

Relazione Compilatori – Progetto “Biblioteca”

Gruppo1:

- Domenico Luciani domenicoleoneluciani@gmail.com
- Daniela Conti danielaconti78@gmail.com
- Andrea Pergola pergoandrea@gmail.com

L'obiettivo del progetto “Biblioteca” è quello di realizzare un'applicazione, con l'ausilio di Flex e Bison, generatori automatici di analizzatori lessicali e sintattici rispettivamente, capace di aiutare l'impiegato di una biblioteca, nella gestione dei libri disponibili, in prestito ed individuare i soggetti “morosi” (soggetti che hanno preso in prestito un libro da più di 60 gg.)

Descrizione del programma

Si fornisce al programma un file di testo in input contenente tre sezioni.

Nella prima sezione, troviamo la data in cui viene effettuata l'analisi nel formato GG/MM/AAAA.

Nella seconda sezione troviamo l'elenco dei libri della biblioteca, raggruppati per autore e le informazioni su di essi (isbn, titolo, numero di pagine, collocazione).

La biblioteca possiede solo una copia di ogni titolo, poiché non specificato e lasciato alla libera interpretazione.

Nella terza sezione, troviamo l'elenco degli individui che hanno preso in prestito dei libri.

Ogni individuo può avere in prestito uno o più libri.

In caso di corretto formato del file in input, il programma restituisce in output l'elenco dei libri ancora disponibili e di quelli che sono in prestito da più di 60 giorni e una statistica sul numero di pagine lette da ogni utente.

Input del programma

Ogni file fornito come input al programma, deve rispettare un formato particolare. Individuiamo tre sezioni fondamentali: data analisi, sezione libri e sezione prestiti. Le tre sezioni sono inoltre divise da separatori simbolici.

Data analisi

La prima sezione contiene l'indicazione sul giorno, mese ed anno in cui si sta facendo l'analisi nel formato GG/MM/AAAA come nell'esempio seguente:

22/12/2015

In questa sezione la data deve rispettare il formato, e il vincolo 01/31 ma non verificare se la data è coerente (30 febbraio 31 novembre).

La sezione della data e quella dei libri transazioni sono divise tra loro dal separatore simbolico “%%”.

Sezione libri

La sezione dei libri contiene il nome dello scrittore, seguito dall'elenco dei libri scritti dall'autore ed possesso dalla biblioteca.

(Ovviamente questa sezione non può essere vuota visto che una biblioteca senza libri non può esistere).

Il formato dei dati è il seguente:

"Hesse Herman" -> 88-17-83457-X: "Narciso e Boccadoro":200:LS SO 127 A, 88-14-24B43-2:"Siddhartha":236:LS SO 127 B, 88-12-34AA3-B: "Il lupo della steppa":262:LS SO 127 C;

"Baricco Alessandro"-> 88-17-10625-9: "Seta":100:LI AV 1, 88-17-86563-X: "City":319:LI AV 2 A;

"Christiane F"-> 88-17-11520-7: "Noi, i ragazzi dello zoo di Berlino":346:LS B0 1;

!!

Il separatore "!!" individua la sezione degli utenti che hanno preso in prestito dei libri

Sezione prestiti

L'ultima sezione contiene un elenco dei soggetti che hanno preso in prestito dei libri
il formato dati è il seguente:

"Stefano Benni" : BNNSFN90A01G999A : 12/04/2015 88-17-83457-X, 20/09/2015 88-14-24B43-2, 29/11/2015 88-17-11520-7;

Questa sezione può essere vuota (evenienza possibile nel caso in cui tutti hanno restituito i libri in biblioteca).

Nella terza sezione, sul codice fiscale non è previsto alcun controllo sulla coerenza e consistenza delle informazioni, ad esempio la corrispondenza tra nomi e codici fiscali, purché tale informazione rispecchi il formato richiesto (come stringa alfanumerica maiuscola e di lunghezza 16 simboli).

Analizzatore lessicale

L'analizzatore lessicale del programma "Biblioteca" è realizzato in Flex, e prevede la tokenizzazione dei seguenti elementi:

Espressione Regolare	Token
{ws}	-
"%%"	SP1
"!!"	SP2
"->"	SEP_SCRIT
" "	SEP_LIST
","	FIN_LIST
":"	CAMPO_LIBRO
\"[A-Za-z0-9]+\"	TEST0
([0-9]{2})"-([0-9]{2})"-([A-F0-9]{5})"-([A-Z0-9]{1})	ISBN
[0-9]+	NUM_PAG
("LI" "LS")	COLL1
<OCC>("AV" "BO" "SO")	COLL2
<OCC2>([0-9]+)([A-Za-z]*)	NUM_COLL
[A-Z0-9]{16}	CF
([0-3][0-9] 10 20 30)"/"([0-1][1-9] 10)"/"([0-9]{4})	DATA

Ad ogni espressione regolare corrisponde un token, separato da spazi.

