UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO
DIPARTIMENTO DI INFORMATICA

Caso di studio, traccia 3:

SOFTWARE PER LA GESTIONE DI MOSTRE D'ARTE

Autori:

Antonio Gravina

Mattia Siragusa

INDICE

SPECIFICHE DEL PROBLEMA

Una prestigiosa galleria d'arte ha necessità di un sistema software per l'organizzazione di mostre permanenti e temporanee a livello nazionale e per la loro fruizione.

Ogni mostra è caratterizzata da:

- **un luogo di esposizione (es. una pinacoteca);**
- **un intervallo temporale, se la mostra è temporanea;**
- **un responsabile locale;**
- **un numero variabile di opere d'arte (dipinti, sculture, disegni, ecc.), ciascuna delle quali caratterizzata da:**
  - **nome;**
  - **autore;**
  - **tipo (es. dipinto);**
  - **genere (es. ritratto);**
  - **periodo storico (es. Barocco);**
  - **anno di produzione.**

Il sistema deve permettere, comunicando con file, di:

- **registrare un nuovo utente interessato a prenotare la visita di una mostra;**
- **cercare una particolare opera d'arte e la mostra in cui è esposta;**
- **fare una ricerca a più ampio respiro che restituisca insiemi di opere e le mostre in cui esse sono esposte, in base ad autore, genere, periodo storico di riferimento, anno/decennio/secolo di produzione;**
- **gestire le prenotazioni per la visita a una particolare mostra in un particolare giorno.**

Il sistema deve permettere all'utente anche di modificare i dati relativi al suo profilo (tranne il nome utente, univoco), finanche eliminarlo, così come di disdire eventuali prenotazioni entro un limite di tempo ragionevole.

ANALISI

Il progetto prevede la realizzazione di un programma per la gestione delle mostre organizzate da una galleria d'arte, nonché delle opere esposte in ogni mostra. In particolare, il software deve consentire la possibilità da parte dell'utente di registrarsi e di effettuare una prenotazione. Per alcune mostre è previsto un limite di partecipanti giornalieri, per altre no.

Inoltre il software prevede che l'utente possa cercare una o più opere d'arte (quindi visualizzare in quale mostra l'opera ricercata è esposta), in base a dei criteri come l'autore, il tipo, il genere, il periodo storico e l'anno di produzione.

L'utente deve essere capace di poter modificare il profilo (eventualmente eliminarlo), e di poter disdire la prenotazione effettuata entro un limite di tempo.

Il programma richiede quindi l'utilizzo di tre file in formato CSV in cui memorizzare le informazioni di ciascuna mostra, di ciascun'opera e di ciascun utente registrato. Queste informazioni devono essere lette all'inizio dell'esecuzione del programma e memorizzate in vettori di struct. Successivamente il software deve chiedere all'utente se vuole registrarsi, effettuare il login (qualora si sia già registrato) oppure uscire.

Qualora l'utente decida di registrarsi, bisogna accedere al file utenti per scrivere su una nuova riga le informazioni del nuovo utente. Per garantire maggiore sicurezza, la password non deve essere scritta sul file "in chiaro": sarebbe appropriato utilizzare un algoritmo di crittazione. Inoltre ci dev'essere un numero minimo di caratteri per la password. Ovviamente l'utente non può scegliere un username già utilizzato.

Subito dopo la fase di registrazione, l'utente accede a un menù da cui è può selezionare le varie operazioni possibili, ad esempio la prenotazione a una mostra.

Se l'utente esegue il login, ha un numero massimo di tentativi per poter sbagliare le credenziali di accesso: nel caso in cui si supera questo valore massimo, l'utente esce automaticamente dal programma (questa soluzione è utile per evitare attacchi hacker del tipo "brute force").

Una feature aggiuntiva, non esplicitamente richiesta dalla traccia, è un utente admin, con credenziali di accesso specifiche. L'admin può modificare le informazioni presenti sui file indirettamente, mediante il programma. Nel caso in cui i file abbiano poche informazioni, l'admin preferirebbe aprire i CSV con un text editor e modificarli, ma se sono memorizzate tante informazioni questa feature è molto utile.

Poiché si suppone che l'admin sia un esperto, gli si dà una totale libertà per le operazioni che può effettuare. Tuttavia, se inserisce dati contraddittori (ad esempio crea una nuova mostra per il 32 gennaio, giorno non esistente) il software potrebbe avere dei problemi che comprometterebbero le sue funzionalità. Per questo motivo l'admin dev'essere capace di eseguire il backup dei file prima di poterli modificare.

Quindi, all'avvio del programma e in seguito all'accesso da parte dell'utente appare un menù, in cui è possibile:

Prenotarsi a una mostra. L'utente può scegliere una tra le mostre disponibili per poi inserire la data di prenotazione, con gli eventuali controlli di inserimento della data. Ad esempio, se l'utente inserisce una data non valida o una data per cui si è già raggiunto il numero massimo di prenotazioni, egli riceve un messaggio di errore. Se l'utente si è già prenotato a una mostra, non può prenotarsi ad un'altra.

Visualizzare e/o modificare le informazioni del profilo. L'utente può leggere le proprie credenziali e modificarle. L'username non è modificabile. Da qui è possibile disdire una prenotazione già effettuata, a meno che non manchino sette giorni all'esposizione della mostra. Inoltre una delle opzioni di questo "sotto-menù" è l'eliminazione del profilo.

Visualizzare le informazioni delle mostre/opere. Da qui l'utente può cercare una o più opere d'arte in base ai parametri richiesti dalla traccia, e se vuole informazioni aggiuntive il programma apre il browser predefinito e ricerca automaticamente su Google l'opera.

Un'altra funzionalità è la possibilità di poter visualizzare le informazioni di una specifica mostra (quindi il luogo di esposizione, l'eventuale data in cui la mostra è visitabile e il responsabile della mostra), nonché la lista delle opere esposte nella mostra selezionata.

Se a fare l'accesso è l'admin, appare un menù diverso in cui è possibile:

Modificare i file. L'admin accede indirettamente ai file CSV e li modifica dal prompt dei comandi, o da una eventuale interfaccia grafica del software. Sebbene gli sia data una totale libertà, l'admin non può modificare le sue credenziali (però può modificare quelle di qualsiasi altro utente). L'admin può anche eliminare una mostra, un'opera d'arte o un utente.

Visualizzare i dati. In particolare può leggere le credenziali di tutti gli utenti, comprese le loro password in chiaro. Ovviamente ciò potrebbe comportare problemi di sicurezza, motivo per cui l'admin deve essere necessariamente una persona fidata nell'azienda per cui si sta progettando il software. In questo caso potrebbe essere il presidente della galleria d'arte o uno dei responsabili.

Aggiungere una nuova mostra/opera/utente.

A seguito di un'operazione è richiesto all'utente o all'admin di inserire il valore 1 per continuare ad usare il programma. Nel caso in cui questo valore predefinito venga accettato, l'utente o l'admin ritorna al suo menù, altrimenti si segnala l'uscita dal software mediante un messaggio del tipo "Grazie per aver usato il programma".

Se l'utente ha cancellato il suo profilo il programma non deve chiedere di uscire, e quindi si esce automaticamente. Analogamente nel caso in cui l'admin ha modificato uno o più file (questo per evitare malfunzionamenti).

A ogni utente sono assegnati i campi "prenotazione", che indica a quale mostra si è prenotato, "giorno prenotazione" e "mese prenotazione". Se "prenotazione" è uguale al valore 0 l'utente non è prenotato a nessuna mostra.

Ma se l'utente si è prenotato a una mostra per il giorno x, cosa succederà quando sarà il giorno x+1?

Se non si interviene, l'utente non può partecipare ad una nuova mostra, perché è possibile prenotarsi ad una per volta, e il suo campo "prenotazione" è ancora pieno. In pratica il suo account diventerebbe inutile.

