

# Fondamenti di Informatica T-1

## modulo 2

---

### Laboratorio 08: ***allocazione dinamica e liste***

1

### Esempio 1: Liste

---

- I risultati di un appello d'esame di Fondamenti di Informatica vengono ***salvati su un file (di testo)***
- Su tale file vengono salvati ***solo i voti, separati da spazi o da newline***
- Scrivere un programma che, facendo uso delle ***liste di elementi interi***:
  - legga i valori salvati nel file, li memorizzi all'interno della lista, e li stampi a video
  - chieda all'utente un valore di soglia
  - memorizzi in ***due liste differenti i valori superiori*** (o uguali) e quelli ***inferiori a tale soglia***, stampando poi a video il contenuto di entrambe le liste

2

## Esempio 1 – main()

---

```
#include <stdio.h>
#include <stdlib.h>
#include "list.h"

int main(void)
{
    FILE * fp;
    List startList, lowList, highList, temp;
    Element voto, soglia;

    if ( (fp=fopen("voti.txt", "r")) == NULL)
    {
        perror("The file does not exist!");
        system("PAUSE"); exit(-1);
    }
    ...
```

3

## Esempio 1 – main() per Lettura File

---

```
...
startList = emptyList();
lowList = emptyList();
highList = emptyList();

while (fscanf(fp, "%d", &voto) > 0)
    startList = cons(voto, startList);
fclose(fp);

showList(startList);
printf("Inserire soglia: ");
scanf("%d", &soglia);
```

4

## Esempio 1 – Costruzione delle due Liste senza Uso di Primitive

---

```
while(startList != NULL)
{
    if (startList->value < soglia)
    {
        temp = (List) malloc(sizeof(Item));
        temp->value = startList->value;
        temp->next = lowList; lowList = temp;
    }
    else
    {
        temp = (List) malloc(sizeof(Item));
        temp->value = startList->value;
        temp->next = highList; highList = temp;
    }
    startList = startList->next;
}
```

5

## Esempio 1 – Costruzione delle due Liste con Uso di Primitive

---

```
while(!empty(startList))
{
    Element value = head(startList);
    if(value < soglia)
        lowList = cons(value, lowList);
    else
        highList = cons(value, highList);
    startList = tail(startList);
}
showList(lowList);
showList(highList);
while(!empty(lowList)) {
    temp = lowList;
    lowList = tail(lowList);
    free(temp);
} //analoga deallocazione serve per highList
system("PAUSE"); return (0);
}
```

6

## Esercizio 2: Intersezione e Differenza fra Liste

---

- Si leggano da standard input **due liste di interi positivi** (l'utente terminerà l'inserimento di ognuna con il valore 0)
- Scrivere le seguenti funzioni:
  - `List intersect(List l1, List l2)` riceve due liste e **restituisce una terza lista contenente i valori presenti in entrambe**, utilizzando le primitive
  - `List diff(List l1, List l2)` restituisce una **lista contenente i valori presenti in l1 che NON sono presenti in l2** ( $l1 - l2$ ), senza usare le primitive
- **Modificare** le soluzioni precedenti facendo in modo che la **lista risultato NON contenga elementi ripetuti**
- Deallocare correttamente le liste utilizzate

7

## Esercizio 2 - Schema di Soluzione

---

- Scomporre in sottoproblemi
- Realizzare una funzione per verificare **se un elemento è contenuto in una lista**
- Realizzare intersezione come insieme degli elementi della prima lista contenuti anche nella seconda
- Realizzare la differenza come l'insieme degli elementi della prima lista NON contenuti nella seconda
- Semplice modifica per evitare ripetizioni nel risultato
  - Aggiungere elementi solo se non già contenuti

8

## Esercizio 3: Lista di Strutture

---

### Gestione di un negozio di videogame

- Un negozio di videogame vuole automatizzare parte della propria gestione
- Il negozio salva mensilmente **lo stato dei propri articoli su un file di testo** e **traccia su un secondo file di testo tutte le vendite** effettuate
- Lo scopo del programma è
  - Costruire **una lista per i videogiochi** aggiornandone il **numero di copie disponibili** in seguito alle vendite
  - Generare una **lista di videogiochi acquistabili da bambini**
  - Salvare in un **file di testo** gli **ordini da effettuare** per riportare il magazzino ad avere un certo numero di copie per ogni gioco

9

## Esercizio 3 – i File di Input

---

- Ogni riga del file memorizza lo stato mensile e contiene i dati di un videogioco
  - codice intero
  - titolo, possibilmente contenente spazi, di esattamente 30 caratteri
  - carattere di identificazione del tipo di gioco ('P' = picchiaduro, 'A' = avventura, 'R' = rompicapo, 'O' = altro)
  - numero di copie disponibili
  - voto medio dato dagli utenti
- In un secondo file di testo il negozio tiene traccia dei **videogiochi venduti** nell'arco del mese
  - Ogni riga del file contiene il codice identificativo di un gioco venduto (una singola copia)
  - Si prevede la possibilità che ci siano delle righe del file con codici errati (vanno saltati)

