

Corso di Laurea in INFORMATICA

Algoritmi e Strutture Dati a.a. 2012-2013

MODULO 1

L'Astrazione in programmazione: dall'algoritmo ai tipi di dati astratti

**Presentazione del corso. Il ruolo delle tecniche di
astrazione nel progetto di programmi**

Questi lucidi sono stati preparati da per uso didattico. Essi contengono materiale originale di proprietà dell'Università degli Studi di Bari e/o figure di proprietà di altri autori, società e organizzazioni di cui e' riportato il riferimento. Tutto o parte del materiale può essere fotocopiato per uso personale o didattico ma non può essere distribuito per uso commerciale. Qualunque altro uso richiede una specifica autorizzazione da parte dell'Università degli Studi di Bari e degli altri autori coinvolti.



OBIETTIVI FORMATIVI

- ❑ Viene introdotta una metodologia di progetto formale basata sui *tipi di dati astratti* (ADT) come base per le tecniche di programmazione orientata ad oggetti.
- ❑ Oltre alla capacità di rappresentare in modo formale diversi tipi di problemi, saranno acquisiti i rudimenti della programmazione per classi attraverso la realizzazione di dati astratti in ambienti di programmazione orientati ad oggetti.
- ❑ Saranno acquisiti i principi della algoritmica, presentati in funzione del modo di utilizzo dello spazio di ricerca: le caratteristiche dei paradigmi selettivo e generativo verranno evidenziate attraverso diversi algoritmi fondamentali.



OBIETTIVI

PROFESSIONALIZZANTI

- ❑ **Capacità di integrare le tecniche di astrazione funzionale con quelle di astrazione dati nella progettazione di programmi per la soluzione di un problema, individuando**
 - **le strutture dati più opportune,**
 - **il metodo solutivo e la tecnica algoritmica appropriata,**
 - **le tecniche di realizzazione più efficienti in un linguaggio di programmazione di riferimento, valutandone i costi ed i benefici in termini di complessità di calcolo.**
- ❑ **Capacità di implementare diverse realizzazioni degli operatori relativi a strutture dati non primitive in un linguaggio imperativo e in uno orientato ad oggetti.**



PROGRAMMA

1. L'astrazione in programmazione
2. Astrarre le strutture di dati: le Algebre di dati
3. Strutture lineari di dati: liste, pile e code
4. Cenni sulla analisi della complessità computazionale
5. Insiemi e Dizionari
6. Alberi binari
7. Code con priorità
8. Alberi n-ari
9. Grafi
10. Le tecniche algoritmiche
11. Esercitazioni su strutture e algoritmi



TESTI DI RIFERIMENTO

A. Bertossi, A. Montresor

Algoritmi e strutture di dati. CittàStudi, 2010

M. Cadoli, M. Lenzerini, P. Naggar, A. Schaerf .

Fondamenti della progettazione dei programmi. Città studi Edizioni, 1997.

C. Demetrescu, I. Finocchi, G. F. Italiano

Algoritmi e strutture dati 2/ed

Seconda edizione, McGraw-Hill, 2008.

P. Crescenzi, G. Gambosi, R. Grossi.

Strutture di Dati e Algoritmi, Addison-Wesley Pearson, 2/ed 2012

T. Cormen, C. Leiserson, R. Rivest, C. Stein.

Introduzione agli algoritmi e strutture dati.

Seconda edizione, McGraw-Hill, 2005.

Clifford A. Shaffer

Data Structures and Algorithm Analysis



Edition 3.2 (C++ Version) 2012

<http://people.cs.vt.edu/~shaffer/Book>

I PREREQUISITI A QUESTO CORSO

IL CORSO PREVEDE CHE SIANO GIA' ACQUISITI:

- *I PRINCIPI DELLA PROGRAMMAZIONE STRUTTURATA E UN LINGUAGGIO DI PROGRAMMAZIONE IMPERATIVA;*
- *IL CONCETTO DI TIPO DI DATI E I PRINCIPI DELLA PROGRAMMAZIONE MODULARE;*
- *I PRINCIPI DELLA ASTRAZIONE FUNZIONALE (FUNZIONI E PROCEDURE) E I MECCANISMI DI PASSAGGIO DEI PARAMETRI.*



I PREREQUISITI A QUESTO CORSO

- ***I PRINCIPI DELLA PROGRAMMAZIONE STRUTTURATA E UN LINGUAGGIO DI PROGRAMMAZIONE IMPERATIVA;***

Linguaggi Imperativi

Un linguaggio imperativo è legato all'architettura della macchina di Von Neumann, ma ad un livello di astrazione vicino a quello dei diagrammi di flusso.

La principale attività del processore è assegnare valori a celle di memoria. Le operazioni che effettuano le istruzioni di un linguaggio imperativo si basano sul concetto di variabile.

