

```

1  ///HEADER
2
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <string.h>
7  #include <stdbool.h>
8  #define NMAX 25
9  #define LENMAX 250
10
11 typedef struct Tdipendente{
12     char codFis[NMAX];
13     char nome[NMAX];
14     char cognome[NMAX];
15     char ruolo[NMAX];
16     float stip;
17 }Tdipendente;
18
19 typedef struct Tnodo{
20     Tdipendente dipendente;
21     struct Tnodo* next;
22 }Tnodo;
23
24 float leggiFloat(char* msg,int vmin,int vmax);
25 int leggiInt(char* msg,int vmin,int vmax);
26 void leggiStr(char* msg,char *s);
27 char leggiChar(char* msg);
28 Tnodo* imp(Tnodo* first);
29 bool contrDati(Tdipendente n,Tnodo* first);
30 void stampa(Tnodo* first);
31 void stampaP(Tnodo* p);
32 Tnodo* rir(Tnodo* first,char *dato);
33 void datiDip(Tnodo* first);
34 void modStip(Tnodo* first);
35 void costoStip(Tnodo* first);
36 void espRuolo(Tnodo* first);
37 Tnodo* cancel(Tnodo* first);
38 Tnodo* cancelMargStip(Tnodo* first);
39
40
41 ///FUNZIONI
42
43
44 #include "header.h"
45
46 //Davide Benedetti 4BI
47
48
49
50 /*
51 descrizione file .csv
52 il file nomeF contiene un elemento per riga, con i dati separati da ";"
53 esempio:
54 FD098;Fabio;Daini;sistemista;1900;
55
56 funzione imp
57
58 par formali
59 first                primo nodo della lista                Tnodo*
60
61 var locali
62 fin                  file                                     FILE*
63 nuovo                contiene temporaneamente i dati di un elemento Tdipendente
64 p                    nuovo nodo della lista                 Tnodo*
65 corretto             ci indica se i dati sono corretti       bool
66 riga                 contiene una riga del file              vettore di

```

```

LENMAX caratteri
67  nomeF                nome del file                                vettore di
NMAX caratteri
68  nodo                puntatore per muoversi all'interno della lista      Tnodo*
69
70  inizio
71      leggi nome del file in nomeF
72      apri il file nomeF in input
73      corretto=true
74      se il file i;% stato aperto
75      allora
76          leggi la prima riga del file in riga
77          mentre file non e finito
78              spezza riga fra i suoi token
79              assegna i token ai campi del record nuovo nel seguente ordine:
codFis,nome,cognome,ruolo;stipendio,
80              se nuovo.codFis non e univoco
81                  corretto=false
82              fse
83              se nuovo.ruolo!=("programmatore", "sistemista", "formatore")
84                  corretto=false
85              fse
86              se(corretto!=false)
87                  istanzia uno spazio per il puntatore p pari alla misura in byte di una struttura
Tdipendente
88                  p->dipendente=nuovo
89                  p->next=NULL
90                  se(first==NULL)
91                      allora
92                          first=p
93                  altrimenti
94                      nodo=first
95                      prec=NULL
96                      mentre(p->dipendente.codFis>nodo->dipendente.codFis && nodo!=NULL)
97                          prec=p
98                          nodo=nodo->next
99                  fmentre
100                  se(nodo==first)
101                      allora
102                          p->next=first
103                          first=p
104                      altrimenti
105                          prec->next=p
106                          p->next=nodo
107                  fse
108              fse
109          fse
110      fmentre
111      altrimenti
112          scrivi "file non trovato"
113      fse
114      ritorna first
115 fine
116
117 */
118 Tnodo* imp(Tnodo* first){
119     char nomeF[NMAX],riga[NMAX];
120     FILE* fin;
121     Tdipendente nuovo;
122     Tnodo* p=NULL;
123     leggiStr("inserisci nome file",nomeF);
124     fin=fopen(nomeF,"r");
125     if(fin!=NULL){
126         fgets(riga,LENMAX,fin);
127         while(!feof(fin)){
128             strcpy(nuovo.codFis,strtok(riga,";"));

