APPELLO SCRITTO DI PROGRAMMAZIONE 1
CORSO DI LAUREA IN MATEMATICA
UNIVERSITÀ DEGLI STUDI DI MILANO

2013–2014

1.VII.2014

VINCENZO MARRA

INDICE

> **Avvertenza.** Il tema d'esame richiede lo sviluppo di un unico programma. I vari passi che portano alla soluzione sono suddivisi in esercizi, ciascuno dei quali richiede lo sviluppo di una o più funzioni che saranno poi integrate nel programma finale. Si raccomanda di leggere interamente il tema d'esame prima di cominciare a scrivere le soluzioni dei singoli esercizi, in modo da avere chiaro l'obiettivo finale.

ESERCIZIO 1

Leggere e scrivere transazioni bancarie da e su file.

Il file `operazioni.txt` allegato al tema d'esame contiene un elenco di operazioni bancarie, una per riga, registrate nel formato:

```
3     -2.50     POS
```

dove il primo valore (`3`) è il numero identificativo dell'operazione, il secondo (`-2.50`) è l'importo dell'operazione e il terzo (`POS`) è la descrizione dell'operazione. Importi negativi indicano uscite; importi positivi indicano entrate. Si veda l'esempio in Fig. 1.

Ultima revisione: 1 luglio 2014.

```
3   -2.50   POS
2   -100    Bancomat
1   -1250   Bonifico
4   -1300   Assegno
5   1200    Assegno
6   200.5   Bonifico
```

FIGURA 1. Un esempio di file contenente 6 operazioni bancarie.

Per memorizzare le operazioni bancarie lette da un file, si definisca tramite `typedef` il tipo di dato `Oper`, i cui valori rappresentano singole transazioni bancarie. Il tipo `Oper` è definito come una struttura di tre campi: `num`, un intero che memorizza il numero identificativo dell'operazione; `imp`, un `double` che rappresenta l'importo dell'operazione; e `desc`, un array di 256 caratteri che contiene il tipo dell'operazione (`POS`, `Bancomat`, ecc.).

Si scriva poi un programma `main` che accetti in ingresso dalla riga di comando due argomenti. Il primo argomento è il nome del file di testo da cui leggere l'elenco delle operazioni bancarie: si assuma che esso sia formattato come l'esempio in Fig. 1. Il secondo argomento è il nome di file sul quale scrivere nuovamente le stesse operazioni lette dal primo file, come dettagliato qui di seguito.

Il programma prima di tutto verifica il numero di argomenti sulla riga di comando: se ve ne sono meno di due esso termina con un messaggio di errore appropriato. Altrimenti il programma tenta di aprire in lettura il file passato come primo argomento dalla riga di comando. Se l'apertura non va a buon fine il programma termina con un messaggio d'errore. Se invece l'apertura va a buon fine, il programma legge le operazioni dal file, una riga alla volta, e le memorizza in un array di operazioni (cioè, un array i cui elementi abbiano tipo `Oper`) dimensionato in modo da poter contenere 1000 elementi. Si richiede che questo array, di nome per esempio `listop` per "lista delle operazioni", sia dichiarato localmente nella funzione `main`, e non globalmente all'intero file sorgente. Nel leggere le operazioni da file il programma tiene traccia con una variabile contatore del numero totale di operazioni, `tot`. (*Suggerimento.* Per leggere i dati dal file usate `fscanf`: sapete esattamente in che formato è composto il file, quindi... .)

Completata la lettura da file e quindi l'assegnazione dei valori appropriati a `listop`, il programma tenta di aprire in scrittura il file passato come secondo argomento dalla riga di comando. Se l'apertura non va a buon fine il programma termina con un messaggio d'errore. In caso contrario, il programma invoca una funzione di prototipo

```
void scriviop(FILE *f, Oper *lop, int tot)
```

che riceve come argomenti un puntatore a file `f`, un puntatore a un array di operazioni `lop`, e un intero `tot` che indica il numero totale di operazioni su cui la funzione agirà, e che scrive ciascuna delle `tot` operazioni contenute nell'array, una per riga, sul file passato in argomento. Fatto ciò, la funzione restituisce il controllo al `main`. La funzione assume che il file `f` sia già correttamente aperto in scrittura. Le operazioni sono scritte sul file seguendo il formato suggerito dalla Fig. 2. (*Suggerimento.* Per scrivere sul file usate `fprintf`.)

```
Num. 3  Imp.  -2.500    Desc. POS
Num. 2  Imp. -100.000   Desc. Bancomat
Num. 1  Imp. -1250.000  Desc. Bonifico
Num. 4  Imp. -1300.000  Desc. Assegno
Num. 5  Imp. 1200.000   Desc. Assegno
Num. 6  Imp. 200.500    Desc. Bonifico
```

FIGURA 2. Un esempio di file prodotto dal programma `main` dell'Esercizio 1.

