Fondamenti di Informatica T-1
Modulo 2

Obiettivo di questa esercitazione

- Files
- Allocazione dinamica della memoria

Esercizio 1
(file)

- Realizzare un programma che, aperto un file di testo di nome “Prova.txt” in modalità “scrittura”, provveda a leggere da input delle parole separate da spazi (stringhe di al più 63 caratteri) e le scriva nel file di testo.
- Il programma termina quando l’utente inserisce la parola “fine”. Si abbbia cura di chiudere il file prima di terminare definitivamente il programma.
- Si controlli il corretto funzionamento del programma accedendo direttamente al file di testo con un editor (ad es. Notepad).

Esercizio 1
(allocazione dinamica)

È dato un file binario, di nome “`valori.dat`”, contenente una sequenza di `int`; non è noto a priori quanti interi siano presenti nel file. I valori sono disposti in ordine casuale.

Si realizzi un programma che, letti dal file tali valori interi, li stampi a video ponendo prima i numeri pari e poi i numeri dispari

A tal scopo si definisca:

Esercizio 1
(allocazione dinamica)

1. Una funzione

```
int readLength(FILE *f, int *even, int *odd)
```

che determini quanti valori sono presenti nel file

In particolare, la funzione deve restituire il numero totale di valori presenti nel file, e tramite i parametri `even` e `odd` deve restituire il numero di valori pari e di valori dispari rispettivamente (la somma di `even` + `odd` deve ovviamente essere uguale al numero totale di valori presenti nel file)

Esercizio 1
(allocazione dinamica)

2. Un programma `main` che, aperto opportunamente il file "`valori.dat`", determini quanti valori sono presenti sul file tramite la funzione `readLength(…)`

Il programma deve allocare dinamicamente memoria sufficiente per leggere tutti i valori, e deve poi procedere a leggere i valori dal file e a disporli nel vettore allocato, prima i pari e poi i dispari.

Ad esempio, se nel file ci sono 13 valori pari e 16 valori dispari, nelle prime 13 posizioni del vettore ci dovranno essere i valori pari, e nelle seguenti 16 i valori dispari.

Si ricorda al candidato l'esistenza della procedura di libreria `void rewind(FILE *f)` che riporta la testina di lettura a inizio file. Il programma stampi infine a video tale vettore

Esercizio 1 - Soluzione

(allocazione dinamica)

```
int readLength(FILE *f, int *even, int *odd) {
   int temp;
   *even = 0;
   *odd = 0;

   while(fread(&temp, sizeof(int), 1, f) == 1) {
      if ((temp%2) == 0) (*even)++;
      else (*odd)++;
   }
   return *even + *odd;
}
```


Esercizio 1 - Soluzione

(allocazione dinamica)

```
int main()
{
    FILE *f;
    int odd, even, lung;
    int *store;
    int i=0, j=0, temp;

    if ((f = fopen("valori.dat", "rb")) == NULL) {
       printf("Error opening the file %s\n", "valori.dat");
       exit(-1);
    }
    lung=readLength(f, &even, &odd);
    rewind(f);
    store = (int *) malloc(sizeof(int) * (lung));
    ...
```

Esercizio 1 - Soluzione

(allocazione dinamica)

```
    ...
    while(fread(&temp, sizeof(int), 1, f) == 1) {
       if (temp%2 == 0) {
              store[i] = temp;
              i++;
       }
       else {
              store[even+j] = temp;
              j++;
       }
    }
    fclose(f);
    for (i=0; i<(lung); i++)
       printf("%d ", store[i]);
    return 0;
}
```

9

Esercizio 2

(allocazione dinamica)

Una società di telefonia cellulare gestisce un programma di premiazione per “utenti fedeli”. In particolare, per ogni cliente viene salvato su un file binario “**punti.dat**” il nome del cliente (al massimo 31 caratteri) e un numero intero che rappresenta i punti accumulati. Tali informazioni sono organizzate come una struttura **user**, opportunamente definita dal candidato

```
#define DIM 32

typedef struct {
   char name[DIM];
   int points;
} user;
```

Esercizio 2
(allocazione dinamica)