```

```

129     strcpy(nuovo.nome, strtok(NULL, ";"));
130     strcpy(nuovo.cognome, strtok(NULL, ";"));
131     strcpy(nuovo.ruolo, strtok(NULL, ";"));
132     nuovo.stip=atof(strtok(NULL, ";"));
133     if(contrDati(nuovo, first)){
134         p=(Tnodo*)malloc(sizeof(Tnodo));
135         p->dipendente=nuovo;
136         p->next=NULL;
137         if(first==NULL){
138             first=p;
139         }else{
140             Tnodo* nodo=first;
141             Tnodo* prec=NULL;
142             while(nodo!=NULL && strcmp(p->dipendente.codFis, nodo->dipendente.codFis)>0){
143                 prec=nodo;
144                 nodo=nodo->next;
145             }
146             if(nodo==first){
147                 p->next=first;
148                 first=p;
149             }else{
150                 prec->next=p;
151                 p->next=nodo;
152             }
153         }
154     }
155     fgets(riga, LENMAX, fin);
156 }
157 fclose(fin);
158 }else
159     printf("file non trovato");
160 return first;
161 }
162 /*
163 funzione contrDati
164
165 par. form.
166 first          primo nodo della lista          Tnodo*
167 n              nuovo elemento da controllare
168 Tdipendente
169 var. locali
170 p              puntatore per muoversi fra i nodi della lista    Tnodo*
171
172 inizio
173     p=first
174     mentre(p!=NULL && corretto)
175         se codFis del nuovo elemento != uguale a codFis del nodo p
176         allora
177             corretto=false
178         fse
179         p=p->next
180     fmentre
181     se ruolo != "programmatore" && "sistemista" && "formatore"
182     allora
183         corretto=false
184     fse
185     ritorna corretto
186 fine
187
188 */
189 bool contrDati(Tdipendente n, Tnodo* first){
190     Tnodo* p=first;
191     bool corretto=true;
192     while(p!=NULL && corretto){
193         if(strcmp(n.codFis, p->dipendente.codFis)==0)

```

```

194         corretto=false;
195         p=p->next;
196     }
197     if(strcmp(n.ruolo,"programmatore")!=0 && strcmp(n.ruolo,"sistemista")!=0 && strcmp(n.ruolo,"formatore"
)!=0)
198         corretto=false;
199     return corretto;
200 }
201 /*
202 procedura stampa
203
204 par. form.
205 p                puntatore per muoversi fra i nodi della lista
Tnodo*
206
207 inizio
208     se i;% stato creato almeno un nodo
209     allora
210         p=first
211         mentre(p!=NULL)
212             stampa i campi del campo dipendente di p: codFis,nome,cognome,ruolo,stip
213             p=p->next
214         fmentre
215     altrimenti
216         scrivi "nessun dato presente"
217     fse
218 fine
219
220 */
221 void stampa(Tnodo* first){
222     if(first!=NULL){
223         Tnodo* p=first;
224         while(p!=NULL){
225             stampaP(p);
226             p=p->next;
227         }
228     }else
229         printf("nessun dato presente");
230
231 }
232 /*
233 procedura stampaP
234
235 par. form.
236 p                nodo da stampare
Tnodo*
237
238 inizio
239     stampa i campi del campo dipendente di p: codFis,nome,cognome,ruolo,stip
240 fine
241 */
242 void stampaP(Tnodo* p){
243     printf("%s, ",p->dipendente.codFis);
244     printf("%s, ",p->dipendente.nome);
245     printf("%s, ",p->dipendente.cognome);
246     printf("%s, ",p->dipendente.ruolo);
247     printf("%f\n",p->dipendente.stip);
248 }
249 /*
250 procedura datiDip
251
252 par. form.
253 first            primo nodo della lista
Tnodo*
254
255 var. locali
256 dato            cod. fiscale da ricercare
vettore di NMAX

```

```

caratteri
257 p                puntatore per muoversi fra i nodi della lista                Tnodo*
258 el                contiene indirizzo del nodo se trovato, altrimenti NULL        Tnodo*
259
260 inizio
261     se !% stato creato almeno un nodo
262     allora
263         p=first
264         leggi codice fiscale da ricercare in dato
265         cerca dato fra i nodi della lista e se lo trovi inserisci l'indirizzo in el
266         se(el!=NULL)
267             allora
268                 stampa i campi del campo dipendente di el
269             altrimenti
270                 scrivi "elemento non trovato"
271         fse
272     altrimenti
273         scrivi "nessun dato presente"
274     fse
275 fine
276 */
277 void datiDip(Tnodo* first){
278     if(first!=NULL){
279         Tnodo* el=NULL;
280         char dato[NMAX];
281         leggiStr("inserisci cod. fiscale da ricercare",dato);
282         el=rir(first,dato);
283         if(el!=NULL)
284             stampaP(el);
285         else
286             printf("elemento non trovato");
287     }else
288         printf("nessun dato presente");
289 }
290 /*
291 procedura modStip
292
293 par. form.
294 first                primo nodo della lista                Tnodo*
295
296 var. locali
297 dato                cod. fiscale da ricercare                vettore di NMAX
caratteri
298 p                puntatore per muoversi fra i nodi della lista                Tnodo*
299 el                contiene indirizzo del nodo se trovato, altrimenti NULL        Tnodo*
300
301 inizio
302     se !% stato creato almeno un nodo
303     allora
304         p=first
305         leggi codice fiscale da ricercare in dato
306         cerca dato fra i nodi della lista e se lo trovi inserisci l'indirizzo in el
307         se(el!=NULL)
308             allora
309                 leggi el->dipendente.stip
310             altrimenti
311                 scrivi "elemento non trovato"
312         fse
313     altrimenti
314         scrivi "nessun dato presente"
315     fse
316 fine
317 */
318 void modStip(Tnodo* first){
319     if(first!=NULL){
320         Tnodo* el=NULL;