Astrazione del concetto di locazione di memoria



I PREREQUISITI A QUESTO CORSO

•IL CONCETTO DI TIPO DI DATI E I PRINCIPI DELLA PROGRAMMAZIONE MODULARE;

In programmazione un **MODULO** è caratterizzato da:

- la struttura interna, cioè l'insieme dei tipi, delle variabili e delle funzioni definiti nel modulo stesso;
- l'insieme dei servizi che esporta, ovvero che offre agli altri moduli;
- la modalità con cui tali servizi possono essere utilizzati (interfaccia del modulo);
- l'insieme dei servizi che esso importa dagli altri moduli e che utilizza per le sue funzioni.



I PREREQUISITI A QUESTO CORSO

- ***I PRINCIPI DELLA ASTRAZIONE FUNZIONALE (FUNZIONI E PROCEDURE) E I MECCANISMI DI PASSAGGIO DEI PARAMETRI.***

L'ASTRAZIONE FUNZIONALE è la tecnica che permette di potenziare il linguaggio di programmazione disponibile introducendo **NUOVI OPERATORI** che fanno uso degli **OPERATORI BASE**, già disponibili. La **INTESTAZIONE** del sottoprogramma/funzione introduce nome dell'operatore/funzione e argomenti mentre il **CORPO** ne descrive il comportamento.



VERRA' APPROFONDITO IL CONCETTO DI ALGORITMO COME STRUMENTO PER RISOLVERE UN BEN DEFINITO PROBLEMA DI CALCOLO.

SI INTRODURRA' IL CONCETTO DI ISTANZA DEL PROBLEMA SI STUDIERA' LA CONNESSIONE TRA ALGORITMO E ORGANIZZAZIONE DEI DATI, SI DESCRIVERANNO GLI ALGORITMI IN PSEUDOCODICE, SI INTRODURRANNO I CONCETTI DI ANALISI DEGLI ALGORITMI.



**LA PROGRAMMAZIONE MODULARE
IMPARATA NEL I° ANNO (PROGETTAZIONE
ORIENTATA ALLE FUNZIONI), TOP-DOWN E
BASATA SULLA DECOMPOSIZIONE DI
PROBLEMI IN SOTTO-PROBLEMI, VERRA'
SOSTITUITA DALLA PROGRAMMAZIONE
ORIENTATA AD OGGETTI.**

**SI UTILIZZERA' UNA DIVERSA METODOLOGIA
DI PROGETTO PASSANDO ATTRAVERSO UNA
TECNICA FORMALE BASATA SULLA
ASTRAZIONE**



NELLA **PROGETTAZIONE DEI PROGRAMMI** SI DEVE TENERE CONTO DI:

- **PRINCIPI**,

CIOE' LE LINEE GUIDA PER PRODURRE
SOFTWARE DI QUALITA'

- **TECNICHE**,

CIOE' I METODI PER PRODURRE
PROGRAMMI COERENTI CON I PRINCIPI

- **STRUMENTI**,

COME I LINGUAGGI DI PROGRAMMAZIONE O,
IN GENERE, GLI AIUTI DI VARIA NATURA
UTILIZZATI NELLA PROGETTAZIONE



***TRA I PRINCIPI DI PROGETTAZIONE
RICORDIAMO LA***

A S T R A Z I O N E

**CHE DA' LUOGO A METODOLOGIE PER LA
PRODUZIONE DI PROGRAMMI DEFINITE**

TECNICHE DI ASTRAZIONE

**(PROGRAMMAZIONE STRUTTURATA,
MODULARIZZAZIONE, ASTRAZIONE DATI,
etc.) CHE HANNO SEGNATO LA
PROGRAMMAZIONE NEGLI ULTIMI 40 ANNI.**



IN GENERALE LA ASTRAZIONE

E' la capacità di costruire teorie astratte a partire da dati empirici ed è una caratteristica fondamentale propria degli esseri umani.

In particolare il pensiero filosofico ha individuato nel tempo molte **categorie astratte** come per esempio il *bello*, il *bene*, il *vero*, il *giusto* etc.

Le **categorie** sono rappresentate e comunicate attraverso il linguaggio e lo sforzo è finalizzato al definire con precisione il significato di ciascuna categoria, disponendo di un alfabeto di **simboli**.

Il simbolo riassume in sé un intero insieme di conoscenze e riferimenti. La capacità di leggere e interpretare i simboli costituisce una potente modalità di ragionamento.

