

```

1  /*
2  * -- M.G. Maffucci --
3  * 10.
4  * 10.1. E' dato un elenco di N studenti con nome, classe, voto
5  * finale.
6  *
7  * 10.2. Stampare il voto piu' basso, quello piu' alto e il nome e
8  * la classe degli studenti ai quali appartengono.
9  * (NOTA: algoritmo di ricerca del massimo e del minimo).
10 *
11 * 10.3. Calcolare la media aritmetica dei voti e verificare se e'
12 * sufficiente o insufficiente.
13 * (NOTA: calcolo aritmetico e condizione).
14 *
15 * 10.4. Stampare nome e classe degli studenti che hanno il voto
16 * finale inferiore alla media aritmetica dei voti.
17 * (NOTA: algoritmo di ricerca degli elementi in un vettore che
18 * rispondono ad un criterio dato).
19 *
20 * 10.5. Contare il numero dei promossi e dei respinti e calcolarne
21 * le rispettive percentuali.
22 * (NOTA: algoritmo di ricerca degli elementi in un vettore che
23 * rispondono ad un criterio dato e calcolo aritmetico).
24 *
25 * 10.6. Stampare l'elenco dei promossi cioe' di coloro che hanno il
26 * voto finale sufficiente.
27 * (NOTA: algoritmo di ricerca degli elementi in un vettore che
28 * rispondono ad un criterio dato; chi volesse fare le cose in modo
29 * elegante potrebbe visualizzare l'elenco in ordine alfabetico).
30 *
31 * NOTA GENERALE: visto che l'esercizio richiede molteplici operazioni
32 * e visualizzazioni, dovete necessariamente svolgerlo nel modo piu'
33 * ordinato possibile, suddividendolo in punti da affrontare uno
34 * per volta. In realta' questo esercizio e' l'unione di piu'
35 * esercizi che avete gia' svolto, l'unico problema che in realta'
36 * potreste avere e' doverli affrontare tutti in un unico esercizio,
37 * ma cio' non vuol dire che dobbiate risolverli tutti insieme ...
38 * ... divide et impera.
39 */
40 #include <stdio.h>
41 #include <stdlib.h>
42 #include <string.h>
43
44 #define NMAX 2000          //numero massimo di studenti
45 #define LMAX 20           //lunghezza massima del nome degli studenti
46
47 int main()
48 {
49     //input
50     char nomi[NMAX][LMAX]; //elenco dei nomi degli studenti
51     int classi[NMAX];      //classi; per semplicita' evito di
52                             //inserire le sezioni
53     int voti[NMAX];        //voto finale di ogni studente
54     int n;                 //numero di studenti che si vogliono
55                             //caricare
56
57     //lavoro
58     int i;                 //indice dei vettori
59     int posMin;            //posizione nel vettore voti[]
60                             //del voto minimo
61     int posMax;            //posizione nel vettore voti[]
62                             //del voto massimo
63     float pPromossi;      //contatore degli studenti promossi
64     float pRespinti;      //contatore degli studenti respinti
65
66     //output
67     float media;          //media aritmetica dei voti
68
69     /*
70     * 10.1. E' dato un elenco di N studenti con nome, classe, voto
71     * finale.
72     *
73     * 10.2. Stampare il voto piu' basso, quello piu' alto e il nome e
74     * la classe degli studenti ai quali appartengono.
75     * (NOTA: algoritmo di ricerca del massimo e del minimo).
76     *
77     * 10.3. Calcolare la media aritmetica dei voti e verificare se e'
78     * sufficiente o insufficiente.
79     * (NOTA: calcolo aritmetico e condizione).
80     *
81     * 10.4. Stampare nome e classe degli studenti che hanno il voto
82     * finale inferiore alla media aritmetica dei voti.
83     * (NOTA: algoritmo di ricerca degli elementi in un vettore che
84     * rispondono ad un criterio dato).
85     *
86     * 10.5. Contare il numero dei promossi e dei respinti e calcolarne
87     * le rispettive percentuali.
88     * (NOTA: algoritmo di ricerca degli elementi in un vettore che
89     * rispondono ad un criterio dato e calcolo aritmetico).
90     *
91     * 10.6. Stampare l'elenco dei promossi cioe' di coloro che hanno il
92     * voto finale sufficiente.
93     * (NOTA: algoritmo di ricerca degli elementi in un vettore che
94     * rispondono ad un criterio dato; chi volesse fare le cose in modo
95     * elegante potrebbe visualizzare l'elenco in ordine alfabetico).
96     *
97     * NOTA GENERALE: visto che l'esercizio richiede molteplici operazioni
98     * e visualizzazioni, dovete necessariamente svolgerlo nel modo piu'
99     * ordinato possibile, suddividendolo in punti da affrontare uno
100    * per volta. In realta' questo esercizio e' l'unione di piu'
101    * esercizi che avete gia' svolto, l'unico problema che in realta'
102    * potreste avere e' doverli affrontare tutti in un unico esercizio,
103    * ma cio' non vuol dire che dobbiate risolverli tutti insieme ...
104    * ... divide et impera.
105    */
106
107     return 0;
108 }