Sarà quindi l'admin a modificare le informazioni del suddetto utente, resettando i campi precedentemente elencati? Se così fosse, sarebbe estremamente scomodo, perché l'amministratore dovrebbe ogni giorno accedere al programma e controllare le prenotazioni di ogni utente.

La soluzione più logica è l'inserimento di una procedura che, subito dopo la lettura da file delle informazioni, controlli le prenotazioni di ogni utente: se il giorno è passato, si resettano i campi di prenotazione.

DESCRIZIONE DEL SISTEMA

Il programma prevede l'astrazione di tre entità:

Mostra, formata dai seguenti campi: nome, luogo di esposizione, giorno inizio esposizione, mese inizio esposizione, giorno fine esposizione, mese fine esposizione, massimo partecipanti giornalieri, responsabile.
Nota: se la mostra è disponibile per tutto l'anno i campi in cui si definisce il periodo di esposizione hanno valore 0.

Opera, formata dai seguenti campi: nome, autore, tipo, genere, periodo artistico, anno di produzione, codice.
Nota: "codice" indica in quale mostra è esposta l'opera. Ad esempio, se l'opera ha codice 1, allora essa è esposta nella seconda mostra memorizzata nel file (quindi per le opere esposte nella prima mostra il codice è 0).

Utente, formato dai seguenti campi: nome, cognome, username, password, prenotazione, giorno prenotazione, mese prenotazione.
Nota: il campo "prenotazione" indica a quale mostra si è prenotato l'utente, in maniera simile al campo "codice" già trattato. Tuttavia, se il numero di prenotazione è 0, non vuol dire che l'utente si è prenotato alla prima mostra del file, ma che l'utente non si è ancora prenotato a nessuna mostra.

Relazioni:

1. Il campo "codice" mette in relazione Opera con Mostra
2. Il campo "prenotazione" mette in relazione Utente con Mostra

MAIN

All'inizio dell'esecuzione del programma, all'interno del main, vengono aperti i tre file contenenti le informazioni sulle entità precedentemente descritte, chiamati "file_mostre.csv", "file_opere.csv" e "file_utenti.csv". Se almeno uno dei tre corrispondenti puntatori è uguale a NULL, l'utente visualizza un messaggio d'errore e il programma si chiude. Ad ogni messaggio d'errore il programma emette un suono, grazie alla sequenza di escape \a.

In seguito si ha la necessità di creare tre array di struct, ognuno relativo ad un'entità. Per stabilire la dimensione dei vettori bisogna leggere dai file quante sono le mostre, quante sono le opere e quanti sono gli utenti, quindi si richiama per tre volte la funzione "conta_righe()", dando come parametro il puntatore relativo al file di cui bisogna, appunto, contare le righe.

Poiché le informazioni di una singola mostra, opera e utente sono memorizzate su una singola riga (divise dalla virgola), ne segue che il valore restituito in int dalla funzione è proprio la dimensione dell'array di struct che si vuole costruire.

In questo modo si hanno "num_mostre", "num_opere" e "num_utenti", tutte e tre variabili di tipo int.
Nota 1: si utilizzano le struct perché bisogna raggruppare in un'unica variabile i campi (di tipo non omogeneo tra loro) di un'entità. Pertanto ci saranno tre struct chiamate in forma abbreviata "mst", "opr", "utn" (rispettivamente mostra, opera e utente).

Nota 2: all'interno dei file, l'ultima riga non deve essere vuota. Ad esempio, se in "file_utente.csv" scrivo le informazioni dell'ultimo utente, il cursore deve rimanere su quella riga, senza andare a capo. Se non si rispetta ciò, il programma crasha in partenza. Per verificarlo si può provare ad aprire con un text editor uno dei tre file, posizionarsi sull'ultima riga, premere "invio" (per andare a capo) e salvare il file: il software non partirà.

Nota 3: per i nomi delle funzioni e delle variabili abbiamo preferito la lingua italiana, perché il programma è destinato ad utenti italiani. Siamo però consapevoli che l'inglese è la lingua franca dell'Informatica.

Dopo la creazione degli array di struct, bisogna popolarli con le informazioni da prendere dai file. Per questo si utilizzano le procedure "popola_mostre (pt_mostra, mst)", "popola_opere (pt_opera, opr)" e "popola_utenti (pt_mostra, utn)".

Nota: pt è l'abbreviazione di "puntatore".

Successivamente si chiudono i tre file. È inutile tenerli tutti aperti se non ce n'è bisogno: quando poi ce ne sarà la necessità, allora verranno riaperti al momento opportuno. In questo modo si evita un appesantimento della RAM. I file pesano pochissimo, ma il programma è pensato per essere funzionante anche su dispositivi poco recenti.

Segue poi la procedura "aggiorna_file (utn, num_utenti)", che ha il compito di cancellare automaticamente le prenotazioni degli utenti se il giorno è ormai passato, come affermato nella parte finale dell'analisi.

Infine è chiamata la procedura “avvio (mst, opr, utn, num_utenti)”, da cui parte il menù iniziale per la registrazione, l’accesso e l’uscita dal programma.

Nota: i parametri di “avvio” sono i tre array di struct e “num_utenti”. Quest’ultimo è fondamentale per le funzioni di accesso, di registrazione (quindi per il funzionamento dell’intero programma) e di quasi tutte le altre funzioni principiali del software, mentre “num_mostre” e “num_opere” no. Perciò non avrebbe senso passare anche queste due variabili come parametri: è meglio re-dichiararle ogni volta che ce ne sarà bisogno, all’interno delle varie funzioni.

La dichiarazione delle tre struct potrebbe avvenire nel file “main.c”. Facendo in questo modo, però, esse dovrebbero essere nuovamente dichiarate all’interno di alcuni degli header creati per il programma. Quindi, per evitare ridondanze e per favorire la leggibilità del codice, la dichiarazione delle struct avviene in “struct.h”, che è incluso in “main.c”.

REQUISITI FUNZIONALI

I requisiti funzionali sono le operazioni necessarie al funzionamento del programma, elencate nella seguente tabella.

CODICE	NOME	DESCRIZIONE
R01	**Caricamento dei dati da file**	**Il programma carica in tre array di struct i dati presenti in “file_mostre.csv”, “file_opere.csv” e “file_utenti.csv”. “file_utenti.csv”.**
R02	**Aggiornamento del file**	**Il programma elimina le prenotazioni di esposizioni già avvenute.**
R03	**Visualizzazione menù iniziale**	**Il programma fa visualizzare all’utente il menù in cui può scegliere se effettuare la registrazione, il login o l’uscita dal programma.**
R04	**Login**	**Il programma consente il login.**
R05	**Registrazione di un nuovo utente**	**Il programma registra sul file le informazioni del nuovo utente.**
R06	**Visualizzazione menù admin**	**Se l’admin ha fatto accesso, il programma mostra le operazioni ad egli riservate.**

R07	Modifica dei file	Il programma consente all'admin di modificare le informazioni memorizzate nei file.
R08	Visualizzazione file	Il programma consente all'admin di visualizzare le informazioni memorizzate nei file.
R09	Aggiunta di informazioni	Il programma consente all'admin di aggiungere una nuova mostra, opera d'arte o utente.
R10	Visualizzazione menù utente	Se l'utente ha fatto accesso, il programma mostra le operazioni ad egli riservate.
R11	Prenotazione	Il programma consente all'utente di prenotarsi a una mostra.
R12	Visualizzazione profilo	Il programma consente all'utente di visualizzare le sue informazioni.
R13	Modifica profilo	Il programma consente all'utente di modificare le informazioni del suo profilo (tranne l'username), fino a poterlo eliminare.
R14	Disdici prenotazione	Il programma consente all'utente di eliminare una prenotazione già effettuata.
R15	Ricerca mostre	Il programma mostra all'utente tutte le mostre disponibili ed egli può visualizzare le informazioni di una a sua scelta.
R16	Ricerca opere	Il programma consente all'utente di cercare una o più opere d'arte memorizzate nel database, in base ai parametri richiesti dalla traccia.