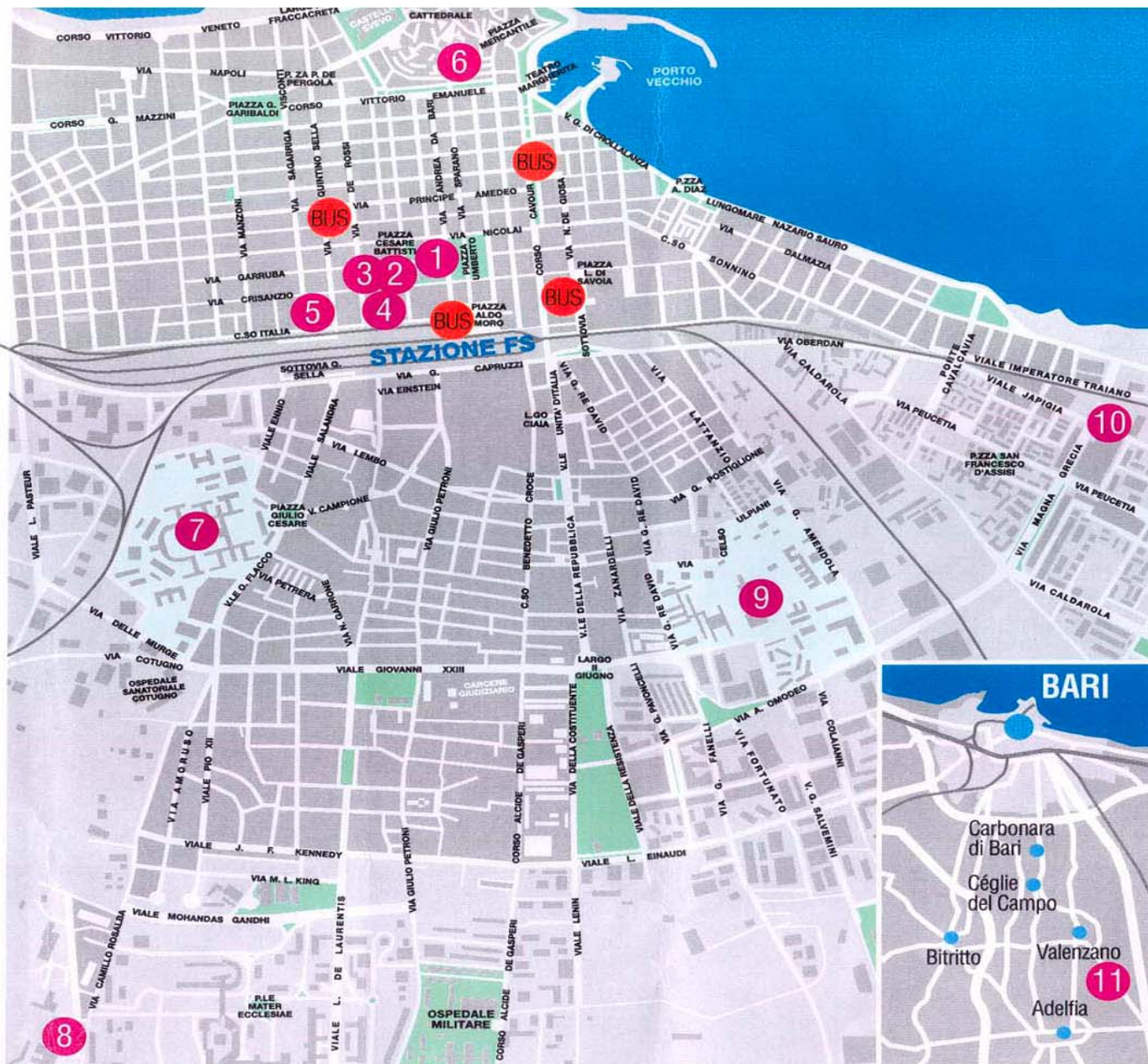
 Costruire una **astrazione** significa costruire un modello



BARI PLESSI UNIVERSITARI

FACOLTÀ E DIPARTIMENTI

- 1 PALAZZO ATENEO
 - Facoltà di Lettere e Filosofia
 - Facoltà di Scienze della Formazione
 - Piazza Umberto I, 1*
- 2 Facoltà di Lingue e Letterature Straniere
Via Garruba, 6/B
- 3 Facoltà di Giurisprudenza
Piazza Cesare Battisti, 1
- 4 Dipartimento di Linguistica, Letteratura e Filologia Moderna
Via De Rossi, 233
- 5 Dipartimento di Scienze Storiche e Geografiche
Via Quintino Sella, 268
- 6 Facoltà di Studi Classici e Cristiani
Strada Torretta, 6
- 7 POLICLINICO
Facoltà di Medicina e Chirurgia
Piazza Giulio Cesare, 11
BUS nn. 20, 27 da Via Q. Sella
- 8 Facoltà di Economia
Via Camillo Rosalba, 53
BUS nn. 6, 27 da Via Q. Sella
- 9 CAMPUS
 - Facoltà di Scienze MM. FF. NN.
Via Orabona, 4
 - Facoltà di Agraria
Via Amendola, 165/A
 - Facoltà di Farmacia
Via Amendola, 173
BUS n. 22 da P.zza A. Moro
- 10 Facoltà di Medicina Veterinaria
Istituti di:
 - Anatomia Normale
 - Anatomia Patologica
 - Patologia Aviare
V. Caduti di tutte le guerre, 1
BUS n. 12/ da P.zza A. Moro
BUS n. 2/ da C.so Cavour
- 11 Facoltà di Medicina Veterinaria
Str. Prov. per Casamassima Km 3 (Valenzano)
BUS n. 4 da P.zza L. di Savoia



