scanf con lo specificatore di formato "%s"
- La variabile da leggere deve essere il nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
 - Non utilizzare la &
- Legge ciò che viene immesso da tastiera, fino al primo spazio o fine linea (esclusi)
 - Non adatta a leggere nomi composti (es. "Pier Paolo")

24

Esempio

```
const int MAX = 20 ;  
char nome[MAX+1] ;  
  
printf("Come ti chiami? ") ;  
scanf("%s", nome) ;
```

25

Esempio

```
const int MAX = 20 ;  
char nome[MAX+1] ;  
  
printf("Come ti chiami? ") ;  
gets(nome) ;
```

27

Esempio

```
printf("Buongiorno, ") ;  
printf("%s", nome) ;  
printf("\n") ;  
  
printf("Buongiorno, %s!\n", nome) ;
```

29

Lettura di stringhe con gets

- La funzione gets è pensata appositamente per acquisire una stringa
- Accetta un parametro, che corrisponde al nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
- Legge ciò che viene immesso da tastiera, fino al fine linea (escluso), e compresi eventuali spazi
 - Possibile leggere nomi composti (es. "Pier Paolo")

26

Scrittura di stringhe con printf

- Utilizzare la funzione printf con lo specificatore di formato "%s"
- La variabile da stampare deve essere il nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
- È possibile combinare la stringa con altre variabili nella stessa istruzione

28

Scrittura di stringhe con puts

- La funzione puts è pensata appositamente per stampare una stringa
- La variabile da stampare deve essere il nome di un vettore di caratteri
 - Non utilizzare le parentesi quadre
- Va a capo automaticamente
 - Non è possibile stampare altre informazioni sulla stessa riga

30

Esempio

```
printf("Buongiorno, ") ;  
puts(nome) ;  
/* No!! printf("\n") ; */
```

31

Conclusione

- Utilizzare sempre la convenzione del terminatore nullo
- Ricordare di allocare un elemento in più nei vettori di caratteri
- Utilizzare quando possibile le funzioni di libreria predefinite
 - In lettura, prediligere gets
 - In scrittura
 - printf è indicata per messaggi composti
 - puts è più semplice se si ha un dato per riga

32

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

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo della frequenza delle lunghezze delle parole */
    int i, indice, lunghezza;
    char s[MAXSIGA];
    char s1[MAXSIGA];
    int lun;

    if(argc != 2)
    {
        printf("Errore: l'unico argomento deve essere il nome del file (*.c)\n");
        exit(1);
    }
    else
    {
        i = fopen(argv[1], "r");
        if(i == NULL)
        {
            printf("Errore: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
        while(fgets(s, MAXSIGA, i) != NULL)
        {
            if(s[lung] == '\0')
            {
                printf("Parola %d: lunghezza = %d\n", indice + 1, lung);
                indice++;
                lung = 0;
            }
            else
                lung++;
        }
        fclose(i);
    }
}
```

Caratteri e stringhe

Operazioni elementari sulle stringhe

- ▶ Lunghezza
- ▶ Copia di stringhe
- ▶ Concatenazione di stringhe
- ▶ Confronto di stringhe
- ▶ Ricerca di sotto-stringhe
- ▶ Ricerca di parole

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

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo della frequenza delle lunghezze delle parole */
    int i, indice, lunghezza;
    char s[MAXSIGA];
    char s1[MAXSIGA];
    int lun;

    if(argc != 2)
    {
        printf("Errore: l'unico argomento deve essere il nome del file (*.c)\n");
        exit(1);
    }
    else
    {
        i = fopen(argv[1], "r");
        if(i == NULL)
        {
            printf("Errore: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
        while(fgets(s, MAXSIGA, i) != NULL)
        {
            if(s[lung] == '\0')
            {
                printf("Parola %d: lunghezza = %d\n", indice + 1, lung);
                indice++;
                lung = 0;
            }
            else
                lung++;
        }
        fclose(i);
    }
}
```

Operazioni elementari sulle stringhe

Lunghezza

Lunghezza di una stringa

- ▶ La lunghezza di una stringa si può determinare ricercando la posizione del terminatore nullo

```
char s[MAX+1] ;
int lun ;
```

s	S	a	l	v	e	Ø	3	r	w	t
0	1	2	3	4	5					

4

```
if(argc != 2)
{
    printf("Errore: l'unico argomento deve essere il nome del file (*.c)\n");
    exit(1);
}
else
{
    const int MAX = 20 ;
    char s[MAX+1] ;
    int lun ;
    int i ;

    ... /* lettura stringa */

    for( i=0 ; s[i] != 0 ; i++ )
        /* Niente */ ;

    lun = i ;
}
```

Calcolo della lunghezza

5

- ▶ Nella libreria standard C è disponibile la funzione `strlen`, che calcola la lunghezza della stringa passata come parametro
- ▶ Necessario includere `<string.h>`

```
const int MAX = 20 ;
char s[MAX+1] ;
int lun ;

... /* lettura stringa */

lun = strlen(s) ;
```

La funzione `strlen`

6

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

#define MAXPAROLA 30
#define MAXC 80

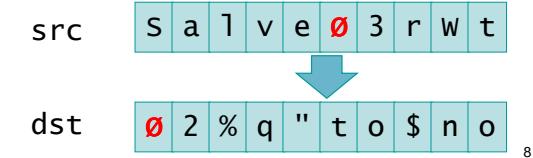
int main(int argc, char *argv[])
{
    int lungMAXPAROLA = 0; // valore di confronto
    float frequenza; // delle frequenze e delle lunghezze delle parole
    FILE *fptr; // apre il file
    int i, nrighe, lunghezza; // contatori
    fptr = fopen(argv[1], "r");
    if(fptr == NULL)
        exit(1);

    for(i=0; i<lungMAXPAROLA; i++)
        fscanf(fptr, "%c", &lunghezza);
    lunghezza = lunghezza - 1;
    if(lunghezza > lungMAXPAROLA)
        lungMAXPAROLA = lunghezza;
    else
        lungMAXPAROLA = lunghezza;

    nrighe = 0;
    while(fscanf(fptr, "%s", argv[1]) != EOF)
    {
        if(strlen(argv[1]) > lungMAXPAROLA)
            printf("Lunghezza della parola %s supera quella massima (%d).", argv[1], lungMAXPAROLA);
        else
            printf("Lunghezza della parola %s e' uguale a %d.", argv[1], strlen(argv[1]));
        nrighe++;
    }
    printf("\n");
    if(nrighe == 0)
        printf("Nessuna parola inserita nel file.");
    else
        printf("Il file contiene %d righe.", nrighe);
}
```

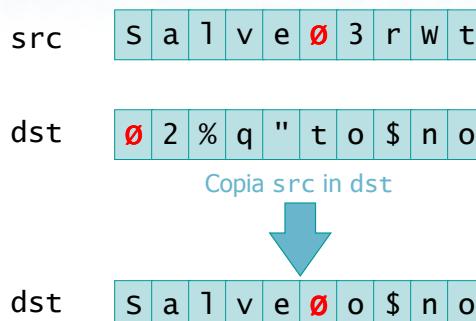
Operazioni elementari sulle stringhe

Copia di stringhe



8

Risultato della copia



```
const int MAXS = 20, MAXD = 30 ;
char src[MAXS+1] ;
char dst[MAXD+1] ;
int i ;
... /* lettura stringa src */
for( i=0 ; src[i] != 0 ; i++ )
    dst[i] = src[i] ; /* copia */
dst[i] = 0 ; /* aggiunge terminatore */
```

10

La funzione strcp

- ▶ Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strcpy` che effettua la copia di stringhe
 - Primo parametro: stringa destinazione
 - Secondo parametro: stringa sorgente

```
const int MAXS = 20, MAXD = 30
char src[MAXS+1] ;
char dst[MAXD+1] ;