1) Si scriva una funzione:

int readPoints (char usersFile[], user results[], int maxDim, int minPoints)

che, ricevuto in ingresso il nome di un file **usersFile**, un array **results** di strutture **user**, la dimensione massima dell'array **maxDim**, e un limite inferiore di punti **minPoints**, copi nell'array **results** i dati dei clienti che hanno almeno i punti specificati da **minPoints**

La funzione deve restituire come risultato il numero di utenti con almeno **minPoints**; si noti che tale risultato rappresenta anche la dimensione logica dell'array **results**. Qualora il file non sia accessibile, la funzione deve restituire il valore -1

Esercizio 2
(allocazione dinamica)

2) Si scriva poi un programma **main()** che chieda all'utente il numero di clienti salvati sul file (tale numero sarà noto solo a tempo di esecuzione), e allochi dinamicamente un vettore **V** di **user** sufficientemente grande per poter contenere, nel caso peggiore, i dati di tutti gli utenti salvati in **usersFile**. Il programma dovrà poi chiedere all'utente il minimo punteggio e, utilizzando la funzione **readPoints()**, leggere da file e memorizzare in **V** i dati degli utenti che hanno almeno il punteggio minimo specificato. Il programma infine deve stampare a video il nome ed il punteggio degli utenti contenuti in **V** se e solo se il nome comincia per "Me"

Il file contiene una quantità indefinita di informazioni:
non è possibile contenerle tutte in un array di dimensione fissata a priori
→ *malloc*

Esercizio 2 - Soluzione

(allocazione dinamica)

```
int readPoints (char usersFile[], user results[], int maxDim,
   int minPoints){

   FILE * f;  int logicDim = 0;

   f = fopen(usersFile, "rb");
   if (f == NULL)  return -1;

   while(logicDim<maxDim &&
     fread( &results[logicDim], sizeof(user), 1, f)  > 0)) {
          if (results[logicDim].points >= minPoints)
               logicDim++;
   }

   fclose(f);
   return logicDim;
}
```

Esercizio 2 - Soluzione

(allocazione dinamica)

```
int main() {
   user * V; int i, maxUtenti; int logicDim, minPoints;

   printf("Inserire numero massimo di utenti da leggere: ");
   scanf("%d", &maxUtenti);
   V = (user *) malloc(sizeof(user) * maxUtenti);

   printf("Inserire punteggio minimo: ");
   scanf("%d", &minPoints);

   logicDim = readPoints("punti.dat", V, maxUtenti, minPoints);
   if (logicDim < 0) exit(-1);

   for (i=0; i<logicDim; i++)
      if ((V[i].name[0] == 'M') && (V[i].name[1] == 'e'))
         printf("L'utente %s ha %d punti.\n", V[i].name,
                                              V[i].points);
   free(V);
   return 0;
}
```

Esercizio 3
(allocazione dinamica)

Un negozio di noleggio CD registra, tramite un PC collegato al registratore di cassa, i dati relativi al noleggio dei Compact Disc. Per ogni utente che restituisce un disco, su un file di testo di nome "**RentedLog.txt**" viene scritto su ogni riga, in ordine:

- un intero **cd_code**, identificativo univoco di un cd
- una stringa, contenente il nome del cliente (al più 64 caratteri, senza spazi)
- un intero **days**, che indica la durata in giorni del noleggio

Dopo aver definito opportunamente una struttura **rent** per contenere tali informazioni, il candidato realizzi un programma che chieda all'utente il **nome di un cliente e il numero massimo di record che si vogliono ottenere**, e stampi a video la lista dei CD noleggiati dal cliente, subito seguito dalla durata media di un noleggio per tale cliente

```
#define DIM 65
typedef struct {
   int cd_code;
   char renter[DIM];
   int days;
} rent;
```


Esercizio 3
(allocazione dinamica)

1) Il candidato scriva una funzione **readRented(…)** che riceve in ingresso il nome di un file di testo, il nome di un utente, un puntatore a strutture **rent** (che punta ad un'area di memoria opportunamente allocata in precedenza) e la dimensione massima di tale area di memoria (in termini di numero di strutture di tipo **rent**). La funzione apra il file e salvi in memoria (tramite il puntatore ricevuto come parametro) i **record relativi all'utente specificato** (per controllare se un record è relativo al cliente specificato, si utilizzi la funzione **strcmp(…)**). La funzione restituisca il **numero di record effettivamente letti**, che deve risultare minore o uguale alla dimensione massima specificata. Qualora si raggiunga la dimensione massima di record letti prima di aver terminato il file, si ignorino i record rimanenti