UN' ASTRAZIONE CHE USIAMO QUOTIDIANAMENTE

Astrazione

E' il processo attraverso il quale ci si dimentica di una parte dell'informazione

– effetto

- cose che sono diverse diventano uguali**

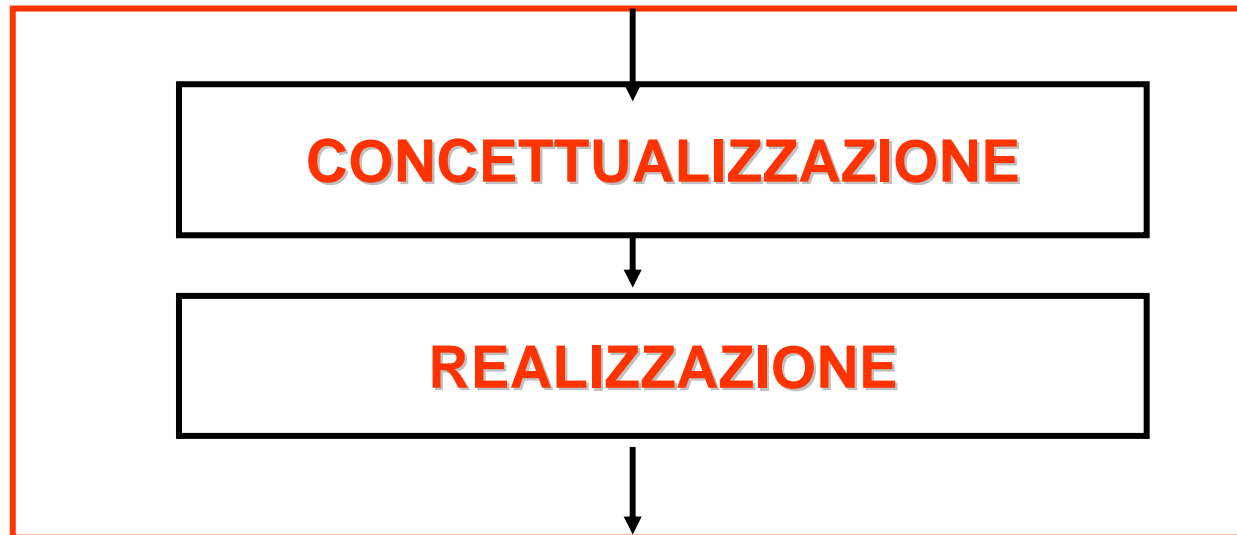
– perché?

- perché si spera di semplificare l'analisi, separando gli attributi che si ritengono rilevanti da quelli che si ritiene possano essere trascurati**
- la rilevanza dipende dal contesto**



I PRINCIPI DI ASTRAZIONE NELLA PROGETTAZIONE DEI PROGRAMMI

**DISTINZIONE TRA IL LIVELLO DI
CONCETTUALIZZAZIONE E QUELLO DI
REALIZZAZIONE**



**NELLA CONCETTUALIZZAZIONE SI ASTRAE PER
DECOMPORRE, RAPPRESENTARE, GENERALIZZARE
E SI ASTRAE SUI DATI E SULLE FUNZIONI**



LE ATTIVITA' RELATIVE ALLA FASE DI

CONCETTUALIZZAZIONE

- **INDIVIDUAZIONE DELLA SOLUZIONE AL PROBLEMA**
- **SPECIFICA DEGLI OGGETTI DA TRATTARE**
- **IDEAZIONE DELLE SCELTE ALGORITMICHE**

IL RISULTATO E' LO “**SCHEMA CONCETTUALE DI PROGETTO**”, OVVERO *la specifica* DI COSA DEVE FARE QUELLO CHE CI ACCINGIAMO A COSTRUIRE

LA FASE DI

REALIZZAZIONE

SI RIFERISCE A QUEL PROCESSO CHE, PARTENDO DALLO SCHEMA CONCETTUALE DI PROGETTO, PERMETTE DI OTTENERE QUANTO “**REALIZZA**” QUELLO CHE E' PREVISTO NELLO SCHEMA

(UNO SPECIFICO **PROGRAMMA** IN UN DEFINITO **LINGUAGGIO DI PROGRAMMAZIONE**).



L'astrazione nella concettualizzazione

NELLA INDIVIDUAZIONE DELLA **STRATEGIA SOLUTIVA** DI UN PROBLEMA SI **ASTRAE** NEL MOMENTO IN CUI SI SCEGLIE UN METODO.

SI CERCA DI RENDERE GENERALE UN METODO SOLUTIVO MESSO A PUNTO PER UN PROBLEMA PARTICOLARE ATTRAVERSO UN PROCESSO DI **GENERALIZZAZIONE**

Primo passo nella è la definizione del problema che va espresso in termini di **obiettivo da raggiungere** partendo da una **situazione iniziale**.