Alcuni token, in grassetto, possiedono un attributo addizionale, passato all'analizzatore sintattico tramite la variabile di Flex `yylval`.

- Il pattern descritto dall'espressione regolare: {ws}, serve ad individuare i caratteri blank ed ignorarli.
- Dopo il token SEP1 iniziano le informazioni sui libri della biblioteca; ogni libro è identificato tramite: autore (**TEST0**), seguito dal token "->" (SEP_SCRIT), codice univoco (ISBN), seguito da ":" (CAMPO_LIBRO), Titolo del libro (**TEST0**), ":" (CAMPO_LIBRO), numero di pagine (**NUM_PAG**), e collocazione (**COLL1**, **COLL2**, **NUM_COLL**), separati dal token "," (SEP_LIST) seguito dagli altri titoli scritti da quell'autore e posseduti dalla biblioteca.
- Il codice ISBN rispetta le specifiche date dal testo e cioè due caratteri numerici, seguiti da un trattino, seguito da due caratteri numerici, trattino, 5 caratteri esadecimali, trattino e 1 carattere alfanumerico.
- ("LI"|"LS") si dà inizio ad una start condition (OCC) che ritorna il token COLL1; attiva anche la start condition (OCC2) che ritorna il token COLL2
- Il token SEP2 dà inizio all'ultima sezione, in cui ogni utente che ha preso in prestito libri è individuato da un nome e cognome (**TEST0**), dal token ":" (CAMPO_LIBRO), dal codice fiscale (**CF**), dal token ":" (CAMPO_LIBRO), dalla data del

prestito (**DATA**), dall' ISBN del libro preso in prestito (**ISBN**), separati dal token “,” (SEP_LIST) a seconda di quanti libri l'utente ha preso in prestito.

- Per il token **DATA** l'analizzatore lessicale, fa in modo che il file input abbia come giorno 01 39, come mese 01 19. Il limite <32 e < 13 viene gestito dal parser.

Per esempio, con un file input ben formato con un solo libro posseduto dalla biblioteca ed un solo libro in prestito, la lista di token che l'analizzatore lessicale genera potrebbe essere la seguente:

```
DATA
SEP1
TEST0  SEP_SCRIT  ISBN  CAMPO_LIBRO  TEST0  CAMPO_LIBRO  NUM_PAG
CAMPO_LIBRO COLL1 COLL2 NUM_COLL FIN_LIST
SEP2
TEST0 CAMPO_LIBRO CF CAMPO_LIBRO DATA ISBN FIN_LIST
```

Si precisa inoltre che i token COLL1 COLL NUM_COLL potrebbero non apparire in quanto opzionali per ogni libro.

Analizzatore sintattico

L'analizzatore sintattico del programma “Biblioteca” è realizzato in Bison, e prevede il riconoscimento di una struttura grammaticale con regole prestabilite a partire dai token forniti dall'analizzatore lessicale.

Avendo necessità di trattare sia stringhe che numeri interi, definiamo un tipo di dato union come puntatore a caratteri char *stringa e intero; definiamo quindi i tipi di token in lettura:

Token	Tipo attributo
TEST0	<stringa>
ISBN	<stringa>
COLL1	<stringa>
COLL2	<stringa>
NUM_COLL	<stringa>
CF	<stringa>
DATA	<stringa>
NUM_PAG	<intero>
SP1	-
SP2	-
SEP_SCRIT	-
SEP_LIST	-
FIN_LIST	-
CAMPO_LIBRO	-

Specifichiamo che vogliamo visualizzare un messaggio d'errore da noi definito con la direttiva %error-verbose e che l'analisi sintattica deve iniziare con il simbolo non terminale Input.

Input: Sezione1 SP1 Sezione2 SP2 Sezione3

La regola iniziale ha la stessa struttura del file in input: sezione della data (Sezione1), sezione libri della biblioteca (Sezione2), sezione utenti che hanno preso in prestito libri (Sezione3) con i relativi separatori (SP1, SP2).

Il non terminale Sezione1 si trasforma nel terminale DATA

```
Sezione1: DATA
```

Il non terminale Sezione2, non può essere vuota e contiene una lista di libri posseduti dalla biblioteca raggruppati per autore.

