

Laboratorio Informatico

Prof. Ing. Daniele Benedetti
d.benedetti@alice.it

**UNIVERSITA' DEGLI STUDI DI
MACERATA**

06/11/2012

Interazione con il docente

- Il docente riceve gli studenti il martedì, dalle 17 alle 19.
- Per ogni comunicazione o avviso, per le dispense e per ulteriori materiali didattici, il docente rimanda gli studenti alla propria pagina personale
- <http://docenti.unimc.it/docenti/daniele-benedetti>
- Il docente è raggiungibile tramite E-Mail d.benedetti@alice.it

Materiale del corso

- Presentazioni illustrate a lezione
- Libro di testo:
 - Informatica di base 5/ed
Dennis P. Curtin, Kim Foley, Kunal Sen, Cathleen Morin

Programma del corso

- Concetti di base dell'ICT
- Architettura degli elaboratori
- **Software: Algoritmi e la programmazione**
- Software di base, i sistemi operativi
- Software applicativi (Office Automation)
- Reti degli Elaboratori

L'istruzione condizionale

- Una istruzione del tipo `if ... then ... else ... end if` è detta istruzione condizionale, in quanto regola l'esecuzione dell'algoritmo al verificarsi di determinate condizioni
- **If** *condizione* **then**
operazione 1, ... , operazione n
else
operazione 1, ... , operazione n
end if

L'iterazione (il ciclo for)

- Una istruzione del tipo for ...
do ... end for è detta iterativa o
ciclo, in quanto permette di
ripetere l'esecuzione di alcune
istruzioni per un certo numero di
volte

L'iterazione (il ciclo while)

- Una istruzione del tipo `while ... do ... end` `while` è ancora una volta iterativa, ma differentemente dal ciclo `for`, il numero di volte per cui alcune istruzioni vengono ripetute dipende da una condizione vero/falso
- Il numero di iterazioni non è quindi noto a priori come nel caso del ciclo `for`, ma dipende dal verificarsi della condizione booleana

Strutture dati

- I tipi di dato che abbiamo considerato finora (booleano, carattere e numerico) sono detti elementari
- Per convenzione, chiamiamo
 - ♦ bool - booleano
 - ♦ char - carattere
 - ♦ int - numero intero
 - ♦ float - numero a virgola mobile
- ♦ In questo corso, ci limiteremo a trattare come strutture dati: Vettori e Record

I vettori

- Un array o vettore è una struttura dati omogenea, che contiene un numero finito n di elementi dello stesso tipo
- Si può immaginare un array come una sorta di casellario, le cui caselle sono dette celle
- Ciascuna delle celle si comporta come una variabile tradizionale; tutte le celle sono variabili di uno stesso tipo, detto tipo base dell'array
- Questi elementi sono individuati attraverso un indice numerico, che tipicamente va da 0 a $n - 1$

Esempio di vettore

- Una stringa capace di contenere un nome di una persona
- `char nome[15]`
- | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|--|--|--|--|--|--|--|--|
| D | a | n | i | e | l | e | | | | | | | | |
|---|---|---|---|---|---|---|--|--|--|--|--|--|--|--|
- Per accedere ad un elemento di un vettore (cioè leggerne il valore o assegnarvi un nuovo valore), utilizziamo l'operatore “[]”
- `char a = nome[0]` → qual è il valore di a?
- `nome[1] = 'i'` → cosa succede?
- `nome[3] = 65` → cosa succede?

I record

- Il record è una struttura dati che può essere eterogenea o omogenea, e quindi può contenere una combinazione di elementi che possono essere di diverso tipo, ad esempio un intero, un numero in virgola mobile e un carattere testuale
- Gli elementi di un record sono detti campi, e sono identificati da un nome

Esempio

- Esempio: record studente

```
struct studente begin
```

```
    int matricola
```

```
    char nome[20]
```

```
    char cognome[30]
```

```
    boolean fuoriCorso
```

```
    boolean erasmus
```

```
end struct
```

Accesso ai contenuti di un record

- L'accesso ai campi di un record avviene usando l'operatore “.”

studente s

int m = s.matricola

s.nome = “Giada”

s.erasmus = **false**

Lo sviluppo dei programmi

- Prima fase: la definizione del problema
- Seconda fase: la progettazione del programma
- Terza fase: la codifica del programma
- Quarta fase: la verifica dei risultati
- Quinta fase: la documentazione

La definizione del problema

- Definire in maniera precisa ed accurata il problema che si intende risolvere, chiedendo ai clienti o utenti finali, di descrivere l'output atteso del programma