... /* lettura stringa src */

strcpy(dst, src) ;
```

Avvertenze

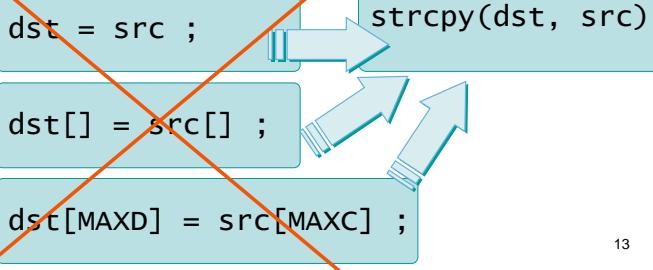
- Nella stringa destinazione vi deve essere un numero sufficiente di locazioni libere
 - `MAXD+1 >= strlen(src)+1`
 - Il contenuto precedente della stringa destinazione viene perso
 - La stringa sorgente non viene modificata
 - Il terminatore nullo
 - Deve essere aggiunto in coda a dst
 - La `strncpy` pensa già autonomamente a farlo

12



Errore frequente

- Per effettuare una copia di stringhe **non** si può assolutamente utilizzare l'operatore =
- Necessario usare `strcpy`



13

Operazioni elementari sulle stringhe

Concatenazione di stringhe

- L'operazione di concatenazione corrisponde a creare una nuova stringa composta dai caratteri di una prima stringa, **seguiti** dai caratteri di una seconda stringa



15

Semplificazione

- Per maggior semplicità, in C l'operazione di concatenazione scrive il risultato **nello stesso vettore** della prima stringa
- Il valore precedente della prima stringa viene così perso
- Per memorizzare altrove il risultato, o per non perdere la prima stringa, è possibile ricorrere a stringhe temporanee ed alla funzione `strcpy`

16

Esempio



17

Algoritmo di concatenazione

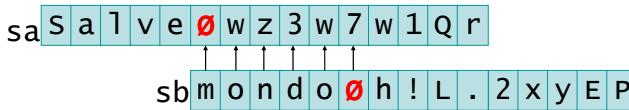
- Trova la fine della prima stringa



18

Algoritmo di concatenazione

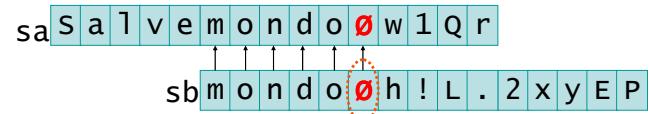
- ▶ Trova la fine della prima stringa
- ▶ Copia la seconda stringa nel vettore della prima, a partire dalla posizione del terminatore nullo (sovrascrivendolo)



19

Algoritmo di concatenazione

- ▶ Trova la fine della prima stringa
- ▶ Copia la seconda stringa nel vettore della prima, a partire dalla posizione del terminatore nullo (sovrascrivendolo)
- ▶ Termina la copia non appena trovato il terminatore della seconda stringa



20

Concatenazione

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int la ;
int i ;

... /* lettura stringhe */

la = strlen(sa) ;
for( i=0 ; sb[i] != 0 ; i++ )
    sa[la+i] = sb[i] ; /* copia */
sa[la+i] = 0 ; /* terminatore */
```

21

La funzione strcat

- ▶ Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strcat`, che effettua la concatenazione di stringhe
 - Primo parametro: prima stringa (destinazione)
 - Secondo parametro: seconda stringa

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;

... /* lettura stringhe */

strcat(sa, sb) ;
```

22

Avvertenze (1/2)

- ▶ Nella prima stringa vi deve essere un numero sufficiente di locazioni libere
 - `MAX+1 >= strlen(sa)+strlen(sb)+1`
- ▶ Il contenuto precedente della prima stringa viene perso
- ▶ La seconda stringa non viene modificata
- ▶ Il terminatore nullo
 - Deve essere aggiunto in coda alla prima stringa
 - La `strcat` pensa già autonomamente a farlo

23

Avvertenze (2/2)

- ▶ Per concatenare 3 o più stringhe, occorre farlo due a due:
 - `strcat(sa, sb);`
 - `strcat(sa, sc);`
- ▶ È possibile concatenare anche stringhe costanti
 - `strcat(sa, "!");`

24

- Il confronto di due stringhe (es.: sa e sb), mira a determinare se:

- Le due stringhe sono uguali
 - hanno uguale lunghezza e sono composte dagli stessi caratteri nello stesso ordine
- Le due stringhe sono diverse
- La stringa sa precede la stringa sb
 - secondo l'ordine lessicografico imposto dal codice ASCII
 - parzialmente compatibile con l'ordine alfabetico
- La stringa sa segue la stringa sb

26

Operazioni elementari sulle stringhe

Confronto di stringhe

Confronto di uguaglianza

- Ogni carattere di sa deve essere uguale al carattere corrispondente di sb
- Il terminatore nullo deve essere nella stessa posizione
- I caratteri successivi al terminatore vanno ignorati

```
sa [ S a 1 v e 0 o 4 d 1 0 w 1 Q r ]
sb [ S a 1 v e 0 h ! L . 2 x y E P ]
```

27

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0 ; i++ )
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

29

Flag: ricerca di universalità della condizione
sa[i]==sb[i]

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0 ; i++ )
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

28

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0 ; i++ )
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

30

Cicla fino al terminatore di
sa o di sb
(il primo che si incontra)

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

Verifica che tutti i caratteri incontrati siano uguali.
Se no, poni a 0 il flag uguali.

31

Confronto di uguaglianza

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int uguali ;
int i ;
...
uguali = 1 ;
for( i=0 ; sa[i]!=0
{
    if(sa[i]!=sb[i])
        uguali = 0 ;
}
if(sa[i]==0 || sb[i]==0)
    uguali = 0 ;
```

In questo punto sicuramente una delle due stringhe è arrivata al terminatore. Se non lo è anche l'altra, allora non sono uguali!

32

Confronto di ordine

- » Verifichiamo se sa "è minore di" sb. Partiamo con i=0
- » Se sa[i]<sb[i], allora sa è minore
- » Se sa[i]>sb[i], allora sa non è minore
- » Se sa[i]=sb[i], allora bisogna controllare i caratteri successivi (i++)
- » Il terminatore nullo conta come "minore" di tutti

sa S a 1 v e 0 o 4 d 1 0 w 1 Q r

sb S a 1 u t e 0 ! L . 2 x y E P

33

Confronto di ordine (1/2)

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int minore ;
int i ;
...
minore = 0 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0
    && minore==0; i++ )
{
    if(sa[i]<sb[i])
        minore = 1 ;
    if(sa[i]>sb[i])
        minore = -1 ;
}
```

34

Confronto di ordine (2/2)

```
if(minore==0 && sa[i]==0 && sb[i]==0)
    minore=1 ;
if(minore==1)
    printf("%s e' minore di %s\n",
           sa, sb ) ;
```

35

Commenti

```
minore = 0 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0
    && minore==0; i++ )
{
    if(sa[i]<sb[i])
        minore = 1 ;
    if(sa[i]>sb[i])
        minore = -1 ;
}

if(minore==0 && sa[i]==0 && sb[i]==0)
    minore=1 ;

if(minore==1)
    ...
```

Ricerca di esistenza della condizione sa[i]<sb[i].

36

Commenti

```
minore = 0 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0
    && minore==0; i++ )
{
    if(sa[i]<sb[i])
        minore
    if(sa[i]>sb[i])
        minore
}
if(minore==0 &&
    minore==1 ;
if(minore==1)
    ...
```

Cicla fino al primo terminatore nullo, oppure fino a che non si "scopre" chi è minore.
In altre parole, continua a ciclare solo finché le stringhe "sembrano" uguali.

37