E' possibile che di quell'autore si possenga solo un libro, o si posseggano più titoli ed è il caso della ricorsione sulla destra di Sezione 2

```
Sezione2: Nome_autore ElencoLibri Sezione2
          | Nome_autore ElencoLibri
          ;
```

Il non terminale Nome_autore si trasforma nei terminali nome dell'autore e il separatore.

```
Nome_autore: TEST0 SEP_SCRIT
```

Il non terminale ElencoLibri può essere uno o più occorrenze di libri scritti da un determinato autore separate da un separatore lista

```
ElencoLibri: StrutturaLibro SEP_LIST ElencoLibri
            | StrutturaLibro FIN_LIST
            ;
```

Il non terminale StrutturaLibro si trasforma nei terminali definiti dalla struttura.

```
StrutturaLibro: ISBN CAMPO_LIBRO TEST0 CAMPO_LIBRO NUM_PAG
                CAMPO_LIBRO COLL1 COLL2 NUM_COLL
```

Il non terminale Sezione3 è una sezione che potrebbe essere vuota (evenienza possibile quando tutti i libri sono disponibili e nessun prestito libro attivo) , oppure contenere uno o più utenti che hanno preso in prestito dei libri.

```
Sezione3:      /* Empty */
              | InizioSezione3
              ;
```

```
InizioSezione3: Nome_cliente Elenco_prenotazioni InizioSezione3
                | Nome_cliente Elenco_prenotazioni
                ;
```

Il non terminale `Nome_cliente` si trasforma nei terminali che identificano l'utente che ha preso in prestito il libro (cognome e nome, cf)

```
Nome_cliente: TESTO CAMPO_LIBRO CF CAMPO_LIBRO
```

Ogni utente poi può avere uno o più libri in prestito, definito con il non terminale `Elenco_prenotazioni`

```
Elenco_prenotazioni: Prenotazioni SEP_LIST Elenco_prenotazioni
                    | Prenotazioni FIN_LIST
                    ;
```

Infine il non terminale `Prenotazioni` si trasforma nei terminali (data prestito e isbn del libro)

```
Prenotazioni: DATA ISBN
              ;
```

In corrispondenza di una struttura sintattica valida, verranno intraprese le azioni semantiche corrispondenti alle varie produzioni della grammatica.

Azioni Semantiche e Tabella dei simboli

Nella sezione delle definizioni dell'analizzatore sintattico, abbiamo dichiarato alcune variabili che utilizziamo per gestire i dati di nostro interesse.

Il puntatore a caratteri `*dataOggi` viene utilizzato per tener traccia della data in cui si effettua l'analisi, corrispondente alla prima sezione del file in input.

Per gestire le date si fa riferimento alla libreria "time.h" che contiene al suo interno delle strutture e funzioni create appositamente per gestire questo tipo di problemi.

Si controlla il formato corretto della data, cioè il per il campo giorno viene considerato valido un valore da 01 a 31 e per il campo mese è considerato valido un valore da 01 a 12, altrimenti viene restituito un messaggio di errore.

Analogamente, si controlla il formato data corretto anche sul puntatore a caratteri `*dataPr`.

L'esercizio, specifica che in biblioteca non sono presenti libri con genere LI BO.

Questo vincolo viene gestito comparando il puntatore a caratteri `*coll1` con la stringa "LI" e `*coll2` con la stringa "BO";

```
if(strcmp(coll1, "LI") == 0 && strcmp(coll2, "BO") == 0)
```

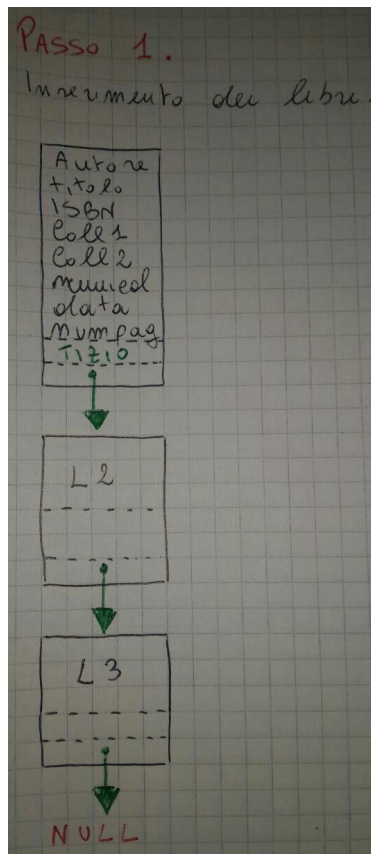
in caso positivo si restituisce un messaggio di errore:

```
{ yyerror("Non esiste il genere LI BO"); }
```