La progettazione del programma

- Obiettivi da raggiungere → Ricerca della soluzione più elegante
- Per scambiarsi idee e descrivere i progetti si usano diverse tecniche, tra cui i **diagrammi di flussi** e gli **pseudocodici**

I diagrammi di flusso

- Grafico per illustrare un processo in tutte le sue fasi
- Le fasi rappresentate con simboli e collegate da linee di flusso
- Il simbolo di terminazione (ovale): inizio o fine
- Il simbolo di decisione (rombo): scelte, if-then-else
- Il simbolo di elaborazione (rettangolo): esecuzione di un calcolo
- Il simbolo di input/output (parallelogramma): immissione o emissione di informazioni
- Il simbolo di connessione (cerchio): unisce due rami del programma

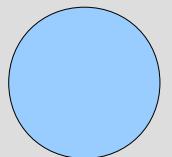
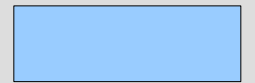
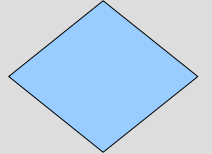


Diagramma di flusso: if then else

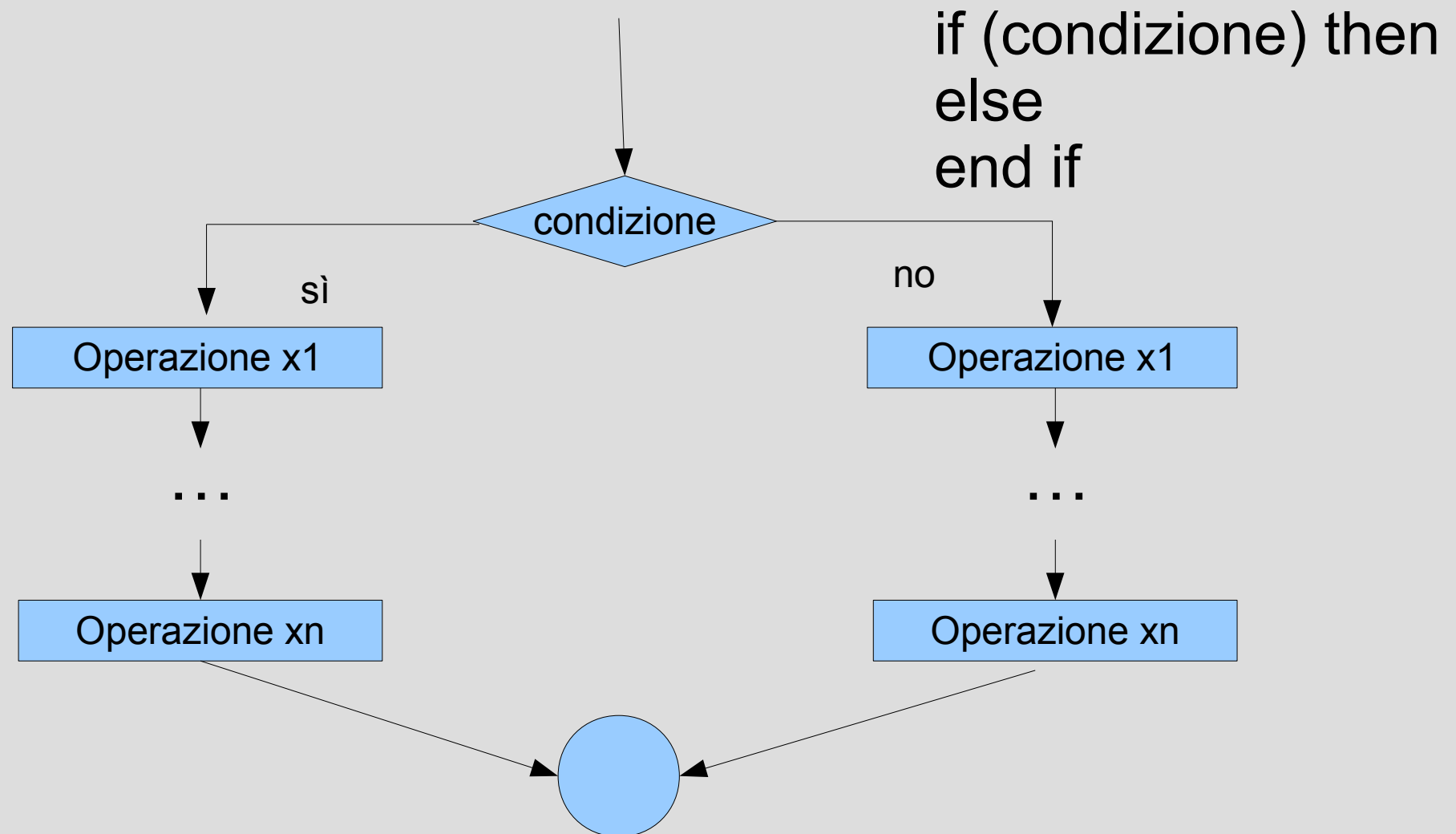


Diagramma di flusso: ciclo for

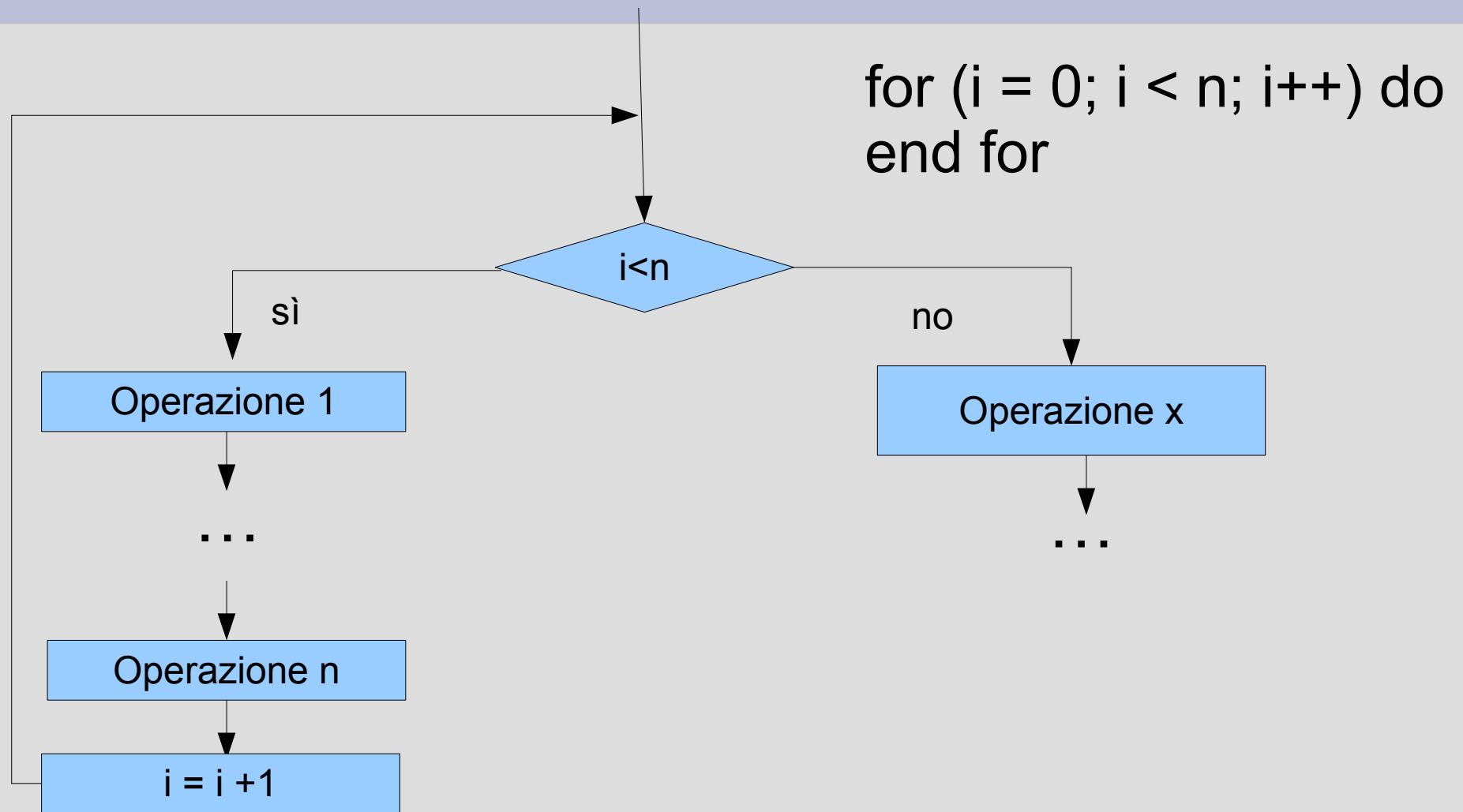
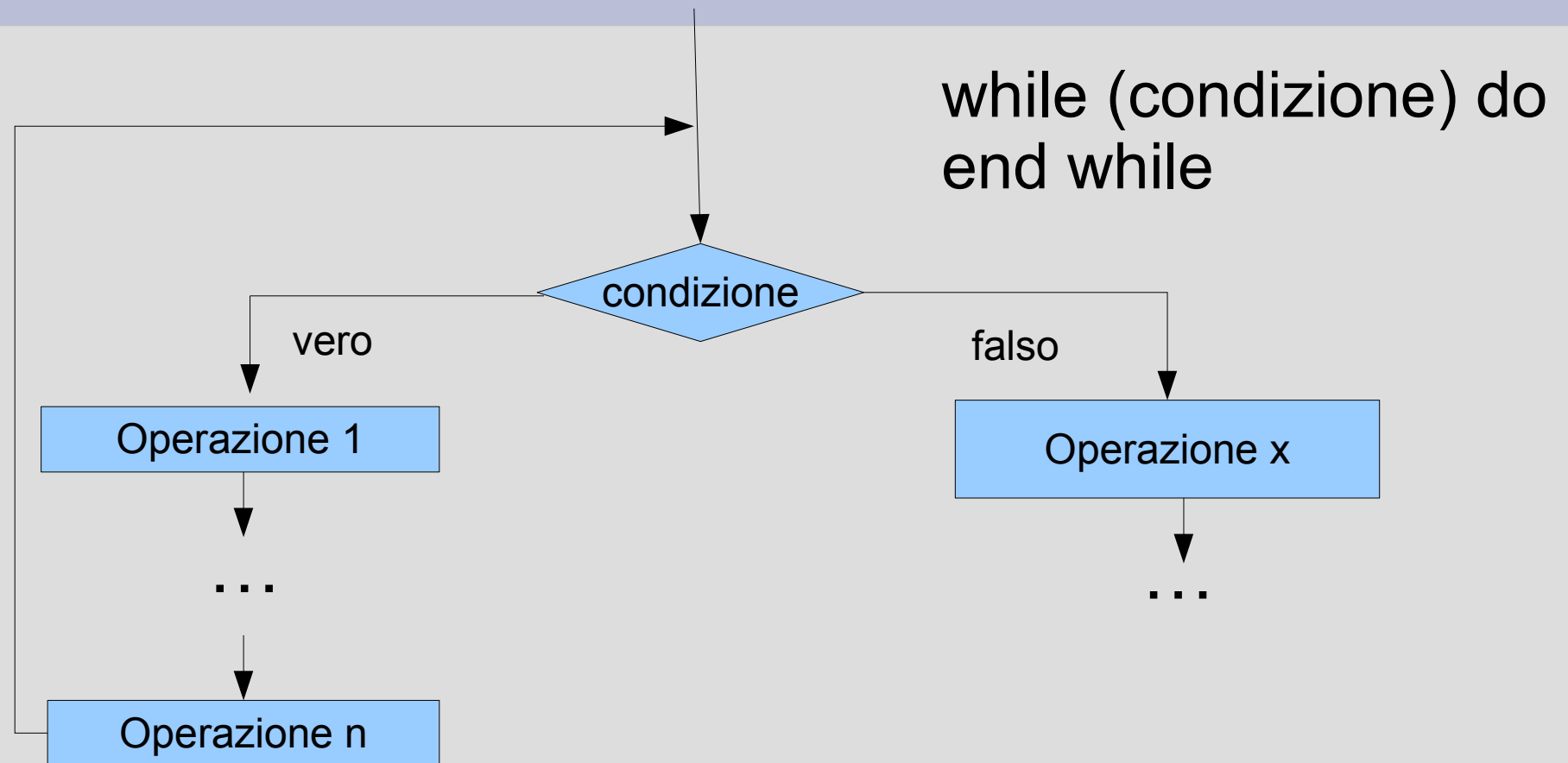


Diagramma di flusso: ciclo while



Gli pseudocodici

- Forma di scrittura del codice affine alla lingua parlata dal programmatore, è simile alla scaletta che ci si prepara nella scrittura di una relazione
- Esempio:

```
Leggi la quantità Q
Leggi l'importo unitario I
Leggi lo sconto S1
Leggi lo sconto S2
Leggi l'aliquotà IVA V
Se  $Q > 100$ 
    Sconto =  $Q * I * S1$ 
Altrimenti
    Sconto =  $Q * I * S2$ 
 $PN = Q * I - \text{Sconto}$ 
 $IVA = PN * V$ 
Prezzo =  $PN + IVA$ 
Stampa Prezzo
```