CASI D'USO

CODICE	NOME
R01	Caricamento dei dati da file
PRE-CONDIZIONI	I file devono: essere in formato CSV; non vuoti; le formattazioni stabilite per i contenuti dei file devono essere rispettate; non sono ammesse righe vuote.
POST-CONDIZIONI	Successo: il programma carica correttamente i dati dei file.
	Fallimento: gli array risultano vuoti.
EVENTO INNESCANTE	Apertura del software.
SCENARIO DI BASE	Il programma legge i file riga per riga e ogni elemento letto viene memorizzato nel corrispondente array di struct.
SCENARIO ALTERNATIVO	Se il programma non riesce ad aprire i file l'utente riceve un messaggio di errore e l'esecuzione termina.

	Se il programma trova i file ma essi non sono formattati in modo corretto, l'esecuzione si interrompe immediatamente.

CODICE	NOME
R02	Aggiornamento del file
PRE-CONDIZIONI	Dev'essere avvenuto con successo il caricamento dei file.
POST-CONDIZIONI	Successo: "file_utenti.csv" mantiene la formattazione corretta.
	Fallimento: le modifiche sul file alterano la sua formattazione originale.
EVENTO INNESCANTE	Caricamento dei dati da file.
SCENARIO DI BASE	Il programma resetta le prenotazioni passate degli utenti.
SCENARIO ALTERNATIVO	Non ci sono prenotazioni passate, quindi non c'è alcuna modifica su "file_utenti.csv".

CODICE	NOME
R03	Visualizzazione menù iniziale
PRE-CONDIZIONI	Dev'essere avvenuto con successo il caricamento dei file.
POST-CONDIZIONI	Successo: - - -
	Fallimento: - - -
EVENTO INNESCANTE	Aggiornamento del file.
SCENARIO DI BASE	L'utente inserisce '1' per effettuare il login e '2' per la registrazione.
SCENARIO ALTERNATIVO	L'utente inserisce un qualsiasi altro valore per uscire dal programma.

CODICE	NOME
R04	Login
PRE-CONDIZIONI	Dev'essere avvenuto con successo il caricamento dei file e l'utente deve aver inserito '1' nel menù.
POST-CONDIZIONI	Successo: le credenziali di accesso inserite combaciano con quelle di uno degli utenti già registrati.
	Fallimento: le credenziali inserite sono errate.
EVENTO INNESCANTE	Visualizzazione menù iniziale.
SCENARIO DI BASE	L'utente riesce ad autenticarsi ed accede ad un secondo menù, che può essere il menù utente oppure il menù admin (nel caso in cui siano state inserite le credenziali dell'amministatore).
SCENARIO ALTERNATIVO	Se l'utente sbaglia per cinque volte l'username e/o la password, il programma si chiude.

CODICE	NOME
R05	Registrazione di un nuovo utente

PRE-CONDIZIONI	Dev'essere avvenuto con successo il caricamento dei file e l'utente deve aver inserito '2' nel menù.
POST-CONDIZIONI	Successo: il nuovo utente inserisce username e password validi. Si crea un secondo array di dimensione 'num_utenti' + 1.
	Fallimento: il nuovo utente inserisce un username già esistente oppure una password che non rispetta i criteri di lunghezza (minimo 8 caratteri).
EVENTO INNESCANTE	Visualizzazione menù iniziale.
SCENARIO DI BASE	Il nuovo utente ha accesso ad un secondo menù.
SCENARIO ALTERNATIVO	Il nuovo utente non riesce a registrarsi per via delle limitazioni imposte. L'unica cosa da fare è uscire dal programma.

CODICE	**NOME**
R06	Visualizzazione menù admin
PRE-CONDIZIONI	L'utente deve aver inserito le credenziali di accesso dell'amministratore.
POST-CONDIZIONI	Successo: - - -
	Fallimento: - - -
EVENTO INNESCANTE	Login.
SCENARIO DI BASE	L'amministratore ha accesso al suo menù, ed inserisce specifici valori per poter eseguire le operazioni ad egli riservate.
SCENARIO ALTERNATIVO	L'admin continua a inserire valori non validi nel menù, causando un loop.

CODICE	**NOME**
R07	Modifica dei file
PRE-CONDIZIONI	L'amministratore deve aver inserito '1', '2' o '3' nel menù per modificare rispettivamente il database delle mostre, delle opere e degli utenti.
POST-CONDIZIONI	Successo: riapertura dei file.
	Fallimento: errori nell'apertura file causano l'interruzione del programma.
EVENTO INNESCANTE	Visualizzazione menù admin.
SCENARIO DI BASE	L'admin modifica i file senza inserire dati contraddittori.
SCENARIO ALTERNATIVO	L'admin inserisce dati contraddittori o insensati, causando malfunzionamenti critici del programma (ossia al prossimo avvio del software esso si chiuderà immediatamente).

CODICE	**NOME**
R08	Visualizzazione dei file
PRE-CONDIZIONI	L'amministratore deve aver inserito '4' nel menù.
POST-CONDIZIONI	Successo: riapertura dei file.
	Fallimento: errori nell'apertura file causano l'interruzione del programma.
EVENTO INNESCANTE	Visualizzazione menù admin.
SCENARIO DI BASE	L'admin seleziona il database che vuole visualizzare.
SCENARIO ALTERNATIVO	Nessuno.

CODICE	**NOME**

R09	Aggiunta di informazioni
PRE-CONDIZIONI	L'admin deve aver inserito '5' nel menù.
POST-CONDIZIONI	Successo: riapertura dei file.
	Fallimento: errori nell'apertura file causano l'interruzione del programma.
EVENTO INNESCANTE	Visualizzazione menù admin.
SCENARIO DI BASE	L'admin seleziona uno dei tre database e aggiunge ad esso un nuovo elemento, rispettando la corretta formattazione dei file.
SCENARIO ALTERNATIVO	L'admin non rispetta la corretta formattazione dei file, oppure inserisce dati contraddittori o insensati, causando malfunzionamenti critici del programma.

CODICE	NOME
R10	Visualizzazione menù utente
PRE-CONDIZIONI	Login o registrazione avvenuti con successo.
POST-CONDIZIONI	Successo: - - -
	Fallimento: - - -
EVENTO INNESCANTE	Login, registrazione.
SCENARIO DI BASE	L'utente visualizza un menù da cui, inserendo dei valori, può effettuare delle operazioni.
SCENARIO ALTERNATIVO	L'utente continua a inserire valori non validi nel menù, causando un loop.

CODICE	NOME
R11	Prenotazione
PRE-CONDIZIONI	L'utente ha inserito '1' nel menù.
POST-CONDIZIONI	Successo: riapertura di "file_opera.csv" e "file_utenti.csv".
	Fallimento: errori nell'apertura file causano l'interruzione del programma.
EVENTO INNESCANTE	Visualizzazione menù utente.
SCENARIO DI BASE	L'utente sceglie una mostra a cui prenotarsi e inserisce una data valida.
SCENARIO ALTERNATIVO	L'utente è già prenotato ad una mostra, quindi deve disdettarla. L'utente inserisce sempre una data non valida, causando loop.

CODICE	NOME
R12	Visualizzazione profilo
PRE-CONDIZIONI	L'utente ha inserito '2' nel menù e poi '1' nel sotto-menù.
POST-CONDIZIONI	Successo: - - -
	Fallimento: - - -
EVENTO INNESCANTE	Visualizzazione menù utente.
SCENARIO DI BASE	L'utente visualizza le informazioni del suo profilo.
SCENARIO ALTERNATIVO	Nessuno.

CODICE	NOME
R13	Modifica profilo
PRE-CONDIZIONI	L'utente ha inserito '2' nel menù e poi '2' oppure '3' nel sotto-menù.
POST-CONDIZIONI	Successo: "file_utenti.csv" si apre correttamente.
	Fallimento: errore nell'apertura del file.
EVENTO INNESCANTE	Visualizzazione menù utente.
SCENARIO DI BASE	L'utente modifica il suo profilo, fino ad eliminarlo.
SCENARIO ALTERNATIVO	Nessuno.