```
minore = 0 ;
for( i=0 ; sa[i]!=0 && minore==0;
    && minore!=1; i++ )
{
    if(sa[i]<sb[i])
        minore = 1 ;
    if(sa[i]>sb[i])
        minore = -1 ;
}
if(minore==0 && sa[i]==0 && sb[i]==0)
    ...
Se flag minore==0 continua a ciclare
...
Sicuramente sa è minore di sb
Flag: minore = 1
Sicuramente sa non è minore di sb
Flag: minore = -1
```

38

Commenti

```
minore = 0 ;
for( i=0 ; sa[i]!=0 && sb[i]!=0
    && minore==0; i++ )
{
    if(sa[i]<sb[i])
        minore = 1
    if(sa[i]>sb[i])
        minore = -1
}
if(minore==0 && sa[i]==0 && sb[i]==0)
    minore=1 ;
if(minore==1)
    ...
...  
Se finora erano uguali, ma sa è più corta di sb, allora sa è minore
```

39

La funzione strcmp

- Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strcmp`, che effettua il confronto di stringhe
 - Primo parametro: prima stringa
 - Secondo parametro: seconda stringa
 - Valore restituito:
 - <0 se la prima stringa è minore della seconda
 - ==0 se le stringhe sono uguali
 - >0 se la prima stringa è maggiore della seconda

40

Confronti vari

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int ris ;
...
ris = strcmp(sa, sb) ;
if(ris<0)
    printf("%s minore di %s\n", sa, sb);
if(ris==0)
    printf("%s uguale a %s\n", sa, sb);
if(ris>0)
    printf("%s maggiore di %s\n", sa, sb);
```

Suggerimento

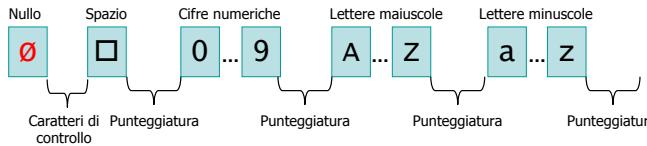
- Per ricordare il significato del valore calcolato da `strcmp`, immaginare che la funzione faccia una "sottrazione" tra le due stringhe
 - `sa - sb`
 - Negativo: sa minore
 - Positivo: sa maggiore
 - Nullo: uguali

```
const int MAX = 20 ;
char sa[MAX] ;
char sb[MAX] ;
int ris ;
...
ris = strcmp(sa, sb) ;
```

42

Ordinamento delle stringhe

- La funzione `strcmp` lavora confrontando tra loro i codici ASCII dei caratteri
- Il criterio di ordinamento è quindi dato dalla posizione dei caratteri nella tabella ASCII



43

Conseguenze

- Ogni lettera maiuscola precede ogni lettera minuscola
 - `ciao` precede `ciao`
 - `zulu` precede `apache`
- Gli spazi contano, e precedono le lettere
 - `Qui Quo Qua` precede `QuiQuoQua`
- I simboli di punteggiatura contano, ma non vi è una regola intuitiva
- L'ordinamento che si ottiene è lievemente diverso da quello "standard" alfabetico

44

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

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di punteggio
    come testo per la parola "ogni parola ha
    un significato proprio" */
    char siglaMAXSIGA;
    int lung, lungMAXSIGA;
    int i, j;
    int n;

    for(i=0; i<MAXPAROLA; i++)
        lungMAXSIGA = i;

    if(argc > 1)
    {
        lungMAXSIGA = strlen(argv[1]);
        if(lungMAXSIGA > MAXSIGA)
            lungMAXSIGA = MAXSIGA;
    }

    if(lungMAXSIGA > 0)
    {
        if(isupper(argv[1][0]))
            lungMAXSIGA = lungMAXSIGA + 1;
        else
            lungMAXSIGA = lungMAXSIGA + 2;
    }

    if(lungMAXSIGA > MAXSIGA)
        lungMAXSIGA = MAXSIGA;

    printf("lungMAXSIGA = %d\n", lungMAXSIGA);
    printf("lungMAXSIGA = %d\n", lungMAXSIGA);
}
```

Operazioni elementari sulle stringhe

Ricerca di sotto-stringhe

46

Ricerca di un carattere (1/2)

- Detti:
 - `s` una stringa arbitraria
 - `ch` un carattere qualsiasi
- Determinare se la stringa `s` contiene (una o più volte) il carattere `ch` al suo interno, in qualsiasi posizione

`s`

S	a	l	v	e	0	o	4	d	1	a	w	1	Q	r
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

`ch`

a

47

Ricerca di un carattere (2/2)

```
const int MAX = 20 ;
char s[MAX] ;
char ch ;
int trovato ;
int i ;

...
trovato = 0 ;
for( i=0 ; s[i]!=0 && trovato==0; i++ )
{
    if( s[i]==ch )
        trovato = 1 ;
}
```

48

La funzione strchr (1/2)

- » Nella libreria standard C, includendo `<string.h>`, è disponibile la funzione `strchr`, che effettua la ricerca di un carattere
 - Primo parametro: stringa in cui cercare
 - Secondo parametro: carattere da cercare
 - Valore restituito:
 - `!=NULL` se il carattere c'è
 - `==NULL` se il carattere non c'è

49

La funzione strchr (2/2)

```
const int MAX = 20 ;
char s[MAX] ;
char ch ;

...