In termini più formali un

Problema computazionale

è la *relazione (formale)* che intercorre fra l'input e l'output desiderato



Problema computazionale: esempi

Minimo

Il minimo di un insieme A è l'elemento di A che è minore o uguale ad ogni elemento di A

$$\min(A) = a \Leftrightarrow \forall b \in A : a \leq b$$

Ricerca

Sia $A = a_1, \dots, a_n$ una sequenza di dati ordinati e distinti, $a_1 < a_2 < \dots < a_n$. Eseguire una ricerca della posizione di un dato v in A consiste nel restituire l'indice corrispondente, se v è presente, oppure 0, se v non è presente

$$ricerca(A, v) = \begin{cases} i & \exists i \in \{1, \dots, n\} : a_i = v \\ 0 & \text{altrimenti} \end{cases}$$

L'astrazione nella concettualizzazione

E' fondamentale la definizione dell'**Algoritmo** che implica:

- o La *individuazione* di una *sequenza di azioni* che un esecutore deve compiere per giungere alla soluzione di un problema
- o La individuazione del modo in cui *descrivere le azioni, rappresentare e organizzare* input, output e tutti i dati intermedi necessari per lo svolgimento

Esempio

Input: ingredienti
Algoritmo: ricetta

Output: piatto cucinato
Esecutore: cuoco

Possibili metodi solutivi (algoritmi)

Minimo

- ✦ Per trovare il minimo di un insieme, confronta ogni elemento con tutti gli altri; l'elemento che è minore di tutti è il minimo.

Ricerca

- ✦ Per trovare un valore v nella sequenza A , confronta v con tutti gli elementi di A , in ordine, e restituisci la posizione corrispondente; restituisci 0 se nessuno degli elementi corrisponde.

Algoritmi

Poiché il metodo solutivo non è unico si pone il problema della

Valutazione

Esistono algoritmi “migliori” per risolvere lo stesso problema?

Dobbiamo definire il concetto di migliore

Ricerca in un array ordinato

Problema

- ✦ Dato un vettore A contenente n elementi, verificare se un certo elemento v è presente
- ✦ Esempi: elenco del telefono, dizionario

Una soluzione “banale”

- ✦ Scorro gli elementi in ordine, finché non trovo un oggetto “maggiore o uguale” a v

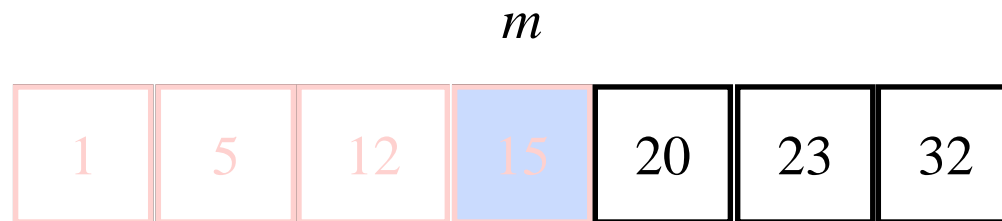
1	5	12	15	20	23	32
---	---	----	----	----	----	----

~~21~~

Ricerca in un array ordinato

Una soluzione efficiente

- Considero l'elemento centrale (indice m) del vettore
 - Se $A[m] = v$, ho finito
 - Se $v < A[m]$, cerco nella “metà di sinistra”
 - Se $A[m] < v$, cerco nella “metà di destra”

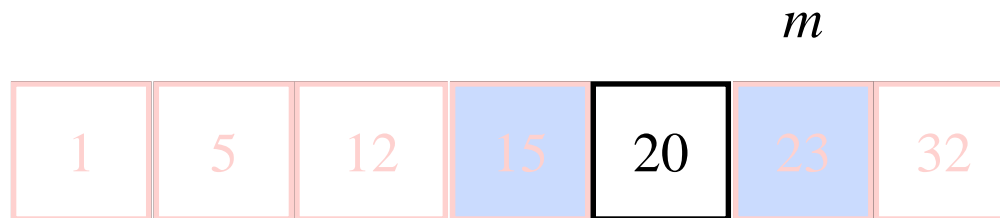


21?

Ricerca in un array ordinato

Una soluzione efficiente

- ✦ Considero l'elemento centrale (indice m) del sottovettore che sto analizzando:
 - ✦ Se $A[m]=v$, ho finito
 - ✦ Se $v < A[m]$, cerco nella “metà di sinistra”
 - ✦ Se $A[m] < v$, cerco nella “metà di destra”



21?