CODICE	NOME
R14	Disdire prenotazione
PRE-CONDIZIONI	L'utente ha inserito '4' nel menù di modifica.
POST-CONDIZIONI	Successo: "file_utenti.csv" si apre correttamente.
	Fallimento: Errore di apertura file.
EVENTO INNESCANTE	Modifica profilo.
SCENARIO DI BASE	L'utente ha disdettato con successo la prenotazione.
SCENARIO ALTERNATIVO	L'utente non si è ancora prenotato ad una mostra, quindi visualizza un messaggio di errore.

CODICE	NOME
R15	Ricerca mostre
PRE-CONDIZIONI	L'utente ha inserito '3' nel menù iniziale e '2' nel sotto-menù di ricerca.
POST-CONDIZIONI	Successo: - - -
	Fallimento: - - -
EVENTO INNESCANTE	Visualizzazione menù utente.
SCENARIO DI BASE	L'utente sceglie una mostra di cui visualizzare le informazioni.
SCENARIO ALTERNATIVO	Nessuno.

CODICE	NOME
R16	Ricerca opere
PRE-CONDIZIONI	L'utente ha inserito '3' nel menù iniziale e '1' nel sotto-menù di ricerca.
POST-CONDIZIONI	Successo: il programma ricerca nel database le opere che presentano il parametro inserito dall'utente.
	Fallimento: - - -
EVENTO INNESCANTE	Visualizzazione menù utente.

SCENARIO DI BASE	**L'utente visualizza una o più opere che presentano il parametro inserito. L'utente può eventualmente inserire '1' per la ricerca su Google dell'opera.**
SCENARIO ALTERNATIVO	**Nessuna tra le opere salvate nel database presenta il parametro inserito dall'utente, che visualizza il relativo messaggio. Il dispositivo non è collegato a Internet e quindi la ricerca su Google non avviene.**

HEADER

Le funzioni utilizzate sono state raggruppate in 11 header: (metti in tabella)

1	Struct.h
2	mostre.h
3	opere.h
4	utenti.h
5	avvio.h
6	admin.h
7	gestione_file.h
8	modifica_file.h
9	prenotazione.h
10	ricerca.h
11	critt.h

La lista degli header è ordinata in base alla logica del programma: i primi cinque sono fondamentali per l'esecuzione del software; quelli dal sesto all'ottavo contengono funzioni necessarie principalmente per le funzioni di admin (in "gestione_file.h" sono presenti anche due funzioni fondamentali per l'utente); il nono e il decimo contengono funzioni utente; l'ultimo header è opzionale, per le funzioni di crypting e decrypting.

STRUCT.H

L'header "struct.h" non contiene funzioni, ma le dichiarazioni delle struct. Inoltre tramite la direttiva #define si definisce la macro "dim" (dimensione) assegnandoli il valore 30.

Questa macro serve per definire la lunghezza degli array di caratteri.

MOSTRE.H

L'header "mostre.h" contiene funzioni che agiscono sull'array di struct mostra chiamato "mst".

CODICE	NOME	PARAMETRI	TIPO
MST01	**popola_mostre**	**FILE *pt_mostra, mostra mst[]**	**void**
MST02	**visualizza_mst**	**mostra mst[], int ind_mst**	**void**
DESCRIZIONE	**MST01**	Legge dal file CSV le informazioni delle mostre e le memorizza nell'array di struct. È una void, quindi non restituisce nulla. Tutte le modifiche effettuate sull'array mst[] all'interno di "popola_mostre" permangono anche al di fuori della funzione.	
	MST02	Fa visualizzare all'utente le informazioni della mostra in posizione ind_mst dell'array.	

OPERE.H

L'header "opere.h" contiene funzioni che agiscono sull'array di struct opera chiamato "opr".

CODICE	NOME	PARAMETRI	TIPO
OPR01	**popola_opere**	**FILE *pt_opera, opera opr[]**	**void**

OPR02	**visualizza_opr**	**mostra mst[], opera opr[], int ind_opr**	**void**
DESCRIZIONE	**OPR01**	**Legge dal file CSV le informazioni delle opere e le memorizza nell'array di struct. È una void, quindi non restituisce nulla. Tutte le modifiche effettuate sull'array opr[] all'interno di "popola_opere" permangono anche al di fuori della funzione.**	
	OPR02	**Fa visualizzare all'utente le informazioni dell'opera in posizione ind_opr dell'array.**	

UTENTI.H

L'header "utenti.h" contiene funzioni che agiscono sull'array di struct utente chiamato "utn".

CODICE	NOME	PARAMETRI	TIPO
UTN01	popola_utenti	FILE *pt_utente, utente utn[]	void
UTN02	profilo	mostra mst[], utente utn[], char username [], int num_utenti	void
UTN03	visualizza_utn	mostra mst[], utente utn[], int ind_utn	void
UTN04	modifica_profilo	utente utn[], int ind_utn, int num_utenti	void
UTN05	cancella_profilo	utente utn[], int ind_utn, int num_utenti	void
DESCRIZIONE	**UTN01**	**Legge dal file CSV le informazioni degli utenti e li memorizza nell'array di struct. È una void, quindi non restituisce nulla. Tutte le modifiche effettuate sull'array utn[] all'interno di "popola_utenti" permangono anche al di fuori della funzione.**	
	UTN02	**Funzione che apre il menù di gestione del profilo per l'utente.**	
	UTN03	**Fa visualizzare all'utente le informazioni dell'utente in posizione ind_utn dell'array.**	

	UTN04	Apre il menù per la modifica del profilo per l'utente. Una funzione simile è "modifica_utn", presente nell'header "modifica_file.h", ma quest'ultima consente all'admin (e non all'utente) di modificare le informazioni.
	UTN05	Cancella il profilo dell'utente.

AVVIO.H

L'header "avvio.h" contiene le prime funzioni che si interfacciano con l'utente. Infatti si visualizza un menù iniziale da cui poter scegliere se effettuare la registrazione, il login oppure l'uscita dal programma.

CODICE	NOME	PARAMETRI	TIPO
AVV01	avvio	mostra mst[], opera opr[], utente utn[], int num_utenti	void
AVV02	menu	mostra mst[], opera opr[], utente utn[], char username[]	void
AVV03	accesso	utente utn[], char username[], int num_utenti	void
AVV04	regist	utente utn[], char username[], int num_utenti	void
DESCRIZIONE	AVV01	Legge dal file CSV le informazioni degli utenti e li memorizza nell'array di struct. È una void, quindi non restituisce nulla. Tutte le modifiche effettuate sull'array utn[] all'interno di "popola_utenti" permangono anche al di fuori della funzione.	
	AVV02	Apre il menù che consente all'utente di scegliere se prenotare una visita a una mostra, visualizzare il proprio profilo oppure cercare informazioni sulle mostre/opere	
	AVV03	Funzione di login	
	AVV04	Funzione di registrazione	

ADMIN.H

La libreria “admin.h” contiene le funzioni per l’amministratore del sistema. Non ci sono controlli sugli input, perché l’admin dovrebbe avere totale libertà. Tuttavia sono presenti più volte dei messaggi di avvertimento, per ricordare che tutte le modifiche eseguite sui file sono irreversibili (pertanto si consiglia di eseguire il backup prima di procedere).

CODICE	NOME	PARAMETRI	TIPO
ADM01	admin_menu	mostra mst[], opera opr[], utente utn[], char username[]	void
ADM02	file_backup	FILE *pt_mostra, FILE *pt_opera, FILE *pt_utente	void
ADM03	admin_mostra	mostra mst[], char username[], int num_mostre	void
ADM04	admin_opera	opera opr[], char username[], int num_opere)	void
ADM05	admin_utente	utente utn[], char username[], int num_utenti	void
ADM06	aggiungi_riga	- - -	void
DESCRIZIONE	ADM01	Apre il menù per l’amministratore.	
	ADM02	Esegue il backup dei file, creandone altri tre chiamati “backup_mostre.csv”, “backup_opere.csv” e “backup_utenti.csv”.	
	ADM03	Apre il menù per la gestione dell’admin delle mostre.	
	ADM04	Apre il menù per la gestione dell’admin delle opere.	
	ADM05	Apre il menù per la gestione dell’admin degli utenti.	
	ADM06	Chiede all’admin se vuole aggiungere ai file una nuova mostra, opera o utente. Le funzioni per aggiungere effettivamente una nuova riga sono presenti in un altro header chiamato “modifica_header.h”.	