Dopo aver scritto le operazioni sul file, il programma visualizza a video l'intero file che ha appena scritto, una riga alla volta, e termina. (*Suggerimenti.* Usate `fgets` per leggere una riga alla volta dal file. Ricordate che il file in questione era stato aperto in scrittura, mentre adesso dovrete leggere dal file. Quindi...)

Si raccomanda di chiudere i file aperti (in lettura o in scrittura) nei punti appropriati del codice sorgente.

ESERCIZIO 2

Calcolare il saldo delle operazioni bancarie.

Scrivete una funzione

```
double saldo(Oper *lop, int tot)
```

che accetti in ingresso una puntatore a un array di operazioni `lop`, e un intero `tot` che indica il numero totale di operazioni su cui la funzione agirà, e che restituisca il saldo delle `tot` operazioni contenute nell'array. Se il puntatore in argomento è `NULL` si assuma che il saldo valga 0.

Si estenda poi la funzione `main` scritta nell'Esercizio 1 in modo che essa scriva sul file in uscita, oltre alla lista delle operazioni, anche il saldo delle operazioni.

ESERCIZIO 3

Calcolare le uscite e le entrate totali delle operazioni bancarie.

Scrivete una funzione

```
double uscite(Oper *lop, int tot)
```

che accetti in ingresso una puntatore a un array di operazioni `lop`, e un intero `tot` che indica il numero totale di operazioni su cui la funzione agirà, e che restituisca l'uscita totale comportata dalle `tot` operazioni. Se il puntatore in argomento è `NULL` si assuma che l'uscita valga 0. Si scriva inoltre l'analoga funzione `entrate` che computa però l'entrata totale comportata dalle `tot` operazioni.

Si estenda poi la funzione `main` scritta nell'Esercizio 2 in modo che essa scriva sul file in uscita, oltre alla lista delle operazioni e al loro saldo, anche l'uscita e l'entrata totale.

Esercizio 4

Estrarre le operazioni di massima uscita e massima entrata.

Scrivete una funzione

```
Oper *maxuscita(Oper *lop, int tot)
```

che accetti in ingresso una puntatore a un array di operazioni `lop`, e un intero `tot` che indica il numero totale di operazioni su cui la funzione agirà, e che restituisca un puntatore all'operazione che, fra le `tot` operazioni in questione, comporta la massima uscita.[1] Se il puntatore in argomento è `NULL` la funzione restituisce `NULL`. Se non è presente alcuna uscita, ma solo entrate, la funzione restituisce `NULL`. Si scriva inoltre l'analoga funzione `maxentrata` che restituisce però un puntatore all'operazione che comporta la massima entrata.

Si estenda poi la funzione `main` scritta nell'Esercizio 3 in modo che essa scriva sul file in uscita, oltre alla lista delle operazioni, al loro saldo, e all'uscita e all'entrata totale, anche le due operazioni di massima uscita e massima entrata. Nel caso in cui l'operazione di massima uscita (o di massima entrata) non esista, il programma scrive sul file la frase "`(Non e' presente alcuna uscita)`" (o "`(Non e' presente alcuna entrata)`").

La Fig. 3 riporta un esempio di esecuzione del programma completo eseguito sul file `operazioni.txt` allegato al tema d'esame.

Istruzioni per la consegna

- Funzioni da consegnare.

  `main()`
  `scriviop()`
  `saldo()`
  `uscite()`
  `entrate()`
  `maxuscita()`
  `maxentrata()`
  Tutte le eventuali altre funzioni ausiliarie
  che abbiate scritto.

- Sito per la consegna:

  `https://upload.mat.unimi.it`

- Autenticarsi con le proprie credenziali di posta elettronica d'ateneo.
- Consegnare solo il codice sorgente (file `*.c`), non l'eseguibile.

(V. Marra) Dipartimento di Matematica *Federigo Enriques*, Università degli Studi di Milano, via Cesare Saldini, 50, I-20133 Milano
E-mail address: `vincenzo.marra@unimi.it`

[1] Ignorate il fatto che vi possano essere più operazioni che comportano massima uscita: è sufficiente che la funzione individui una qualunque di esse.

```
pc-marra:Soluzione enzo$ ./a.out operazioni.txt rapporto.txt
Num. 3  Imp.   -2.500    Desc. POS
Num. 2  Imp. -100.000    Desc. Bancomat
Num. 1  Imp. -1250.000   Desc. Bonifico
Num. 4  Imp. -1300.000   Desc. Assegno
Num. 5  Imp. 1200.000    Desc. Assegno
Num. 6  Imp. 200.500     Desc. Bonifico
Saldo  :-1252.000
Uscite : -2652.500
Entrate: 1400.500
Massima uscita:
Num. 4  Imp. -1300.000   Desc. Assegno
Massima entrata:
Num. 5  Imp. 1200.000    Desc. Assegno
pc-marra:Soluzione enzo$
```

FIGURA 3. Un esempio di esecuzione dell'intero programma.