Esercizio 3
(allocazione dinamica)

2) Il candidato realizzi poi un programma C che chieda inizialmente all'utente il nome di un cliente e il numero massimo di elementi su cui si vuole effettuare la statistica. Dopo aver allocato dinamicamente memoria sufficiente secondo le istruzioni ricevute dall'utente, il programma utilizzi la funzione **readRented(…)** per ottenere i dati relativi al determinato cliente. **Si stampi a video poi, in ordine, per ogni CD noleggiato, il nome del cliente, il codice del CD e la durata del noleggio**. Si stampi infine la durata media del noleggio

Esercizio 3 - Soluzione
(allocazione dinamica)

```
int readRented(char * fileName, char * name, rent * stat, int maxDim) {
    int dim = 0; FILE * f;

    if ( (f = fopen(fileName, "r")) == NULL ) {
           printf("Error opening the file %s\n", fileName);
           exit(-1);
    }

    while ( fscanf(f, "%d %s %d", &(stat[dim].cd_code),
        stat[dim].renter, &(stat[dim].days)) != EOF
                                        && dim<maxDim) {
            if (strcmp(stat[dim].renter, name) == 0)
                  dim = dim + 1;
    }

    fclose(f);
    return dim;
}
```

Esercizio 3 - Soluzione

(allocazione dinamica)

```
int main() {
    char userName[DIM]; int maxDim = 0; int realDim = 0;
    int i; float total = 0; rent * stat;

    printf("Inserire nome e dimensione massima: ");
    scanf("%s %d", userName, &maxDim);
    stat = (rent *) malloc(sizeof(rent) * maxDim);

    realDim = readRented("RentedLog.txt",userName,stat,maxDim);

    for (i=0; i< realDim; i++) {
            printf("User: %s, CD: %d, Rented for: %d days\n",
                stat[i].renter, stat[i].cd_code, stat[i].days);
            total = total + stat[i].days;
    }

    printf("\nAverage length of a rent: %6.2f\n\n",total/realDim);

    free(stat);
    return 0;
}
```


Esercizio 4

(allocazione dinamica)

Un negoziante tiene traccia del prezzo degli articoli in vendita e dell'elenco degli articoli già venduti in due file di testo distinti. In particolare,

- il file **listino.txt** specifica in ogni riga, separati tra loro da uno spazio, la tipologia di articolo in vendita (al più dieci caratteri senza spazi), la sua marca (al più 10 caratteri senza spazi) e il suo prezzo in euro (float).
- Il file **venduti.txt** elenca gli articoli già venduti, con una riga con tipologia e marca per ogni articolo venduto.

listino.txt
acqua fiuggi 7.0
acqua recoaro 6.0
pasta barilla 0.3
pasta dececco 0.5

venduti.txt
acqua recoaro
acqua recoaro
pasta barilla
pasta barilla
acqua recoaro
pasta dececco

Esercizio 4

(allocazione dinamica)

- Ovviamente possono esserci più occorrenze di uno stesso articolo in **venduti.txt** e
- non è detto che ogni articolo presente in listino.txt sia presente anche in **venduti.txt**
- invece se un articolo è presente in **venduti.txt** allora è sicuramente presente anche in **listino.txt**

Dopo aver realizzato una struttura dati item in cui sia possibile specificare la tipologia di un articolo, la sua marca, il prezzo in euro e la quantità di articoli già venduti tramite un intero, il candidato realizzi una funzione:

```
item* articoli(FILE* listino,FILE* venduti,
        char* tipologia, int* len)
```


Esercizio 4

(allocazione dinamica)

Tale funzione:

- ricevuti in ingresso due puntatori a file ed una tipologia di articolo (ad esempio **`pasta`**)
- legga il file **`listino`** per calcolare quanti articoli sono presenti del tipo **`tipologia`** e sfrutti tale valore per allocare dinamicamente memoria sufficiente per contenere tutti gli articoli di quel tipo presenti nel file **listino.txt**
- per ogni marca in vendita della tipologia di articolo richiesta, la funzione inserisca nello spazio di memoria allocata dinamicamente un **`item`**
  - Per ogni **`item`** specificare, oltre a tipologia, marca e prezzo, anche il numero di articoli venduti (ovvero il numero di occorrenze in **`venduti`**).