GESTIONE_FILE.H

Questo header contiene funzioni che lavorano sui file, leggendo e/o scrivendo su di essi.

CODICE	NOME	PARAMETRI	TIPO
GEF01	conta_righe	FILE *ptr	int
GEF02	stampa_utn	utente utn[], int num_utenti	void
GEF03	aggiorna_file	mostra mst[], opera opr[], utente utn[],	void
GEF04	visualizza_file	int num_mostre, int num_opere, int num_utenti	void
DESCRIZIONE	GEF01	Conta le righe di un file aperto.	
	GEF02	Funzione di stampa del contenuto di “file_utenti.csv”.	
	GEF03	Aggiorna “file_utenti.csv” prendendo la data odierna ed eliminando eventuali prenotazioni passate.	
	GEF04	Funzione che consente all'utente di visualizzare informazioni su un utente, una mostra o un'opera in particolare	

MODIFICA_FILE.H

Questo header contiene le funzioni che consentono all’admin di modificare i file e di aggiungere una nuova mostra/opera/utente. È quindi in stretta relazione con l’header “admin.h”.

Queste funzioni potrebbero anche essere messe nell’header “gestione_file.h”, tuttavia esso sarebbe stato troppo lungo, con circa 500 righe di codice. Per favorire il principio di divide et impera si è quindi deciso di realizzare due header separati.

CODICE	NOME	PARAMETRI	TIPO
MOF01	modifica_mst	mostra mst[], int ind_mst, int num_mostre, int selez, bool *elimina	void
MOF02	nuova_data	mostra mst[], int ind_mst	void
MOF03	modifica_opr	opera opr[], int ind_opr, int num_opere, int selez, bool *elimina	void
MOF04	modifica_utn	utente utn[], int ind_utn, int num_utenti, int selez, bool *elimina	void
MOF05	aggiungi_mostra	- - -	void
MOF06	aggiungi_opera	- - -	void
MOF07	aggiungi_utente	- - -	void

DESCRIZIONE	MOF01	Consente all'admin di modificare le informazioni di una mostra, e di salvare le modifiche nel file.
	MOF02	Funzione richiamata da "modifica_mst", serve per modificare la data della mostra in posizione ind_mst nell'array.
	MOF03	Consente all'admin di modificare le informazioni di un'opera d'arte, e di salvare le modifiche nel file.
	MOF04	Consente all'admin di modificare le informazioni di un utente a sua scelta, e di salvare le modifiche nel file. L'admin può anche modificare l'username (l'utente non può farlo, in base ad una specifica richiesta della traccia). Questa funzione è simile a "modifica_profilo", presente in un altro header, che permette invece all'utente di modificare le proprie credenziali.
	MOF05	Richiamata da "aggiungi_riga", permette all'admin di inserire nel file una nuova mostra.
	MOF06	Richiamata da "aggiungi_riga", permette all'admin di inserire nel file una nuova opera d'arte.
	MOF07	Richiamata da "aggiungi_riga", permette all'admin di inserire nel file un nuovo utente.

PRENOTAZIONE.H

CODICE	NOME	PARAMETRI	TIPO
PRE01	prenotazione	mostra mst[], utente utn[], char username[], int num_mostre	void
PRE02	inserisci_data	int data[]	void
PRE03	data_odierna	opera opr[], int ind_opr, int num_opere, int selez, bool *elimina	void
PRE04	controllo_mese	mostra mst[], int data[], int ind_mst, int *mese_inserito)	void
PRE05	controllo_giorno	mostra mst[], utente utn[], int data[], int ind_mst, int mese_inserito, int *giorno_inserito	void
PRE06	disdici_prenotazione	utente utn[], int ind_utn	void
PRE07	giorni_mensili	- - -	int
DESCRIZIONE	PRE01	Consente all'utente di prenotarsi a una mostra.	
	PRE02	Funzione richiamata da "prenotazione", chiede all'utente di inserire la data di prenotazione per la	

		mostra scelta. Si richiede prima il mese e poi il giorno.
	PRE03	Funzione che memorizza in un array di tre int, chiamato “data[3]”, la data corrente (formato giorno, mese, anno). È stata utilizzata la funzione time() presente nella libreria <time.h>.
	PRE04	In base al mese inserito dall’utente, si controlla che esso sia valido o no. Ad esempio, se l’utente inserisce come mese febbraio ma la mostra è aperta da marzo a giugno, si visualizza un messaggio d’errore. Inoltre se al momento dell’esecuzione del programma è luglio, non si può inserire un mese antecedente a luglio.
	PRE05	In base al giorno di prenotazione inserito dall’utente si verifica che esso sia valido o meno. Ad esempio, se l’utente inserisce come mese febbraio e come giorno 29 il programma restituisce un messaggio di errore (a meno che l’anno corrente non sia bisestile).
	PRE06	Consente all’utente di disdire una prenotazione effettuata. Impossibile disdire se mancano meno di sette giorni all’evento.
	PRE07	Funzione richiamata da “disdici_prenotazione” per stabilire da quanti giorni è composto il mese corrente, in modo tale da stabilire correttamente quanti giorni mancano all’evento nel caso in cui esso ricada nel mese successivo.

RICERCA.H

Le funzioni presenti in questo header consentono all’utente di cercare un’opera d’arte in base a determinati parametri, oppure di visualizzare informazioni sulle mostre.

CODICE	NOME	PARAMETRI	TIPO
RIC01	ricerca	mostra mst[], opera opr[], int num_mostre, int num_opere	void
RIC02	ricerca_opr	mostra mst[],opera opr[], int num_opere	void
RIC03	ricerca_nome	mostra mst[], opera opr[], int num_opere	void
RIC04	ricerca_autore	mostra mst[], opera opr[], int num_opere	void
RIC05	ricerca_tipo	mostra mst[], opera opr[], int num_opere	void

RIC06	ricerca_genere	mostra mst[], opera opr[], int num_opere utente utn[], int ind_utn	void
RIC07	ricerca_periodo	mostra mst[], opera opr[], int num_opere	void
RIC08	ricerca_anno	mostra mst[], opera opr[], int num_opere	void
RIC09	ricerca_mst	mostra mst[], opera opr[], int num_opere	void
RIC10	ricerca_google	opera opr[], int i	void

DESCRIZIONE	**RIC01**	Chiede all'utente se vuole cercare informazioni su un'opera oppure su una mostra.
	RIC02	Chiede all'utente in base a quale criterio vuole cercare un'opera.
	RIC03	Esegue la ricerca delle opere in base al loro nome. La ricerca non è case sensitive (cioè se l'opera si chiama "Gioconda" e l'utente cerca "GIOconDA" la ricerca ha comunque esito positivo), in modo tale da facilitare l'interazione tra l'utente e il programma. Non sono valide sottostringhe.
	RIC04	Esegue la ricerca delle opere in base al loro autore. Anche qui la ricerca non è case sensitive e non sono valide sottostringhe.
	RIC05	Esegue la ricerca delle opere in base al loro tipo. Anche qui la ricerca non è case sensitive e non sono valide sottostringhe.
	RIC06	Esegue la ricerca delle opere in base al loro genere. Anche qui la ricerca non è case sensitive e non sono valide sottostringhe.
	RIC07	Esegue la ricerca delle opere in base al loro periodo artistico. Anche qui la ricerca non è case sensitive e non sono valide sottostringhe.
	RIC08	Esegue la ricerca delle opere in base all'anno di produzione. Anche qui la ricerca non è case sensitive e non sono valide sottostringhe.
	RIC09	Fa visualizzare all'utente la lista di tutte le mostre memorizzate nell'array "mst" e le informazioni di quella visualizzata (compreso il responsabile della mostra e le opere in essa esposte).
	RIC10	Ogni volta che si ottiene un'opera a seguito delle funzioni di ricerca, si chiede all'utente se vuole avere più informazioni su quell'opera. Se acconsente, il programma apre il browser di default e cerca automaticamente su Google la stringa "nome opera + autore opera".