```

```

321     char dato[NMAX];
322     leggiStr("inserisci cod. fiscale da ricercare",dato);
323     el=rir(first,dato);
324     if(el!=NULL)
325         el->dipendente.stip=leggiFloat("inserisci nuovo stipendio",0,10000);
326     else
327         printf("elemento non trovato");
328 }else
329     printf("nessun dato presente");
330 }
331
332 /*
333 procedura costoStip
334
335 par. form.
336 first          primo nodo della lista          Tnodo*
337
338 var. locali
339 p              puntatore per muoversi fra i nodi della lista      Tnodo*
340 totStip        somma di tutti gli stip          reale
341
342 inizio
343     se i½ stato creato almeno un nodo
344     allora
345         p=first
346         totStip=0
347         mentre(p!=NULL)
348             totStip=totStip+p->dipendente.stip
349             p=p->next
350         fmentre
351         stampa totStip
352     altrimenti
353         scrivi "nessun dato presente"
354     fse
355 fine
356
357
358 */
359 void costoStip(Tnodo* first){
360     if(first!=NULL){
361         Tnodo* p=first;
362         float totStip=0;
363         while(p!=NULL){
364             totStip=totStip+p->dipendente.stip;
365             p=p->next;
366         }
367         printf("%f",totStip);
368     }else
369         printf("nessun dato presente");
370 }
371
372 /*
373 procedura espRuolo
374
375 par. form.
376 first          primo nodo della lista          Tnodo*
377
378 var. locali
379 fin            file                          FILE*
380 p              nodo della lista              Tnodo*
381 dato          ruolo da esportare             vettore di NMAX caratteri
382
383 inizio
384     se(first!=NULL)
385     allora
386         p=first

```

```

387     apri il file binario file.bin in scrittura
388     leggi dato e controlla che sia=="programmatore" o "sistemista" o "formatore"
389     mentre(p!=NULL)
390         se(dato==p->dipendente.ruolo)
391             scrivi i campi del campo dipendente di p nel file fin con lunghezza massima LENMAX
392         fse
393         p=p->next
394     fmentre
395     altrimenti
396         scrivi "nessun dato trovato"
397     fse
398 fine
399 */
400
401 void espRuolo(Tnodo* first){
402     if(first!=NULL){
403         Tnodo* p=first;
404         char dato[NMAX];
405         FILE* fin=fopen("file.bin","w");
406         do{
407             leggiStr("inserisci ruolo(programmatore, sistemista, formatore)",dato);
408             }while(strcmp(dato,"programmatore")!=0 && strcmp(dato,"sistemista")!=0 && strcmp(dato,"formatore"
409 )!=0);
410             while(p!=NULL){
411                 if(strcmp(dato,p->dipendente.ruolo)==0)
412                     fwrite(&(p->dipendente),LENMAX,1,fin);
413                 p=p->next;
414             }
415             fclose(fin);
416         }else
417             printf("nessun dato da esportare trovato");
418     }
419     /*
420     funzione rir
421     par. form.
422     first      primo nodo della lista                Tnodo*
423     dato       dato da ricercare                     char*
424
425     var. locali
426     p          nodo della lista                      Tnodo*
427     el         contiene indirizzo del nodo se trovato, altrimenti NULL    Tnodo*
428
429     inizio
430     p=first
431     mentre(lista non e finita && el non trovato)
432         se(p->dipendente.codFis==dato)
433             el=p
434         fse
435         p=p->next
436     fmentre
437     ritorna el
438 fine
439
440 */
441 Tnodo* rir(Tnodo* first, char* dato){
442     Tnodo* p=first;
443     Tnodo* el=NULL;
444     while(p!=NULL && el==NULL){
445         if(strcmp(p->dipendente.codFis,dato)==0)
446             el=p;
447         p=p->next;
448     }
449     return el;
450 }
451 /*