11

Esercizio 4
(allocazione dinamica)

La funzione **`articoli(...)`** restituisca alla funzione chiamante un puntatore all'area di memoria che contiene gli **`item`** e il numero di elementi restituiti tramite **`len`**

Si ricorda l'esistenza della funzione **`void rewind(*FILE)`** che riporta la testina di lettura a inizio file e della funzione **`int strcmp(char* st, char * ct)`** per il confronto tra stringhe

Infine si scriva un main di esempio dove viene invocata la funzione articoli(…)

Esercizio 4 - Soluzione
(allocazione dinamica)

```
typedef struct{
   char tipo[11];
   char marca[11];
   float prezzo;
   int totaleVenduti;
}item;

item* articoli(FILE* listino, FILE* venduti, char* tipologia, int* len){
   char tipo[11], marca[11];
   item *res,*resTemp, temp;
   *len=0;

   while(fscanf(listino,"%s %s %f\n", temp.tipo, temp.marca,
                                          (temp.prezzo))!=EOF){
         if(strcmp(tipologia,temp.tipo)==0)
                  (*len)=(*len)+1;
   res=(item*)malloc( sizeof(item)*(*len) );
   resTemp=res;
   ...
```

Esercizio 4 - Soluzione

(allocazione dinamica)

```
    ...
    rewind(listino);
    while(fscanf(listino,"%s %s %f\n", temp.tipo, temp.marca,
                                        &(temp.prezzo))!=EOF){
                if(strcmp(tipologia,temp.tipo)==0){
                        temp.totaleVenduti=0;
                        while(fscanf(venduti,"%s %s\n", tipo, marca)!=EOF){
                                if(strcmp(tipologia,tipo)==0){
                                        if(strcmp(temp.marca,marca)==0){
                                                (temp.totaleVenduti)++;
                                        }
                                }
                        }
                        rewind(venduti);
                        (*resTemp)=temp;
                        resTemp++;
                }
    }
    return res;
}
```

25

Esercizio 5

(allocazione dinamica)

Gestione degli esami di uno studente

- Realizzare un programma che permetta di gestire gli esami di uno studente
- Funzionalità richieste
  - Caricamento degli esami sia da file di testo che da file binario
    - Si assuma che la prima riga (il primo campo) del file da cui leggere gli esami contenga **IL NUMERO DI ESAMI** presenti nel file
  - Stampa degli esami
  - Calcolo della media pesata sul numero di crediti
  - Salvataggio su file di testo dell'elenco degli esami la cui dicitura contiene una stringa data, unitamente alla media calcolata solo su questi esami

26

Esercizio 5
(allocazione dinamica)

- Un esame è caratterizzato da
  - Dicitura, contenente anche degli spazi (ma, sempre, solo ed esattamente 35 caratteri)
  - Numero di crediti (intero)
  - Voto (intero)
- *Ancora non gestiamo liste…*
- Quindi modelliamo l'insieme degli esami come un vettore (di strutture opportune) la cui dimensione NON È NOTA A PRIORI
  - Suggerimento: utilizzare una struttura definita in termini di
    - Dimensione LOGICA (e FISICA) dell'array
    - Un array di strutture esami

Esercizio 5
(allocazione dinamica)

- Salvataggio su file dell'elenco degli esami la cui dicitura CONTIENE una stringa data, unitamente alla media calcolata solo su questi esami
  - È un tipico caso di filtro (sulla dicitura dell'esame)
    - Ho un vettore di esami, ottengo un nuovo vettore di esami
    - Poi lo salvo ricorrendo ad un'ALTRA funzione
- Come realizzo il filtro?
  - Riguardare **`string.h`**
- Come posso conoscere la dimensione del vettore filtrato?
  - Suggerimento: scandire l'elenco di partenza due volte
    - La prima per calcolare la dimensione del secondo elenco
    - La seconda per riempirlo