CRITT.H

Questo header contiene funzioni per il crypting e il decrypting delle password degli utenti memorizzate in "file_utenti.csv".

Ci sono migliaia di algoritmi di crypting diversi, ma abbiamo scelto il più semplice (Cifrario di Cesare, già affrontato durante le esercitazioni del corso) per ragioni di tempistiche, in quanto ci siamo concentrati su altre funzionalità del nostro programma.

Certo è che se il software fosse realmente destinato ad un'azienda avremmo adoperato algoritmi più complessi per tutelare la sicurezza degli utenti.

CODICE	NOME	PARAMETRI	TIPO
CRI01	critt	char password[]	int
CRI02	decritt	char password[]	void
DESCRIZIONE	CRI01	Funzione per crittare la password dell'utente, nel momento in cui egli si registra nel sistema. In questo modo il file contiene la password crittata, e non quella in chiaro, così chiunque abbia accesso al file deve conoscere la chiave di decrypting.	
	CRI02	Funzione per decrittare la password dell'utente. Serve principalmente quando egli vuole visualizzare le sue credenziali, e quindi deve vedere la sua password in chiaro.	

PRINCIPALI STRUTTURE DATI, FILE E VARIABILI

NOME	TIPO	DESCRIZIONE	CAMPI
mostra	Struct	Tipo di dato definito per descrivere le informazioni delle mostre.	char nome[] char luogo[] int giorno_inizio int mese_inizio int giorno_fine int mese_fine int max_partecipanti char responsabile
opera	Struct	Tipo di dato definito per descrivere le informazioni delle opere.	char nome[] char autore[] char tipo[] char genere[] char periodo[] int anno

			int codice
utente	Struct	Tipo di dato definito per descrivere le informazioni degli utenti.	char nome[] char cognome[] char username[] char password[] int prenotazione int giorno_prenot int mese_prenot
tm	Struct	Tipo di dato definito per descrivere una data.	int tm_sec int tm_min int tm_hour int tm_day int tm_mon int tm_year int tm_wday int tm_yday int tm_isdst

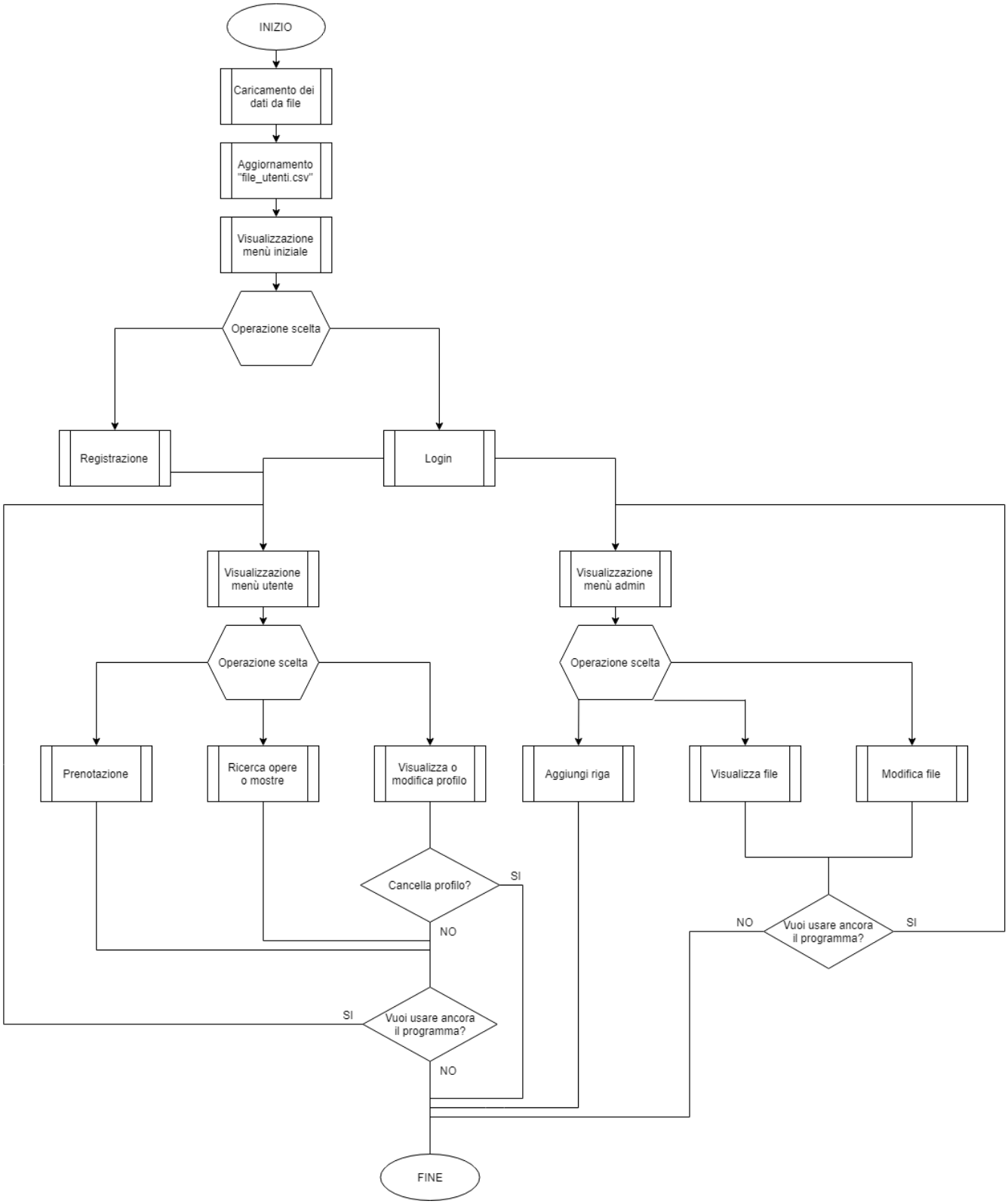
NOME	TIPOLOGIA	DESCRIZIONE	FORMATTAZIONE
file_mostre.csv	File	File contenente tutte le informa zioni delle mostre.	Nome; luogo; data di inizio esposizione; data di fine esposizione; massimo partecipanti; responsabile.
file_opere.csv	File	File contenente tutte le informa zioni delle opere.	Nome; autore; tipo; genere; periodo artistico; anno di produzione; codice mostra.
file_utenti.csv	File	File contenente tutte le informa zioni degli utenti.	Nome;cognome; username; password; mostra prenotata; data prenotazione.