if(strchr(s, ch)!=NULL)
    printf("%s contiene %c\n", s, ch) ;
```

50

Ricerca di una sotto-stringa

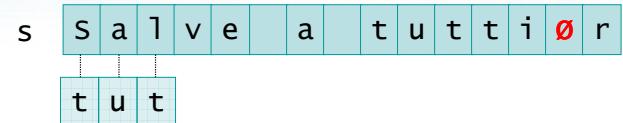
- » Detti:
 - `s` una stringa arbitraria
 - `r` una stringa da ricercare
- » Determinare se la stringa `s` contiene (una o più volte) la stringa `r` al suo interno, in qualsiasi posizione

s 

r 

51

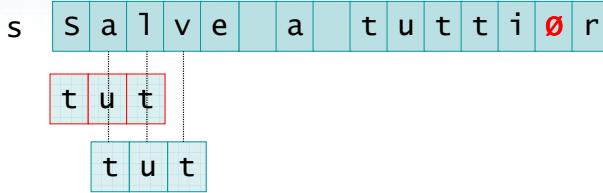
Esempio



r 

52

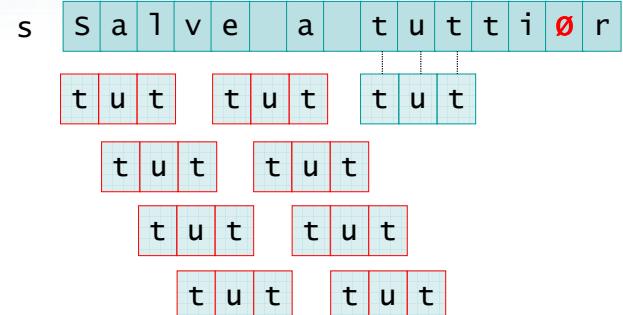
Esempio



r 

53

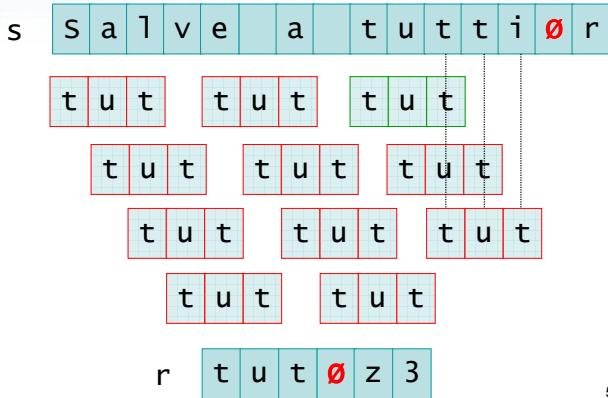
Esempio



r 

54

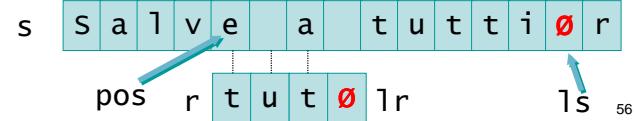
Esempio



55

Algoritmo di ricerca

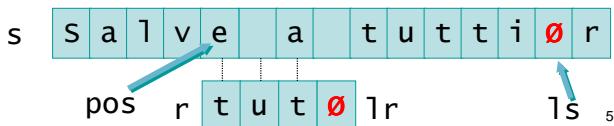
- lr = strlen(r); ls = strlen(s)
- trovato = 0
- Per ogni posizione possibile di r all'interno di s:
pos = 0...ls-lr (compresi)



56

Algoritmo di ricerca

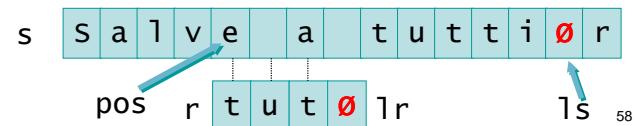
- lr = strlen(r); ls = strlen(s)
- trovato = 0
- Per ogni posizione possibile di r all'interno di s:
pos = 0...ls-lr (compresi)
 - Controlla se i caratteri di r, tra 0 e lr-1, coincidono con i caratteri di s, tra pos e pos+lr-1
 - Se sì, trovato = 1



57

Algoritmo di ricerca

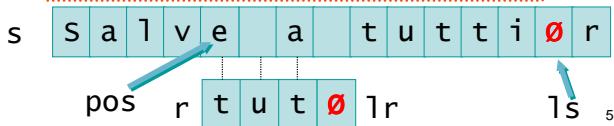
- lr = strlen(r); ls = strlen(s)
- trovato = 0
- Per ogni posizione possibile di r all'interno di s:
pos = 0...ls-lr (compresi)
 - Controlla se i caratteri di r, tra 0 e lr-1, coincidono con i caratteri di s, tra pos e pos+lr-1
 - Se sì, trovato = 1



58

Algoritmo di ricerca

- lr = strlen(r); ls = strlen(s)
- trovato = 0
- Per ogni posizione possibile di r all'interno di s:
pos = 0...ls-lr
 - Controlla se i caratteri di r, tra 0 e lr-1, coincidono con i caratteri di s, tra pos e pos+lr-1
 - Se sì, trovato = 1



59

Ricerca di una sotto-stringa (1/2)

```
const int MAX = 20 ;
char s[MAX] ;
char r[MAX] ;
int lr, ls, pos ;
int i ;
int trovato, diversi ;

...
ls = strlen(s);
lr = strlen(r);
```



60

Ricerca di una sotto-stringa (2/2)

```
trovato = 0 ;
for(pos=0; pos<=ls-1r; pos++)
{
    /* confronta r[0...lr-1] con s[pos...pos+lr-1] */
    diversi = 0 ;
    for(i=0; i<lr; i++)
        if(r[i]!=s[pos+i])
            diversi = 1 ;

    if(diversi==0)
        trovato=1 ;
}

if(trovato==1)
    printf("Trovato!\n");
```



substr.c

61

La funzione strstr (1/2)

- Nella libreria standard C, includendo <string.h>, è disponibile la funzione **strstr**, che effettua la ricerca di una sottostringa
 - Primo parametro: stringa in cui cercare
 - Secondo parametro: sotto-stringa da cercare
 - Valore restituito:
 - !=NULL se la sotto-stringa c'è
 - ==NULL se la sotto-stringa non c'è

62

La funzione strstr (2/2)

```
const int MAX = 20 ;
char s[MAX] ;
char r[MAX] ;

...
if(strstr(s, r)!=NULL)
    printf("Trovato!\n");
```

63

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

#define MAXPAROLA 30
#define MAXRIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* variezza di parola */
    const char *parola; /* parola da riconoscere nella stringa */
    char riga[MAXRIGA];
    int i, j, lungMAX;
    i=0;
}

for(;; i++) riga[i]=parola[i]; /* copia parola in riga */

/* scrive riga su schermo */
if(argc>1) {
    if(strlen(argv[1])>0) {
        printf("%s\n", argv[1]);
    }
}

/* legge riga da tastiera */
if(argc>1) {
    if(strlen(argv[1])>0) {
        fgets(riga, MAXRIGA, stdin);
    }
}

/* stampa riga */
if(strlen(riga)>0) {
    printf("%s\n", riga);
}

/* cerca parola */
while(1) {
    if(strlen(riga)>MAXRIGA-1) break;
    if(strstr(riga, parola)) {
        /* trova parola */
        printf("TROVATA parola \"%s\" all'interno della riga \"%s\"\n", parola, riga);
        /* pulisce riga */
        i=0;
        for(;; i++) riga[i]=parola[i];
    }
}
```

Operazioni elementari sulle stringhe

Ricerca di parole

Ricerca di parole

- Talvolta non interessa trovare una qualsiasi sotto-stringa, ma solamente verificare se una **parola completa** è presente in una stringa

s1	c	i	a	o	n	n	n	o	Ø	t	2	"	r
s2	o	g	g	i	n	n	n	c	'	e	'	Ø	4

r	n	o	n	Ø	z	3
---	---	---	---	---	---	---

65

Definizioni (1/2)

- Lettera:** carattere ASCII facente parte dell'alfabeto maiuscolo ('A'...'Z') o minuscolo ('a'...'z')
- Parola:** insieme consecutivo di lettere, separato da altre parole mediante spazi, numeri o simboli di punteggiatura

66

Definizioni (2/2)

➤ **Inizio di parola:** lettera, prima della quale non vi è un'altra lettera

- Non vi è un altro carattere (inizio stringa)
- Vi è un altro carattere, ma non è una lettera

➤ **Fine di parola:** lettera, dopo la quale non vi è un'altra lettera

- Non vi è un altro carattere (fine stringa)
- Vi è un altro carattere, ma non è una lettera

67

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, $s[pos]$, è un inizio di parola

- Controlla se i caratteri di r, tra 0 e $l_r - 1$, coincidono con i caratteri di s, tra pos e $pos + l_r - 1$
- Controlla se $s[pos + l_r - 1]$ è una fine di parola
- Se entrambi i controlli sono ok, allora trovato=1

68

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, $s[pos]$, è un inizio di parola

- Controlla se i caratteri di r, tra 0 e $l_r - 1$, coincidono con i caratteri di s, tra pos e $pos + l_r - 1$
- Controlla se $s[pos - 1]$ non è una lettera
- Se entrambi i `if(pos == 0 || s[pos-1] non è una lettera)`

69

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, $s[pos]$, è un inizio di parola

- Controlla se i caratteri di r, tra 0 e $l_r - 1$, coincidono con i caratteri di s, tra pos e $pos + l_r - 1$
- Controlla se $s[pos - 1]$ non è una lettera
- Se entrambi i `if(pos == 0 || s[pos-1] non è una lettera)`

```
if( pos == 0 ||  
    !( (s[pos-1]>='a' && s[pos-1]<='z') ||  
        (s[pos-1]>='A' && s[pos-1]<='Z') )  
)
```

70

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, $s[pos]$, è un inizio di parola

- Controlla se i caratteri di r, tra 0 e $l_r - 1$, coincidono con i caratteri di s, tra pos e $pos + l_r - 1$
- Controlla se $s[pos + l_r - 1]$ è una fine di parola
- Se entrambi i controlli sono ok, allora trovato=1

