



Università degli Studi di Bologna  
Facoltà di Ingegneria

## Corso di Fondamenti di Informatica L-B

*Corso di Laurea in Ingegneria  
Elettronica e dell'Automazione*

Prof. Enrico Lodolo

Anno accademico 2002/2003

Introduzione 1

## Contenuti acquisiti (I PARTE)

### Costruzione di componenti software

- algoritmi e linguaggi di *programmazione*
- metodologie per il *progetto* della soluzione di problemi *su piccola scala*
- Linguaggio C

### Programmazione nel piccolo (in-the-small)

Introduzione 2

## Brevi Richiami ..... Risoluzione dei Problemi

La risoluzione di un problema è il processo che dato un problema, e individuato un opportuno metodo risolutivo trasforma i dati iniziali nei corrispondenti risultati finali.

Affinché la risoluzione di un problema possa essere realizzata attraverso l'uso del calcolatore, tale processo deve poter essere definito come sequenza di azioni elementari.

Introduzione 3

## Algoritmo e Programma

Un algoritmo è una sequenza **finita** di mosse che risolve **in un tempo finito** una classe di problemi.

L'esecuzione delle azioni nell'ordine specificato dall'algoritmo consente di ottenere, a partire dai dati di ingresso, i risultati che risolvono il problema

Dato un algoritmo, un **programma** è la sua **descrizione in un particolare linguaggio** di Programmazione

PROGRAMMA = DATI + CONTROLLO

Introduzione 4

## Metodologie di Progettazione

### Metodologia Top-Down:

Procede per **decomposizione** del problema in sottoproblemi, per **passi di raffinamento successivi**. Si scompone il problema in sottoproblemi. Si risolve ciascun sottoproblema con lo stesso metodo, fino a giungere a sottoproblemi risolvibili con mosse elementari

### Metodologia Bottom-Down:

Procede per **composizione di componenti e funzionalità elementari**, fino alla **sintesi** dell'intero algoritmo ("dal dettaglio all'astratto")

Introduzione 5

## FUNZIONI COME COMPONENTI SOFTWARE

- Una funzione è un *componente software (servitore) riutilizzabile*
- che costituisce una unità di traduzione:
  - può essere definita in un unico file e compilata per proprio conto
  - pronta per essere usata da chiunque

Introduzione 6

## FUNZIONI COME COMPONENTI SOFTWARE

Per usare tale componente software, il cliente:

- non ha bisogno di sapere *come è fatto* (cioè, di conoscerne la *definizione*)
- deve conoscerne solo l'interfaccia:
  - nome
  - numero e tipo dei parametri
  - tipo del risultato

Introduzione 7

## DICHIARAZIONE DI FUNZIONE

La dichiarazione di una funzione è costituita dalla sola interfaccia, *senza corpo* (sostituito da un `;`)

<dichiarazione-di-funzione> ::=  
<tipoValore> <nome> (<parametri>) ;

Introduzione 8

## DICHIARAZIONE DI FUNZIONE

```
int max(int a, int b)
{ if (a>b) return a;
  else return b;
}
```

DEFINIZIONE

```
int max(int a, int b);
```

DICHIARAZIONE  
o *prototipo*  
o *interfaccia*

Introduzione 9

## DICHIARAZIONE DI FUNZIONI

Dunque,

- per usare una funzione non occorre conoscere tutta la *definizione*
- basta conoscere la *dichiarazione*, perché essa specifica proprio il *contratto di servizio*

Introduzione 10

## DICHIARAZIONE DI FUNZIONI

- La definizione di una funzione costituisce l'effettiva realizzazione del componente
  - Dice come è fatto il componente
- La dichiarazione specifica il contratto di servizio fra cliente e servitore, esprimendo le proprietà essenziali della funzione.
  - Dice come si usa il componente
  - Per usare una funzione non è necessario sapere come è fatta, anzi è controproducente

Introduzione 11

## DICHIARAZIONE DI FUNZIONI

- La dichiarazione specifica:
  - il nome della funzione
  - numero e tipo dei parametri (non necessariamente il nome)
  - il tipo del risultato

Il nome avrebbe significato *solo nell'environment della funzione, che qui non c'è!*

Introduzione 12

## DICHIARAZIONE vs. DEFINIZIONE

- La definizione di una funzione costituisce l'effettiva realizzazione del componente
  - Non può essere duplicata
  - Ogni applicazione deve contenere una e una sola definizione per ogni funzione utilizzata
  - La compilazione della definizione genera il codice macchina che verrà eseguito ogni volta che la funzione verrà chiamata.