NOME	TIPOLOGIA	DESCRIZIONE
pt_mostra	FILE *	Puntatore per “file_mostre.csv”.
pt_opera	FILE *	Puntatore per “file_opere.csv”.
pt_utente	FILE *	Puntatore per “file_utenti.csv”.
mst[]	Array di mostra	Array contenente tutte le informazioni delle mostre lette dal file.
opr[]	Array di opera	Array contenente tutte le informazioni delle opere lette dal file.
utn[]	Array di utente	Array contenente tutte le informazioni degli utenti lette dal file.
data[3]	Array di int	Contiene il giorno, il mese e l’anno corrente.

num_mostre	int	Variabile contenente il numero di mostre presenti nel file.
num_opere	int	Variabile contenente il numero di opere presenti nel file.
num_utenti	int	Variabile contenente il numero di utenti presenti nel file.
ind_mst	int	Variabile indice per l'array 'mst'.
ind_opr	int	Variabile indice per l'array 'opr'.
ind_utn	int	Variabile indice per l'array 'utn'.
ind	int	Variabile indice per un array generico.
selez	int	Variabile per la selezione delle operazioni presenti in un menù.
ripeti	int	Variabile per la ripetizione del menù iniziale a seguito dell'inserimento del valore '1' da parte dell'utente.
conta_err	int	Contatore di errori per il login.
nuova_data	int	Variabile in cui è memorizzata la nuova data di esposizione di una mostra.
no_scadenza	int	Variabile che indica se la nuova mostra inserita è disponibile sempre o no.
mese_inserito	int	Variabile che indica il mese di prenotazione inserito dall'utente.
giorno_inserito	int	Variabile che indica il giorno di prenotazione inserito dall'utente.
num_prenotati	int	Variabile che contiene il numero di prenotati a una mostra.
lung_mese	int	Variabile che contiene la lunghezza del mese corrente.
anno_cercato	int	Variabile in cui è contenuto il parametro di ricerca opera: anno.
mostra_selez	int	Variabile in cui è memorizzata la mostra di cui l'utente vuole leggere le informazioni.
info_opere	int	Variabile che indica se l'utente vuole vedere l'elenco delle opere di una mostra.
ch	char	Memorizza un singolo carattere.
username[]	array di char	Memorizza l'username dell'utente loggato.
password[]	array di char	Memorizza la password dell'utente loggato.
nome[]	array di char	Memorizza il nome dell'utente loggato.

cognome[]	array di char	Memorizza il cognome dell'utente loggato.
riga[]	array di char	Memorizza una riga di caratteri.
x_cercato[]	array di char	Memorizza il parametro cercato dell'opera inserito dall'utente.
x_opera[]	array di char	Memorizza il parametro dell'opera.
elimina	bool	Indica se l'admin deve eliminare una mostra/opera/utente dal database.
accesso_err	bool	Indica se c'è stato un errore di login.
regist_err	bool	Indica se c'è stato un errore di registrazione.
aggiorna	bool	Indica se "file_utenti.csv" dev'essere aggiornato.
data_err	bool	Indica se la data inserita dall'utente non è valida.
tempo	time_t	Parametro di time() e di localtime(), usata per ottenere la data odierna.
info_tempo	tm	Struttura in cui è memorizzata la data data odierna.

FLOWCHART


INIZIO
Caricamento dei dati da file
Aggiornamento "file_utenti.csv"
Visualizzazione menù iniziale
Operazione scelta
Registrazione
Login
Visualizzazione menù utente
Visualizzazione menù admin
Operazione scelta
Operazione scelta
Prenotazione
Ricerca opere o mostre
Visualizza o modifica profilo
Aggiungi riga
Visualizza file
Modifica file
Cancella profilo?
SI
NO
NO
Vuoi usare ancora il programma?
SI
SI
Vuoi usare ancora il programma?
NO
FINE

PSEUDOCODICE

Begin

```
{
  Call Caricamento dei dati da file;

  Call Aggiornamento “file_utenti.csv”;
  Call Visualizzazione menù iniziale;
  Read Operazione scelta;
  Switch (Operazione scelta);
  {
    Case 1:
    Call Registrazione;
    Break;


    Case 2;
    Call Login;
    Break,

  } //end_switch


  If (username = ‘admin’)
  {
    Do
    {
      Call Visualizzazione menù admin;
      Read Operazione scelta;
      Int ripeti = 0;
      Switch (Operazione scelta);
      {
      Case 1:
      Call Aggiungi riga;
      Break;

      Case 2:
      Call Visualizza file;
      Print “Vuoi usare ancora il programma? Se sì inserisci 1”;
      Read ripeti;

      Break;

      Case 3:
      Call Modifica file;
```

```
        Print "Vuoi usare ancora il programma? Se sì inserisci 1";

        Read ripeti;

        Break;
        } //end_switch
      } //end_do
      While (ripeti = 1);
    } //end_if
    Else
    {
      Do
      {
        Call Visualizzazione menù utente;
        Read Operazione scelta;
        Int ripeti = 0;
        Switch (Operazione scelta);
        {
        Case 1:
        Call Prenotazione;
        Print "Vuoi usare ancora il programma? Se sì inserisci 1";
        Read ripeti;

        Break;

        Case 2:
        Call Ricerca opere o mostre;
        Print "Vuoi usare ancora il programma? Se sì inserisci 1";

        Read ripeti;

        Break;

        Case 3:
        Call Visualizza o modifica profilo;

        If (!Cancella profilo)
        {
          Print "Vuoi usare ancora il programma? Se sì inserisci 1";
          Read ripeti;

        } //end_if
        Break;
        } //end_switch
      } //end_do
      While (ripeti = 1);
} //end_else
```

} //end_begin
End;

TESTING

CODICE FUNZIONE	NOME	TIPO TEST	ESEMPIO	RISULTATO
MST01	popola_mostre	File esistente	- - -	Caricamento dati
MST01	popola_mostre	File non esistente	- - -	Messaggio di errore
MST02	visualizza_mst	Array pieno	- - -	Visualizzazione mostra
MST02	visualizza_mst	Array vuoto	- - -	Se l'array è vuoto vuol dire che il file è vuoto, quindi messaggio di errore
OPR01	popola_opere	File esistente	- - -	Caricamento dati
OPR01	popola_opere	File non esistente	- - -	Messaggio di errore
OPR02	visualizza_opr	Array pieno	- - -	Visualizzazione opera
OPR02	visualizza_opr	Array vuoto	- - -	Se l'array è vuoto vuol dire che il file è vuoto, quindi messaggio di errore
UTN01	popola_utenti	File esistente	- - -	Caricamento dati
UTN01	popola_utenti	File non esistente	- - -	Messaggio di errore
UTN02	profilo	Utente loggato	- - -	Visualizzazione menù profilo
UTN02	profilo	Inserimento carattere	'a'	Programma va in loop
UTN03	visualizza_utn	Array pieno	- - -	Visualizzazione utente
UTN03	visualizza_utn	Array vuoto	- - -	Se l'array è vuoto vuol dire che il file è vuoto, quindi messaggio di errore
UTN04	modifica_profilo	Inserimento carattere	'a'	Programma va in loop
UTN05	cancella_profilo	- - -	- - -	Profilo utente eliminato
AVV01	avvio	Inserimento valore valido	'1'	Programma esegue l'opzione scelta dall'utente
AVV01	avvio	Inserimento valore non valido	spazio	Programma continua ad attendere l'input
AVV02	menu	Inserimento valore valido	'1'	Programma esegue l'opzione scelta dall' utente
AVV02	menu	Inserimento carattere	'a'	Programma va in loop
AVV03	accesso	Login riuscito	- - -	Programma apre il menù utente
AVV03	accesso	Troppi tentativi di login	- - -	Programma si chiude

AVV04	regist	Registrazione riuscita	- - -	Programma apre il menù utente
AVV04	regist	Inserimento username non valido	‘admin’	Programma continua a chiedere di inserire un altro username
AVV04	regist	Inserimento password troppo corta	‘prova’	Programma continua a chiedere di inserire un’altra password
ADM01	admin_menu	Inserimento valore valido	‘1’	Programma esegue l’operazione scelta dall’utente
ADM01	admin_menu	Inserimento carattere	‘a’	Programma va in loop
ADM02	file_backup	File di backup inesistenti	- - -	Vengono creati nuovi file di backup
ADM02	file_backup	File di backup già esistenti	- - -	I file di backup vengono sovrascritti
ADM03	admin_mostra	Inserimento valore valido	‘1’	Programma esegue l’operazione scelta dall’utente
ADM03	admin_mostra	Inserimento carattere	‘a’	Programma va in loop
ADM04	admin_opera	Inserimento valore valido	‘1’	Programma esegue l’operazione scelta dall’utente
ADM04	admin_opera	Inserimento carattere	‘a’	Programma va in loop
ADM05	admin_utente	Inserimento valore valido	‘1’	Programma esegue l’operazione scelta dall’utente
ADM05	admin_utente	Inserimento carattere	‘a’	Programma va in loop
ADM06	aggiungi_riga	Inserimento valore valido	‘1’	Programma esegue l’operazione scelta dall’utente
ADM06	aggiungi_riga	Inserimento carattere	‘a’	Programma va in loop
GEF01	conta_righe	File pieno	- - -	La funzione restituisce il numero di righe
GEF01	conta_righe	File vuoto	- - -	La funzione restituisce 1
GEF02	stampa_utn	array utenti pieno	- - -	Si riscrivono le informazioni di tutti gli utenti su “file_utenti.csv”
GEF02	stampa_utn	array utenti vuoto	- - -	“file_utenti.csv” si svuota, e quindi il programma presenterà dei bug
GEF03	aggiorna_file	data del dispositivo corretta	- - -	Il programma aggiornerà correttamente “file_utenti.csv”