71

Algoritmo di ricerca

➤ Per ogni possibile posizione pos di r all'interno di s

- Se il carattere in quella posizione, $s[pos]$, è un inizio di parola

- Controlla se i caratteri di r, tra 0 e $l_r - 1$, coincidono con i caratteri di s, tra pos e $pos + l_r - 1$
- Controlla se $s[pos + l_r - 1]$ è una fine di parola
- Se entrambi i controlli sono ok, allora trovato=1

72

Ricerca di una parola (1/2)

```
trovato = 0 ;  
for(pos=0; pos<=ls-1r; pos++)  
{  
    if( pos==0 ||  
        !( s[pos-1]>='a' &&  
            s[pos-1]<='z') ||  
        (s[pos-1]>='A' &&  
            s[pos-1]<='Z') ) )  
    {  
        diversi = 0 ;  
        for(i=0; i<lr; i++)  
            if(r[i]!=s[pos+i])  
                diversi = 1 ;  
    }  
}
```

parola.c

73

Ricerca di una parola (2/2)

```
if( diversi==0 &&  
    ( pos == ls-1r ||  
        !( s[pos+1r]>='a' &&  
            s[pos+1r]<='z') ||  
        (s[pos+1r]>='A' &&  
            s[pos+1r]<='Z') ) )  
{  
    trovato=1 ;  
}  
}
```

parola.c

74

La funzione strparola

- » Nella libreria standard C non esiste alcuna funzione che svolga automaticamente la ricerca di una parola intera!!!
- » Occorre identificare, ogni volta, se il compito da svolgere è riconducibile ad una o più funzioni di libreria
- » Eventualmente si combinano tra loro più funzioni di libreria diverse
- » In alcuni casi occorre però ricorrere all'analisi carattere per carattere

75

- ▶ Introduzione
- ▶ Lunghezza di stringhe
- ▶ Classificazione di caratteri
- ▶ Trasformazione di caratteri
- ▶ Copia e concatenazione
- ▶ Confronto di stringhe
- ▶ Ricerca in stringhe
- ▶ Conversione numero-stringa

2

Caratteri e stringhe

Funzioni di libreria

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

```
#define MAXPAROLA 30
#define MAXSIGA 80
```

```
int main(int argc, char *argv[])
{
    int lung(MAXPAROLA); /* valore di controllo delle frequenze delle parole */
    char test[MAXSIGA]; /* stringa testo da analizzare */
    char sig(MAXSIGA);
    int i, lunghezza;
    FILE *fptr;
```

```
for(i=0; i<MAXPAROLA; i++)
    lung[i]=0;
for(i=0; i<MAXSIGA; i++)
    test[i]=0;
if((fptr=fopen(argv[1], "r"))==NULL)
    {
        printf("ERRORE, impossibile aprire il file %s", argv[1]);
        exit(1);
    }

```

Funzioni di libreria

Introduzione

4

```
(Merge 4/3)
```

Librerie sulle stringhe

- ▶ La libreria standard C dispone di molte funzioni predisposte per lavorare su caratteri e stringhe
- ▶ Tali funzioni si trovano prevalentemente in due librerie:
 - **<ctype.h>** funzioni operanti su caratteri
 - **<string.h>** funzioni operanti su stringhe
- ▶ Tutte le funzioni di libreria accettano e generano stringhe correttamente terminate



Suggerimenti

- ▶ Quando possibile, utilizzare sempre le funzioni di libreria
 - Sono più veloci
 - Sono maggiormente collaudate
- ▶ In ogni caso, ricordare che è sempre possibile effettuare le operazioni direttamente:
 - Sui caratteri, ricorrendo alla codifica ASCII
 - Sulle stringhe, ricorrendo alla rappresentazione come vettori di caratteri

5

```
(Merge 4/3)
```

Rappresentazione

Nome funzione	strlen
Libreria	#include <string.h>
Parametri in ingresso	s : stringa
Valore restituito	int : la lunghezza della stringa
Descrizione	Calcola la lunghezza della stringa s
Esempio	lun = strlen(s) ;

6

Convenzioni

- ▶ Assumiamo che nel seguito di questa lezione siano valide le seguenti definizioni

```
const int MAX = 20 ;
char s[MAX] ;
char s1[MAX] ;
char s2[MAX] ;
char r[MAX] ;
int lun ;
int n ;
char ch ;
float x ;
```

1

Lunghezza di stringhe

- Definite in `<string.h>`
 - Determina la lunghezza di una stringa data

strlen

Funzioni di libreria

Lunghezza di stringhe

Funzioni di libreria

Classificazione di caratteri

Funzioni di libreria

Lunghezza di stringhe

strlen	
Nome funzione	strlen
Libreria	#include <string.h>
Parametri in ingresso	s : stringa
Valore restituito	int : la lunghezza della stringa
Descrizione	Calcola la lunghezza della stringa s
Esempio	lun = strlen(s) ;

10

Classificazione di caratteri

- Definite in <ctype.h>
 - Analizzano un singolo carattere, identificandone la tipologia
 - Lettera
 - Maiuscola
 - Minuscola
 - Cifra
 - Punteggiatura

- `isalpha`
 - `isupper`
 - `islower`
 - `isdigit`
 - `isalnum`
 - `isxdigit`
 - `ispunct`
 - `isgraph`
 - `isprint`
 - `isspace`
 - `iscntrl`

12

isalpha

Nome funzione	isalpha
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una lettera maiuscola o minuscola (A...Z, a...z), "falso" altrimenti
Esempio	if(isalpha(ch)) { ... }

13

isupper

Nome funzione	isupper
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una lettera maiuscola (A...Z), "falso" altrimenti
Esempio	if(isupper(ch)) { ... }

14

islower

Nome funzione	islower
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una lettera minuscola (a...z), "falso" altrimenti
Esempio	if(islower(ch)) { ... }

15

isdigit

Nome funzione	isdigit
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una cifra numerica (0...9), "falso" altrimenti
Esempio	if(isdigit(ch)) { ... }

16

isalnum

Nome funzione	isalnum
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una lettera oppure una cifra numerica, "falso" altrimenti. Equivalenti a <code>isalpha(ch) isdigit(ch)</code>
Esempio	if(isalnum(ch)) { ... }

17

isxdigit

Nome funzione	isxdigit
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è una cifra numerica oppure una lettera valida in base 16 (a...f, A...F), "falso" altrimenti.
Esempio	if(isxdigit(ch)) { ... }

18

ispunct

Nome funzione	ispunct
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è un simbolo di punteggiatura (!"#\$%&'()*+,-./;:<=>?@[\]^_`{ }~), "falso" altrimenti.
Esempio	if(ispunct(ch)) { ... }

19

isgraph

Nome funzione	isgraph
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è un qualsiasi simbolo visibile (lettera, cifra, punteggiatura), "falso" altrimenti.
Esempio	if(isgraph(ch)) { ... }

20

isprint

Nome funzione	isprint
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è un qualsiasi simbolo visibile oppure lo spazio, "falso" altrimenti.
Esempio	if(isprint(ch)) { ... }

21

isspace

Nome funzione	isspace
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se il carattere ch è invisibile (spazio, tab, a capo), "falso" altrimenti.
Esempio	if(isspace(ch)) { ... }

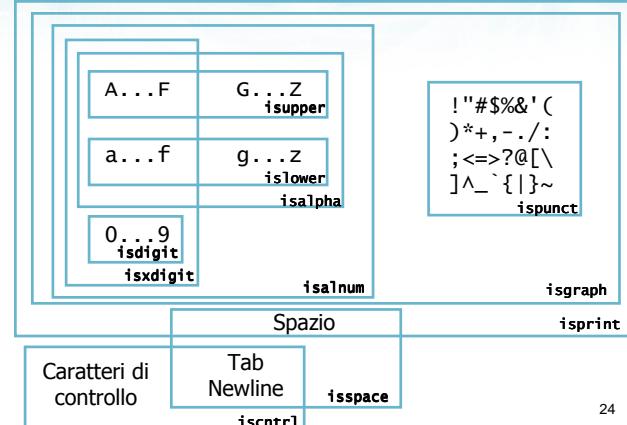
22

iscntrl

Nome funzione	iscntrl
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	Vero/falso
Descrizione	Ritorna "vero" se ch è un carattere di controllo (ASCII 0...31, 127), "falso" altrimenti.
Esempio	if(iscntrl(ch)) { ... }