Ricerca in un array ordinato

Una soluzione efficiente

- ✦ Considero l'elemento centrale (indice m) del sottovettore che sto analizzando:
 - ✦ Se $A[m]=v$, ho finito
 - ✦ Se $v < A[m]$, cerco nella “metà di sinistra”
 - ✦ Se $A[m] < v$, cerco nella “metà di destra”



~~21?~~

Come descrivere un algoritmo

Particolare attenzione va dedicata alla descrizione del metodo solutivo e al livello di dettaglio

Da una ricetta di Pan di Spagna leggo:

“...montare le chiare a neve...” “incorporare il composto col resto degli ingredienti”

Cosa significa “montare le chiare a neve”? Cosa significa “incorporare il composto”?

E' necessario utilizzare una descrizione il più possibile formale, comprensibile e non ambigua.

La descrizione del metodo deve essere indipendente dal particolare linguaggio di programmazione che verrà scelto per la realizzazione, va, cioè, espressa in

“*Pseudo-codice*”

PSEUDOCODICE

CONVENZIONI

- ❑ L'INDENTAZIONE (RIENTRO VERSO DESTRA DELLE RIGHE) SERVE AD INDICARE LA STRUTTURA A BLOCCHI DELLO PSEUDOCODICE.
- ❑ I COSTRUTTI ITERATIVI SONO *while*, *for* (E *repeat*)
- ❑ I COSTRUTTI CONDIZIONALI SONO *if*, E *if.. then.. else*
- ❑ L'ASSEGNAZIONE E' INDICATA DA ←
- ❑ LE VARIABILI SONO DA INTENDERSI SEMPRE LOCALI A MENO DI ESPLICITA INDICAZIONE
- ❑ L'ELEMENTO DI UNA TABELLA INDICIZZATA E' INDICATO DA NOME SEGUITO DALL'INDICE IN PARENTESI
- ❑ GLI OPERATORI *and* E *or* SONO CORTOCIRCUITATI: SE SI VALUTA L'ESPRESSIONE "*x and y*" VA VALUTATO *x* E SE QUESTO E' FALSO NON VA VALUTATO *y*



PSEUDOCODICE

- $a \leftarrow b$
- $a \leftrightarrow b \quad \equiv \quad temp \leftarrow a; a \leftarrow b; b \leftarrow temp$
- **if** *condizione* **then** istruzione
- **if** *condizione* **then** istruzione1 **else** istruzione2
- **while** *condizione* **do** istruzione
- **for** *indice* \leftarrow *estremoInferiore* **to** *estremoSuperiore* **do** istruzione
- **for** *indice* \leftarrow *estremoSuperiore* **downto** *estremoInferiore* **do** istruzione

indice \leftarrow *estremoInferiore*

while *indice* \leq *estremoSuperiore* **do**

 | *istruzione*

 | *indice* \leftarrow *indice* + 1

indice \leftarrow *estremoSuperiore*

while *indice* \geq *estremoInferiore* **do**

 | *istruzione*

 | *indice* \leftarrow *indice* - 1

- $\text{iif}(\text{condizione}, v_1, v_2)$
- **foreach** *elemento* \in *insieme* **do** istruzione
- % *commento*
- **integer, real, boolean**
- **and, or, not**
- $+, -, \cdot, /, \lfloor x \rfloor, \lceil x \rceil, \log, x^2, \dots$

Esempio: ricerca del minimo in un vettore

❑ LE FUNZIONI, IDENTIFICATE DA UN NOME, SONO INDICATE SPECIFICANDO NOME E TIPO DEGLI ARGOMENTI

ITEM min(ITEM[] A , integer n)

ITEM $min \leftarrow A[1]$ % Minimo parziale

for integer $i \leftarrow 2$ to n do

 if $A[i] < min$ then
 $min \leftarrow A[i]$ % Nuovo minimo parziale

return min

Ricerca in un array ordinato

ITEM `binarySearch`(ITEM[] A , ITEM v , **integer** i , **integer** j)

```
if  $i > j$  then  
    | return 0  
else  
    | integer  $m \leftarrow \lfloor (i + j) / 2 \rfloor$   
    | if  $A[m] = v$  then  
    | | return  $m$   
    | else if  $A[m] < v$  then  
    | | return binarySearch( $A, v, m + 1, j$ )  
    | else  
    | | return binarySearch( $A, v, i, m - 1$ )  
    |
```


Problemi aperti sugli Algoritmi

- ♦ **Descrizione**

Passaggio dalla descrizione in linguaggio naturale, ad un linguaggio più formale

- ♦ **Valutazione**

L'algoritmo è corretto? Esistono algoritmi “migliori” per risolvere lo stesso problema?

- ♦ **Progettazione**

Problemi più complessi devono essere affrontati con opportune tecniche di programmazione

L'astrazione nella programmazione

Anche in **programmazione** vi è una fase di concettualizzazione: l'**astrazione** è un meccanismo presente nei **linguaggi ad alto livello** che consente di usare direttamente i costrutti del linguaggio invece che una delle numerosissime sequenze di istruzioni in linguaggio macchina “equivalenti”

Più in generale nella **programmazione**, l'**astrazione** allude alla distinzione che si fa tra:

- **cosa** fa un pezzo di codice
- **come** esso è implementato

L'utente del codice è interessato a conoscere solo cosa fa il codice e non i dettagli della implementazione.



