Informatica ed Elementi di Informatica Medica

A.A. 2017-18

Laboratorio n°5

Dott. Michele Zanella

Ing. Gian Enrico Conti

HEAP LAB

Info Logistiche (cont'd)

- Calendario laboratori

Data	Orario	Squadra	Aula	Resp.	Programma
16/03/18	14:15-17:15	A	L26.14	Conti	Lab 1
19/03/18	15:15-18:15	B	L26.14	Zanella	
23/03/18	14:15-17:15	A	L26.14	Conti	Lab 2
26/03/18	15:15-18:15	B	L26.14	Zanella	
06/04/18	14:15-17:15	A	L26.14	Conti	Lab 3
09/04/18	15:15-18:15	B	L26.14	Zanella	
13/04/18	14:15-17:15	A	L26.14	Conti	Lab 4
16/04/18	15:15-18:15	B	L26.14	Zanella	
20/04/18	14:15-17:15	A	L26.14	Conti	Lab 5
23/04/18	15:15-18:15	B	L26.14	Zanella	

Esercizio 5.1: Salvataggio partite Mastermind con File I/O

Aggiungere al gioco Mastermind la possibilità di salvare la partita e riprendere il gioco.

Requisiti:

- Il gioco deve salvare in un file di testo (`.txt`) ogni round di una partita
- L'utente sceglie se riprendere l'esecuzione di una partita o se iniziarne una nuova
- Il gioco deve poter riprendere l'esecuzione di una partita dall'ultimo round giocato salvato su un file di testo (`.txt`)

Hints:

- Se la partita termina (vittoria o sconfitta) svuotare il file di testo
- Aprire e CHIUDERE il file nelle modalità corrette!
- Se si inizia una partita nuova, sovrascrivere il file precedente
- Salvo solo l'ultimo round giocato (ulteriore esercizio: salvare tutta la partita fino all'ultimo round giocato)

Ricorsione

- La **ricorsione** è un approccio per la risoluzione di problemi che consiste nel riapplicare uno stesso algoritmo su un insieme di dati semplificato o suddiviso.
- Le soluzioni trovate per le chiamate ricorsive vengono poi combinate per formare la soluzione del problema iniziale.
- Una *funzione* è detta **ricorsiva** se richiama se stessa.
  Se due funzioni, invece, si chiamano l'un l'altra sono dette **mutuamente ricorsive**.
- Una funzione ricorsiva è formata da:
  - **Caso base**: è un caso particolare del problema che è possibile risolvere direttamente
  - **Passo ricorsivo:** la funzione viene richiamata passando un sottoinsieme dei dati originali.
- **L'obiettivo è quello di semplificare ad ogni passo ricorsivo i dati/il problema in modo da raggiungere un caso base.**
- La sequenza di chiamate ricorsive termina quando quella *annidata* (più interna) incontra uno dei casi basi, a questo punto si risale la sequenza ottenendo il risultato

Esempio Ricorsione: Fattoriale

- Funzione ricorsiva per calcolare il fattoriale di un numero:

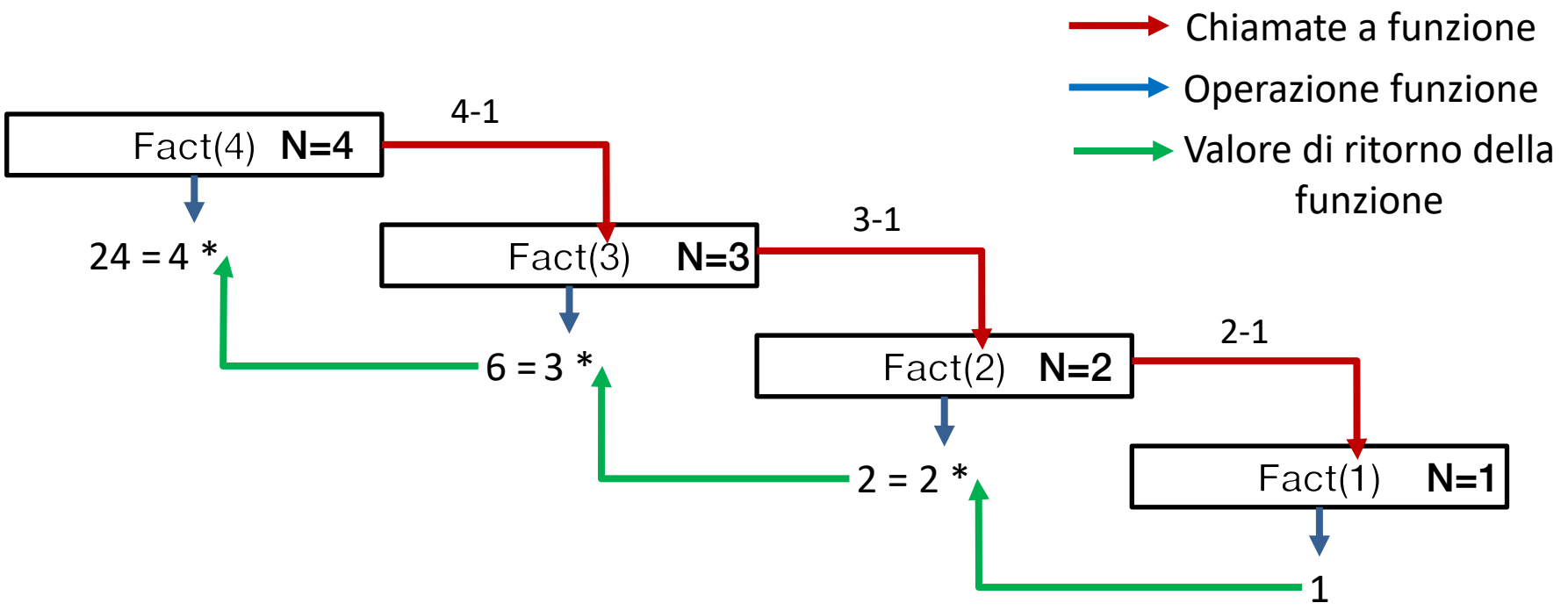
```
int fact(int a){
    if (n<=1)
        return 1;
    else
        return n*fact(n-1);
}
```

Caso base

Passo ricorsivo

- Cosa succede (chiamo `fact(4)`):
  1. In fact_1 n=4, non è il caso base e quindi per eseguire il calcolo `4*fact(3)`, la funzione chiama `fact(3)` e aspetta la sua terminazione.
  2. In fact_2 n=3, non è il caso base e quindi per eseguire il calcolo `3*fact(2)`, la funzione chiama `fact(2)` e aspetta la sua terminazione.
  3. In fact_3 n=2, non è il caso base e quindi per eseguire il calcolo `2*fact(1)`, la funzione chiama `fact(1)` e aspetta la sua terminazione.
  4. In fact_4 n=1, è il caso base, essa quindi ritorna subito il valore 1 a fact_3
  5. fact_3 riceve il valore 1, esegue il calcolo sospeso e ritorna 2 a fact_2
  6. fact_2 riceve il valore 2, esegue il calcolo sospeso e ritorna 6 a fact_3
  7. fact_1 riceve il valore 6, esegue il calcolo sospeso e ritorna 24.

Esempio Ricorsione: Fattoriale (cont'd)



Ricorsione

- Pro & Cons:
  - Problemi complessi -> si risolvono con poche righe di codice
  - Non è efficiente perché richiama molte volte una funzione e alloca sullo *stack* i parametri e le variabili ad ogni chiamata
  - Qualsiasi problema ricorsivo può essere svolto in modo *iterativo* (non ricorsivo), ma spesso quest'ultimo è molto più complesso.
  - Non bisognerebbe utilizzarla quando essa esegue a sua volta più di una chiamata ricorsiva

Esercizio 5.2: Stringa inversa (con RICORSIONE)

Si implementi una funzione ricorsiva che stampi la stringa ricevuta come parametro, in ordine inverso.

```
void stampa_inversa(char* s);
```

Esercizio 5.3: Sudoku ricorsivo

Scrivere un programma per la soluzione RICORSIVA di un **Sudoku**.

https://1sudoku.com/play/sudoku-kids-free/sudoku-4x4/

Regole di base:

- 4 Regioni 2x2
- La regione va riempita con i numeri da 1 a 4
- Sulla stessa colonna e sulla stessa riga un numero deve comparire una sola volta
- All'interno della regione un numero deve comparire una sola volta
- Riempire le celle vuote
- La soluzione deve essere RICORSIVA