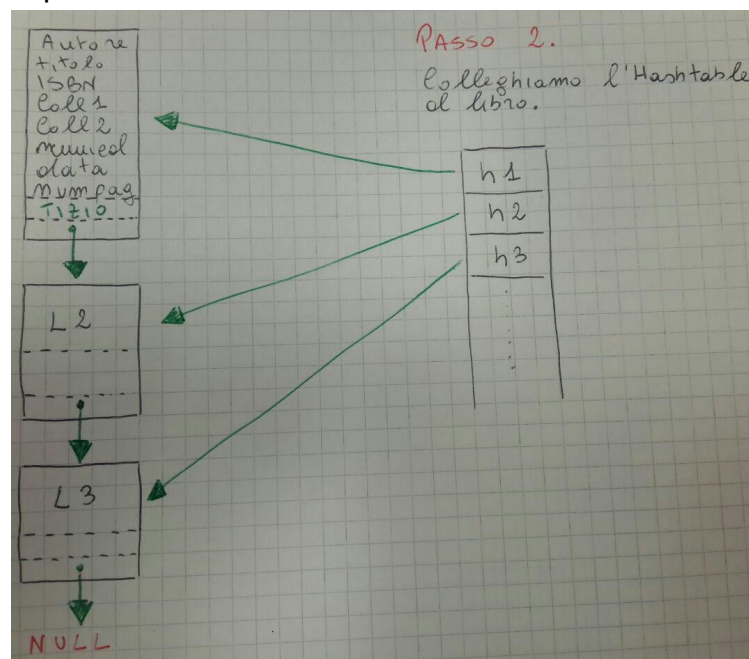
per poi continuare ad analizzare la restante parte.

Usiamo 2 hashtable, una per i clienti e una per i libri.

L'insieme dei libri è rappresentato da una lista concatenata.



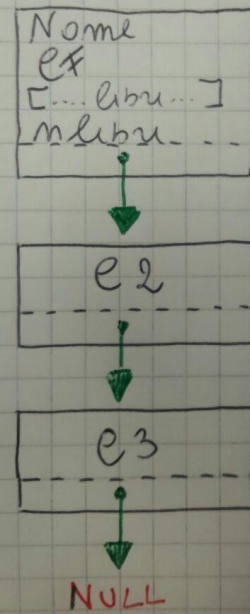
Dopo l'inserimento del libro viene generato un hash usando il suo ISBN che useremo per identificare la sua posizione all'interno dell'hashtable collegando la stessa all'elemento della lista dei libri corrispondente.



L'insieme dei clienti è rappresentata da una lista concatenata. Ogni cliente può avere più libri in prestito, per risolvere questo problema abbiamo dotato ogni cliente di un array dinamico contenente gli hash dei libri che ha prenotato.

PASSO 3.

