

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 // struct per memorizzare ogni riga
5 struct data
6 {
7     int n;
8     float f;
9     char s[6 + 1]; // ogni stringa deve avere spazio per \0
10 };
11
12 int main()
13 {
14     // tmp struttura di supporto. vett puntatore a vettore di strutture Data
15     // i variabile di ciclo, N numero di righe lette, size capacità del vettore
16     // fp puntatore al file, prima di input e poi di output
17     struct data tmp, *vett = NULL;
18     int i, N = 0, size = 1;
19     FILE *fp;
20
21     // Apro il file di input in lettura
22     fp = fopen("data_es2_input", "r");
23     if (fp == NULL)
24         return (EXIT_FAILURE);
25     // Memorizzo la riga sulla variabile temporanea fino alla fine del file
26     while (fscanf(fp, " %d %f %6s", &(tmp.n), &(tmp.f), tmp.s) != EOF)
27     {
28         // Controllo se necessaria duplicazione della capacità vettore
29         if (size == N + 1)
30         {
31             size *= 2;
32             vett = (struct data *)realloc(vett, size * sizeof(struct data));
33         }
34         // Copio variabile temporanea nel vettore e incremento numero di righe lette
35         vett[N] = tmp;
36         N++;
37     }
38     // Chiudo il file di input e apro il file di output in scrittura
39     fclose(fp);
40     fp = fopen("data_es2_output", "w");
41     if (fp == NULL)
42         return (EXIT_FAILURE);
43     // Scrivo il vettore su disco iterando in senso inverso
44     for (i = N - 1; i >= 0; i--)
45         fprintf(fp, " %d %f %6s\n", vett[i].n, vett[i].f, vett[i].s);
46
47     // Chiudo il file di output. Libero memoria e termino
48     fclose(fp);
49     free(vett);
50     return 0;
51 }
52

```