

## Caso di Studio: Traccia 3

# Negozio online di videogiochi

Laboratorio di Informatica  
a.a. 2016/2017

Autore:  
Walter MAIO

---

## 1. Analisi

Il sistema realizzato simula alcune funzionalità di uno store di videogiochi (Steam, Playstore, Appstore).

Nello specifico, il software dovrà permettere di gestire con efficienza l'inserimento di giocatori sulla piattaforma e la gestione di essi attraverso alcuni dati quali il **nickname** (che sarà univoco per ogni giocatore registrato sulla piattaforma),

l'**e-mail** associata al giocatore (anch'essa sarà univoca per ogni giocatore), i *dati anagrafici* del giocatore come il **cognome**, il **nome** ed il **sesso**, inoltre il sistema predisponde la memorizzazione del **livello** di un giocatore, delle **ore di gioco** effettuate da parte di esso e del **credito** attualmente in possesso.

Il livello ed il credito di un giocatore sono determinati dal totale delle ore che hanno trascorso sulla piattaforma di gioco, il credito può anche essere aggiunto tramite un mezzo di pagamento alla registrazione di un nuovo giocatore o durante l'esecuzione del programma.

Oltre alle funzionalità di base che sono: l'aggiunta di un nuovo giocatore sulla piattaforma, la modifica dei dati di un giocatore già presente (la modifica non è possibile sul nickname), l'aggiunta di un videogioco ad un giocatore tramite l'utilizzo dei crediti, l'aggiunta del credito in euro al giocatore sotto forma di numero decimale, la visualizzazione della lista dei videogiochi che un giocatore possiede, l'incremento delle ore di gioco di un giocatore (il sistema accetta solo le ore intere e non comprensive di minuti), la visualizzazione dei giocatori presenti sulla piattaforma ordinati decrescentemente in base al livello del giocatore, la ricerca di un videogioco nello store tramite l'inserimento di un codice chiave di ricerca ;  
il sistema permette inoltre come funzionalità aggiuntive la visualizzazione di tutti i dati dei giocatori che sono presenti sulla piattaforma fino a quel dato momento,  
la visualizzazione di tutti i giochi che sono presenti sulla piattaforma e la stampa su file della libreria dei videogiochi che un utente possiede.

### 1.2.1 Requisiti Funzionali

codice	nome	descrizione
00	Caricamento dei dati dai file	Il programma deve caricare i dati dei giocatori, dei videogiochi e delle relazioni dai file
01	Visualizzazione del Menù di scelta	Il programma deve visualizzare all'utente un menù per inserire la scelta desiderata
02	Acquisizione dei dati di un nuovo giocatore	Il programma deve acquisire i dati relativi al nuovo giocatore
03	Verifica nickname presente	Il programma deve verificare che il nickname non sia presente sul database
04	Verifica e-mail valida	Il programma deve verificare che l'e-mail inserita dall'utente sia valida
05	Verifica e-mail presente	Il programma deve verificare che l'e-mail inserita dall'utente non sia presente nel database
06	Modifica dei dati di un giocatore	Il programma deve modificare i dati relativi ad un giocatore presente
07	Aggiunta credito al giocatore	Il programma deve aggiungere del credito ad un giocatore
08	Aggiunta ore di gioco al giocatore	Il programma deve aggiungere delle ore di gioco ad un giocatore
09	Verifica incremento del livello	Il programma deve verificare se può o non può incrementare il livello
10	Ordinamento per livello del giocatore	Il programma deve visualizzare i dati dei giocatori per livello in ordine decrescente

## 1.2.2 Requisiti Funzionali

codice	nome	descrizione
11	Stampa dei giocatori presenti nel database	Il programma deve visualizzare tutti i giocatori presenti nel database
12	Visualizza la lista dei giochi di un giocatore	Il programma deve visualizzare tutti i giochi di un giocatore
13	Ricerca del videogioco nel database	Il programma deve verificare se il videogioco è presente o no nel database
14	Aggiunta gioco ad un giocatore	Il programma deve aggiungere un gioco ad un giocatore se possibile
15	Stampa su file tutti i dati dei giocatori	Il programma deve scrivere su file tutti i dati che riguardano i giocatori
16	Stampa su file tutte le relazioni	Il programma deve scrivere su file tutte le relazioni che ci sono tra un giocatore e un gioco
17	Stampa su file tutti i giochi di un giocatore	Il programma deve scrivere su file la libreria dei giochi di un giocatore
18	Modifica dei colori	La procedura permette di modificare i colori delle stampe a video

### 1.3.1 Casi d'Uso

codice	nome	descrizione
00	Caricamento dei dati dai file	Il programma deve caricare i dati dei giocatori e dei videogiochi dai file
Pre-Condizioni		Il file deve essere valido: - formato .csv - deve contenere almeno un giocatore e un videogioco
Post-Condizioni	Successo	Il sistema carica i dati presenti sui file
	Fallimento	Il sistema mostra un messaggio di errore sul caricamento
Evento innescante	L'utente deve aver aperto il software	
Scenario di Base	Il funzionamento del programma è quello atteso, i dati vengono prelevati dai file.	
Scenario Alternativo	Il funzionamento del programma non è quello atteso, non viene prelevato alcun dato.	

### 1.3.2 Casi d'Uso

codice	nome	descrizione
<b>01</b>	Visualizzazione del Menù di scelta	Il programma deve visualizzare all'utente un Menù per inserire la scelta desiderata
<b>Pre-Condizioni</b>	Le informazioni lette dai file devono essere valide:	
<b>Post-Condizioni</b>	Successo	Il sistema mostra a video il Menù di scelta
	Fallimento	Il sistema non mostra nulla a video
<b>Evento innescante</b>	L'utente deve aver aperto il software ed il requisito "00" deve essere andato a buon fine	
<b>Scenario di Base</b>	Il funzionamento del programma è quello atteso, il Menù viene mostrato all'utente.	
<b>Scenario Alternativo</b>	Il funzionamento del programma non è quello atteso, non viene mostrato alcun Menù, il programma si blocca.	

### 1.3.3 Casi d'Uso

codice	nome	descrizione
02	Acquisizione dei dati di un nuovo giocatore	Il programma deve acquisire i dati relativi al nuovo giocatore
Pre-Condizioni	Affinchè l'operazione vada a buon fine, i dati inseriti devono essere validi	
Post-Condizioni	Successo	Il sistema inserisce il nuovo giocatore nel database
	Fallimento	Il sistema mostra un messaggio di errore
Evento innescante	L'utente deve aver effettuato la scelta corretta: "Inserisci un nuovo giocatore"	
Scenario di Base	Il funzionamento del programma è quello atteso, il giocatore entra a far parte della community.	
Scenario Alternativo	Il funzionamento del programma non è quello atteso, il giocatore non viene accettato e viene mostrato un messaggio di errore.	