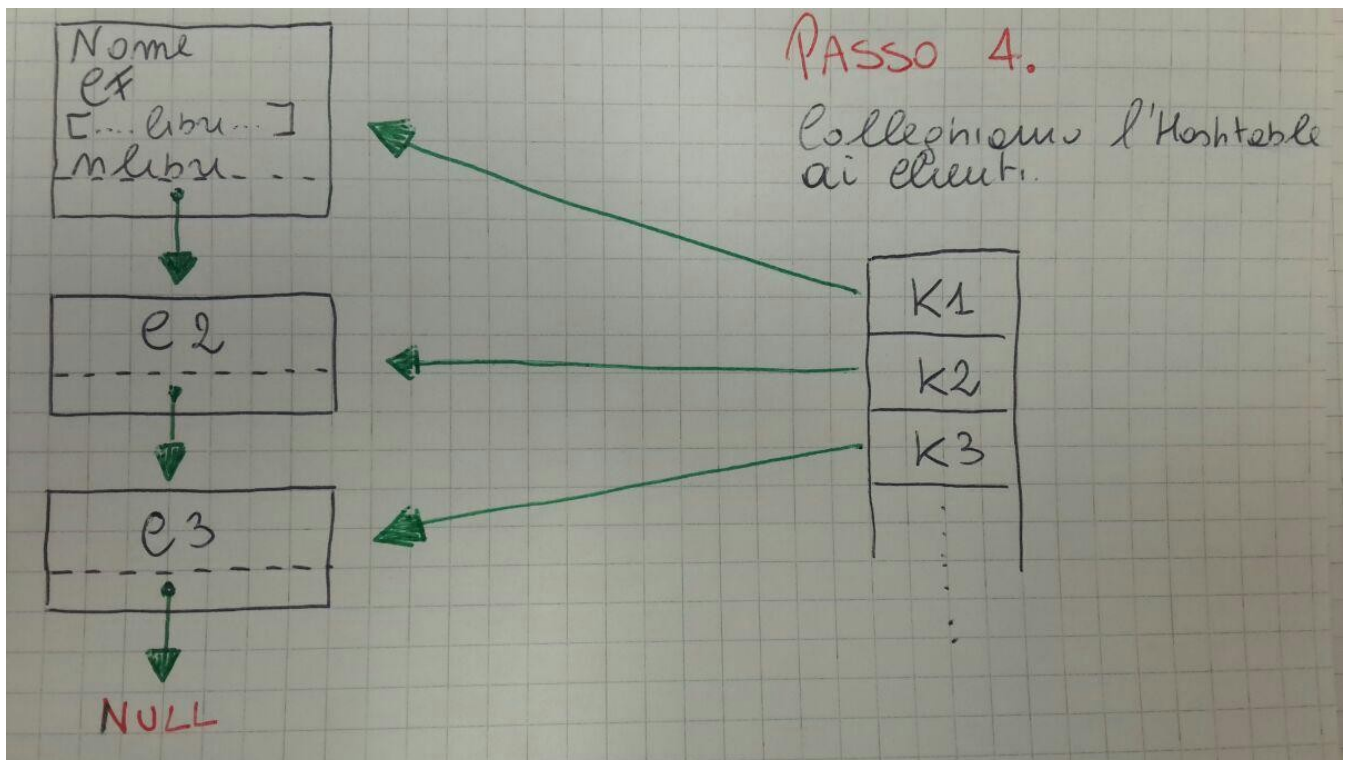
Inserimento di un cliente



Durante l'inserimento di un cliente effettuiamo un controllo all'interno dell'hashtable per verificare se è già presente nel nostro database, qualora non lo fosse generiamo un hash usando il suo CF che useremo per identificare la sua posizione all'interno dell'hashtable collegando la stessa all'elemento della lista dei clienti corrispondente, altrimenti non viene creato.

PASSO 4.

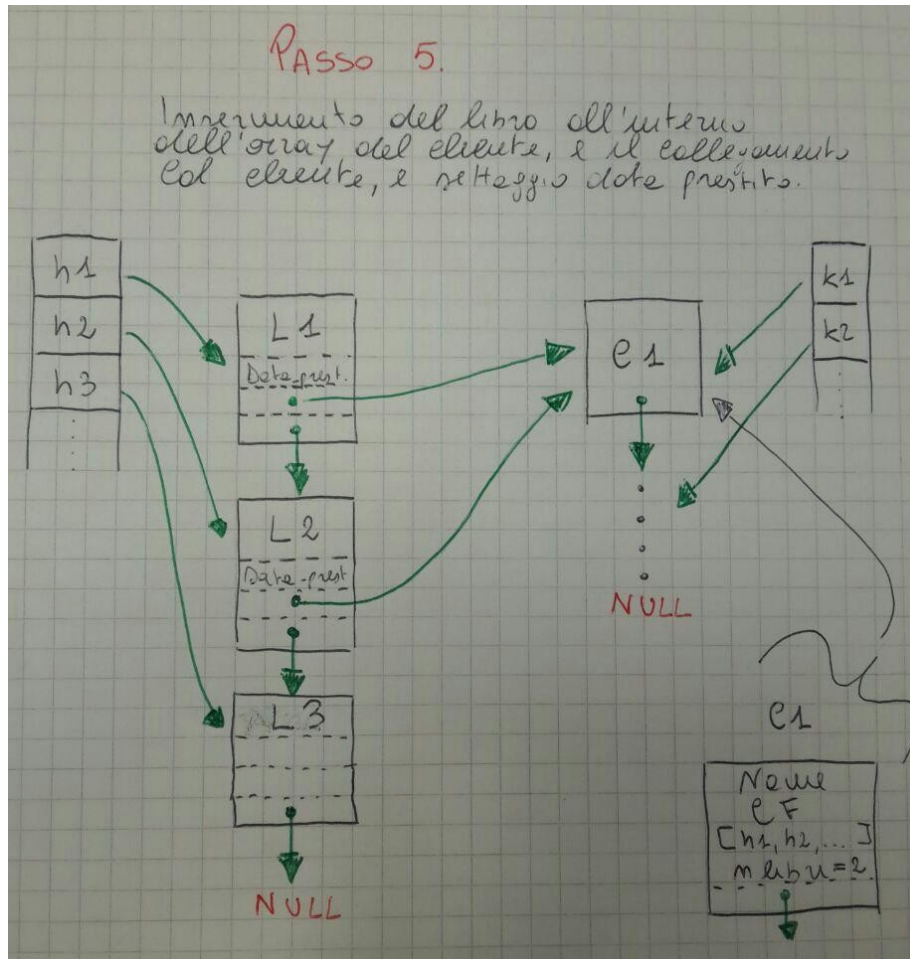
Colleghiamo l'Hashtable ai clienti.



Questa operazione è utile per non creare duplicati all'interno del database, collegando così più libri ad un singolo cliente.