Hints:

- Utilizzare una matrice 4x4
- Identificare il caso base
- Identificare il passo iterativo

Esercizio 5.3: Sudoku ricorsivo (cont'd)

Organizzazione delle funzioni:

- `risolvi`: funzione ricorsiva che risolve il Sudoku
- `valido`: funzione che controlla se la configurazione è valida (e.g., nessuna ripetizione nelle righe, nelle colonne e nelle regioni)
- `completo`: funzione che controlla se la configurazione non contiene zeri (e.g., celle vuote)
- `controlla_riga`: funzione per controllare se la riga i-esima non contiene numeri >0 ripetuti
- `controlla_colonna`: funzione per controllare se la colonna i-esima non contiene numeri >0 ripetuti
- `controlla_regione`: funzione per controllare se la regione con riquadro in alto a sinistra `(i,j)` non ha ripetizioni

Elaborazione delle immagini

Procedimento con il quale data un'immagine si eseguono delle operazioni per modificarne l'aspetto.

https://it.wikipedia.org/wiki/Elaborazione_digitale_delle_immagini

Immagine: file contenente delle informazioni circa la dimensione, i singoli pixel (colore, luminosità, posizione, ecc…)

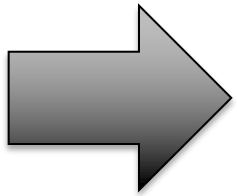
Può essere vista come una matrice alla quale si applicano delle funzioni:

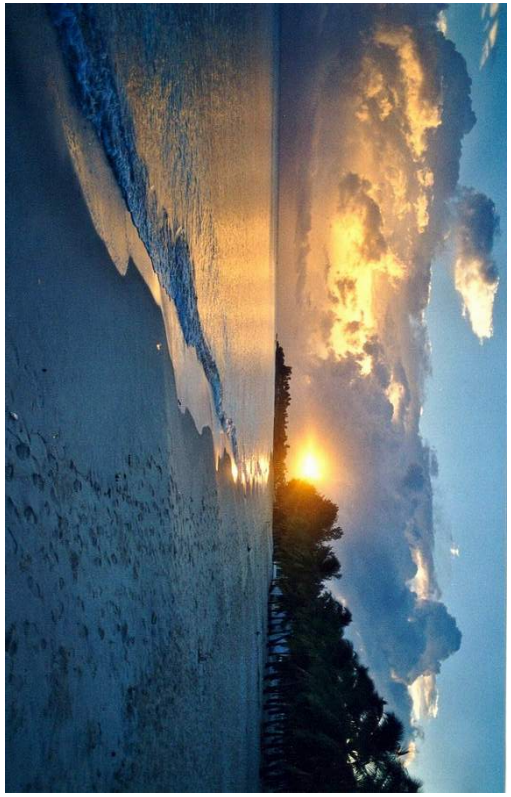
- Traslazione
- Rotazione
- Inversione
- Riflessione
- ….

Fino ad operazioni più complesse come: classificazione, edge detection, regressione, pattern recognition…

Elaborazione delle immagini (cont'd)