### 1.3.4 Casi d'Uso

codice	nome	descrizione
03	Verifica nickname presente	Il programma deve verificare che il nickname sia presente sul database o no
Pre-Condizioni	Affinchè l'operazione vada a buon fine, bisogna inserire un nickname in input	
Post-Condizioni	Successo	Il sistema accetta il nickname inserito in input e va avanti
	Fallimento	Il sistema mostra un messaggio di nickname non trovato / non disponibile
Evento innescante	L'utente deve aver effettuato una scelta che riguarda il confronto del nickname	
Scenario di Base	Il nickname risulta disponibile e si può procedere con le operazioni: 02, 07, 08	
Scenario Alternativo	il nickname non viene accettato e viene mostrato un messaggio di errore.	

### 1.3.5 Casi d'Uso

codice	nome	descrizione
<b>04</b>	Verifica e-mail valida	Il programma deve verificare che l'e-mail inserita dall'utente sia valida
<b>Pre-Condizioni</b>	Affinchè l'operazione vada a buon fine, bisogna inserire un e-mail in input	
<b>Post-Condizioni</b>	Successo	Il sistema accetta l'e-mail inserita in input e va avanti
	Fallimento	Il sistema mostra un messaggio di e-mail non valida
<b>Evento innescante</b>	L'utente deve aver effettuato una scelta che riguarda l'inserimento dell'e-mail	
<b>Scenario di Base</b>	L'e-mail risulta valida e si può procedere con le operazioni: 02, 06	
<b>Scenario Alternativo</b>	L'e-mail non viene accettata e viene mostrato un messaggio di errore.	

### 1.3.6 Casi d'Uso

codice	nome	descrizione
05	Verifica e-mail presente	Il programma deve verificare che l'e-mail inserita dall'utente non sia presente nel database
Pre-Condizioni	Affinchè l'operazione vada a buon fine, bisogna inserire un e-mail in input	
Post-Condizioni	Successo	Il sistema accetta l'e-mail inserita in input e va avanti
	Fallimento	Il sistema mostra un messaggio di e-mail non disponibile
Evento innescante	L'utente deve aver effettuato una scelta che riguarda l'inserimento dell'e-mail	
Scenario di Base	L'e-mail risulta disponibile e si può procedere con le operazioni: 02, 06	
Scenario Alternativo	L'e-mail non viene accettata e viene mostrato un messaggio di errore.	

### 1.3.7 Casi d'Uso

codice	nome	descrizione
<b>06</b>	Modifica dei dati di un giocatore	Il programma deve modificare i dati relativi ad un giocatore presente
<b>Pre-Condizioni</b>	Affinchè l'operazione vada a buon fine, bisogna inserire un nickname valido in input	
<b>Post-Condizioni</b>	Successo	Il sistema procede alla modifica dei dati del giocatore
	Fallimento	Il sistema mostra un messaggio di errore
<b>Evento innescante</b>	L'utente deve aver effettuato la scelta corretta per la modifica dei dati del giocatore	
<b>Scenario di Base</b>	Si modificano i dati inerenti al giocatore, tutti eccetto il nickname che è permanentemente una volta inserito	
<b>Scenario Alternativo</b>	Il nickname non viene accettato e viene mostrato un messaggio di errore.	

### 1.3.8 Casi d'Uso

codice	nome	descrizione
07	Aggiunta credito al giocatore	Il programma deve aggiungere del credito ad un giocatore
Pre-Condizioni	Affinchè l'operazione vada a buon fine, bisogna inserire un nickname e il credito validi in input	
Post-Condizioni	Successo	Il sistema procede alla modifica del credito del giocatore
	Fallimento	Il sistema mostra un messaggio di errore
Evento innescante	L'utente deve aver effettuato la scelta corretta per la modifica del credito del giocatore	
Scenario di Base	Viene modificato il credito in possesso al giocatore	
Scenario Alternativo	Il credito inserito e' errato e non viene accettato, viene richiesto l'inserimento	

### 1.3.9 Casi d'Uso

codice	nome	descrizione
<b>08</b>	Aggiunta ore di gioco al giocatore	Il programma deve aggiungere delle ore di gioco ad un giocatore
<b>Pre-Condizioni</b>	Affinchè l'operazione vada a buon fine, bisogna inserire un nickname e delle ore validi in input	
<b>Post-Condizioni</b>	Successo	Il sistema procede alla modifica delle ore di gioco del giocatore
	Fallimento	Il sistema mostra un messaggio di errore
<b>Evento innescante</b>	L'utente deve aver effettuato la scelta corretta per la modifica delle ore di gioco del giocatore	
<b>Scenario di Base</b>	Vengono modificate le ore di gioco totali del giocatore	
<b>Scenario Alternativo</b>	Le ore inserite sono errate e non vengono accettate, viene richiesto l'inserimento	

### 1.3.10 Casi d'Uso

codice	nome	descrizione
09	Verifica incremento del livello	Il programma deve verificare se può o non può incrementare il livello
Pre-Condizioni	Affinchè l'operazione vada a buon fine, bisogna inserire delle ore valide in input	
Post-Condizioni	Successo	Il sistema procede alla modifica del livello del giocatore
	Fallimento	Il sistema ritorna il livello posseduto dal giocatore in precedenza
Evento innescante	L'utente deve aver inserito delle ore di gioco al giocatore	
Scenario di Base	Viene modificato il livello del giocatore se le ore sono necessarie all'incremento	
Scenario Alternativo	Il livello non viene incrementato perchè le ore raggiunte non sono sufficienti viene ritornato il livello in precedenza	

### 1.3.11 Casi d'Uso

codice	nome	descrizione
10	Ordinamento per livello del giocatore	Il programma deve visualizzare i dati dei giocatori per livello in ordine decrescente
Pre-Condizioni	Affinchè l'operazione vada a buon fine, bisogna inserire la scelta valida dal menù	
Post-Condizioni	Successo	Il sistema visualizza i giocatori dal livello maggiore al minore
	Fallimento	-
Evento innescante	Scelta da parte dell'utente	
Scenario di Base	Vengono mostrati i dati dei giocatori ordinati dal livello maggiore a quello minore	
Scenario Alternativo	Se non ci sono giocatori, la lista sarà vuota	

## 1.3.12 Casi d'Uso

codice	nome	descrizione
11	Stampa dei giocatori presenti nel database	Il programma deve visualizzare tutti i giocatori presenti nel database
<b>Pre-Condizioni</b>	Affinchè l'operazione vada a buon fine, bisogna inserire la scelta valida dal menù	
<b>Post-Condizioni</b>	Successo	Il sistema procede alla visualizzazione dei dati di tutti i giocatori
	Fallimento	-
<b>Evento innescante</b>	Scelta da parte dell'utente	
<b>Scenario di Base</b>	Vengono mostrati tutti i dati dei giocatori presenti sul database	
<b>Scenario Alternativo</b>	Se non ci sono giocatori, la lista sarà vuota	