Dopo di ciò l'array di libri all'interno della struttura del cliente viene ingrandito inserendo all'interno di esso Hash del libro in questione, incrementando un contatore che tiene conto del numero di libri presi in prestito dall'utente in questione.

Dopo questa operazione non ci resta altro che collegare l'elemento libro -trovato tramite il suo Hash, usando l'Hashtable- al cliente, inoltre settiamo il campo data del libro in questione con la data del prestito.



Output

Per visualizzare i libri disponibili scorriamo la lista dei libri, stampiamo i dati del libro che non ha nessun collegamento con un cliente. (Così da effettuare solo n controlli al massimo, dove n è il numero di libri)

Per visualizzare i libri scaduti (presi in prestito per più di 60 giorni) scorriamo la lista dei clienti (quindi al più n controlli al massimo dove n è il numero di clienti) e per ognuno di esso analizziamo il suo array di libri presi in prestito (quindi al più m controlli al massimo dove m è il numero di libri presi in prestito dal singolo utente n-esimo); per ogni Hash presente all'interno dell'array troviamo il libro corrispondente tramite l'Hashtable (accedendo direttamente ad esso), a questo punto compariamo la data del prestito con la data attuale; se la loro differenza è maggiore di 60 stampiamo il CF del cliente seguito da tutti i suoi prestiti scaduti.

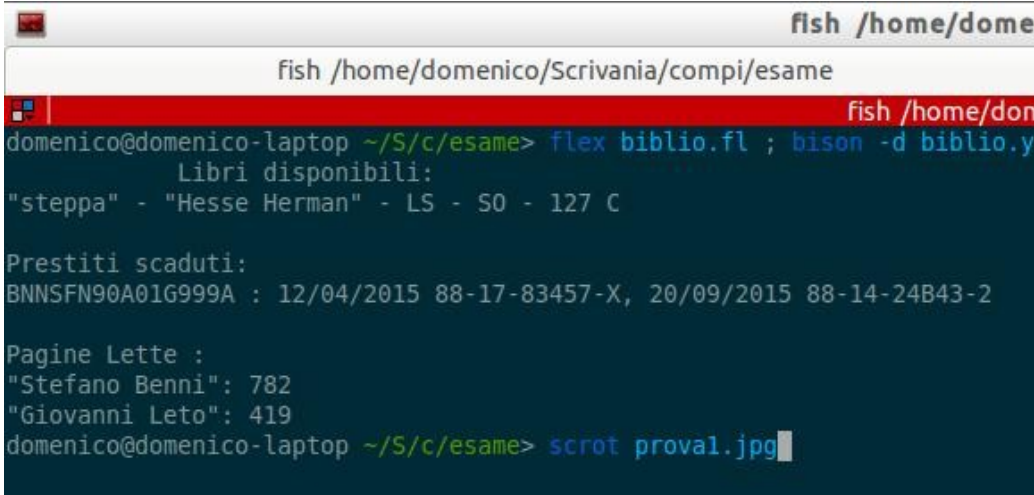
In questo passaggio usiamo la funzione difftime della libreria time.h che permette di fare la differenza tra due date opportunamente inserite all'interno di apposite strutture dati.

Per visualizzare la somma complessiva delle pagine lette da ogni singolo cliente scorriamo la lista dei clienti, per ogni cliente analizziamo l'array di libri presi in prestito, come fatto in precedenza. Dopo di ciò sommiamo il numero di ogni libro preso in prestito dal cliente e in fine stampiamo il risultato.

File di esempio

Si allegano i seguenti file di esempio da sottoporre all'applicazione "Biblioteca" con descrizione dell'esito di lettura:

"prova1.txt" – Input di esempio fornito con la consegna del progetto;