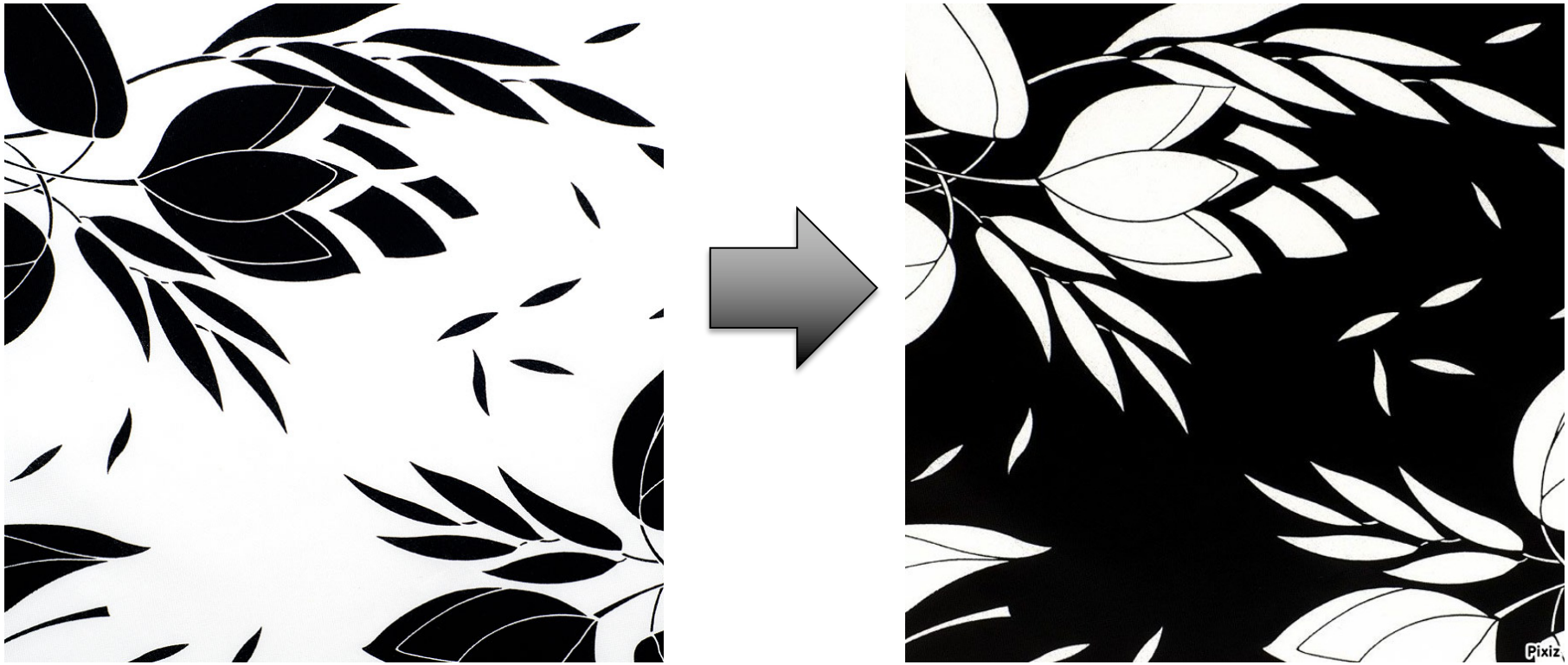
Rotazione





Elaborazione delle immagini (cont'd)

Negativo



Elaborazione delle immagini: PBM

Portable BitMap (PBM): è un formato immagine non compresso contenente solo informazioni sul colore monocromatico (bianco o nero) e la posizione dei singoli pixel.

http://netpbm.sourceforge.net/doc/pbm.html

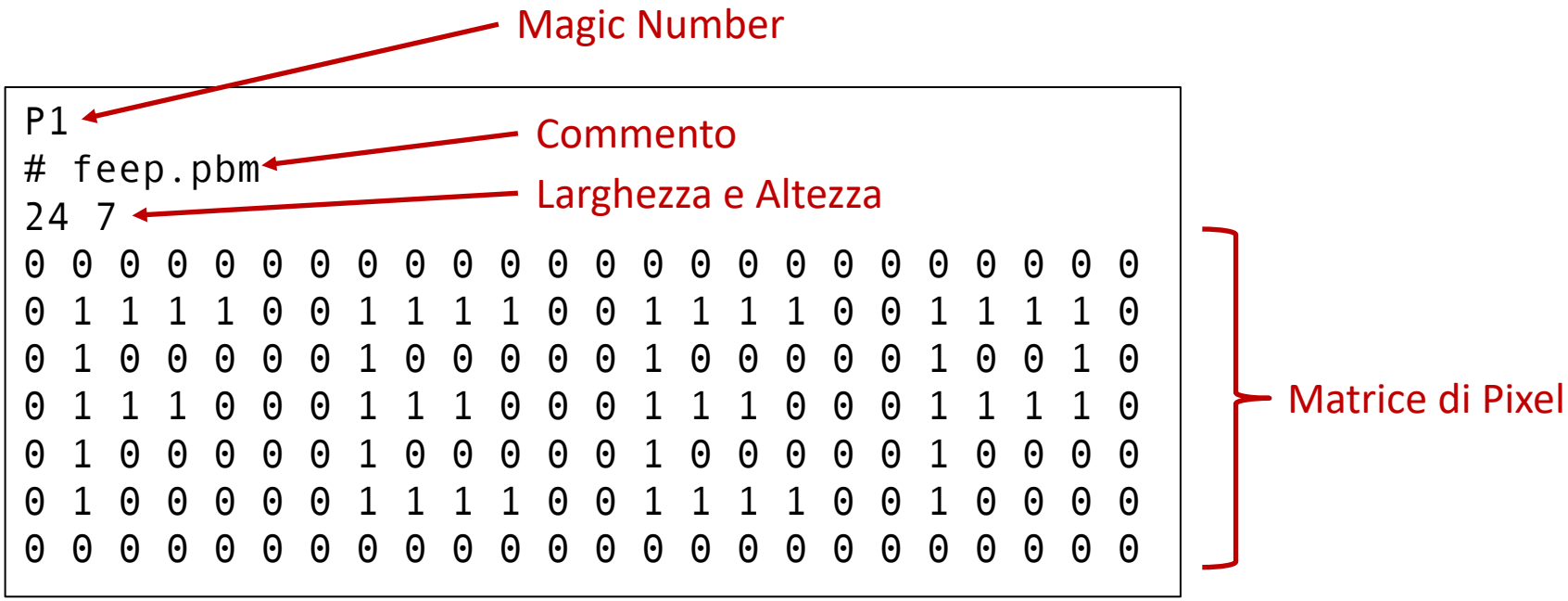
Elaborazione delle immagini: PBM (cont'd)