### 1.3.13 Casi d'Uso

codice	nome	descrizione
12	Visualizza la lista dei giochi di un giocatore	Il programma deve visualizzare tutti i giochi di un giocatore
Pre-Condizioni	L'utente deve aver effettuato la scelta valida dal menù e deve aver inserito un nickname valido	
Post-Condizioni	Successo	Il sistema visualizza la lista dei giochi del giocatore
	Fallimento	Il sistema richiede l'inserimento
Evento innescante	L'utente deve aver effettuato la scelta valida dal menù	
Scenario di Base	Il programma mostra la libreria dei giochi di un giocatore	
Scenario Alternativo	La lista sarà vuota	

## 1.3.14 Casi d'Uso

codice	nome	descrizione
13	Ricerca del videogioco nel database	Il programma deve verificare se il videogioco è presente o no nel database
<b>Pre-Condizioni</b>	L'utente deve aver effettuato la scelta valida dal menù e deve aver inserito un codice di gioco valido	
<b>Post-Condizioni</b>	Successo	Il sistema visualizza i dati del videogioco
	Fallimento	Il sistema richiede l'inserimento
<b>Evento innescante</b>	L'utente deve aver effettuato la scelta valida dal menù	
<b>Scenario di Base</b>	Il programma mostra i dati di un videogioco	
<b>Scenario Alternativo</b>	-	

### 1.3.15 Casi d'Uso

codice	nome	descrizione
<b>14</b>	Aggiunta gioco ad un giocatore	Il programma deve aggiungere un gioco ad un giocatore se possibile
<b>Pre-Condizioni</b>	Inserimento di un codice valido per il videogioco	
<b>Post-Condizioni</b>	Successo	Il sistema aggiunge il videogioco alla libreria del giocatore
	Fallimento	Il sistema richiede l'inserimento
<b>Evento innescante</b>	L'utente deve aver effettuato la scelta valida dal menù	
<b>Scenario di Base</b>	Il programma permette di visualizzare la lista dei giochi e aggiunge un videogioco al giocatore	
<b>Scenario Alternativo</b>	Il codice di gioco è errato, richiede l'inserimento	

### 1.3.16 Casi d'Uso

codice	nome	descrizione
15	Stampa su file tutti i dati dei giocatori	Il programma deve scrivere su file tutti i dati che riguardano i giocatori
Pre-Condizioni	Il file deve essere correttamente aperto	
Post-Condizioni	Successo	Il sistema scrive su file i dati dei giocatori
	Fallimento	Il sistema mostra un messaggio di errore
Evento innescante	L'utente deve aver effettuato una operazione di manipolazione dei dati su un giocatore: 02, 06, 07, 08, 14	
Scenario di Base	Il programma aggiunge al file nuovi giocatori	
Scenario Alternativo	Messaggio di errore sul file	

### 1.3.17 Casi d'Uso

codice	nome	descrizione
<b>16</b>	<b>Stampa su file tutte le relazioni</b>	Il programma deve scrivere su file tutte le relazioni che ci sono tra un giocatore e un gioco
<b>Pre-Condizioni</b>	Il file deve essere correttamente aperto	
<b>Post-Condizioni</b>	Successo	Il sistema scrive su file le relazioni tra un videogioco e un giocatore
	Fallimento	Il sistema mostra un messaggio di errore
<b>Evento innescante</b>	L'utente deve aver effettuato l'aggiunta di un videogioco ad un giocatore: 14	
<b>Scenario di Base</b>	Il programma aggiunge al file delle nuove relazioni	
<b>Scenario Alternativo</b>	Messaggio di errore sul file	

### 1.3.18 Casi d'Uso

codice	nome	descrizione
17	Stampa su file tutti i giochi di un giocatore	Il programma deve scrivere su file la libreria dei giochi di un giocatore
Pre-Condizioni	Il file deve essere correttamente aperto	
Post-Condizioni	Successo	Il sistema scrive su file la libreria dei giochi di un giocatore
	Fallimento	Il sistema mostra un messaggio di errore
Evento innescante	L'utente deve aver effettuato la scelta valida dal menù	
Scenario di Base	Il programma aggiunge al file la libreria dei videogiochi di un giocatore	
Scenario Alternativo	Messaggio di errore sul file	

### 1.3.19 Casi d'Uso

codice	nome	descrizione
18	Modifica dei colori	La procedura permette di modificare i colori delle stampe a video
<b>Pre-Condizioni</b>	Libreria importata correttamente	
<b>Post-Condizioni</b>	Successo	Il sistema cambia il colore del testo
	Fallimento	-
<b>Evento innescante</b>	Richiamo della procedura	
<b>Scenario di Base</b>	Il programma modifica il colore del testo visualizzato	
<b>Scenario Alternativo</b>	-	

## **1.4 Strumenti di Sviluppo**

Il sistema è stato realizzato utilizzando un Notebook Asus N552VW con processore i7 - 2.6 GHz - 16 GB di RAM.

Il Software è stato scritto in linguaggio C.

L'ambiente di sviluppo adottato è stato Eclipse con il plug-in per Doxygen.

Per utilizzare la procedura di verifica dell'email si è fatto riferimento a del materiale aggiuntivo reperibile qui :