```

```

67     * 10.1. E' dato un elenco di N studenti con nome, classe, voto
68     * finale.
69     * Inserisco il numero di studenti da caricare,
70     * controllando la validita' dell'input.
71     */
72     do{
73         printf("Inserisci il numero di studenti da caricare (1-2000): ");
74         scanf("%d", &n);
75     } while( (n < 1) || (n > NMAX));
76
77     /*
78     * Ciclo di caricamento dei tre vettori.
79     * Il caricamento viene fatto parallelamente, in un unico ciclo.
80     */
81     for(i = 0; i < n; i++){
82         /*
83         * Il while seguente serve solo per ripulire lo standard input (stdin)
84         */
85         while(getchar() != '\n');
86         printf("Inserisci il nome dello studente: ");
87         fgets(nomi[i], LMAX, stdin);
88         nomi[i][strlen(nomi[i]) - 1] = '\0';
89         printf("Inserisci la classe frequentata dallo studente (1-5): ");
90         scanf("%d", &classi[i]);
91         printf("Inserisci il voto finale dello studente: ");
92         scanf("%d", &voti[i]);
93     }
94     /*
95     * 10.2. Stampare il voto piu' basso, quello piu' alto e il nome e
96     * la classe degli studenti ai quali appartengono.
97     * (NOTA: algoritmo di ricerca del massimo e del minimo).
98     */
99     for(i = 1, posMin = 0; i < n; i++){
100         if(voti[i] < voti[posMin]){
101             posMin = i;
102         }
103     }
104     printf("\nVoto piu' basso: %d\nElenco studenti\n", voti[posMin]);
105     printf("\tNome\t\t\t\t\t|Classe|\n");
106     for(i = 0; i < n; i++){
107         if(voti[i] == voti[posMin]){
108             /*
109             * La stringa di conversione %-20.20s indica che si visualizzera'
110             * una stringa allineata a sinistra (-) in uno spazio complessivo
111             * di 20 caratteri (20) e si considereranno solo i primi 20
112             * caratteri della stringa visualizzata (.20).
113             */
114             printf("|%-20.20s|%d\t\t\t\t\t|\n", nomi[i], classi[i]);
115         }
116     }
117     for(i = 1, posMax = 0; i < n; i++){
118         if(voti[i] > voti[posMax]){
119             posMax = i;
120         }
121     }
122     printf("\nVoto piu' alto: %d\nElenco studenti\n", voti[posMax]);
123     printf("\tNome\t\t\t\t\t|Classe|\n");
124     for(i = 0; i < n; i++){
125         if(voti[i] == voti[posMax]){
126             printf("|%-20.20s|%d\t\t\t\t\t|\n", nomi[i], classi[i]);
127         }
128     }
129     /*
130     * 10.3. Calcolare la media aritmetica dei voti e verificare se e'
131     * sufficiente o insufficiente.
132     * (NOTA: calcolo aritmetico e condizione).

```

```

133     */
134     for(i = 0, media = 0; i < n; i++){
135         media += voti[i];
136     }
137     media /= n;
138     printf("\nLa media scolastica e' %.2f e quindi ", media);
139     if(media < 6){
140         printf("insufficiente\n");
141     } else {
142         printf("sufficiente\n");
143     }
144     /*
145     * 10.4. Stampare nome e classe degli studenti che hanno il voto
146     * finale inferiore alla media aritmetica dei voti.
147     * (NOTA: algoritmo di ricerca degli elementi in un vettore che
148     * rispondono ad un criterio dato).
149     */
150     printf("\nStudenti con voto inferiore alla media scolastica %.2f\n", media);
151     printf("\tNome\t\t\t\t\t|Classe|\n");
152     for(i = 0; i < n; i++){
153         if(voti[i] < media){
154             printf("|%-20.20s|%d\t\t\t\t|\n", nomi[i], classi[i]);
155         }
156     }
157     /*
158     * 10.5. Contare il numero dei promossi e dei respinti e calcolarne
159     * le rispettive percentuali.
160     * (NOTA: algoritmo di ricerca degli elementi in un vettore che
161     * rispondono ad un criterio dato e calcolo aritmetico).
162     */
163     for(i = 0, pPromossi = 0, pRespinti = 0; i < n; i++){
164         if(voti[i] >= 6)
165             pPromossi++;
166         else
167             pRespinti++;
168     }
169     /*
170     * Nella stringa costante la sequenza %% permette di visualizzare il
171     * carattere %.
172     */
173     printf("\nPercentuale promossi: %.2f%%\n", (pPromossi*100/n));
174     printf("Percentuale respinti: %.2f%%\n", (pRespinti*100/n));
175     /*
176     * 10.6. Stampare l'elenco dei promossi cioe' di coloro che hanno il
177     * voto finale sufficiente.
178     * (NOTA: algoritmo di ricerca degli elementi in un vettore che
179     * rispondono ad un criterio dato; chi volesse fare le cose in modo
180     * elegante potrebbe visualizzare l'elenco in ordine alfabetico).
181     */
182     printf("\nStudenti promossi\n");
183     for(i = 0; i < n; i++){
184         if(voti[i] >= 6){
185             printf("%s\n", nomi[i]);
186         }
187     }
188
189     return 0;
190 }

```