Introduzione 13

## DICHIARAZIONE vs. DEFINIZIONE

- La dichiarazione di una funzione costituisce solo una specifica delle proprietà del componente:
  - Può essere duplicata senza danni
  - Un'applicazione può contenerne più di una
  - La compilazione di una dichiarazione non genera codice macchina

Introduzione 14

## DICHIARAZIONE vs. DEFINIZIONE

- La definizione è *molto più* di una dichiarazione  
definizione = dichiarazione + corpo



La definizione funge anche da dichiarazione (ma non viceversa)

Introduzione 15

## CONTENUTI (II PARTE)

### Costruzione di sistemi software

#### Programmazione nel largo (in-the-large)

- Progettazione di *sistemi* per composizione modulare di *componenti software*
- metodologie e linguaggi *a oggetti*
- applicazioni diverse come esempi  
*ad esempio grafica, file*
- **Linguaggio Java**

Introduzione 16

## Ambiente di programmazione

### Dagli *Algoritmi* ai *Sistemi Software*

- **Approccio a oggetti, linguaggio Java**
  - Componenti Software
  - Metodologie e linguaggi *a oggetti*
  - Java: Linguaggio e Architettura
  - Concetti base della programmazione a oggetti: *incapsulamento, composizione, ereditarietà, ...*
  - Interfacce grafiche e programmazione "event-driven"
  - Strutture dati dinamiche

Introduzione 17

## OBIETTIVI

- Conoscere i principi e gli strumenti di un supporto ad oggetti per progettare soluzioni a piccoli problemi con dimensioni crescenti
- Progettare *un singolo componente software*
- Progettare *un (piccolo) sistema software* usando componenti esistenti e aggiungendone nuovi

Introduzione 18

## VALUTAZIONE

La valutazione finale del modulo fatta con una prova di laboratorio integrata

- Progetto
  - Risoluzione di un problema mediante costruzione di un componente in Java
  - Produzione del programma eseguibile
- Conoscenza
  - Conoscenza delle parti "teoriche" di progetto

Introduzione 19

## VALUTAZIONE in ITINERE

### Laboratori

- Ogni sessione ha un obiettivo
  - Assimilazione degli elementi delle parti del corso con una prova di progetto
  - Produzione di programmi eseguibili
- Conoscenza
  - Comprensione (in crescita) delle parti del corso

Introduzione 20

## PREREQUISITI...

- Conoscenze del modulo precedente
- Manualità da migliorare

## LABORATORIO

- Attività di esercitazione assistita da tutore
- Attività di esercitazione libera

Introduzione 21

## LABORATORIO

- ORARI LAB:
- Turno 1 (A-K) - Martedì 9-11
- Turno 2 (L-Z) - Martedì 11-13

Introduzione 22

## INFORMAZIONI UTILI

- Ricevimento studenti
    - Giovedì 17-18
  - Posta elettronica
    - [elodolo@deis.unibo.it](mailto:elodolo@deis.unibo.it)
  - Telefono e Fax
    - 051 371018
  - Tutor
    - Ing. Stefano Contadini
    - Posta elettronica: [s.contadini@quindi.it](mailto:s.contadini@quindi.it)
- Usare il laboratorio per tutta la durata del corso

Introduzione 23

## IL SITO WEB DEL CORSO

<http://lia.deis.unibo.it/Courses/FondB0203-ELE/>

- Il vostro punto di riferimento per
  - materiale didattico (lezioni, esercizi)
  - software gratuito
  - testi degli esami e loro soluzione
- Iscrizione agli esami e esiti delle prove
  - via Internet

**uniwex.unibo.it**

Introduzione 24

## AMBIENTI DI PROGRAMMAZIONE

- Linguaggio Java
  - Strumenti a linea di comando
    - Sun JDK 1.3** ("Java 2 platform")
  - Ambiente di sviluppo **Borland JBuilder**

Introduzione 25

## ESERCITAZIONI DI LABORATORIO

- **Sala Terminali del Centro di Calcolo**  
(seminterrato dell'edificio "aule nuove")
  - **Martedì 9-11 (A-K) e 11-13 (L-Z)**
  - ..e in ogni momento in cui ci siano postazioni libere
- **Ricordate**
  - all'ingresso del laboratorio è **obbligatorio** lasciare un documento al personale

Introduzione 26

## TESTI DI RIFERIMENTO

- **Diapositive proiettate a lezione**
  - consultabili sul sito Web
  - disponibili al centro fotocopie della biblioteca
- **Manuali Linguaggio Java**
  - John Lewis - William Loftus  
Java - Fondamenti di progettazione software  
Addison - Wesley
  - H.M. Deitel, P.J. Deitel: "Java, Fondamenti di Programmazione", Apogeo, Milano, 2000
- **Generali**
  - D. Flanagan: "Java in a nutshell", seconda edizione, O'Reilly
  - B. Eckel: "Thinking in Java", seconda edizione, Prentice-Hall

Introduzione 27