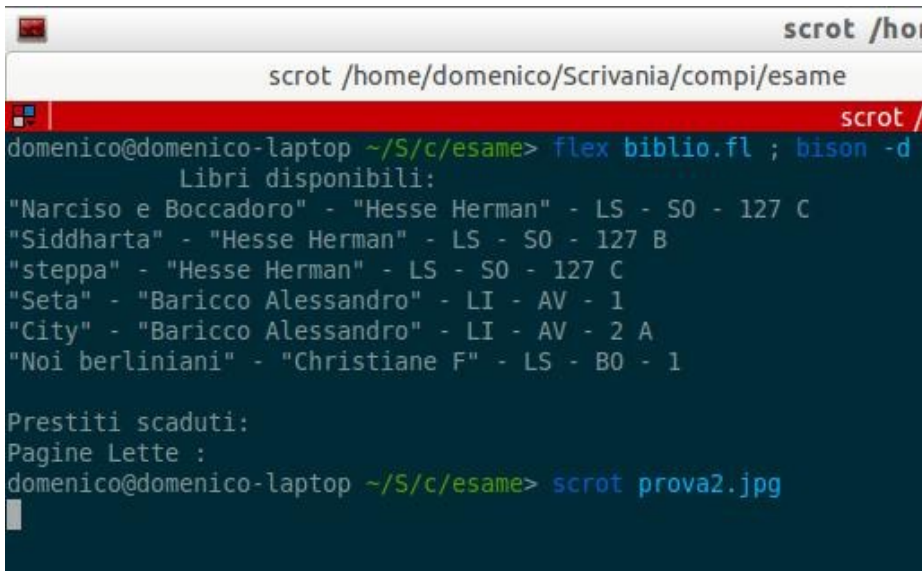


```
fish /home/domenico/Scrivania/compi/esame
domenico@domenico-laptop ~/S/c/esame> flex biblio.fl ; bison -d biblio.y
Libri disponibili:
"steppa" - "Hesse Herman" - LS - S0 - 127 C

Prestiti scaduti:
BNNSFN90A01G999A : 12/04/2015 88-17-83457-X, 20/09/2015 88-14-24B43-2

Pagine Lette :
"Stefano Benni": 782
"Giovanni Leto": 419
domenico@domenico-laptop ~/S/c/esame> scrot prova1.jpg
```

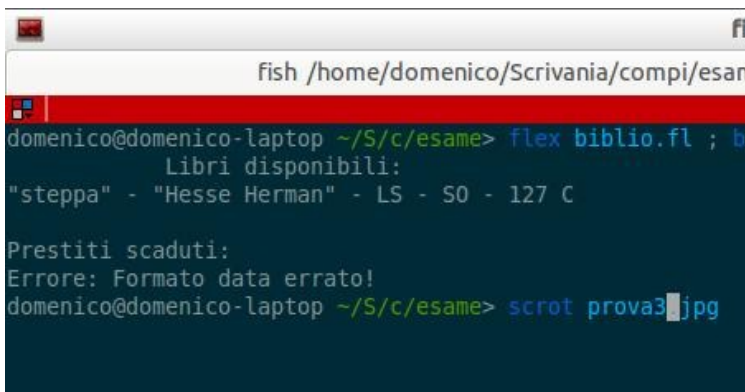
"prova2.txt" – Variante senza prestiti in corso;



```
scrot /home/domenico/Scrivania/compi/esame
domenico@domenico-laptop ~/S/c/esame> flex biblio.fl ; bison -d
Libri disponibili:
"Narciso e Boccadoro" - "Hesse Herman" - LS - S0 - 127 C
"Siddharta" - "Hesse Herman" - LS - S0 - 127 B
"steppa" - "Hesse Herman" - LS - S0 - 127 C
"Seta" - "Baricco Alessandro" - LI - AV - 1
"City" - "Baricco Alessandro" - LI - AV - 2 A
"Noi berliniani" - "Christiane F" - LS - B0 - 1

Prestiti scaduti:
Pagine Lette :
domenico@domenico-laptop ~/S/c/esame> scrot prova2.jpg
```