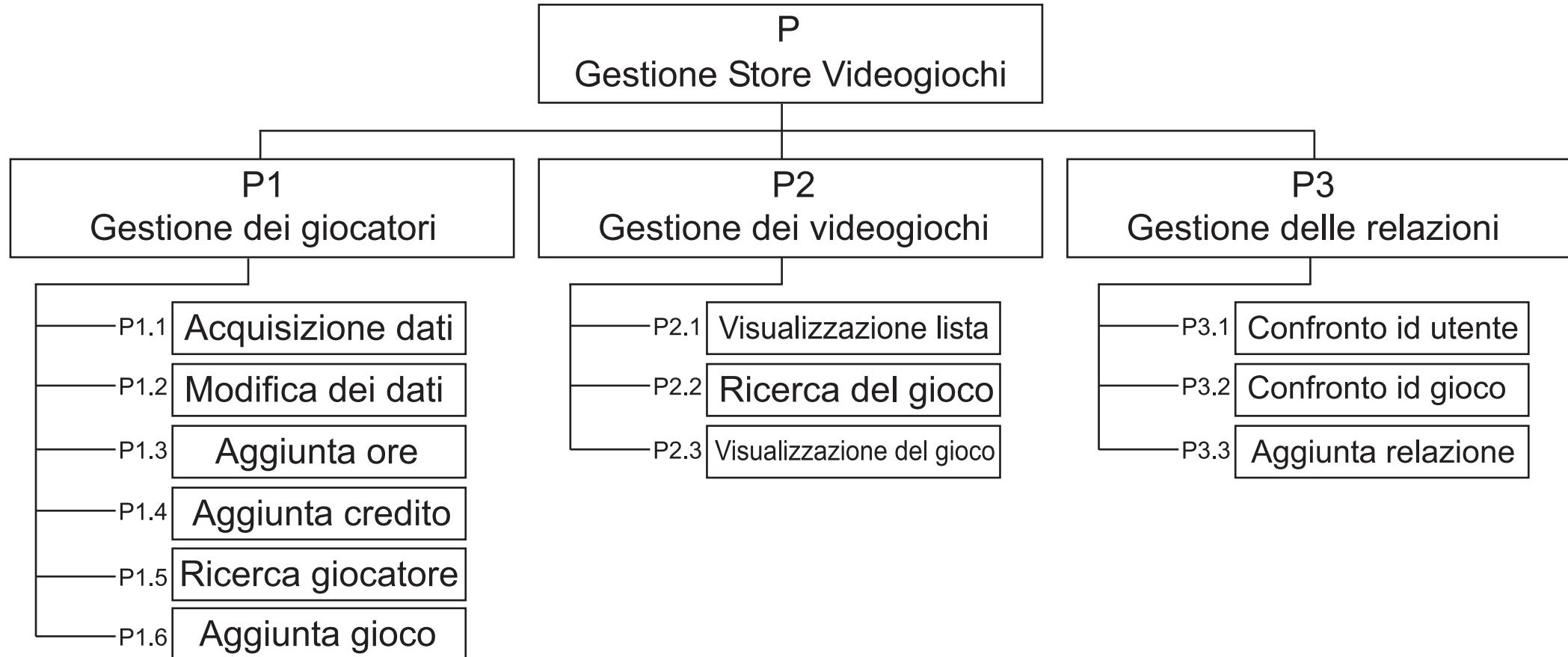
<https://stackoverflow.com/questions/2948043/regular-exp-to-validate-email-in-c>

Inoltre per la modifica del colore del testo sono state utilizzate le librerie <conio.h> e <windows.h> messe a disposizione dal linguaggio stesso.

Per eseguire il Software correttamente tramite l'eseguibile bisogna utilizzare macchine con Sistema Operativo da Windows xp in poi fino ad oggi.

Per eseguire il Software su altre macchine si ha la necessità di ricompilare l'intera unità con un compilatore adatto alla macchina di cui si sta facendo uso.

## 2. Progettazione



## 2.1 Progettazione dei tipi di dato e delle strutture dati

nome	tipologia	descrizione	tipi/campi/valori
gamer	struct	tipo di dato definito per descrivere gli attributi del giocatore	nickname: <b>char nickname[25]</b> e-mail: <b>char email[25]</b> cognome: <b>char surname[20]</b> nome: <b>char name[20]</b> sesto: <b>char gender</b> ore di gioco: <b>int played</b> livello: <b>int level</b> credito: <b>float credit</b>

nome	tipologia	descrizione	tipi/campi/valori
game	struct	tipo di dato definito per descrivere gli attributi dei videogiochi	codice gioco: <b>char code[25]</b> nome del gioco: <b>char name[35]</b> casa produttrice: <b>char production[50]</b> categoria: <b>char category[25]</b> prezzo del gioco: <b>float price</b> età minima di gioco: <b>int agemin</b>

nome	tipologia	descrizione	tipi/campi/valori
owned	struct	tipo di dato definito per associare un videogioco ad un giocatore	nome dell'utente <b>char iduser[25]</b> nome del gioco: <b>char idgame[35]</b>

## 2.1.1 Progettazione dei tipi di dato e delle strutture dati

nome	tipologia	descrizione	tipi/campi/valori
MAX	costante	tipo di dato definito per stabilire la dimensione degli array dei giocatori e dei videogiochi	#define MAX 100
users.csv	file	file utilizzato per salvare tutti i dati dei giocatori	#define FILEUSERS "users.csv"
store.csv	file	file utilizzato per acquisire tutti i dati dei videogiochi	#define FILEGAMES "store.csv"
relat.csv	file	file utilizzato per salvare tutte le relazioni videogioco-giocatore	#define FILERELAT "relat.csv"
games.csv	file	file utilizzato per salvare la libreria di un giocatore	#define FILEGAMELIST "games.csv"

## 2.2 Progettazione delle librerie/ funzioni

Sono state progettate due librerie:

## users.h utility.h

**Le funzioni/procedure utilizzate nella libreria users.h sono le seguenti:**

```
void newUser(gamer gamers[], int* contUs);

void modUser(gamer gamers[], int contUs);

void addCredit(gamer gamers[], int contUs);

void addHours(gamer gamers[], int contUs);

int searchUser(gamer gamers[], int contUs, char* nickname);

void addGame(gamer gamers[], game store[], owned relat[], int* relat_len, int contUs, int contGa);

int buyGame(gamer* pplayer, game app, owned relat[], int* relat_len, float priceG);
```

**Le funzioni/procedure utilizzate nella libreria utility.h sono le seguenti:**

```
int searchMail(gamer gamers[], int contUs, char* email);

int verEmail(char* email);

int levelCheck(int played, int levelOld);

void ordLevel(gamer gamers[], int contUs);

void gameListU(gamer gamers[], game store[], owned relat[], int relat_len, int contUs, int contGa);

int searchGame(game store[], int contGa, char* codeGa);

void searchGameStore(game store[], int contGa);

void printUser(gamer gamers[], int i);

void printGame(game store[], int i);

void printFile(gamer gamers[], int contUs);

void printFileRelat(owned relat[], int relat_len);

void printFileGamesUser(gamer gamers[], game store[], owned relat[], int relat_len, int contUs, int contGa);

void SetColor(short);
```

*Le librerie sono state create secondo il criterio per cui si acquisiscono e gestiscono dei dati di utenti in "users.h", e si effettuano delle operazioni su questi dati tramite alcune strategie in "utility.h". Per ulteriori informazioni sulle librerie consultare la documentazione Doxygen*

## 2.3 Dipendenza tra funzioni

Le procedure di aggiunta di un nuovo giocatore o modifica dei suoi dati devono accertarsi che i dati non siano già presenti, per questo sono state implementate le seguenti funzioni/procedure.

```
int searchUser(gamer gamers[], int contUs, char* nickname);
int searchMail(gamer gamers[], int contUs, char* email);
int verEmail(char* email);
```

La procedura di aggiunta di un videogioco al giocatore si basa su due fattori, un'interfaccia con l'utente per l'inserimento del nickname del giocatore e del codice del gioco addGame() ed un successivo controllo da parte della funzione buyGame() che verifica se il giocatore può comprare o no il gioco.