23

Vista d'insieme



24

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di confronto delle frequenze delle parole */
    char parolaAXINGAL;
    int i, indice, lunghezza;
    FILE *fp;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        perror("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    printf("Aperto file: %s\n", MAXIGRA, i) == NULL;
}

```

Funzioni di libreria

Trasformazione di caratteri

Trasformazione di caratteri

- Definite in `<ctype.h>`
- Convertono tra lettere maiuscole e lettere minuscole
- toupper
- tolower

26

toupper

Nome funzione	toupper
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	char : carattere maiuscolo
Descrizione	Se ch è una lettera minuscola, ritorna l'equivalente carattere maiuscolo, se no ritorna ch stesso
Esempio	for (i=0; s[i]!=0; i++) s[i] = toupper(s[i]) ;

27

tolower

Nome funzione	tolower
Libreria	#include <ctype.h>
Parametri in ingresso	ch : carattere
Valore restituito	char : carattere maiuscolo
Descrizione	Se ch è una lettera minuscola, ritorna l'equivalente carattere maiuscolo, se no ritorna ch stesso
Esempio	for (i=0; s[i]!=0; i++) s[i] = tolower(s[i]) ;

28

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freqMAXPAROLA; /* valore di confronto delle frequenze delle parole */
    char parolaAXINGAL;
    int i, indice, lunghezza;
    FILE *fp;

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }

    fp = fopen(argv[1], "r");
    if(fp == NULL)
    {
        perror("ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    printf("Aperto file: %s\n", MAXIGRA, i) == NULL;
}

```

Funzioni di libreria

Copia e concatenazione

Copia e concatenazione

- Definite in `<string.h>`
- Trasferiscono il contenuto di una stringa in un'altra
 - Sostituendolo
 - Accodandolo
- strcpy
- strncpy
- strcat
- strncat

30

strcpy

Nome funzione	strcpy
Libreria	#include <string.h>
Parametri in ingresso	dst : stringa src : stringa
Valore restituito	nessuno utile
Descrizione	Copia il contenuto della stringa src all'interno della stringa dst (che deve avere lunghezza sufficiente).
Esempio	strcpy(s1, s2) ; strcpy(s, "") ; strcpy(s1, "ciao") ;

31

strncpy

Nome funzione	strncpy
Libreria	#include <string.h>
Parametri in ingresso	dst : stringa src : stringa n : numero max caratteri
Valore restituito	nessuno utile
Descrizione	Copia il contenuto della stringa src (massimo n caratteri) all'interno della stringa dst.
Esempio	strncpy(s1, s2, 20) ; strncpy(s1, s2, MAX) ;

32

strcat

Nome funzione	strcat
Libreria	#include <string.h>
Parametri in ingresso	dst : stringa src : stringa
Valore restituito	nessuno utile
Descrizione	Accoda il contenuto della stringa src alla fine della stringa dst (che deve avere lunghezza sufficiente).
Esempio	strcat(s1, s2) ; strcat(s1, " ") ;

33

strncat

Nome funzione	strncat
Libreria	#include <string.h>
Parametri in ingresso	dst : stringa src : stringa n : numero max caratteri
Valore restituito	nessuno utile
Descrizione	Accoda il contenuto della stringa src (massimo n caratteri) alla fine della stringa dst.
Esempio	strncat(s1, s2) ;

34

```

#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int lenMAXPAROLA; /* vettore di dimensione massima delle parole */
    char strMAXPAROLA[MAXPAROLA]; /* stringa MAXPAROLA */
    int i, indice, lunghezza;
    i = 1;
    indice = 0;
    lunghezza = 0;
    if(argc > 1)
        for(i=1; i<argc; i++)
            if(argv[i][0] == '-')
                if(argv[i][1] == 'p')
                    if(argv[i][2] == '\0')
                        lenMAXPAROLA = atoi(argv[i]+3);
                    else
                        lenMAXPAROLA = atoi(argv[i]+3)+1;
                else
                    printf("ERRORE: Impostazione parola %s non corretta\n", argv[i]);
            else
                if(argv[i][0] == 'i')
                    if(argv[i][1] == '\0')
                        MAXIGRA = atoi(argv[i]+3);
                    else
                        MAXIGRA = atoi(argv[i]+3)+1;
                else
                    printf("ERRORE: Impostazione riga %s non corretta\n", argv[i]);
    }
    printf("lunghezza MAXIGRA (%d)\n", MAXIGRA);
}

```

Funzioni di libreria

Confronto di stringhe

Confronto di stringhe

- Definite in <string.h>
- Confrontano due stringhe sulla base dell'ordine lessicografico imposto dalla tabella dei codici ASCII
- strcmp
- strncmp

36

strcmp

Nome funzione	strcmp
Libreria	#include <string.h>
Parametri in ingresso	s1 : stringa s2 : stringa
Valore restituito	int : risultato confronto
Descrizione	Risultato <0 se s1 precede s2 Risultato ==0 se s1 è uguale a s2 Risultato >0 se s1 segue s2
Esempio	if (strcmp(s, r)==0) {...} while (strcmp(r, "fine")!=0) {...}

3

Funzioni di libreria

Ricerca in stringhe

Ricerca

- Definite in `<string.h>`
 - Ricercano all'interno di una stringa data
 - Se compare un carattere
 - Se compare una sotto-stringa
 - Se compare una sequenza qualsiasi composta di caratteri dati

40

strchr

Nome funzione	strchr
Libreria	#include <string.h>
Parametri in ingresso	s : stringa ch : carattere
Valore restituito	==NULL oppure !=NULL
Descrizione	Risultato !=NULL se il carattere ch compare nella stringa. Risultato ==NULL se non compare.
Esempio	if(strchr(s, '.'))!=NULL ... if(strchr(s, ch)==NULL) ...

4

strstr

Nome funzione	strstr
Libreria	#include <string.h>
Parametri in ingresso	s : stringa r : stringa
Valore restituito	==NULL oppure !=NULL
Descrizione	Risultato !=NULL se la sotto-stringa r compare nella stringa s. Risultato ==NULL se non compare.
Esempio	if(strstr(s, "xy")!=NULL)... if(strstr(s, s1)==NULL)...

42

strspr

Nome funzione	strspn
Libreria	#include <string.h>
Parametri in ingresso	s : stringa r : stringa
Valore restituito	int : lunghezza sequenza iniziale
Descrizione	Calcola la lunghezza della parte iniziale di s che è composta esclusivamente dei caratteri presenti in r (in qualsiasi ordine).
Esempio	lun = strspn(s, " ") ; lun = strspn(s, " :,.;") ;

4

strcspn

Nome funzione	strcspn
Libreria	#include <string.h>
Parametri in ingresso	s : stringa r : stringa
Valore restituito	int : lunghezza sequenza iniziale
Descrizione	Calcola la lunghezza della parte iniziale di s che è composta esclusivamente da caratteri non presenti in r (in qualsiasi ordine).
Esempio	lun = strcspn(s, " ") ; lun = strcspn(s, " :,.;") ;

44

Funzioni di libreria

Conversione numero-stringa

atoi

ato-

Nome funzione	atoi
Libreria	#include <stdlib.h>
Parametri in ingresso	s : stringa
Valore restituito	int : valore estratto
Descrizione	Analizza la stringa s ed estraе il valore intero in essa contenuto (a partire dai primi caratteri).
Esempio	n = atoi(s) ; n = atoi("232abc") ;