Astrazione e linguaggi

Al fine di agevolare la progettazione, lo sviluppo e la manutenzione di sistemi software sempre più complessi, i **linguaggi di programmazione** offrono diverse tecniche di **astrazione**.

In linea di principio, un **linguaggio simbolico** è di per sé una astrazione della particolare macchina.

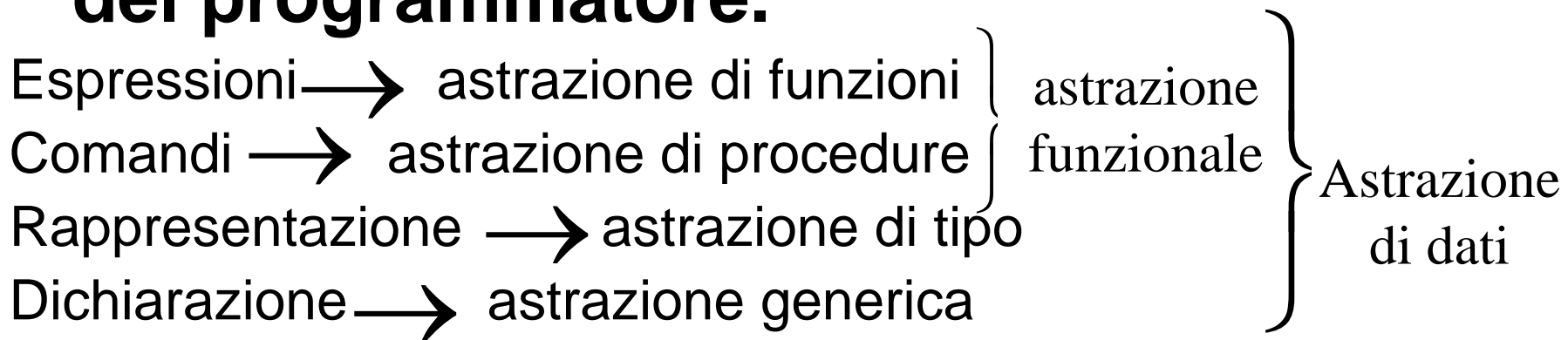
- Anni '50: introduzione di nomi mnemonici per le istruzioni e per l'accesso alla memoria (assembler)
- Anni '60: si inizia ad utilizzare l'astrazione come tecnica di organizzazione dei programmi: per es. tipi di dati predefiniti.



Astrazione e linguaggi

Alla fine degli anni '60 si cominciò a pensare che ***è possibile costruire astrazioni su una qualsiasi classe sintattica che sottintendesse una computazione.***

1967: definizione di tipi di dati da parte del programmatore.



Astrazione e linguaggi

Livello di astrazione



Programmazione	Dati	Algoritmi
Linguaggio macchina	Bits	Bits
Assemblers	Symbolic Words	Op-code
Compilatori	Variables & Types	Statements
Linguaggi strutturati	Data structures	Subroutines
Ada (Modula)	Abstract Data Types	Packages (Modules)
Object Oriented	Objects	Objects



Astrazione e linguaggi

**GLI ATTUALI LINGUAGGI DI PROGRAMMAZIONE
CONSENTONO DI:**

ASTRARRE SUI DATI

**LA ASTRAZIONE E' USATA PER COGLIERE GLI ASPETTI
ESSENZIALI DI UNA SITUAZIONE DEL MONDO REALE
ALLO SCOPO DI DEFINIRE OPPORTUNE STRUTTURE DI
RAPPRESENTAZIONE CHE EVIDENZINO ALCUNI ASPETTI
E NE TRASCURINO ALTRI. (*TABELLE, ELENCHI, ALBERI,
MAPPE, GRAFI, IMMAGINI etc.*).**

**SI FA RIFERIMENTO A STRUTTURE ALGEBRICO-
MATEMATICHE, CARATTERIZZATE DA VALORI E DA
OPERAZIONI SU TALI VALORI, PRESCINDENDO DAL MODO
IN CUI QUESTE STRUTTURE SONO EFFETTIVAMENTE
ORGANIZZATE E REALIZZATE.**



Astrazione e linguaggi

I LINGUAGGI DI PROGRAMMAZIONE CONSENTONO DI:

ASTRARRE SULLE FUNZIONI

CONCENTRANDO L'ATTENZIONE SU "COSA" FA UNA PARTICOLARE OPERAZIONE PIUTTOSTO CHE SUL "COME" E' FATTA.

SI ASTRAE DALLE MODALITA' DI REALIZZAZIONE, CI SI CONCENTRA SUL COMPITO O SULLE FUNZIONALITA'.