14

Esercizio 5
(allocazione dinamica)

Funzionalità da realizzare:

- `Boolean leggiEsamiTxt(char *nomeFile, VettoreEsami* vett);`
- `Boolean leggiEsamiBin(char *nomeFile, VettoreEsami* vett);`
- `void stampaEsami(VettoreEsami vett);`
- `float media(VettoreEsami vett);`
- `VettoreEsami filtra(VettoreEsami vett, char *pattern);`
- `Boolean salvaReport(VettoreEsami vett, char* nomeFile);`

Esercizio 5 - Soluzione
(allocazione dinamica)

Tipi di dato utilizzati

```
typedef struct
{
   char dicitura[36];
   int crediti;
   int voto;
} Esame;

typedef struct
{
   int dim;
   Esame* esami;
} VettoreEsami;
```

35 caratteri fissi + terminatore

15

Esercizio 5 - Soluzione

(allocazione dinamica)

```
boolean leggiEsamiTxt(char *nomeFile, VettoreEsami* vett)
{
  FILE *fp;
  int i;
  if((fp = fopen(nomeFile, "r")) == NULL)
  {
      perror("Errore di accesso al file: ");
      return false;
  }
  fscanf(fp,"%d",&vett->dim);
  vett->esami =
      (Esame*) malloc(vett->dim * sizeof(Esame));
```

Lettura del numero di esami (che corrisponde alla dimensione del vettore)

Allocazione dello spazio necessario per contenere dim esami

31

Esercizio 5 - Soluzione

(allocazione dinamica)

```
  ...

  for(i = 0; i < vett->dim; i++)
  {
      fgetc(fp);
      fscanf(fp, "%35c%d%d",
                 vett->esami[i].dicitura,
                 &vett->esami[i].crediti,
                 &vett->esami[i].voto);
       vett->esami[i].dicitura[35]= '\0';
  }
  fclose(fp);
  return true;
}
```

Lettura del newline (necessaria perché la stringa viene letta a caratteri)

Poiché la stringa viene letta a caratteri, il terminatore va aggiunto esplicitamente!

32

Esercizio 5 - Soluzione

(allocazione dinamica)

```
boolean leggiEsamiBin(char *nomeFile, VettoreEsami *vett)
{
   FILE *fp; int i;
   if((fp = fopen(nomeFile, "rb")) == NULL)
   {
       perror("Errore di accesso al file: ");  return false;
   }
   fread(&vett->dim, sizeof(int), 1, fp);
   vett->esami = (Esame*) malloc(vett->dim * sizeof(Esame));
   for(i = 0; i < vett->dim; i++)
   {
       fread(&vett->esami[i], sizeof(Esame), 1, fp);
       vett->esami[i].dicitura[35]= '\0';
       //per sicurezza! Quando sarebbe invece necessario?
   }
   fclose(fp);
   return true;
}
```

33

Esercizio 5 - Soluzione

(allocazione dinamica)

```
void stampaEsame(Esame esame)
{
   printf("%s (%d): %d\n",
              esame.dicitura, esame.crediti, esame.voto);
}

void stampaEsami(VettoreEsami vett)
{
   int i;
   for(i = 0; i < vett.dim; i++)
       stampaEsame(vett.esami[i]);
}
```

34

Esercizio 5 - Soluzione

(allocazione dinamica)

```
float media(VettoreEsami vett)
{
   int num = 0, den = 0, i;
   for(i = 0; i < vett.dim; i++)
   {
       num = num + vett.esami[i].crediti * vett.esami[i].voto;
       den = den + vett.esami[i].crediti;
   }
   return ((float) num) / den;
}

boolean matches(char* str, char* pattern)
{
   return (strstr(str, pattern) != NULL);
}
```

35

Esercizio 5 - Soluzione

(allocazione dinamica)

```
VettoreEsami filtra(VettoreEsami vett, char* pattern)
{
   int i, j = 0, dimFiltro = 0;
   VettoreEsami filtro;
   for(i = 0; i < vett.dim; i++)
   {
       if( matches(vett.esami[i].dicitura, pattern) )
             dimFiltro++;
   }
   filtro.dim = dimFiltro;
   …
```