4

atof

Nome funzione	atof
Libreria	#include <stdlib.h>
Parametri in ingresso	s : stringa
Valore restituito	double/float : valore estratto
Descrizione	Analizza la stringa s ed estrae il valore reale in essa contenuto (a partire dai primi caratteri).
Esempio	x = atof(s) ; x = atof("2.32abc") ;

48

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

#define MAXPAROLA 30
#define MAXSIGA 80
```

Caratteri e stringhe

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

#define MAXPAROLA 30
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /* valore di controllo delle frequenze delle parole */
    char lungMAXSIGA; /* valore di controllo delle lunghezze delle parole */
    int i, indice, lungParola;
    FILE *fp;

    fp=fopen(argv[1], "r");
    lungMAXSIGA=0;
    lungMAXPAROLA=0;
    for(i=0; i<MAXSIGA; i++)
        lungMAXSIGA+=lungMAXSIGA;
    for(i=0; i<MAXPAROLA; i++)
        lungMAXPAROLA+=lungMAXPAROLA;
    if(argc > 1)
    {
        fp=fopen(argv[1], "r");
        if(fp==NULL)
            printf("ERRORE, nonso un file valido\n");
        else
        {
            i=0;
            while(fgets(sigla, MAXSIGA, fp) != NULL)
            {
                lungParola=0;
                for(indice=0; sigla[indice] != '\n'; indice++)
                    lungParola++;
                if(lungParola > lungMAXSIGA)
                    lungMAXSIGA=lungParola;
                if(lungParola > lungMAXPAROLA)
                    lungMAXPAROLA=lungParola;
            }
        }
    }
    printf("lungMAXSIGA = %d\n", lungMAXSIGA);
    printf("lungMAXPAROLA = %d\n", lungMAXPAROLA);
    fclose(fp);
}
```

Esercizi proposti

Esercizi proposti

Esercizi proposti

- Esercizio “Parola palindroma”
- Esercizio “Iniziali maiuscole”
- Esercizio “Alfabetto farfallino”

2

Esercizio “Parola palindroma”

Esercizio “Parola palindroma”

- Sia data una parola inserita da tastiera.
- Si consideri che la parola può contenere sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al massimo 30 caratteri
- Il programma deve svolgere le seguenti operazioni:
 - Visualizzare la parola inserita
 - Aggiornare la parola in modo che tutti i caratteri siano minuscoli, e visualizzarla
 - Verificare se la parola è palindroma

4

Palindromia

- Una parola è detta **palindroma** se può essere letta indifferentemente da sinistra verso destra e da destra verso sinistra

- Esempi:

o	t	t	o
---	---	---	---

m	a	d	a	m
---	---	---	---	---

5

Analisi

- Acquisisci parola
- Stampa parola
- Converti in minuscolo
- Stampa minuscolo
- Verifica se è palindroma
- Stampa se è palindroma

6

Analisi

- » Acquisisci parola
- » Stampa parola
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

```
const int MAX = 30 ;  
char parola[MAX+1] ;  
printf("Inserisci parola: ") ;  
scanf("%s", parola) ;
```

7

8

Analisi

- » Acquisisci parola
- » Stampa parola
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

```
char minusc[MAX+1] ;  
int i ;  
  
strcpy(minusc, parola) ;  
  
for(i=0; minusc[i]!=0; i++)  
{  
    minusc[i] = tolower( minusc[i] ) ;  
}
```

- » Acquisisci parola
- » Stampa parola
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

10

Analisi

- » Acquisisci parola
- » Stampa parola
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

```
int i, j, palin, lun ;  
i = 0 ;  
j = strlen( minusc ) - 1 ;  
palin = 1 ;  
  
while( i<j && palin==1 )  
{  
    if(minusc[i] != minusc[j])  
        palin = 0 ;  
    i++ ; j-- ;  
}
```

- » Acquisisci parola
- » Stampa se è palindroma
- » Converti in minuscolo
- » Stampa minuscule
- » Verifica se è palindroma
- » Stampa se è palindroma

12

Soluzione

```
Quincy 2005
Parola palindroma
Inserisci una parola: Otto
La parola inserita e': Otto
La parola in minuscolo e': otto
La parola e' palindroma
Press Enter to return to Quincy...
```



palindroma.c

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

#define MAXPAROLA 80
#define MAXSIGA 80

int main(int argc, char *argv[])
{
    int lungMAXPAROLA; /** valore di controllo
    delle frequenze delle lunghezze delle parole */
    int lungMAXSIGA; /** lunghezza della parola più
    lunga inserita */
    int i, j, k;
    char orig[100]; /** stringa inserita dall'utente */
    char sig[100]; /** stringa inserita dall'utente */

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: inserisci un programma da eseguire\n");
        exit(1);
    }
    else
    {
        if(argv[1] == NULL)
        {
            fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
            exit(1);
        }
    }

    lungMAXSIGA = lungMAXPAROLA = 1;
    for(i = 0; orig[i] != '\0'; i++)
    {
        if(lungMAXSIGA <= lungMAXPAROLA)
        {
            lungMAXSIGA++;
        }
    }

    for(j = 0; j < lungMAXSIGA; j++)
    {
        sig[j] = '\0';
    }

    for(i = 0; orig[i] != '\0'; i++)
    {
        if(orig[i] >='A' & orig[i] <='Z')
        {
            sig[i] = orig[i];
        }
        else
        {
            sig[i] = orig[i] - 32;
        }
    }

    if(sig[lungMAXSIGA - 1] != '\0')
    {
        printf("Inserisci %s", sig);
    }
}
```

Esercizi proposti

Esercizio “Iniziali maiuscole”

Esercizio “Iniziali maiuscole” (1/2)

- Scrivere un programma che legga una frase introdotta da tastiera
 - La frase è terminata dall'introduzione del carattere di invio
 - La frase contiene sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al più 100 caratteri

13

Esercizio “Iniziali maiuscole” (2/2)

- Il programma deve svolgere le seguenti operazioni:
 - Visualizzare la frase inserita
 - Costruire una nuova frase in cui il primo carattere di ciascuna parola nella frase di partenza è stato reso maiuscolo. Tutti gli altri caratteri devono essere resi minuscoli
 - Visualizzare la nuova frase

15

16

Esempio

cHe bE1LA gIORnATA

Che Bella Giornata

Analisi

- La frase inserita può contenere degli spazi: occorrerà usare `gets` e non `scanf`
- Ogni lettera iniziale di parola va convertita in maiuscolo
- Ogni lettera non iniziale di parola va convertita in minuscolo
- Ogni altro carattere va lasciato immutato

17

18

Conversione delle lettere

```
for(i=0; frase[i]!=0; i++)
{
    if( isalpha(frase[i]) &&
        ( i==0 || !isalpha(frase[i-1]) ) )
    {
        frase[i] = toupper( frase[i] ) ;
    }
    else
    {
        frase[i] = tolower( frase[i] ) ;
    }
}
```

iniziali.c



iniziali.c

19

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

#define MAXPAROLA 30
#define MAXSIGA 60

int main(int argc, char *argv[])
{
    int lunghezzaParola; /* valore di controllo della frequenza delle lunghezze delle parole */
    int lunghezzaSiga; /* lunghezza della parola */
    int i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z;
    char frase[MAXPAROLA]; /* frase inserita dall'utente */
    char siga[MAXSIGA]; /* sigla inserita dall'utente */
    char output[MAXSIGA]; /* sigla ottenuta dalla conversione */

    if(argc != 2)
    {
        fprintf(stderr, "ERRORE: inserire un parametro sulla linea di comando che sia l'input");
        exit(1);
    }

    if((frase = fopen(argv[1], "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file %s", argv[1]);
        exit(1);
    }

    if((siga = fopen("sighe.txt", "r")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file sighe.txt");
        exit(1);
    }

    if((output = fopen("output.txt", "w")) == NULL)
    {
        fprintf(stderr, "ERRORE: impossibile aprire il file output");
        exit(1);
    }
```