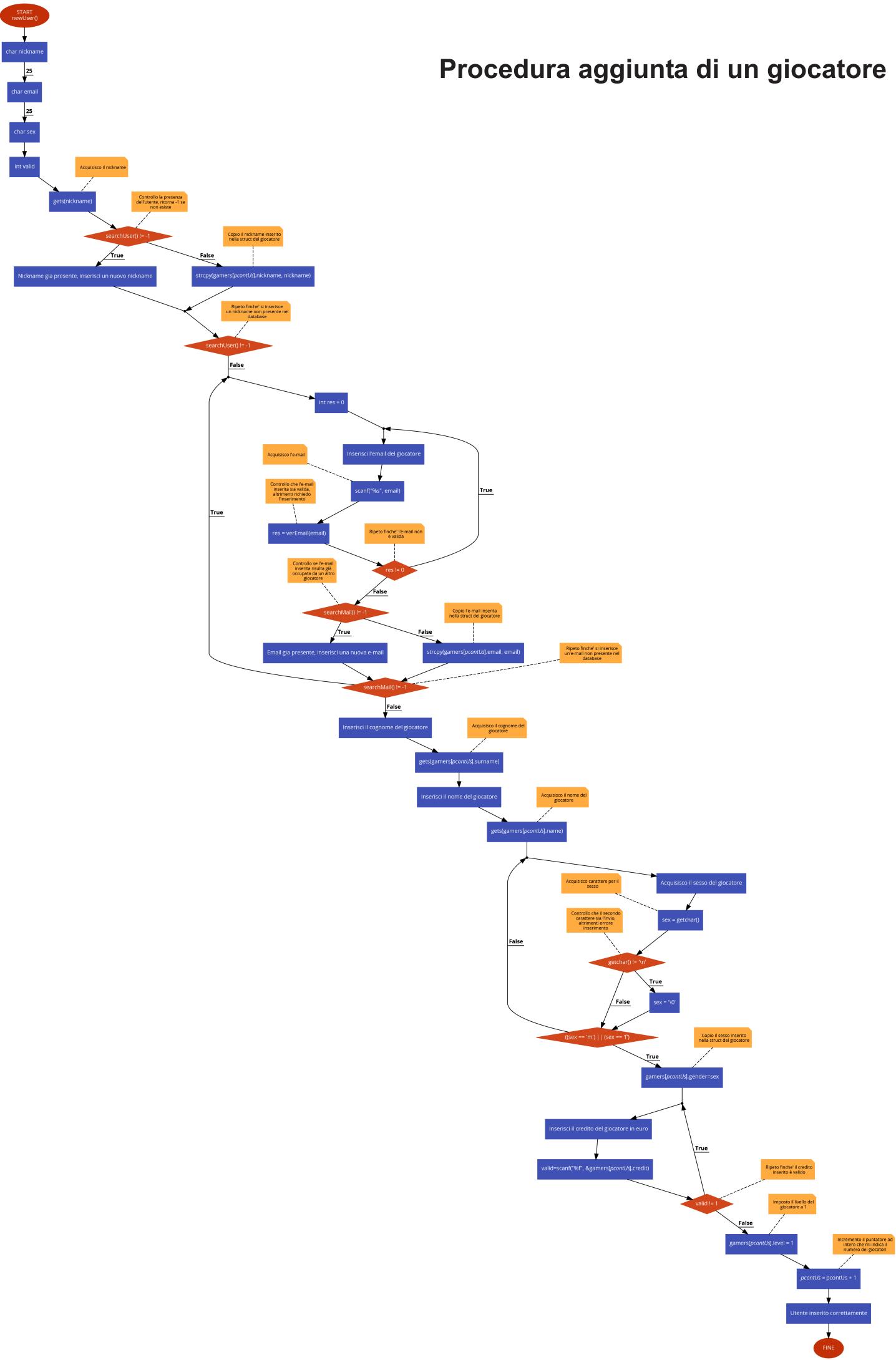
```
void addGame(gamer gamers[], game store[], owned relat[], int* relat_len, int contUs, int contGa);
int buyGame(gamer* pplayer, game app, owned relat[], int* relat_len, float priceG);
```

Esse dipendono l'una dall'altra per l'effettivo funzionamento.

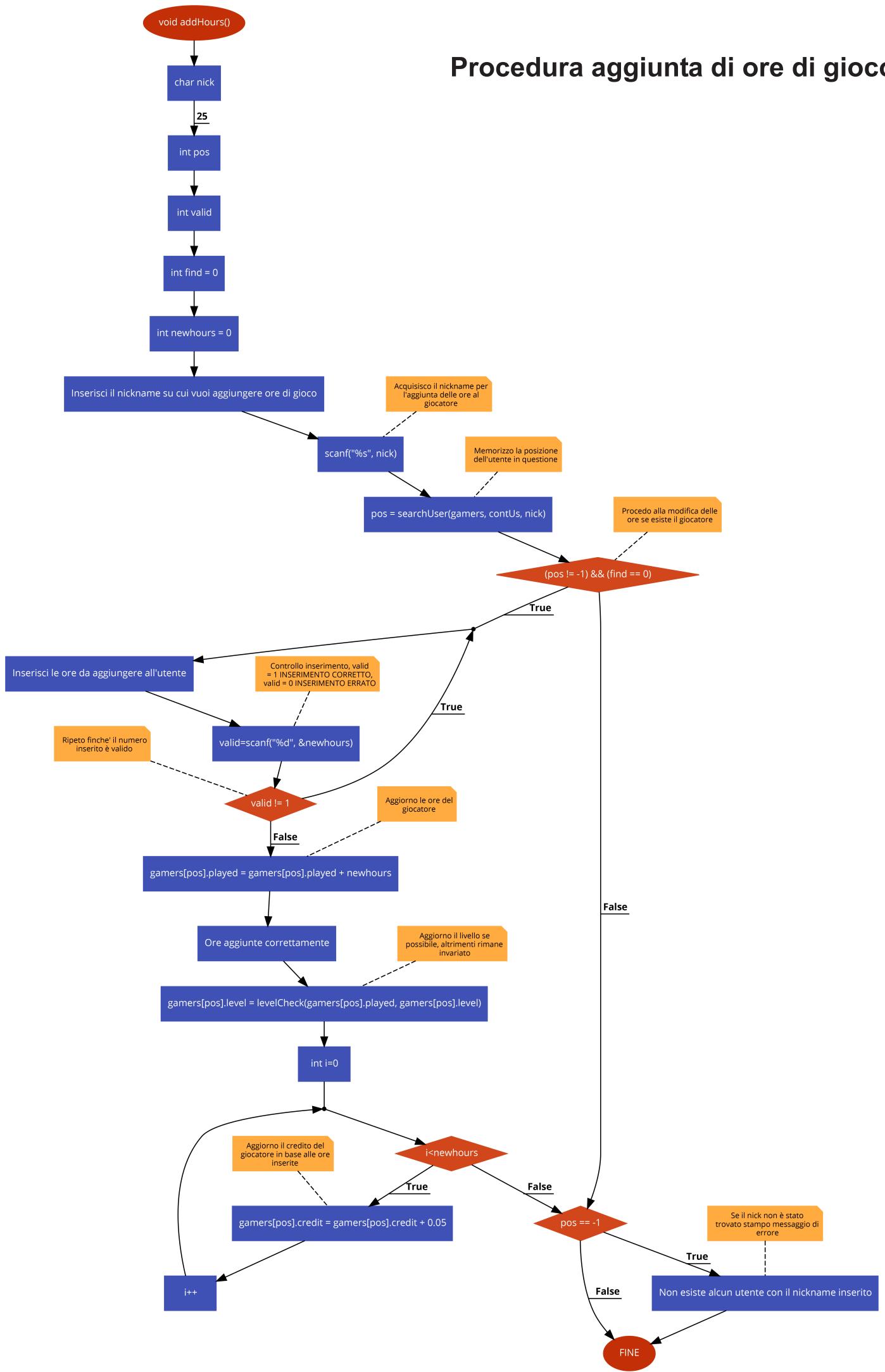
La procedura di aggiunta di ore di gioco ad un giocatore potrebbe determinare l'incremento del livello del giocatore stesso, per questo sono state realizzate una procedura che aggiunge le ore di gioco ed una funzione che tramite quelle ore gestisce l'aumento oppure no del livello. Esse hanno una dipendenza tra loro perchè non avendo l'aggiunta di ore, non può essere chiamata la funzione di controllo del livello.

```
void addHours(gamer gamers[], int contUs);
int levelCheck(int played, int levelOld);
```