Ogni file di immagine plain PBM è formato da due parti con alcune sezioni:

- HEADER:
    - "Magic Number": formato da due caratteri che identificano il tipo di file ("P1")
    - Whitespace (spazio bianco)
    - Larghezza dell'immagine: formattata in ASCII decimale
    - Whitespace
    - Altezza dell'immagine
    - Newline (a capo)
    - Eventuali commenti preceduti dal simbolo '#'
- DATA:
    - Una matrice di pixel: ciascun byte rappresenta un pixel, 1 se nero, 0 se bianco (in ASCII). Ciascuna linea è lunga al massimo 70 caratteri

Elaborazione delle immagini: PBM (cont'd)

Esempio: feep.pbm



Essendo un formato di immagine è apribile con un visualizzatore di immagini!

Esercizio 5.4: Elaborazione di un'immagine PBM

Scrivere un programma che permetta di eseguire delle semplici elaborazioni di un'immagine in formato PBM.

Requisiti:

- Il programma deve caricare e leggere un'immagine da file in formato PBM
- Deve implementare la rotazione di 90° dell'immagine
- Deve mostrare il negativo dell'immagine
- Per ogni trasformazione deve salvare un nuovo file immagine in formato plain PBM

Hints:

- Salvare l'immagine caricata in una matrice e lavorare sulla matrice
- Scaricare lo zip con l'immagine "feep.pbm" dal sito del laboratorio ed estrarla nella stessa cartella dove andrete ad implementare il codice

Esercizio 5.4: Elaborazione di un'immagine PBM (cont'd)

Opzioni avanzate:

- Possibilità di passare come argomenti da command line le operazioni da eseguire sull'immagine
- Possibilità di passare come primo argomento il nome del file (senza estensione) su cui operare

Esercizio 5.4: Elaborazione di un'immagine PBM (cont'd)

Modello e tipi di dato:

Immagine: -> *struct*

- Tipo (i.e., il magic number) -> *char[2]*
- Larghezza -> *int*
- Altezza -> *int*
- Pixel -> *int[ALTEZZA][LARGHEZZA]*

Esercizio 5.4: Elaborazione di un'immagine PBM (cont'd)

Organizzazione delle funzioni:

- `apri_immagine`: Funzione che apre l'immagine, legge l'header e legge i dati dei pixel (suddivisa in sotto funzioni specifiche)
- `salva_immagine`: Funzione che scrive l'header, scrive i dati dei pixel e salva l'immagine (suddivisa in sotto funzioni specifiche)
- `copia_immagine`: Funzione che crea una copia dell'immagine originale per effettuare le operazioni
- `ruota_immagine_90`: Funzione che ruota l'immagine originale di 90 gradi
- `negativo_immagine`: Funzione che crea il negativo dell'immagine originale

Conclusione

Per dubbi, info e domande contattatemi via mail

michele.zanella@polimi.it

Per i codici sorgenti e le slide:

zanella.faculty.polimi.it