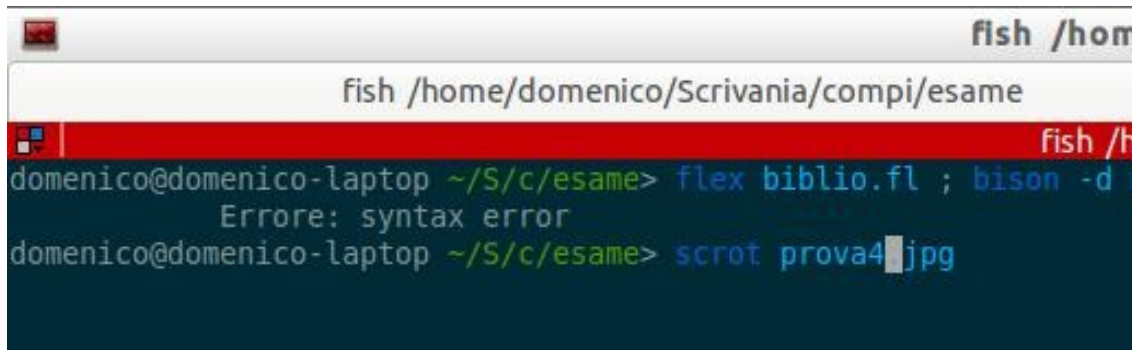
"prova3.txt" – Variante con input in formato errato (data errata);



```
fish /home/domenico/Scrivania/compi/esame
domenico@domenico-laptop ~/S/c/esame> flex biblio.fl ; b
Libri disponibili:
"steppa" - "Hesse Herman" - LS - S0 - 127 C

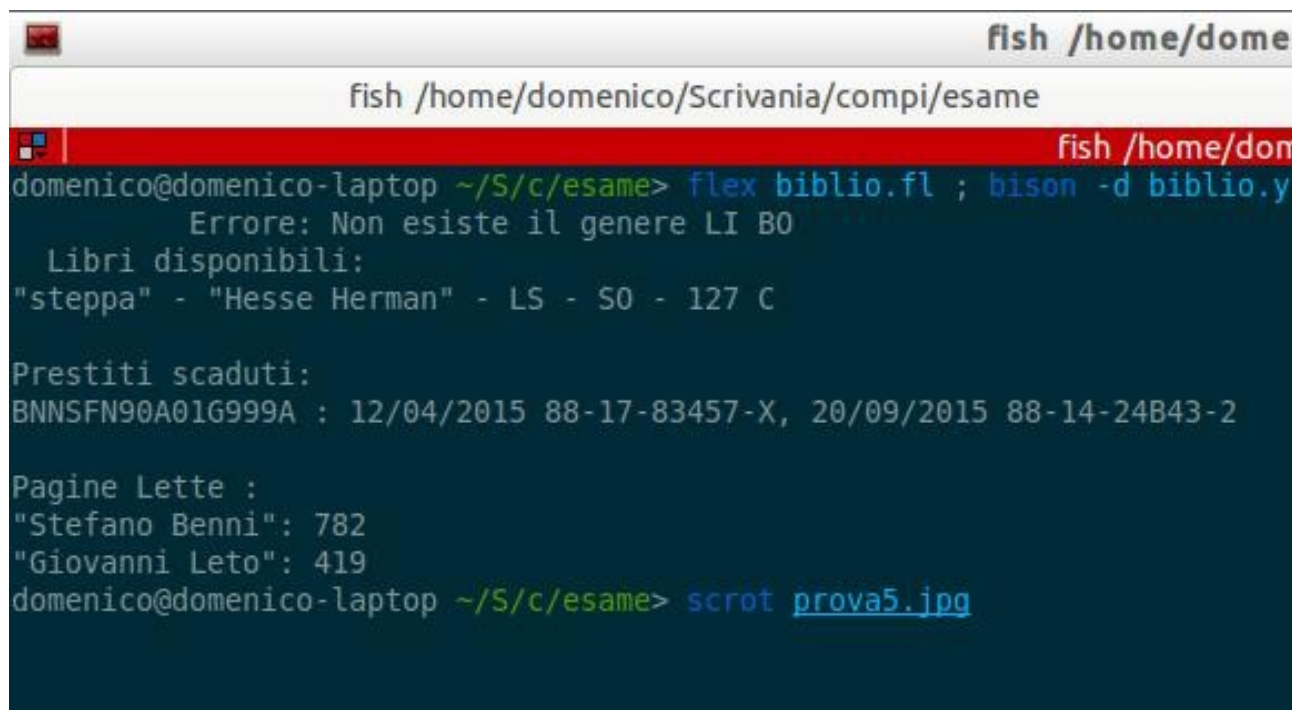
Prestiti scaduti:
Errore: Formato data errato!
domenico@domenico-laptop ~/S/c/esame> scrot prova3.jpg
```

“prova4.txt” – Variante in assenza separatore (%%);



```
fish /home/domenico/Scrivania/compi/esame
domenico@domenico-laptop ~/S/c/esame> flex biblio.fl ; bison -d
        Errore: syntax error
domenico@domenico-laptop ~/S/c/esame> scrot prova4.jpg
```

“prova5.txt” – Variante con libro genere LI BO



```
fish /home/domenico/Scrivania/compi/esame
domenico@domenico-laptop ~/S/c/esame> flex biblio.fl ; bison -d biblio.y
        Errore: Non esiste il genere LI BO
        Libri disponibili:
        "steppa" - "Hesse Herman" - LS - S0 - 127 C

        Prestiti scaduti:
        BNNSFN90A01G999A : 12/04/2015 88-17-83457-X, 20/09/2015 88-14-24B43-2

        Pagine Lette :
        "Stefano Benni": 782
        "Giovanni Leto": 419
domenico@domenico-laptop ~/S/c/esame> scrot prova5.jpg
```

Possibili Miglioramenti

Il programma può essere migliorato in futuro prevedendo più copie per un determinato titolo, basterebbe sostituire il puntatore ad un singolo cliente con un array di hash clienti (un po' come fatto nei clienti per i libri) e mettendo un campo numeroCopie decrementandolo ogni volta; se il numero di copie è 0 non deve essere possibile prenotare il libro.