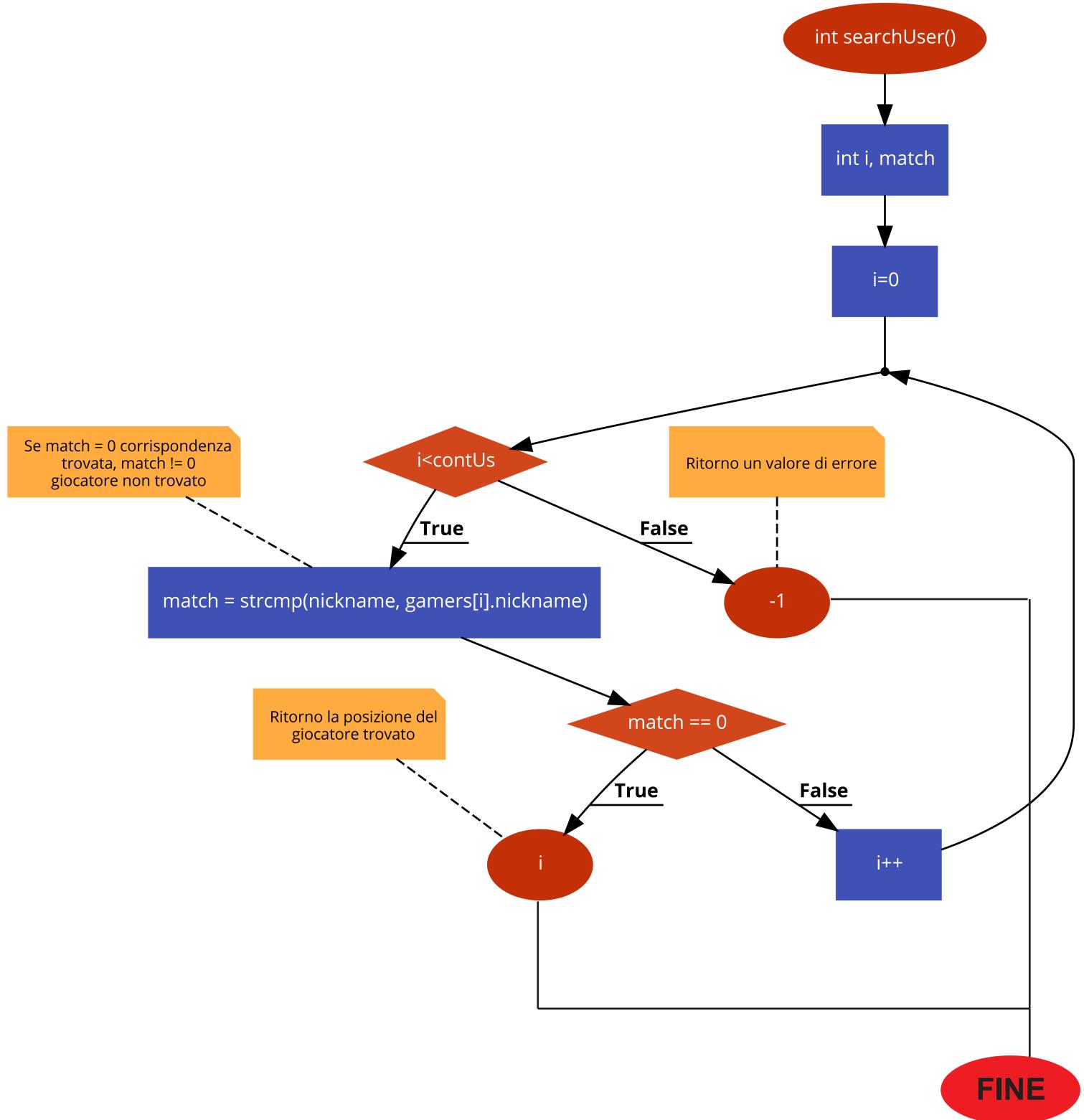
# Procedura aggiunta di un giocatore



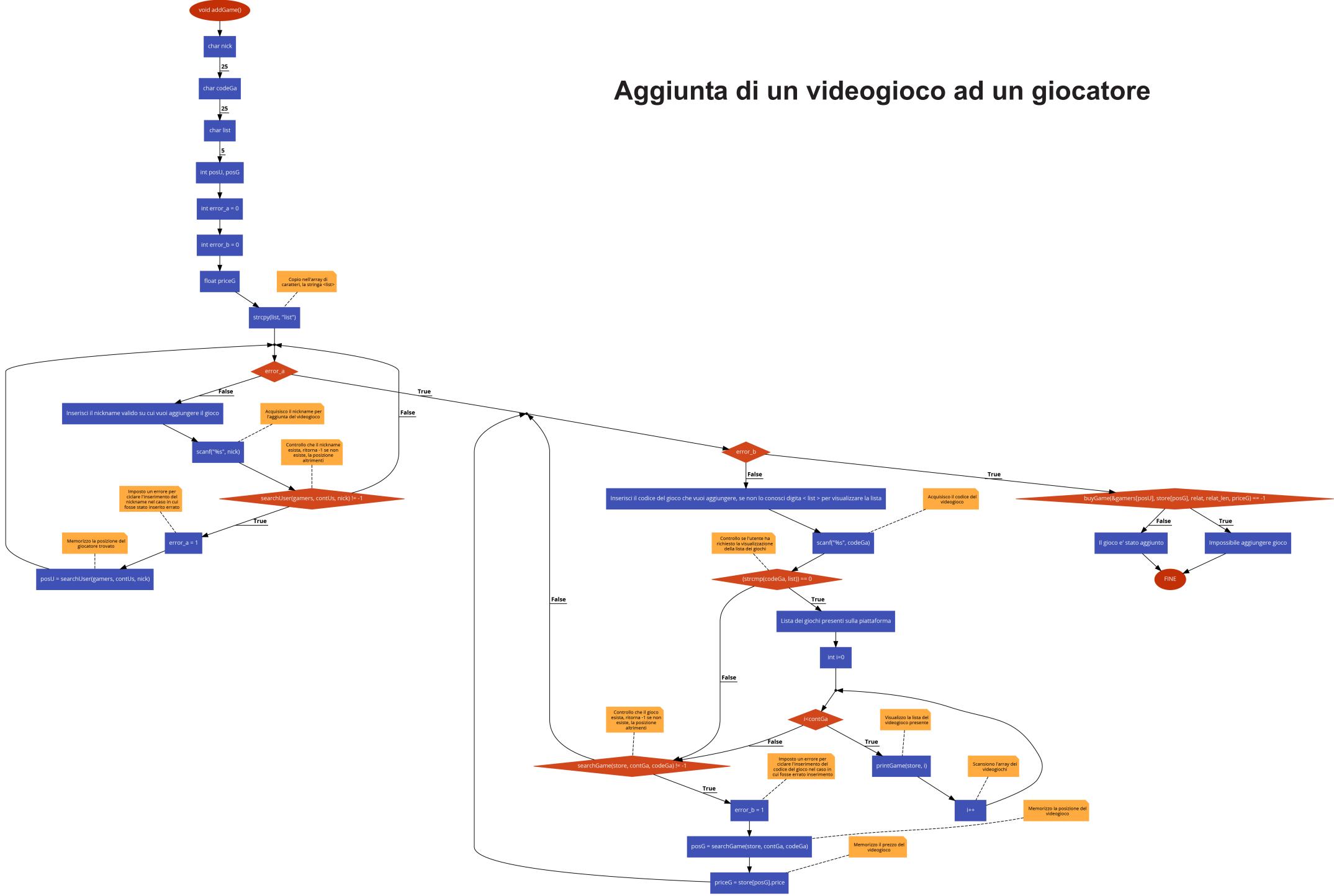
# Procedura aggiunta di ore di gioco

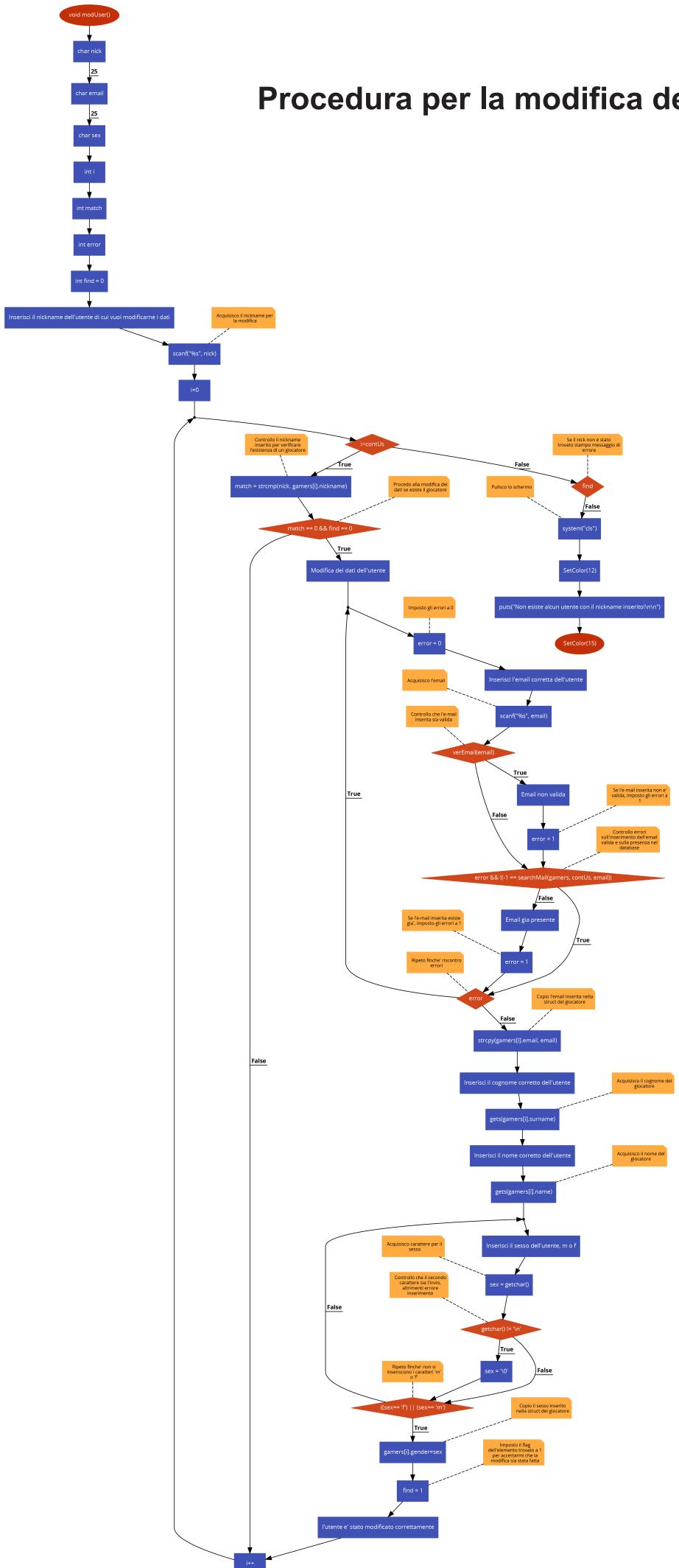


## Procedura di ricerca di un giocatore

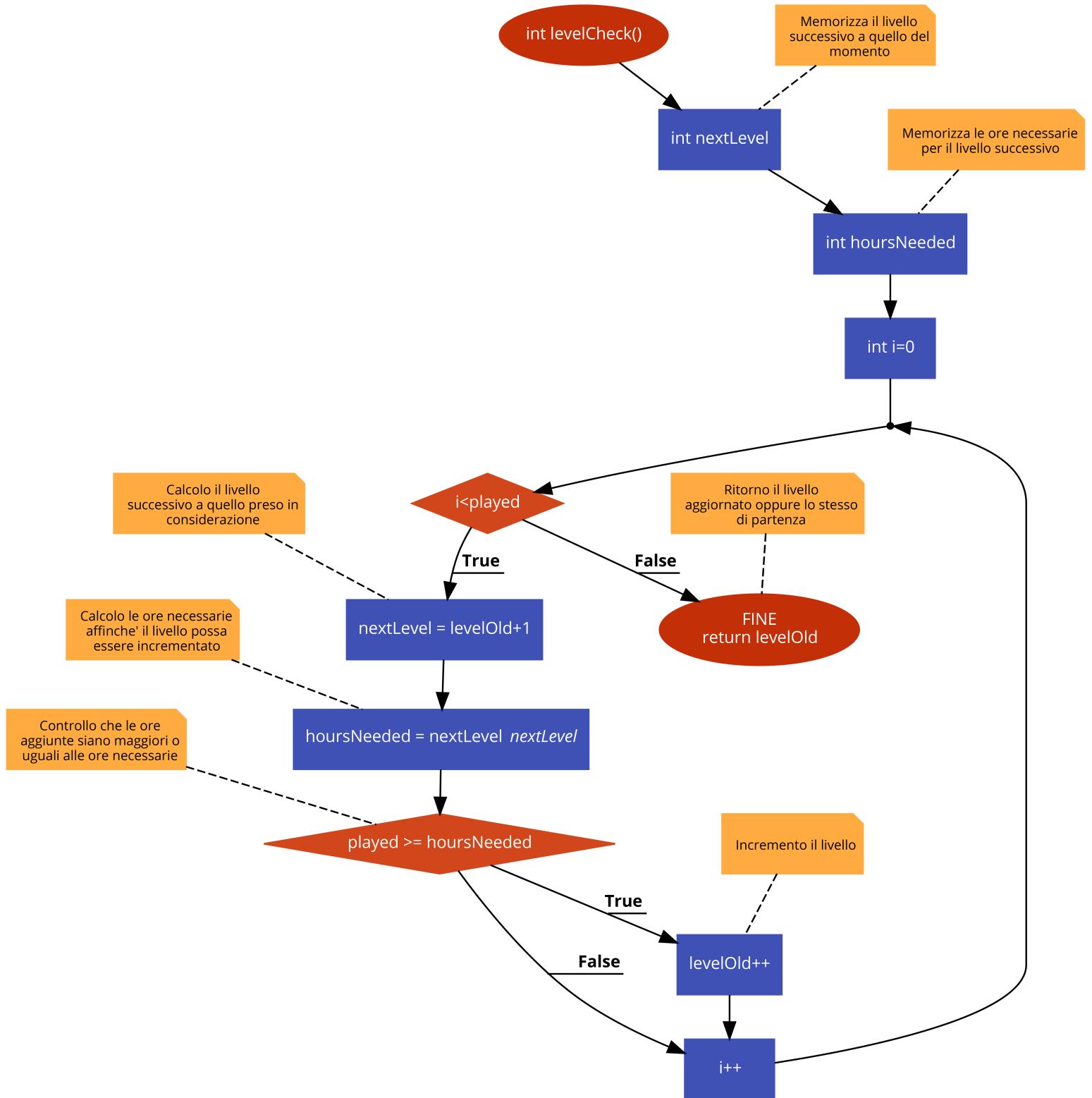


# Aggiunta di un videogioco ad un giocatore





## Funzione di controllo incremento del livello



## 4. Testing

Codice Requisito	Codice Test	Nome	Descrizione Test	Eventuale Input	Risultato atteso	Risultato ottenuto
00	1.0	Caricamento dati da file	File non esistente	a	Creazione file e aggiunta del nuovo utente	Creazione file e aggiunta del nuovo utente
01	1.0	Visualizzazione menù di scelta	Scelta errata	aa	Messaggio errato inserimento	Messaggio errato inserimento
02	1.0	Acquisizione dati nuovo giocatore	Scelta esatta	a	Messaggio corretto inserimento	Messaggio corretto inserimento
02	1.1	Acquisizione dati nuovo giocatore	Inserimento nickname esistente	---	Messaggio errore inserimento nickname	Messaggio errore inserimento nickname
02	1.2	Acquisizione dati nuovo giocatore	Inserimento e-mail esistente	---	Messaggio e-mail già presente	Messaggio e-mail già presente
03	1.0	Verifica nickname presente	Aggiunta del credito	nickname valido	Messaggio credito aggiunto credito aggiornato	Messaggio credito aggiunto credito aggiornato