```

```

452 procedura cancel
453
454 par. form.
455 first          primo nodo della lista          Tnodo*
456
457 var. locali
458 dato          cod. fiscale da ricercare          vettore di
NMAX caratteri
459 p             serve per ricollegare i nodi adiacenti al nodo che viene cancellato          Tnodo*
460 el            contiene indirizzo del nodo se trovato, altrimenti NULL          Tnodo*
461 prec          serve per ricollegare i nodi adiacenti al nodo che viene cancellato          Tnodo*
462
463 inizio
464     se i½ stato creato almeno un nodo
465     allora
466         p=first
467         leggi codice fiscale da ricercare in dato
468         cerca dato fra i nodi della lista e se lo trovi inserisci l'indirizzo in el
469         se(el!=NULL)
470             allora
471                 se(el==first)
472                     allora
473                         first=first->next
474                         cancella il nodo puntato da el
475                     altrimenti
476                         p=el
477                         prec=first
478                         mentre(prec->next!=el)
479                             prec=prec->next
480                         fmentre
481                         cancella il nodo puntato da el
482                         prec->next=p->next
483                     fse
484                 altrimenti
485                     scrivi "elemento non trovato"
486                 fse
487             altrimenti
488                 scrivi "nessun dato presente"
489             fse
490         ritorna first
491 fine
492 */
493 Tnodo* cancel(Tnodo* first){
494     if(first!=NULL){
495         Tnodo* el=NULL;
496         char dato[NMAX];
497         leggiStr("inserisci cod. fiscale dell'el. da cancellare",dato);
498         el=rir(first,dato);
499         if(el!=NULL){
500             if(el==first){
501                 first=first->next;
502                 free(el);
503             }else{
504                 Tnodo* p=el;
505                 Tnodo* prec=first;
506                 while(prec->next!=el)
507                     prec=prec->next;
508                 prec->next=p->next;
509                 free(el);
510             }
511         }else
512             printf("elemento non trovato");
513     }else
514         printf("nessun dato presente");
515     return first;
516 }

```



```

517
518  /*
519  procedura cancel
520
521  par. form.
522  first          primo nodo della lista          Tnodo*
523
524  var. locali
525  p              serve per muoversi fra i nodi della lista e per riagganciare i nodi          Tnodo*
526  aus            serve per eliminare un nodo inizializzandolo con l'indirizzo di esso          Tnodo*
527  prec           serve per ricollegare i nodi adiacenti al nodo che viene cancellato          Tnodo*
528  a              intervallo minore (per ricerca stipendio)          reale
529  b              intervallo maggiore (per ricerca stipendio)          reale
530  trovato        ci indica se abbiamo trovato almeno un el          bool
531
532  inizio
533      se i;% stato creato almeno un nodo
534      allora
535          p=first
536          trovato=false
537          leggi intervallo(a,b) da ricercare e controlla che a sia <b
538          se(p!=NULL)
539          allora
540              se(p->dipendente.stip si trova nell'intervallo)
541              allora
542                  aus=p
543                  se(p==first)
544                      first=first->next
545                      cancella il nodo puntato da aus
546                  altrimenti
547                      prec=first;
548                      mentre(prec->next!=p)
549                          prec=prec->next;
550                      fmentre
551                      prec->next=p->next;
552                      cancella il nodo puntato da aus
553                  fse
554                  trovato=true
555              fse
556              p=p->next
557          altrimenti
558              scrivi "nessun elemento nel margine e stato trovato"
559          fse
560      altrimenti
561          scrivi "nessun dato presente"
562      fse
563      ritorna first
564  fine
565  */
566
567  Tnodo* cancelMargStip(Tnodo* first){
568      if(first!=NULL){
569          Tnodo* p=first;
570          float a,b;
571          bool trovato=false;
572          do{
573              a=leggiFloat("inserisci intervallo a(<b e <10000)",0,10000);
574              b=leggiFloat("inserisci intervallo b(>a e <10000)",0,10000);
575          }while(a>b);
576          while(p!=NULL){
577              if(p->dipendente.stip>=a && p->dipendente.stip<=b){
578                  Tnodo* aus=p;
579                  if(p==first){
580                      first=first->next;
581                      free(aus);
582                  }else{