GEF03	aggiorna_file	data del dispositivo errata	- - -	Il programma aggiornerà il file in modo errato
GEF04	visualizza_file	Inserimento valore valido	‘1’	Programma esegue l’operazione scelta dall’utente
GEF04	visualizza_file	Inserimento carattere	‘a’	Programma va in loop
MOF01	modifica_mst	Admin rispetta la formattazione di “file_mostre.csv”	- - -	Le modifiche non portano a problemi per il funzionamento del programma (bug)
MOF01	modifica_mst	Admin non rispetta la formattazione di “file_mostre.csv”	- - -	Le modifiche portano a problemi per il funzionamento del programma (bug)
MOF02	nuova_data	Mostra sempre disponibile	‘1’	In “file_mostre.csv” i campi delle date sono formattati a 0
MOF02	nuova_data	Mostra disponibile in un intervallo di tempo	‘0’	L’admin sceglie le date di inizio e di fine esposizione
MOF03	modifica_opr	Admin rispetta la formattazione di “file_opere.csv”	- - -	Le modifiche non portano a problemi per il funzionamento del programma (bug)
MOF03	modifica_opr	Admin non rispetta la formattazione di “file_opere.csv”	- - -	Le modifiche portano a problemi per il funzionamento del programma (bug)
MOF04	modifica_utn	Admin rispetta la formattazione di “file_utenti.csv”	- - -	Le modifiche non portano a problemi per il funzionamento del programma (bug)
MOF04	modifica_utn	Admin non rispetta la formattazione di “file_mostre.csv”	- - -	Le modifiche portano a problemi per il funzionamento del programma (bug)
MOF05	aggiungi_mostra	Stesso testing di MOF01		
MOF06	aggiungi_opera	Stesso testing di MOF03		
MOF07	aggiungi_utente	Stesso testing di MOF04		
PRE01	prenotazione	Inserimento valore valido	‘1’	Programma esegue l’operazione scelta dall’utente
PRE01	prenotazione	Inserimento carattere	‘a’	Programma va in loop
PRE02	inserisci_data	Inserimento data valida	mese: 8; giorno: 10	L’utente si prenota correttamente

PRE02	**inserisci_data**	**Inserimento carattere**	**‘a’**	**Programma va in loop**
PRE03	**data_odierna**	**Data del dispositivo corretta**	**- - -**	**Programma funziona correttamente**
PRE03	**data_odierna**	**Data del dispositivo errata**	**- - -**	**Bug per la prenotazione**
PRE04	**controllo_mese**	**Inserimento mese valido**	**‘5’**	**Mese accettato**
PRE04	**controllo_mese**	**Inserimento mese non valido**	**‘13’**	**Mese non accettato**
PRE04	**controllo_mese**	**Inserimento carattere**	**‘a’**	**Programma va in loop**
PRE05	**controllo_giorno**	**Inserimento giorno valido**	**‘10’**	**Giorno accettato**
PRE05	**controllo_giorno**	**Inserimento giorno non valido**	**‘32’**	**Giorno non accettato**
PRE05	**controllo_giorno**	**Inserimento carattere**	**‘a’**	**Programma va in loop**
PRE06	**disdici_prenotazione**	**Utente prenotato**	**- - -**	**Prenotazione cancellata**
PRE06	**disdici_prenotazione**	**Utente non prenotato**	**- - -**	**Messaggio di errore**
PRE06	**disdici_prenotazione**	**Utente prenotato**	**- - -**	**Messaggio di errore a meno di sette giorni dalla mostra**
PRE07	**giorni_mensili**	**Data del dispositivo corretta**	**- - -**	**La funzione restituisce la lunghezza del mese corrente**
PRE07	**giorni_mensili**	**Data del dispositivo errata**	**- - -**	**La funzione può restituire la lunghezza di un mese diverso da quello corrente**
RIC01	**ricerca**	**Inserimento valore valido**	**‘1’**	**Programma esegue l’operazione scelta dall’utente**
RIC01	**ricerca Inserimento**	**carattere**	**‘a’**	**Programma va in loop**
RIC02	**ricerca_opr**	**Stesso testing di RIC01**		
RIC03	**ricerca_nome**	**Ripetizione ricerca**	**‘1’**	**L’utente ricerca nuovamente in base allo stesso parametro**
RIC03	**ricerca_nome**	**Inserimento carattere**	**‘a’**	**Crash del programma**
RIC04	**ricerca_autore**	**Stesso testing di RIC03**		
RIC05	**ricerca_tipo**	**Stesso testing di RIC03**		
RIC06	**ricerca_genere**	**Stesso testing di RIC03**		
RIC07	**ricerca_periodo**	**Stesso testing di RIC03**		

RIC08	ricerca_anno	Inserimento di un valore valido	'1504'	Il programma mostra all'utente le opere prodotte nell'anno inserito
RIC08	ricerca_anno	Inserimento di un carattere	'a'	Crash del programma
RIC09	ricerca_mst	Inserimento valore valido	'1'	Programma esegue l'operazione scelta dall'utente
RIC09	ricerca_mst	Inserimento di un carattere	'a'	Crash del programma
RIC09	ricerca_mst	Array vuoto	- - -	Non viene visualizzata nessuna mostra
RIC10	ricerca_google	Utente vuole cercare su Google	'1'	La ricerca avviene
RIC10	ricerca_google	Utente non vuole cercare su Google	'2'	La ricerca non avviene
RIC10	ricerca_google	Inserimento di un carattere	'a'	Crash del programma
CRI01	critt	Carattere valido	't' (116)	Si ottiene il carattere successivo, cioè 'u' (117)
CRI01	critt	Ultimo simbolo ASCII	255	Sul file si stampano caratteri speciali
CRI01	critt	Simboli ASCII non stampabili	Dallo 00 al 30; 126	Sul file si stampano caratteri speciali
CRI02	decritt	Carattere valido	'u' (117)	Si ottiene il carattere precedente, cioè 't' (116)
CRI02	decritt	Primo simbolo ASCII	00	Sul file si stampano caratteri speciali
CRI02	decritt	Simboli ASCII non stampabili	Da 01 a 32; 128	Sul file si stampano caratteri speciali

ULTIME CONSIDERAZIONI

Il programma può essere migliorato aggiungendo ulteriori controlli sugli input, in particolare quando l'utente inserisce dei caratteri alfanumerici che (come già osservato nella fase di testing) causano crash e bug vari.

Inoltre si potrebbe creare una piattaforma online per consentire agli utenti di registrarsi e/o di accedere in remoto, con le dovute conoscenze del linguaggio HTML.

A causa della funzione di ricerca su Google, il software potrebbe essere bloccato dal firewall di Windows perché è riconosciuto come un virus. Per evitare ciò, si può inserire il file eseguibile nella whitelist del firewall, in modo tale da non essere considerato come un malware.

Ulteriore problema è la semplicità del Cifrario di cesare. Come già scritto in precedenza, se il software fosse destinato a un'azienda si sarebbero utilizzati algoritmi di crypting più complessi.

Da notare l'inserimento dei doc -comments all'interno degli header.