36

Esercizio 5 - Soluzione

(allocazione dinamica)

```
    …
    filtro.esami =
        (Esame*) malloc(dimFiltro * sizeof(Esame));
    for(i = 0; i < vett.dim; i++)
    {
        if(matches(vett.esami[i].dicitura, pattern))
        {
              filtro.esami[j] = vett.esami[i];
              j++;
        }
    }
    return filtro;
 }
```


Esercizio 5 - Soluzione

(allocazione dinamica)

```
void salvaEsame(FILE *fp, Esame *esame)
{
   fprintf(fp, "%s (%d): %d\n",
              esame->dicitura, esame->crediti, esame->voto);
}

boolean salvaReport(VettoreEsami vett, char* nomeFile)
{
   int i;
   FILE *fp = fopen(nomeFile, "w");
   if(fp == NULL)
   {
       perror("errore durante il salvataggio: ");
       return false;
   }
…
```

Esercizio 5 - Soluzione
(allocazione dinamica)

```
   …
   for(i = 0; i < vett.dim; i++)
       salvaEsame(fp, &vett.esami[i]);
   fprintf(fp, "MEDIA: %f", media(vett) );
   fclose(fp);
   return true;
}
```

Filtraggio & Salvataggio (esempio):

```
VettoreEsami filtro = filtra(vett, "L-A");
salvaReport(filtro, "report.txt");
free(filtro); //DEALLOCAZIONE!
```


Esercizio 5 - Soluzione
(allocazione dinamica)

```
boolean serialize(VettoreEsami vett, char* nomeFile)
{
   FILE *fp = fopen(nomeFile, "wb");

   if(fp == NULL)
   {
       perror("errore durante il salvataggio: ");
       return false;
   }
   fwrite(&vett.dim, sizeof(int), 1, fp);
   fwrite(vett.esami, sizeof(Esame), vett.dim, fp);
   fclose(fp);
   return true;
}
```


20

Esercizio 6

(allocazione dinamica)

Gestione articoli in vendita

Realizzare un programma che permetta di gestire gli articoli in vendita con le seguenti funzionalità:

- Caricamento del prezzo e della quantità degli articoli già venduti da due file di testo **`listino.txt`** e **`venduti.txt`**
  - Ciascuna riga di **`listino.txt`** specifica, separati tra loro da uno spazio, la tipologia di articolo in vendita (al più dieci caratteri senza spazi), la sua marca (al più 10 caratteri senza spazi) e il suo prezzo in euro (**`float`**)
  - Ciascuna riga di **`venduti.txt`** specifica, separati tra loro da uno spazio, la tipologia e la marca di ciascun articolo venduto
- Stampa dell'elenco degli articoli già venduti suddivisi per marca e tipo con prezzo unitario e quantità totale
- Salvataggio su file binario dell'elenco precedente
- Calcolo dell'incasso ottenuto suddiviso per **marca**

41

Esercizio 6

(allocazione dinamica)

- Ovviamente possono esserci più occorrenze di uno stesso articolo in **`venduti.txt`**
- Non è detto che ogni articolo presente in **`listino.txt`** sia presente anche in **`venduti.txt`**
- Invece se un articolo è presente in **`venduti.txt`** allora è sicuramente presente anche in **`listino.txt`**

`listino.txt`

```
acqua fiuggi 7.0
acqua recoaro 6.0
pasta barilla 0.3
pasta dececco 0.5
acqua dececco 0.2
pasta fiuggi 0.1
```

`venduti.txt`

```
acqua recoaro
acqua recoaro
pasta barilla
pasta barilla
acqua recoaro
pasta dececco
pasta fiuggi
pasta fiuggi
acqua dececco
```

42

Esercizio 6
(allocazione dinamica)

- Realizzare ***una struttura dati*** ***`item`*** in cui sia possibile specificare la tipologia di un articolo, la sua marca, il prezzo in euro e la quantità di articoli già venduti
- Realizzare ***una struttura dati*** ***`income`*** in cui sia possibile specificare la marca di un articolo e l'incasso relativo a tale marca
- Si ricorda l'esistenza della funzione **`void rewind(*FILE)`** che riporta la testina di lettura a inizio file e della funzione **`int strcmp(char* st, char * ct)`** per il confronto tra stringhe