10

## Esercizio 3 - Requisiti

---

Implementare le seguenti funzioni

- `void printGames(list games)`
  - Deve stampare il contenuto di una lista di videogiochi (si realizzi la funzione ricorsivamente usando le primitive)
- `boolean loadFromFile(char *fileName, list *games)`
  - Realizza la lista di videogiochi di partenza prendendo i dati dal file di nome `fileName` (si utilizzino le primitive)
- `boolean updateAvailability(char *fileName, list games)`
  - Aggiorna il numero di copie dei videogiochi, leggendo le vendite dal file di nome `fileName`
    - Se trova un codice non “riconosciuto”, stampa un messaggio di warning e va avanti
  - (si utilizzi notazione a puntatori, no primitive)

11

## Esercizio 3 - Requisiti

---

- `list gamesForKids(list games, float threshold)`
  - Restituisce la lista di videogiochi acquistabili da un bambino nel mese corrente (si realizzi la funzione ricorsivamente e usando le primitive)
  - Si adotti la seguente politica per scegliere se un videogioco è acquistabile da un bambino
    - Sono acquistabili unicamente i giochi disponibili (ovvero quelli per cui il numero di copie è positivo)
    - I picchiaduro non sono acquistabili da un bambino
    - Tutti i giochi di avventura sono acquistabili da un bambino
    - Per gli altri giochi, si indichino solo quelli per cui il voto medio dagli utenti è superiore alla soglia data

12

## Esercizio 3 - Requisiti

---

- Boolean `saveOrdersToFile`

`(char *fileName, list games, int qty)`

- Salva i codici, i titoli e il numero di copie da ordinare per ogni videogioco (si realizzi la funzione iterativamente usando le primitive)
- Si adotti la seguente politica
  - I quantitativi da ordinare devono riportare tutti i giochi alla quantità decisa `qty`
  - Ovviamente, quindi, un videogioco è presente nella lista solo se per esso è effettivamente necessario ordinare copie aggiuntive

13

## Esercizio 3 – main()

---

```
int main()
{
    List games;
    List kids;
    loadFromFile("lista.txt", &games);
    printGames(games);
    printf("-----UPDATE-----\n");
    updateAvailability("acquisti.txt", games);
    printGames(games);
    printf("-----G4KIDS-----\n");
    kids = gamesForKids(games, 4);
    printGames(kids);
    saveOrdersToFile("ordine.txt", games, 5);
    //DEALLOCAZIONE DELLE LISTE!!!
    return 0;
}
```

14

## Esercizio 3 - Output atteso

---

|                             |       |          |
|-----------------------------|-------|----------|
| 787) dis-avventura          | (A) 2 | 2.200000 |
| 981) noia e ancora noia     | (O) 3 | 3.100000 |
| 421) the incredible machine | (R) 7 | 8.500000 |
| 753) super pang             | (O) 4 | 8.600000 |
| 642) zak mckracken          | (A) 6 | 9.500000 |
| 574) ti spiezzo in due      | (P) 4 | 4.500000 |
| 125) monkey island 1        | (A) 9 | 9.200000 |
| 123) tekken 3               | (P) 7 | 8.100000 |

-----UPDATE-----

CODICE NON RICONOSCIUTO: 111

|                             |       |          |
|-----------------------------|-------|----------|
| 787) dis-avventura          | (A) 2 | 2.200000 |
| 981) noia e ancora noia     | (O) 1 | 3.100000 |
| 421) the incredible machine | (R) 3 | 8.500000 |
| 753) super pang             | (O) 0 | 8.600000 |
| 642) zak mckracken          | (A) 5 | 9.500000 |
| 574) ti spiezzo in due      | (P) 1 | 4.500000 |
| 125) monkey island 1        | (A) 4 | 9.200000 |
| 123) tekken 3               | (P) 5 | 8.100000 |

-----G4KIDS-----

|                             |       |          |
|-----------------------------|-------|----------|
| 787) dis-avventura          | (A) 2 | 2.200000 |
| 421) the incredible machine | (R) 3 | 8.500000 |
| 642) zak mckracken          | (A) 5 | 9.500000 |
| 125) monkey island 1        | (A) 4 | 9.200000 |

15

## Esercizio 3 - Output atteso

---

### ■ Contenuto del file `ordine.txt`

|                            |   |
|----------------------------|---|
| 787 dis-avventura          | 3 |
| 981 noia e ancora noia     | 4 |
| 421 the incredible machine | 2 |
| 753 super pang             | 5 |
| 574 ti spiezzo in due      | 4 |
| 125 monkey island 1        | 1 |

16