Terza fase: la codifica del programma

- Scegliere un linguaggio di programmazione
- Esempio:

```
PRINT "Inserisci quantità"  
INPUT Q  
PRINT "Inserisci importo unitario"  
INPUT I  
PRINT "Inserisci sconto S1"  
INPUT S1  
PRINT "Inserisci sconto S2"  
INPUT S2  
PRINT "Inserisci aliquota IVA"  
INPUT V  
IF Q > 100 THEN Sconto = Q * I * S1  
ELSE Sconto = Q * I * S2  
PN = Q * I - Sconto  
IVA = PN * V  
Prezzo = PN + IVA  
PRINT Prezzo  
END
```

Esercizi:

- Dato un numero intero, calcolare se è numero primo e stampare di conseguenza
- Dato in input un array di 10 elementi, stampare gli elementi ordinati in ordine crescente
 - Prendere in input i 10 valori e salvarli in un vettore
 - Ordinare gli elementi
 - Stampare elementi ordinati

Esercizio per la prossima volta

- Conversione base:
 - Vogliamo un algoritmo che operi la conversione di un numero da una rappresentazione in base 2 alla rappresentazione in base 10.
 - $\text{valore} = C_{n-1} \cdot b^{n-1} + \dots + C_1 \cdot b^1 + C_0 \cdot b^0$
 - Elevamento a potenza $x \wedge y = x$ elevato alla y

Programmare per il Web: HTML

- HTML: HyperText Markup Language
- Sistema per descrivere e strutturare un documento ipertestuale
- Browser leggono markup e ci presentano le pagine come le vediamo
- Attraverso dei tag nascosti tra parentesi angolari descrive precisamente ogni aspetto del documento (colori, stili, posizione, immagini, link, ...)

Come è fatta una pagina Web

- Per visualizzare il codice HTML:
 - In Internet Explorer:
 - Menù visualizza
 - “HTML”
 - In Mozilla Firefox:
 - Menù visualizza
 - “Sorgente pagina”
 - In Google Chrome:
 - Menù strumenti
 - “Visualizza sorgente”

Documento HTML

- Editor di testo o applicazioni (NVU, EditPad, FrontPage, Dreamweaver, Quanta)
- Salvare il documento in formato “solo testo” o “ASCII”, con l'estensione “.html” o “.htm”

Struttura di un documento HTML

`<html>`

`<head>`

`</head>`

`<body>`

`</body>`

`</html>`

Tag di chiusura,
segno “/”

The diagram illustrates the structure of an HTML document. It lists the tags: `<html>`, `<head>`, `</head>`, `<body>`, `</body>`, and `</html>`. Arrows point from the closing tags `</head>`, `</body>`, and `</html>` to the text 'Tag di chiusura, segno "/>

- Tag **html** identifica il documento
- Tag **head** racchiude l'intestazione
- Tag **body** la parte del documento visualizzata dal browser

HTML: title e contenuto

- Per aggiungere un titolo alla finestra:
`<title>Prova titolo</title>`
- Questo tag va inserito all'interno del tag head
- Per scrivere all'interno della finestra del browser:
`<body>Sto scrivendo nella finestra</body>`
- Esempio pagina.html

HTML: collegamenti e immagini (1)

- Per aggiungere un collegamento (link) ad un sito
`<a href="http://docenti.unimc.it/docenti/daniele-benedetti/2012/laboratorio-informatico-2012">Pagina del corso</a>`
`<a href="pagina.html">Vai al primo esempio</a>`
`<a href="mailto:d.benedetti@alice.it">Contattami</a>`
- Per aggiungere un collegamento (link) ad un file
`<a href="sms.gif">Immagine sms</a>`
`<a href="presentazione 3010.pdf">Presentazione</a>`
`<a href="presentazione.zip">Download Presentazione</a>`
- Esempio pagina1.html

HTML: collegamenti e immagini (2)

- Per aggiungere un'immagine

`` ← Senza tag di chiusura

Attributi WIDTH (larghezza in pixel), HEIGHT (altezza in pixel)

- Per aggiungere spazi/righe vuote

racchiudere paragrafo: `<p></p>`

riga vuota: `<br>` ←

- Esempio pagina2.html, pagina3.html

Attributi del tag body

- Sfondo della pagina:

`<BODY BGCOLOR= "green">` ← sfondo verde

Esempio pagina5.html

`<BODY BACKGROUND="sms.gif">` ← immagine di sfondo

Esempio pagina6.html

- Colore del testo:

`<BODY TEXT="red">` ← colore testo rosso

Esempio pagina7.html

Paragrafi: tag br

- Tag **br**:
 - è un'interruzione di riga che non spezza il paragrafo.
 - Testo seguente con tutte le caratteristiche del testo precedente.
 - Nessuno spazio vuoto tra le righe.
 - Non richiede tag di chiusura

Paragrafi: tag p

- Tag p:
 - Interruzione di paragrafo
 - Testo successivo va a capo
 - Riga di spazio vuoto
 - Tag di chiusura non obbligatorio, ma consigliato

Paragrafi: tag div

- Tag **div**:
 - Blocco di testo che va a capo
 - Può avere formattazioni diverse dal testo precedente
 - Non crea spazio vuoto tra il nuovo testo e il precedente
 - Ha il tag di chiusura

Paragrafi: intestazioni

- Tag **H**:
 - Usati per dare un titolo ad una sezione
 - Definiti in 6 livelli, specificati dal numero di seguito alla H (H1, H2, ..., H6)

Allineamento del testo

- ALIGN è un attributo del tag p e del tag div
- I possibili valori di ALIGN sono:
 - left = allineamento a sinistra (default)
 - center = allineamento al centro
 - right = allineamento a destra
 - justify = allineamento giustificato
- Esempio paragrafo.html

Formattazione testo

- Grassetto: `<B></B>`
- Corsivo: `<I></I>`
- Sottolineato: `<u></u>`
- Barrato: `<strike></strike>`
- Apice: `<sup></sup>`
- Pedice: `<sub></sub>`
- Esempio pagina4.html

Gestione del testo

- `<FONT FACE="tipo di carattere"> testo </FONT>`
- `<FONT COLOR="colore"> testo </FONT>`
- `<FONT SIZE="valore"> testo </FONT>` (da 1 a 7)
- Esempio pagina8.htm

Elenchi (non ordinato)

- Tag **UL**:
 - Elenco puntato
 - Si chiude con `</UL>`
 - Ogni voce dell'elenco è identificata con `<LI>`
 - Attributo `TYPE=` “nome simbolo” per scegliere un simbolo personalizzato
 - `disc` = punto pieno (default)
 - `circle` = punto vuoto
 - `square` = quadrato nero
- Esempio `pagina9.html`

Elenchi (ordinato)

- Tag **OL** (Esempio pagina10.html):
 - Elenco numerato
 - Si chiude con `</OL>`
 - Ogni voce dell'elenco è identificata con `<LI>`
 - Attributo `TYPE=` “tipo numerazione” per scegliere numeri o lettere
 - 1 = numeri arabi (default)
 - I = numeri romani maiuscoli
 - i = numeri romani minuscoli
 - A = lettere maiuscole
 - a = lettere minuscole
 - Attributo `START=` “numero” per far iniziare il conteggio da numero

Le tabelle

```
<TABLE BORDER =2 WIDTH=600>
```

```
<TR>
```

```
<TD VALIGN="top"> cella 1 </TD>
```

```
<TD> cella 2 </TD>
```

```
</TR>
```

```
<TR>
```

```
<TD> cella 3 </TD>
```

```
<TD> cella 4 </TD>
```

```
</TR>
```

```
</TABLE>
```

WIDTH anche in
percentuale, esempio
WIDTH=90%

Le tabelle: attributi

- ALIGN= “valore”: allineamento (sia su table che su td)
- VALIGN= “valore”: allineamento verticale (valido su td), top, middle, bottom
- WIDTH= “valore”: larghezza tabella (sia su table che su td)
- HEIGHT= “valore”: altezza cella (valido su td)
- BORDER= “valore”: spessore della tabella
- BGCOLOR= “colore”: colore sfondo tabella (table e td)
- BACKGROUND= “immagine”: immagine di sfondo (table e td)
- CELLSPACING= “numero pixel”: spaziatura tra celle
- CELLPADDING= “numero pixel”: spaziatura interna della cella tra margine e contenuti
- ROWSPAN= “numero” E COLSPAN= “numero”: unire celle (su td)

Homework

- Con gli elementi visti a lezione costruire una pagina (o più pagine collegate tra loro) di un sito web a vostro piacimento
- Consegna via e-mail entro il 20 novembre