Esercizi proposti

Esercizio “Alfabeto farfallino”

Esercizio “Alfabeto farfallino” (1/2)

- » Scrivere un programma che legga una frase introdotta da tastiera
- La frase è terminata dall'introduzione del carattere di invio
 - La frase contiene sia caratteri maiuscoli che caratteri minuscoli, e complessivamente al più 100 caratteri

21

Esercizio “Alfabeto farfallino” (2/2)

- » Il programma deve svolgere le seguenti operazioni:
- Visualizzare la frase inserita
 - Costruire una nuova frase nel cosiddetto “alfabeto farfallino”
 - Visualizzare la nuova frase

22

L’alfabeto farfallino

- » La traduzione nell’alfabeto farfallino di una parola segue le seguenti regole:
- Tutte le consonanti sono tradotte in modo identico
 - Ogni vocale è tradotta uguale a se stessa, seguita da altri due caratteri:
 - la lettera ‘f’ (se la vocale è minuscola) o la lettera ‘F’ (se la vocale è maiuscola)
 - una copia della vocale stessa
- » Esempi:
- a → afa
 - con → cofon

23

Vacanze di NATALE

vafacafanzefe difi NAFATAFALEFE

24

Approccio risolutivo

- » Copiamo la stringa frase in una nuova stringa farfa
- » lun=0
- » Per ogni carattere frase[i]
 - Se non è una vocale, va accodato a farfa[lun]
 - Se è una vocale, occorre accodare a farfa[lun] i 3 caratteri: frase[i], poi 'f' o 'F', poi ancora frase[i]
 - Incrementare lun (di 1 oppure 3)
- » Infine aggiungere a farfa[lun] il terminatore nullo

25

Conversione alfabeto (1/2)

```
lun = 0 ;  
for(i=0; frase[i]!=0; i++)  
{  if( isalpha(frase[i]) )  
    { /* lettera alfabetica */  
      ch = tolower(frase[i]) ;  
      if( ch=='a' || ch=='e' || ch=='i'  
          || ch=='o' || ch=='u' )  
      { /* vocale: trasforma */  
        1 farfa[lun] = frase[i] ;  
        if(isupper(frase[i]))  
          2 farfa[lun+1] = 'F' ;  
        else farfa[lun+1] = 'f' ;  
        3 farfa[lun+2] = frase[i] ;  
  
        lun = lun + 3 ;  
      }  
    }  
  }  
}
```

26

Conversione alfabeto (2/2)

```
else  
{  
  /* consonante: copia */  
  farfa[lun] = frase[i] ;  
  lun++ ;  
}  
}  
else  
{  
  /* altro carattere: copia */  
  farfa[lun] = frase[i] ;  
  lun++ ;  
}  
}  
farfa[lun] = 0 ; /* terminatore */
```

27

```
#include <stdio.h>
#include <ctype.h>
#define MAXPAROLA 30
#define MAXIGRA 80

int main(int argc, char *argv[])
{
    int freq[MAXPAROLA]; /* vettore di contatori delle frequenze delle lunghezze delle parole */
    int i, max, lunghezza; /* i: indice, lunghezza */
    FILE *fp;

    fp=fopen(argv[1], "r");
    if(fp==NULL)
    {
        fprintf(stderr, "ERRORE: non è possibile aprire il file %s", argv[1]);
        exit(1);
    }
    max=lunghezza=0;
    for(i=0; i<MAXIGRA; i++)
        freq[i]=0;
    while(fgets(argv[2], MAXIGRA, fp))
    {
        if(argv[2][0]=='.')
            break;
        lunghezza=strlen(argv[2]);
        if(lunghezza>max)
            max=lunghezza;
        freq[lunghezza]++;
    }
    fclose(fp);
    printf("Parola con la maggiore lunghezza: %s\n", argv[2]);
    printf("Lunghezza massima: %d\n", max);
    printf("Frequenze: %d\n", freq[0]);
    printf("Frequenze: %d\n", freq[1]);
    printf("Frequenze: %d\n", freq[2]);
    printf("Frequenze: %d\n", freq[3]);
    printf("Frequenze: %d\n", freq[4]);
    printf("Frequenze: %d\n", freq[5]);
    printf("Frequenze: %d\n", freq[6]);
    printf("Frequenze: %d\n", freq[7]);
    printf("Frequenze: %d\n", freq[8]);
    printf("Frequenze: %d\n", freq[9]);
    printf("Frequenze: %d\n", freq[10]);
    printf("Frequenze: %d\n", freq[11]);
    printf("Frequenze: %d\n", freq[12]);
    printf("Frequenze: %d\n", freq[13]);
    printf("Frequenze: %d\n", freq[14]);
    printf("Frequenze: %d\n", freq[15]);
    printf("Frequenze: %d\n", freq[16]);
    printf("Frequenze: %d\n", freq[17]);
    printf("Frequenze: %d\n", freq[18]);
    printf("Frequenze: %d\n", freq[19]);
    printf("Frequenze: %d\n", freq[20]);
    printf("Frequenze: %d\n", freq[21]);
    printf("Frequenze: %d\n", freq[22]);
    printf("Frequenze: %d\n", freq[23]);
    printf("Frequenze: %d\n", freq[24]);
    printf("Frequenze: %d\n", freq[25]);
    printf("Frequenze: %d\n", freq[26]);
    printf("Frequenze: %d\n", freq[27]);
    printf("Frequenze: %d\n", freq[28]);
    printf("Frequenze: %d\n", freq[29]);
}
```

Caratteri e stringhe

Sommario

(Argomento 14/21)

Argomenti trattati

- Caratteri e stringhe
- Il tipo `char`
- Vettori di `char`
- Stringhe: parole, frasi
- Operazioni fondamentali sulle stringhe
 - Classificazione
 - Ricerca
 - Copia e concatenazione
 - Conversione

2

(Argomento 14/21)

Tecniche di programmazione

- Terminatore nullo
- Librerie `<string.h>` e `<ctype.h>`
- Manipolazione di stringhe come vettori di caratteri
- Manipolazione di stringhe attraverso le funzioni di libreria
- Identificazione di parole all'interno di frasi

3

(Argomento 14/21)

Errore frequente

- L'input/output nelle stringhe è spesso problematico
- Utilizzando la funzione `scanf`, in presenza di spazi interni alla stringa rimarranno dei caratteri "non letti", che daranno fastidio alle successive `scanf`
- Quando possibile, ricorrere alla funzione `gets` per la lettura di una stringa

5

(Argomento 14/21)

Stringhe e vettori

- Molte operazioni sulle stringhe sono ricondotte ad analoghe operazioni sui vettori
- Molti problemi "astratti" su vettori numerici assumono forma più "concreta" nel caso delle stringhe
- I cicli sono solitamente governati dal controllo del terminatore di fine stringa

4

(Argomento 14/21)

Errore frequente

- Le stringhe hanno lunghezza variabile; i vettori che le contengono hanno lunghezza fissa
- È possibile, con una chiamata a `strcpy` o `strcat`, scrivere oltre la dimensione del vettore
 - Grave errore di programmazione, che può portare alla corruzione di dati in altre variabili
- Abituarsi a verificare la lunghezza prima di copiare le stringhe

```
if(strlen(a) + strlen(b) + 1 <= MAX)
    strcat(a,b) ;
else
    ERRORE!!!
```

6

Materiale aggiuntivo

» Sul CD-ROM

- Testi e soluzioni degli esercizi trattati nei lucidi
- Scheda sintetica
- Esercizi risolti
- Esercizi proposti

» Esercizi proposti da altri libri di testo