POICHE' **LA DECOMPOSIZIONE** DI UN PROBLEMA IN SOTTOPROBLEMI E' UN METODO SEMPLICE E INTUITIVO PER AFFRONTARE PROBLEMI COMPLESSI, LE PROCEDURE/FUNZIONI CONSENTONO DI ASTRARRE DALLE CARATTERISTICHE DEI SOTTOPROBLEMI CONCENTRANDO L'ATTENZIONE SULLA LORO FUNZIONALITA' E SULLA LORO INTERAZIONE.



Decomposizione e astrazione

- **necessaria quando si devono sviluppare programmi abbastanza grandi**
 - **decomporre il problema in sotto-problemi**
 - **i moduli che risolvono i sotto-problemi devono riuscire a cooperare nella soluzione del problema originale**
- **persone diverse possono/devono essere coinvolte**
 - **si deve poter lavorare in modo indipendente (ma coerente) nello sviluppo dei diversi moduli**
 - **deve essere possibile eseguire “facilmente” (da parte di persone diverse da quelle coinvolte nello sviluppo) modifiche e aggiornamenti (manutenzione) a livello dei singoli moduli, senza influenzare il comportamento degli altri**
- **i programmi devono essere decomposti in moduli, in modo che sia facile capirne le interazioni**



Decomposizione e astrazione

- **caratteristiche**
 - i sotto-problemi devono avere lo stesso livello di dettaglio
 - ogni sotto-problema può essere risolto in modo indipendente
 - una combinazione delle soluzioni ai sotto-problemi risolve il problema originale
- la decomposizione può essere effettuata in modo produttivo ricorrendo all'astrazione
 - cambiamento del livello di dettaglio, nella descrizione di un problema, limitandosi a “considerare” solo alcune delle sue caratteristiche
 - si passa ad un problema più semplice
 - su questo si effettua la decomposizione in sotto-problemi
- il passo astrazione-decomposizione si può ripetere più volte finché non si arriva a sottoproblemi per cui si conosce una soluzione



Il più comune tipo di astrazione

- ❑ l'astrazione procedurale
 - presente in tutti i linguaggi di programmazione
- ❑ la separazione tra “definizione” e “chiamata” rende disponibili nel linguaggio i due meccanismi fondamentali di astrazione
 - **l'astrazione attraverso parametrizzazione**
 - si astrae dall'identità di alcuni dati, rimpiazzandoli con parametri
 - si generalizza un modulo per poterlo usare in situazioni diverse
 - **l'astrazione attraverso specifica**
 - si astrae dai dettagli dell'implementazione del modulo, per limitarsi a considerare il comportamento che interessa a chi utilizza il modulo (ciò che fa, non come lo fa)
 - si rende ogni modulo indipendente dalle implementazioni dei moduli che usa



Astrazione procedurale

STIMOLA GLI SFORZI PER EVIDENZIARE OPERAZIONI RICORRENTI O BEN CARATTERIZZATE ALL'INTERNO DELLA SOLUZIONE DI UN PROBLEMA.

CONSENTE, ATTRAVERSO L'USO DI *SOTTOPROGRAMMI*, DI POTENZIARE IL LINGUAGGIO DISPONIBILE INTRODUCENDO *NUOVI OPERATORI* CHE FANNO USO DEGLI *OPERATORI BASE*, GIÀ DISPONIBILI.

- fornita da tutti i linguaggi ad alto livello
- aggiunge nuove operazioni a quelle del linguaggio di programmazione (la macchina astratta)
- la specifica descrive le proprietà della nuova operazione
- oscura il “come” è ottenuto il risultato
- stabilisce le modalità di comunicazione tra l'unità di programma creata e il resto del programma



Astrazione via parametrizzazione

- l'introduzione dei parametri permette di descrivere un insieme (anche infinito) di computazioni diverse con un singolo programma che le astrae tutte

$x * x + y * y$

- descrive una computazione

$\text{lx,y:int.}(x * x + y * y)$

- descrive tutte le computazioni che si possono ottenere chiamando la procedura, cioè applicando la funzione ad una opportuna n-upla di valori



Astrazione via specifica

la procedura si presta a meccanismi di astrazione più potenti della parametrizzazione

- possiamo astrarre dalla specifica computazione descritta nel corpo della procedura,
 - associando ad ogni procedura una **specific**a semantica “intesa” della procedura
 - e derivando la semantica della chiamata dalla specifica invece che dal corpo della procedura
- non è di solito supportata dal linguaggio di programmazione
 - se non in parte (vedi specifiche di tipo)
- si realizza con specifiche semi-formali
 - sintatticamente, commenti



Altri tipi di astrazione

Parametrizzazione e specifica permettono di definire vari tipi di astrazione

- **iterazione astratta**

- permette di iterare su elementi di una collezione, senza sapere come questi vengono ottenuti

- **gerarchie di tipo**

- permette di astrarre da specifici tipi di dato a famiglie di tipi correlati

- **tipi di dati astratti**

- si aggiungono nuovi tipi di dato a quelli della macchina astratta del linguaggio di programmazione