43

Esercizio 6
(allocazione dinamica)

Funzionalità da realizzare:

- **`item* articoli(FILE* listino, FILE* venduti, int* len);`**
- **`Boolean scriviArticoli(char* nomeFileBinario, item* vett, int len);`**
- **`income* calcolaIncasso(item* vett, int lenVett, int* lenIncasso);`**

44

22

Esercizio 6 - Soluzione
(allocazione dinamica)

- Tipi di dato utilizzati

```
typedef struct
{
   char tipo[11];
   char marca[11];
   float prezzo;
   int totaleVenduti;
} item;

typedef struct
{
   char marca[11];
   float euro;
} income;
```

10 caratteri fissi + terminatore

Esercizio 6 - Soluzione
(allocazione dinamica)

```
item* articoli(FILE* listino, FILE* venduti, int* len)
{
    char tipo_temp[11], marca_temp[11];
    item *res, *resTemp, temp;

    *len = 0;

    while(fscanf(listino,"%s %s %f\n", temp.tipo,
                         temp.marca, &(temp.prezzo))==3)
        (*len)++;
    res = (item*) malloc( sizeof(item)*(*len) );
    resTemp = res;
    rewind(listino);
```

Lettura del numero di oggetti nel listino prezzi (che corrisponde alla dimensione del vettore)

*Allocazione dello spazio necessario per contenere *len item*

23

Esercizio 6 - Soluzione

(allocazione dinamica)

```
while( fscanf( listino,"%s %s %f\n",
            temp.tipo, temp.marca, &(temp.prezzo))==3)
    {
      temp.totaleVenduti=0;
      while( fscanf(venduti, "%s %s\n",
                          tipo_temp, marca_temp) == 2 )
        if( strcmp(temp.tipo, tipo_temp) ==0 )
            if( strcmp(temp.marca, marca_temp) ==0 )
                (temp.totaleVenduti)++;
        rewind(venduti);
        (*resTemp) = temp;
        resTemp++;
    }
    return res;
}
```

Iterazione su tutti gli oggetti presenti nel file listino

Iterazione su tutti gli oggetti presenti nel file venduti

Stessa marca e stessa tipologia, quindi aumento di uno il contatore degli oggetti venduti

Esercizio 6 - Soluzione

(allocazione dinamica)

```
Boolean scriviArticoli( char* nomeFileBinario,
                                    item* vett, int len)
{
    int i;
    FILE*bin;

    bin = fopen(nomeFileBinario,"wb");

    if(bin == NULL)
        return false;
    for(i=0; i<len; i++)
        fwrite(vett+i, sizeof(item), 1, bin);
    close(bin);
    return true;
}
```

ricordarsi di chiudere il file

Esercizio 6 - Soluzione

(allocazione dinamica)

```
income* calcolaIncasso(item vett[], int lenVett,
                                    int* lenIncasso)
{
    income *res;  int i, j, trovato;
    *lenIncasso = 0;
    res = (income*)malloc(sizeof(income)*lenVett);

    for(i=0; i<lenVett; i++)
    {
        trovato=0;
        for(j=0; j<*lenIncasso && trovato==0 ;j++)
           if( strcmp((res+j)->marca, vett[i].marca)==0)
               trovato=1;
```

*Allocazione dello spazio necessario per contenere *lenVett income*

Iterazione sul vettore res per cercare un elemento di marca vett[i].marca

Il vettore res contiene già un elemento con marca uguale a quella in esame

49

Esercizio 6 - Soluzione

(allocazione dinamica)

```
    if(trovato == 0)
    {
      strcpy( (res+(*lenIncasso))->marca, vett[i].marca );
      (res+(*lenIncasso))->euro =
                    (vett[i].prezzo*vett[i].totaleVenduti);
      (*lenIncasso)++;
    }
    else
      (res+j-1)->euro +=
                    (vett[i].prezzo*vett[i].totaleVenduti);
    }
    return res;
}
```

Prima volta in cui viene esaminato un oggetto di marca vett[i].marca

Un oggetto di marca vett[i].marca è già stato esaminato

50

25