03	1.1	Verifica nickname presente	Aggiunta del credito	nickname errato	Messaggio nickname inesistente	Messaggio nickname inesistente
08	1.0	Aggiunta ore di gioco al giocatore	nick esistente aggiungo le ore	ore dieci	Messaggio errore inserimento	Messaggio errore inserimento
08	1.1	Aggiunta ore di gioco al giocatore	nick esistente aggiungo le ore	30a	Messaggio errore inserimento	Ore aggiunte, viene troncata la parte non numerica
09	1.0	Controllo incremento livello	nick esistente aggiungo le ore necessarie	81	Messaggio ore aggiunte correttamente livello aggiornato	Messaggio ore aggiunte correttamente livello aggiornato
10	1.0	Ordinamento per livello maggiore	Scelta esatta ma lista utenti vuota	---	Messaggio impossibile ordinare lista vuota	Visualizzazione lista dei giocatori vuota
14	1.0	Aggiunta gioco ad un giocatore	Nickname esatto, non so cosa fare	---	Aggiunta del gioco alla libreria del giocatore	in attesa di un codice che non ricordo, termino il programma
14	1.1	Aggiunta gioco ad un giocatore	Nickname esatto, non so cosa fare	list	Aggiunta del gioco alla libreria del giocatore	Aggiunta del gioco alla libreria del giocatore
07	1.0	Aggiunta credito ad un giocatore	nick esistente aggiungo del credito	35 euro	Messaggio errore inserimento	Aggiunta del credito al giocatore viene troncata la parte non numerica