```

```

583         Tnodo* prec=first;
584         while(prec->next!=p)
585             prec=prec->next;
586         prec->next=p->next;
587         free(aus);
588     }
589     trovato=true;
590 }
591 p=p->next;
592 }
593 if(!trovato)
594     printf("nessun elemento nel margine e stato trovato");
595 }else
596     printf("nessun dato presente");
597 return first;
598 }
599
600
601 //MAIN
602
603 #include "header.h"
604 /*
605 Davide Benedetti 4BI
606
607
608 Si devono gestire i dipendenti di un'azienda informatica.
609 Per ogni dipendente si conosce:
610 i_c% Codice Fiscale (univoco)
611 i_c% Nome
612 i_c% Cognome
613 i_c% Ruolo ("programmatore", "sistemista", "formatore")
614 i_c% Stipendio
615 Realizzare un programma che permetta di:
616
617 1. Importazione dei dipendenti da file csv (con controllo unicita i_c% codice fiscale) in una lista (inserimento
ordinato crescente in base al codice fiscale); i dipendenti non importati saranno inseriti sul file log.csv
618 2. Stampare tutti i dati di tutti i dipendenti dell'azienda;
619 3. Fornito in input il Codice Fiscale, stampare i dati del dipendente;
620 4. Fornito in input il Codice Fiscale, modificare lo stipendio;
621 5. Stampare il costo totale che l'azienda dovra i_c% versare per il pagamento degli stipendi.
622 6. Esportare su file binario tutti i dipendenti di un determinato ruolo
623 7. Fornito in input il Codice Fiscale, cancellare il relativo dipendente dalla lista.
624 8. Cancellare dalla lista i dipendenti aventi uno stipendio compreso tra due valori inseriti in input
dall'i_c%utente.
625
626 Obblighi/Limiti:
627 - Ogni singolo punto del menu deve essere preceduto dallo pseudocodice con descrizione dei parametri, delle
variabili locali e delle strutture dei file creati.
628 - Il progetto dovra i_c% essere realizzato dividendo il codice in piu i_c% file.
629 - Per gli input si dovranno utilizzare le apposite funzioni realizzate.
630 */
631 int main()
632 {
633     int m;
634     Tnodo* first=NULL;
635     do{
636         m=leggiInt("Inserisci:\n1. Importazione dei dipendenti da file csv\n2. Stampare tutti i dati di
tutti i dipendenti dell'azienda\n3. Fornito in input il Codice Fiscale, stampare i dati del dipendente\n4.
Fornito in input il Codice Fiscale, modificare lo stipendio\n5. Stampare il costo totale che l'azienda dovra i_c%
versare per il pagamento degli stipendi\n6. Esportare su file binario tutti i dipendenti di un determinato
ruolo\n7.Fornito in input il Codice Fiscale, cancellare il relativo dipendente dalla lista\n8.Cancellare dalla
lista i dipendenti aventi uno stipendio compreso tra due valori inseriti in input dall'i_c%utente.\n9.termina 0,9);
637         switch(m){
638             case 1:
639                 first=imp(first);
640                 break;

```

```

641         case 2:
642             stampa(first);
643             break;
644         case 3:
645             datiDip(first);
646             break;
647         case 4:
648             modStip(first);
649             break;
650         case 5:
651             costoStip(first);
652             break;
653         case 6:
654             espRuolo(first);
655             break;
656         case 7:
657             first=cancEl(first);
658             break;
659         case 8:
660             first=cancMargStip(first);
661             break;
662     }
663 }while(m!=9);
664 return 0;
665 }
666
667
668 ///FUNZIONI PER INSERIMENTO
669
670 #include "header.h"
671 void leggiStr(char* msg,char *s){
672     printf("%s\n",msg);
673     do{
674         gets(s);
675         if(strcmp(s,"")==0)
676             printf("errore,reinserire");
677     }while(strcmp(s,"")==0);
678 }
679
680 int leggiInt(char* msg,int vmin,int vmax){
681     int n;
682     printf("%s\n",msg);
683     do{
684         scanf("%d",& n);
685         if(n<vmin || n>vmax)
686             printf("errore,reinserire");
687     }while(n<vmin || n>vmax);
688     fflush(stdin);
689     return n;
690 }
691
692 float leggiFloat(char* msg,int vmin,int vmax){
693     float n;
694     printf("%s\n",msg);
695     do{
696         scanf("%f",& n);
697         if(n<vmin || n>vmax)
698             printf("errore,reinserire");
699     }while(n<vmin || n>vmax);
700     return n;
701 }
702 char leggiChar(char* msg){
703     char s;
704     printf("%s\n",msg);
705     do{
706         scanf("%c",& s);

```

```
707         if(s=='\0')
708             printf("\nerrore, reinserire\n");
709     }while(s!='\0');
710     return s;
711 }
```