#### **4.1 Comportamento ed esiti finali**

Al primo avvio in assoluto, il software presenterà dei messaggi di errore su dei file perchè non sono stati trovati, si consiglia di proseguire con l'esecuzione del programma in quanto i file verranno creati in automatico non appena ci sarà la necessità.

Con il requisito 14 si era riscontrato un problema, una volta inserito il nickname su cui si desiderava aggiungere il gioco e non ci si ricordava il codice del gioco, si era costretti a chiudere forzatamente il programma. Ho risolto aggiungendo la possibilità di visionare la lista dei giochi a runtime infatti dopo aver inserito il nickname esatto, si ha la possibilità digitando < list > di visionare tutti i giochi e i relativi codici.

Sono stati effettuati vari test per stabilire la stabilità del software e si può affermare che il software è abbastanza stabile, si è cercato di ricondurre al minimo i bug tramite l'utilizzo delle stampe ausiliarie e dei controlli sugli inserimenti.

Si pianifica per il futuro una migliore gestione della memoria, ottimizzazioni prestazionali, ulteriori controlli su inserimenti di valori logicamente impossibili e implementazione di nuove funzionalità.



UNIVERSITÀ  
DEGLI STUDI DI BARI  
ALDO MORO

**dib**

DIPARTIMENTO  
DI INFORMATICA

---

**FINE**